Unpacking Developer Intentions: Assessing the Behavioural Factors Influencing ChatGPT Adoption in Software Development

الكشف عن نوايا المطورين: تقييم العوامل التي تؤثر على اعتماد شات جي بي تي في تطوير البرمجيات

by

MOHAMMAD KHALEEL JABER SALAMEH

Dissertation submitted in partial fulfilment

of the requirements for the degree of

MSc INFORMATICS

at

The British University in Dubai

November 2023

DECLARATION

I warrant that the content of this research is the direct result of my own work and that any use made in it of published or unpublished copyright material falls within the limits permitted by international copyright conventions.

I understand that a copy of my research will be deposited in the University Library for permanent retention.

I hereby agree that the contents of this dissertation for which I am author and copyright holder may be copied and distributed by The British University in Dubai for the purposes of research, private study or education and that The British University in Dubai may recover from purchasers the costs incurred in such copying and distribution, where appropriate.

I understand that The British University in Dubai may make a digital copy available in the institutional repository.

I understand that I may apply to the University to retain the right to withhold or to restrict access to my thesis for a period which shall not normally exceed four calendar years from the congregation at which the degree is conferred, the length of the period to be specified in the application, together with the precise reasons for making that application.

_____

Signature of the student

COPYRIGHT AND INFORMATION TO USERS

The author whose copyright is declared on the title page of the work has granted to the British University in Dubai the right to lend his/her research work to users of its library and as an open source and to make partial or single copies for educational and research use.

The author has also granted permission to the University to keep or make a digital copy for similar use and for the purpose of preservation of the work digitally.

Multiple copying of this work for scholarly purposes may be granted by either the author, the Registrar or the Dean only.

Copying for financial gain shall only be allowed with the author's express permission.

Any use of this work in whole or in part shall respect the moral rights of the author to be acknowledged and to reflect in good faith and without detriment the meaning of the content, and the original authorship.

ABSTRACT

Emerging AI tools such as ChatGPT have the potential to revolutionize software development. To provide a comprehensive study, our research combined two influential frameworks: The Unified Theory of Acceptance and Use of Technology 2 (UTAUT2) and the Task-Technology Fit (TTF). This convergence allowed for a comprehensive evaluation that encompassed individual behavioral motivations as well as the compatibility between the technology's capabilities and the particulars of development tasks. Our research methodology was multifaceted. Beginning with a Measurement Model Assessment to confirm our constructs, we moved on to a Structural Model Assessment to uncover underlying relationships. Using the capabilities of Artificial Neural Networks (ANN), we implemented additional Root Mean Squared Error (RMSE) and Sensitivity Analysis evaluations to enhance the accuracy of our insights. Ten-fold cross-validation was rigorously applied to a dataset containing 461 observations. Our comprehensive study sought to understand the factors influencing the adoption of ChatGPT in the software development domain. Central to our findings was the pivotal role of Performance Expectancy (PE), indicating developers' inclination towards tools that enhance their efficiency and streamline processes. Similarly, the importance of Social Influence (SI) underscored the collective nature of the developer community, where endorsements from peers or influential figures can significantly bolster adoption. Habit Behavior (HB) emerged as a defining factor, emphasizing the relevance of ingrained routines in the adoption of new technologies. Moreover, Task-Technology Fit (TTF) and its interplay with PE highlighted the importance of aligning AI tools with specific developer tasks to amplify expected outcomes. Conversely, factors traditionally deemed significant in technology

adoption, such as Effort Expectancy (EE), Facilitating Conditions (FC), and Hedonic Motivation (HM), did not exert a considerable influence in the context of ChatGPT.

خلاصة

أدوات الذكاء الاصطناعي الناشئة مثل ChatGPT لديها القدرة على احداث ثورة في تطوير البرمجيات. لتقديم دراسة شاملة، دمج بحثنا بين إطارين: النظرية الموحدة للقبول واستخدام التكنولوجيا 2 (UTAUT2) وملاءمة المهمة-التكنولوجيا .(TTF) هذا الدمج سمح بتقييم شامل شمل الدوافع السلوكية الفردية وكذلك التوافق بين قدرات التكنولوجيا وخصوصيات مهام التطوير. كانت منهجيتنا البحثية متعددة الأوجه. بدأنا بتقييم نموذج القياس لتأكيد سلامة بنية النموذج، ثم انتقلنا إلى تقييم النموذج الهيكلي لكشف العلاقات بين المتغيرات. باستخدام قدرات الشبكات العصبية الاصطناعية(ANN) ، قمنا بتنفيذ تقييمات إضافية للخطأ التربيعي الوسطي الجذري (RMSE) وتحليل الحساسية لتعزيز نتائج البحث. سعت دراستنا الشاملة لفهم العوامل التي تؤثر على تبني ChatGPT في مجال تطوير البرمجيات. كان لمعيار ال (PE) دور محوري في النتيجة النهائية، مما يشير إلى ميل المطورين نحو الأدوات التي تعزز كفاءتهم. بالمثل، أكدت أهمية التأثير الاجتماعي (SI) على الطبيعة الجماعية لمجتمع المطورين، حيث يمكن لتأييدات من الزملاء أو الشخصيات المؤثرة أن تعزز التبني بشكل ملحوظ. ظهر (HB) كعامل محدد، مؤكدًا على أهمية الروتين في تبني التقنيات الجديدة. علاوة على ذلك، أبرز ملاءمة المهمة-التكنولوجيا (TTF) وتفاعلها مع PE أهمية مواءمة أدوات الذكاء الاصطناعي مع مهام المطورين المحددة لتعزيز النتائج المتوقعة. على عكس المتوقع، لم تُظهر العوامل التي تعتبرمهمة في تبني التقنيات الجديدة مثل(EE) ، (FC)، (HM)، تأثيرا مهما في قرار التبني.

ACKNOWLEDGEMENTS

I wish to extend my heartfelt gratitude to my thesis mentors, Prof. Khaled Shaalan and Dr. Piyush Maheshwari. Their persistent guidance, unwavering support, and constructive feedback have been instrumental in shaping this research. Their expertise has consistently illuminated my path throughout this endeavor.

My deepest thanks go to my family and my wife whose steadfast support and encouragement have been my anchor during this rigorous journey. The warmth of their faith in me has been a continual source of strength and motivation. Special mentions to my classmates, Inas Abou Sharkha, Salam Al Sarayrah, and Ahmad Al Zammar for their assistance and insights.

My gratitude is also extended to the British University for equipping me with the essential tools and an environment conducive to scholarly research.

Lastly, my sincere appreciation goes out to all the participants who devoted their time and shared their invaluable insights. Their contributions have been the cornerstone of this research, making every discovery and conclusion possible

TABLE OF CONTENTS

Chapter 1: INTRODUCTION 1

1.1 Background 1

1.2 Research Motivation 4

1.3 Problem Statement 5

1.4 Research Objectives 6

1.5 Research Questions 7

1.6 Research Model 7

Chapter 2: LITERATURE REVIEW 9

2.1 Overview 9

2.2 AI in Requirements Gathering, and Software Design 11

2.3 AI in Code Generation and Development 14

2.4 AI in Software Testing & Quality 16

2.5 ChatGPT in SDLC 22

2.6 Future Trends and Challenges in AI for SDLC 26

Chapter 3: RESERACH MODEL AND HYPOTHESES DEVELOPMENT…..... 31

3.1 Research Model 31

3.2 Research Hypothesis 35

3.2.1 UTAUT2 Hypothesis 35

3.2.2 Task Technology Fit (TFF) Hypothesis 40

Chapter 4: RESERACH METHODOLOGY 45

4.1 Data Collection and Method 45

4.2 Research Instrument Development 46

4.3 Data Analysis 52

Chapter 5: RESULTS 55

5.1 Descriptive Analysis 55

5.1.1 Distribution of Males and Females 56

5.1.2 Age Distribution Analysis 57

5.1.3 Educational Perspectives 58

5.1.4 Professional Function Assignment 59

5.2 Measurement Model Assessment 60

5.2.1 Construct Reliability (Internal Consistency) 61

5.2.2 Convergent Validity 62

5.2.3 Discriminant Validity 64

5.3 Structural Model Assessment 72

5.3.1 Assessment of the Path Coefficient 72

5.3.2 Assessment of the Coefficient of determination (R2) 74

5.3.3 Assessment of the Effect Size (f2) 75

5.3.4 Assessment of the Predictive Relevance (Q2) 76

5.4 Importance-Performance Map Analysis (IPMA) 78

5.5 Artificial Neural Network (ANN) 80

5.6 Sensitivity Analysis 85

Chapter 6: Discussion 88

6.1 Overview 88

6.2 UTAUT2 Factors Effects 89

6.3 Joint Effects Between TTF and UTAUT2 94

Chapter 7: Conclusion 98

7.1 Research Summary 98

7.2 Key Findings 99

7.2.1 ChatGPT and AI Tools Support SDLC 100

7.2.2 UTAUT2 & TTF Integration Introduced an Effective Adoption Model 100

7.2.3 Main Factors Influence Developers Adoption 101

7.2.4 Model Accuracy 102

7.2.5 Organizations Adoption Strategies 102

7.3 Theoretical Contributions 104

7.4 Practical Implications 105

7.5 Limitations and Future work 106

LIST OF TABLES

Table 1: Research Instrument's Constructs and Related Items 48

Table 2: Participants Demographics 55

Table 3: Convergent Validity, Internal Consistency, and Reliability results 63

Table 4: Discriminant Validity (Fornell-Larcker Criterion Test Results) 66

Table 5: Discriminant Validity (Cross-Loading) 68

Table 6: Discriminant Validity (HTMT) 71

Table 7: Path Coefficient result 73

Table 8: Coefficient of determination (R2) result 75

Table 9: F2 Results 76

Table 10: Prediction Summary Result 77

Table 11: Importance-Performance Map Analysis (IPMA) Assessment 79

Table 12: RMSE values of ANN models 81

Table 13: Sensitivity analysis for model 1 86

LIST OF FIGURES

Figure 1: Software Development Life Cycle 1

Figure 2: Conceptual Framework 34

Figure 3: Participants Gender 57

Figure 4: Age Groups 58

Figure 5: Education Level 59

Figure 6: Participants Roles 60

Figure 7: Importance-Performance Map Analysis (IPMA) Assessment 80

Figure 8: ANN Model 1 83

Figure 9: ANN Model 2 83

Figure 10: ANN Model 3 84

Figure 11: ANN Model 4 84

Figure 12: Sensitivity Analysis 87

LIST OF ABBREVIATIONS

Abbreviation	Meaning
SDLC	Software Development Life Cycle
AI	Artificial intelligence
GPT	Generative Pre-trained Transformer (GPT)
NLP	Natural Language Processing
DDS	Design Document Specification (DDS)
SRS	Software Requirements Specifications (SRS)
CI	Computational Intelligence (CI)
CM-Builder	Class-Model Builder (CM-Builder)
NL	Neural Network
CBR	Case-Based Reasoning (CBR)
LLMs	Language Learning Models (LLMs)
DOL	Division of Labor (DOL)
DL	Deep Learning (DL)
ANN	Artificial Neural Networks (ANN)
RNNs	Recurrent Neural Networks (RNNs)
SVR	Support Vector Regression (SVR)
HGA	Hybrid Genetic Algorithms (HGA)
GUIs	graphical user interfaces (GUIs)
SVM	Support Vector Machines (SVM)
ANI	Artificial Narrow Intelligence (ANI)
AST	Abstract Syntax Tree (AST)

UTAUT2	Unified Theory of Acceptance and Use of Technology 2 (UTAUT2)
BI	Behavioral Intention (BI)
PE	Performance Expectancy (PE)
EE	Effort Expectancy (EE)
SI	Social Influence (SI)
FC	Facilitating Conditions (FC)
HM	Hedonic Motivation (HM)
HB	Habit Behavior (HB)
TTF	Task Technology Fit (TTF)
TC	Technology Characteristics (TC)
TaskC	Task Characteristics (TaskC)
SEM-ANN	Structural Equation Modeling-Artificial Neural Network (SEM-ANN)
SEM	Structural Equation Modeling (SEM)
PLS-SEM	Partial Least Squares-Structural Equation Modeling (PLS-SEM) techniques.
CA	Cronbach's Alpha (α)
CR	Composite Reliability (CR)
AVE	Average Variance Extracted (AVE)
HTMT	Heterotrait-Monotrait Ratio of Correlations (HTMT)
IPMA	Importance-Performance Map Analysis (IPMA)
RMSE	Root Mean Square Error (RMSE)
NI	Normalized importance

Chapter 1: Introduction

1.1 Background

Software development has become an integral part of contemporary society, with applications spanning business, entertainment, healthcare, and more. With increasing project complexity and demands for rapid development, it has become increasingly essential to investigate innovative software development life cycle (SDLC) optimization strategies. As shown in Figure 1, SDLC is a structured process that includes planning, design, implementation, testing, deployment, and maintenance to ensure the successful development and delivery of software applications.

Figure 1: Software Development Life Cycle

(Source: https://scoutapm.com/)

Numerous methodologies, including the waterfall model, agile development, and DevOps, have emerged over time to enhance the efficiency and effectiveness of the SDLC. However, due to the increasing complexity of modern software systems and the growing demand for high-quality applications, there is an ongoing need for innovative tools and methods to further improve the software development process.

Recent advancements in artificial intelligence (AI), particularly in natural language processing, have created new opportunities for AI-driven software development tools and methodologies. The increasing interest in AI-driven software development tools can be attributed to their potential to increase productivity and decrease the likelihood of human error. AI-powered tools can, for instance, assist developers in producing code from natural language descriptions, identifying defects and vulnerabilities in the code, and even recommending software architecture. However, the adoption of AI-powered software development tools come with obstacles. For instance, developers may encounter a steep learning curve when integrating these tools into their operations, and there may be concerns regarding the accuracy and dependability of AI-generated outputs. Moreover, the perceived utility and usability of AI-powered tools and the fear from getting them replaced by AI tools can have a significant impact on their adoption among developers (Kumari et al., 2018). ChatGPT, which stands for Generative Pre-trained Transformer (GPT), launched in November 2022 is one of these tools. It is a potent NLP model developed by OpenAI. It is based on the GPT-4 architecture, a cutting-edge language model that demonstrates human-like text comprehension and generation. ChatGPT is trained with a large corpus of text data and is able to generate coherent and contextually pertinent responses, making it an ideal tool for a variety of natural language understanding and generation tasks (Feng et al., 2023).

Given ChatGPT's advanced comprehension and generation of human-like text, it is plausible that its integration into the SDLC could result in significant enhancements to the software development life cycle. ChatGPT has the potential to revolutionize several facets of the SDLC, including requirements elicitation and analysis, design and system analysis, code generation, documentation and code commenting, test case generation and testing automation, bug detection and code review. In requirements elicitation and analysis stage, ChatGPT can assist developers in understanding and interpreting the requirements specified by stakeholders by identifying ambiguities, inconsistencies, and redundancies in the requirements documentation. This enables developers to clarify and refine the specifications prior to moving on to the design and implementation phases (Sharma et al., 2019). ChatGPT can give suggestions for system architecture, design patterns, and data flow diagrams during design and system analysis. This capability can help developers make informed design decisions and streamline the design process as a whole. ChatGPT can be used for code generation to generate code fragments or entire modules based on natural language descriptions of the desired functionality (Feng et al., 2023). This can substantially reduce the time and effort required for manual coding, allowing developers to concentrate on more intricate and creative aspects of software development. ChatGPT can generate meaningful documentation and code remarks based on the existing code, ensuring that developers and maintainers have a complete understanding of the system's functionality and structure. In terms of test case generation and testing automation, ChatGPT can generate test cases and testing scripts based on system requirements and specifications, thereby reducing the amount of manual effort required for test cases development. In addition, it can help automate the testing process by producing test scripts that are compatible with testing frameworks and tools. ChatGPT can be used in bug detection

and code review to identify potential software defects, and vulnerabilities by analyzing the codebase and comparing it to best practices and known issue patterns. This capability can assist developers in proactively addressing prospective issues prior to their escalation into critical problems (Geliştirme, 2021).

1.2 Research Motivation

The integration of Artificial Intelligence (AI) and Natural Language Processing (NLP) technologies into software development practices has garnered substantial attention in recent years. One such AI-driven tool, ChatGPT, has emerged as a promising candidate for enhancing various aspects of the software development lifecycle. However, understanding the factors that influence its adoption within the software development community remains a critical research endeavor. Motivated by the rapid evolution of AI technologies and their potential to reshape traditional software development practices, this research seeks to investigate and measure the adoption of ChatGPT. In doing so, we aim to address several key motivations:

Advancing AI in Software Development: The software development landscape is constantly evolving, with AI technologies at its forefront. Understanding how ChatGPT, a powerful language model, can be effectively incorporated into this environment has the potential to advance AI's role in the industry.

Enhancing Productivity: ChatGPT has the capability to streamline various software development tasks, from code generation to requirement analysis. By comprehensively assessing its adoption, we can uncover opportunities to enhance developer productivity.

Identifying Adoption Barriers: Potential barriers, such as concerns about job displacement, technical complexities, or the alignment of ChatGPT with software development tasks, need to be identified and addressed. This research aims to shed light on these barriers.

Theoretical Contributions: The study draws on established models such as the UTAUT2 (Unified Theory of Acceptance and Use of Technology 2) and TTF (Task-Technology Fit) model, providing a theoretical framework for understanding technology adoption. By applying these models, we contribute to the theoretical foundation of technology adoption research.

Practical Insights: Findings from this research can offer practical insights to software development organizations, guiding their decisions on the integration of ChatGPT and similar AI tools.

Preparing for the Future: As AI continues to evolve, staying ahead of the curve is essential. This research equips software development professionals and organizations with knowledge about adopting cutting-edge AI technologies.

1.3 Problem Statement

As AI-driven tools like ChatGPT continue to advance, it is essential to investigate their potential software development life cycle applications. Existing empirical evidence regarding the effectiveness of ChatGPT in software development tasks is limited. This research seeks to provide valuable data on ChatGPT's performance in software development from software developers point view, which can inform the decisions of organizations considering the adoption of AI-powered development tools. Understanding the factors that influence software developers' acceptance and use of ChatGPT is crucial for integrating this technology into existing workflows. This research will provide insights into the main adoption drivers and barriers, allowing organizations to develop strategies to overcome obstacles and promote the successful implementation of ChatGPT. The results of this study will contribute to the expanding body of knowledge on AI-driven software development tools, informing both the tool's creators and

organizations that employ them. This information can help shape the future development of ChatGPT and similar AI-driven tools, ensuring that they effectively address the industry's requirements and challenges. Organizations that successfully adopt and integrate AI-powered tools, such as ChatGPT, into their software development processes may obtain a competitive advantage in terms of software development efficiency, productivity, and quality. By understanding the potential advantages and disadvantages of adopting ChatGPT, organizations can make well-informed decisions to remain competitive in a market that is becoming increasingly intense.

1.4 Research Objectives

Ultimately, the expected outcomes of this research endeavor aim to accomplish the following objectives:

- Explore the prospective applications of AI tool and ChatGPT throughout the SDLC, including requirements elicitation, design, coding, documentation, and testing.
- Determine the factors influencing the acceptance and utilization of ChatGPT among software developers.
- Provide recommendations for organizations contemplating the incorporation of ChatGPT in their software development processes, addressing the critical success factors and potential obstacles.
- Contribute to the expanding corpus of knowledge on AI-powered software development tools, informing both the tool's creators and the organizations that adopt them.

These findings could inspire additional innovation in the field of AI-driven software development tools, resulting in the creation of new approaches and methodologies to improve the software development life cycle.

1.5 Research Questions

The following questions should be addressed by the research in order to accomplish the research objectives:

- **Q1**: How can AI tools and ChatGPT be effectively utilized across different phases of the Software Development Life Cycle (SDLC), including requirements elicitation, design, coding, documentation, and testing?
- **Q2**: What is the effective model that captures developers' intention to use ChatGPT in software development?
- **Q3**: What are the main factors that influence software developers' acceptance and utilization of ChatGPT?
- **Q4)** To what extent the proposed model will able to explain the determinants of ChatGPT adoption in software development?
- **Q5**: What strategies can organizations employ to overcome adoption barriers and promote the successful implementation of ChatGPT in software development?

1.6 Research Model

This research will integrate the Unified Theory of Acceptance and Use of Technology 2 (UTAUT2) and the Task-Technology Fit (TTF) model to provides a comprehensive framework for analyzing the factors that influence the acceptance and use of ChatGPT in software development. Focusing on performance expectancy, effort expectancy, social influence, facilitating conditions, hedonic motivation, and habit, the UTAUT2 model provides an improved comprehension of the factors that influence users' behavioral intentions for

technology. By incorporating these constructs, the research can encompass the various aspects of technology adoption from the standpoint of software developers (Venkatesh et al., 2012).

The TTF model, on the other hand, emphasizes the significance of aligning the capabilities of technology with the specific tasks and needs of users (Goodhue et al., 1995). In the context of software development, the TTF model can help determine the compatibility of ChatGPT's features and capabilities with the requirements and challenges encountered by developers. By evaluating the compatibility between ChatGPT's functionality and the tasks performed by developers, the research can shed light on the potential benefits and drawbacks of adopting ChatGPT in software development tasks.

Combining the UTAUT2 and TTF models permits a more nuanced comprehension of ChatGPT's adoption in software development. The integrated model captures both the individual and organizational factors that influence the acceptance and use of ChatGPT, as well as the alignment between the capabilities of the technology and the particular tasks conducted by software developers. By analyzing the relationships between these factors, the research will be able to identify the primary drivers and barriers for ChatGPT's adoption and use in software development as confirmed in similar research (Sharif et al., 2019; Beier et al., 2020; Faqih et al., 2021). In addition, this integrated approach can assist organizations considering the adoption of ChatGPT receive more targeted recommendations. By understanding the factors that influence acceptance, organizations can develop strategies to resolve potential obstacles, facilitate the successful integration of ChatGPT into existing workflows, and maximize the benefits derived from its use. Finally, the insights generated by this integrated model can inform the future development of ChatGPT and similar AI-driven tools, ensuring that they address the requirements and challenges of the software development industry.

Chapter 2: Literature Review

2.1 Overview

This lecturer review investigates the transformative impact of artificial intelligence (AI) on the Software Development Life Cycle (SDLC), with a particular emphasis on the prospective contributions of OpenAI's ChatGPT. Through an in-depth analysis of recent scholarly literature, this article endeavors to reveal the pivotal role that AI, and specifically ChatGPT, can play in revolutionizing the various stages of SDLC. According to (Dong et al., 2023), software development is a complex process that requires cooperation and collaboration. This process, which is traditionally divided into stages such as requirement analysis, planning & design, coding, testing, deployment, and maintenance, is structured by methodologies such as Waterfall, Agile, and DevOps. On the other side, (Arora et al., 2021) examined the complexities of software development life cycles (SDLC) and how computational strategies, particularly artificial intelligence (AI), can be employed to simplify them, they recommended employing AI at various phases of the SDLC, including software requirement prioritization, cost estimation, defect prediction, vulnerability prediction, and size prediction. The same has been confirmed by (Kumari et al., 2018) when they explored the intersection of Artificial Intelligence (AI) and Software Engineering, highlighting the evolving relationship between these two fields. They emphasized the convergence of artificial intelligence and software engineering, which has resulted in the emergence of new research fields, they examined the application of AI techniques throughout the SDLC's numerous phases. Initially, during the Software Planning and Requirements Analysis phase, AI contributes to accurate decision-making by assisting with project planning and feasibility studies. Next, during the Defining Requirements phase, AI plays a crucial role in generating precise Software Requirement Specification (SRS) documents,

ensuring a thorough comprehension of the software's functional and non-functional requirements. According to (Kumari et al., 2018), AI techniques facilitate the development of optimal software architectures based on previously defined SRS. AI also contributes to the compilation of Design Document Specification (DDS)-compliant programming code during the Constructing or Developing the Product phase, thereby improving the efficiency and accuracy of software development. Moving forward, during the Testing the Product phase, AI techniques aid in the identification and correction of product defects, ensuring conformity with the SRS-specified quality standards. Lastly, during Product Market Deployment and Maintenance, AI plays a crucial role in staging the deployment of software, conducting User Acceptance Testing (UAT), and providing ongoing maintenance and support to existing consumers (Kumari et al., 2018).

This review's key findings and themes below answer the first research question on how can AI tools and ChatGPT be effectively utilized across different phases of the SDLC (Q1):

- **AI in Requirements Gathering, and Software Design**: Discuss how AI technologies can improve the process of gathering software requirements. Explore the role of AI in aiding the design and planning phases of SDLC, from creating more efficient software architectures to streamlining development sprints.
- **AI in Code Generation and Development**: Discuss how AI can assist in the coding phase of SDLC, including automated code generation, efficient debugging, and optimization.
- **AI in Software Testing & Quality**: Highlight how AI technologies can contribute to effective and efficient software testing, and security testing from generating test cases to predictive bug detection.

- **ChatGPT in SDLC**: To delve into the specific role of ChatGPT in SDLC, outlining how this particular model could be instrumental in transforming software development.
- **Future Trends and Challenges in AI for SDLC**: Explore the potential future developments and challenges of AI in SDLC, considering aspects such as continual learning, model interpretability, and the need for human oversight.

2.2 AI in Requirements Gathering, and Software Design

(Kharade et al., 2023) have examined the function and potential of artificial intelligence (AI) in software requirement engineering. Requirement engineering is concerned with software requirements elicitation, analysis, specification, validation, and management throughout the software product life cycle, with AI offering promise improvements to this process. (Belani et al., 2019; Habib, 2019; Naikwadi et al., 2021) have explained how AI techniques such as morphological analysis, tokenization, and grammar extraction can be advantageous for analyzing and managing software requirements. Besides that, (Giray, 2021) discussed how machine learning (ML) has changed the face of software development, shifting it from the traditional hard-coded algorithm model to systems created by learning from data. (Kharade et al., 2023) have indicated that AI's application in requirement engineering may present obstacles, particularly when used in the systematic method for developing AI-based software, which was the primary focus of their paper.

(Arora et al., 2021) have discussed the role of AI in Software Requirements Specifications (SRS) in software design. They explained that SRS is a document that describes the capabilities and limitations of a software system and how the SRS document plays a vital role in ensuring that the software development meets its intended requirements. They presented the concept of Computational Intelligence (CI), which refers to the use of heuristic computations

to solve complex problems in a reasonable amount of time. They proposed using Case-Based Reasoning (CBR), an artificial intelligence (AI) technique that solves novel problems based on past experiences to improve the quality of SRS. This method can substantially accelerate the SRS quality investigation process due to the fact that many tasks are similar to those that have already been completed. (Arora et al., 2021)contend that artificial intelligence could be instrumental in addressing these issues by translating natural language requirements into formal specifications, as is the case with frameworks such as TELL and FORSEN. Other noteworthy frameworks include LOLITA and the Class-Model Builder (CM-Builder), which are capable of creating OO models from NL requirements. Moreover, (Arora et al., 2021) highlighted the role of AI in software architecture design, noting the use of Genetic Algorithms and Constraint Programming in defining system structure, and how these approaches can improve software quality properties such as isolation, complexity, changeability, understandability, and reusability.

According to (Sharma et al., 2013; Sharma et al., 2019), the incorporation of AI techniques into requirement elicitation offers substantial potential for overcoming the limitations and complexities of conventional approaches. By employing AI techniques such as Keyword Mapping, Natural Language Processing, Ontology, Speech Understanding, Neural Networks, Expert Systems, Sketch-Based Modeling, Knowledge Level Ethnography, Case-Based Reasoning, Intelligent Agents, Knowledge-Based Systems, Fuzzy Logic, and an Integrating Model, practitioners can improve the precision, thoroughness, and clarity of requirements. These AI techniques enhance communication, analysis, and context-awareness, resulting in more successful software development projects and a greater likelihood of meeting user needs and expectations according to (Sharma et al., 2013; Sharma et al., 2019). In their

comprehensive analysis of the requirement elicitation phase in software development, they examined the difficulties and complexities associated with collecting accurate and exhaustive requirements. They discussed various requirement elicitation techniques, including communicative, collaborative, observational, cognitive, and innovative methods. However, they emphasized that these traditional techniques are not exempt from limitations and challenges, such as ambiguity, communication barriers, diverse interpretations, and resource planning issues. To surmount these obstacles and enhance the efficacy of requirement elicitation, Sharma et al. proposed incorporating artificial intelligence (AI) techniques. They identified particular AI methodologies that can overcome the limitations of conventional techniques and improve the quality of requirements. These AI techniques include Keyword Mapping, which assists in comprehending the "observable" words that are crucial to system requirements. By extracting meaning from textual data, Natural Language Processing (NLP) enables greater comprehension and analysis of requirements. According to (Sharma et al., 2013; Sharma et al., 2019), Ontology-based approaches facilitated the semantic representation and organization of knowledge, thereby facilitating the specification of precise requirements. On other side, Speech Understanding Methodology enabled more efficient capture and interpretation of verbal specifications. These techniques offered a novel approach for identifying user needs, comprehending the environment, and deriving context-specific requirements.

Overall, adopting Artificial Intelligence (AI) in the Software Development Life Cycle (SDLC) has the potential to improve requirements gathering and software design. As noted by (Sharma et al., 2013; Sharma et al., 2019; Arora et al., 2021; Kharade et al., 2023), artificial intelligence mitigates obstacles such as ambiguity, communication barriers, and diverse interpretations. AI enhances the precision, clarity, and quality of Software Requirements

Specifications (SRS) by utilizing techniques such as Natural Language Processing (NLP), Case-Based Reasoning (CBR), and Computational Intelligence (CI). Moreover, AI-driven frameworks are revolutionizing the design of software architecture, thereby improving software quality characteristics.

2.3 AI in Code Generation and Development

(Dong et al., 2023) provided a thorough overview of code generation, highlighting its significance in software development and artificial intelligence research. A self-collaboration paradigm for code generation based on large language models (LLMs) such as ChatGPT has been proposed by them. Focusing on functionality, readability, and maintainability, this framework incorporated additional software development phases into the code generation process to improve the quality of the final product. They have examined the self-collaboration framework in order to produce the desired output based on provided requirements. The framework included two major components: Division of Labor (DOL) and collaboration, complex tasks are decomposed into a series of phases and various roles are assigned based on LLMs and role instructions in DOL. The use of role instructions facilitates the assignment of identities and responsibilities to LLMs, thereby enhancing the clarity and effectiveness of subsequent communications and collaborations. In order to illustrate a practical application of the proposed framework, They presented an example of a virtual team employing the cascade model from software development methodology. The team consists of three roles: an analyst, a programmer, and a tester, each of whom is responsible for a distinct phase of software development. The method ensured that problems discovered at any stage can be communicated to the antecedent stage for correction, thereby preserving the overall quality of the software development process. They have compared their methods to a number of cutting-edge models,

including CodeX, CodeT, PaLM Coder, GPT-4, and ChatGPT. They demonstrated that the self-collaboration framework substantially enhanced the performance of the base LLM, outperforming even GPT-4. The datasets with extended test cases, namely HumanEval-ET and MBPP-ET, demonstrated a significant relative improvement in performance. They indicated that their strategy could be enhanced by employing a more potent model, such as GPT-4 (Dong et al., 2023).

On the other side, (Priya et al., 2017) have investigated the use of Deep Learning (DL) and Artificial Neural Networks (ANN) for automated code generation and programming language comprehension in their research. Due to the inherent complexities and structures of various programming languages, they identified this as a unique difficulty. They have assessed the efficacy of Recurrent Neural Networks (RNNs) in automatic code generation. This evaluation took into account three crucial factors: the programming language used for RNN training, the extent of the training dataset, and the training time. Java, Python, and C# are the three distinct programming languages chosen by the researchers, each with its own syntax and structure. In addition, datasets of varying sizes were utilized, and each language's training spanned over 20 iterations. Using a character-based Recurrent Neural Network (RNN) model, a series of exhaustive experiments were conducted to generate code fragments in Java, Python, and C#. The model was trained on datasets of diverse sizes and evaluated at various training stages to determine its performance in relation to programming language, dataset size, and training duration. Multiple hardware platforms were used for model implementation, training, and computational resources, along with Python and Tensorflow to conduct the experiments. The training datasets were meticulously compiled from prominent Java, Python, and C# Github repositories, and included both large and small datasets. Each training data character was

encoded as an integer, and the RNN model parameters were consistently defined across all experiments. Multiple training sessions were conducted, and Tensorboard was used to evaluate the model's development based on batch accuracy and loss measurements. The results indicated that the Java and C# models performed better than the Python model, and there was little difference in performance between large and small datasets. The efficacy of the automatic code generation for all languages was satisfactory, with Java and C# producing superior code. However, none of the languages were capable of producing code devoid of errors (Priya et al., 2017).

Overall, AI is considerably revolutionizing code generation and software development by providing productive and high-quality innovative models. These models, such as large language models (LLMs), can play multiple roles, including those of an analyst, programmer, and tester, and promote effective collaboration to enhance output. In addition, the use of deep learning and neural networks in automated code generation paves the way for software that can comprehend multiple programming languages. However, while these developments are promising, there are still obstacles to generating error-free code. In conclusion, the intersection of artificial intelligence with code generation and development presents transformative opportunities and challenges, and has the potential to profoundly influence the future of software development.

2.4 AI in Software Testing & Quality

(Hourani et al., 2019) described the necessity of software testing, which ensures that business requirements are met, resulting in customer satisfaction and a productive software development lifecycle. They have provided a comprehensive overview of the intersection between artificial intelligence (AI) and software testing, concentrating on the roles of machine

learning (ML) and natural language processing (NLP) in this context. They contend that AI and its fundamental components, such as machine learning and natural language processing, can substantially enhance software testing. Utilizing these techniques can result in time savings, improved precision, and intelligent test case generation. This can ultimately lead to improved coverage of the software being tested, resulting in a higher ROI. The researchers have discovered AI algorithms and techniques applied in software testing, these algorithms and techniques demonstrated that AI has the potential to enhance various aspects of the testing process, thereby addressing obstacles and enhancing productivity. As documented by (Hourani et al., 2019), C4.5 decision tree algorithms were used to enhance black-box test specifications and category-partition test specifications. In Huber Regression, Support Vector Regression (SVR), and multi-layer perceptron techniques were used to predict automated testing coverage, using these algorithms. They also investigated the use of Hybrid Genetic Algorithms (HGA) to autonomously test graphical user interfaces (GUIs) by optimizing test sequences and test case selection. These algorithms intended to streamline GUI testing procedures, increase test coverage, and reduce the number of duplicate test cases. In the context of regression testing, K-means clustering was utilized to classify test cases and improve the efficacy of regression testing, by categorizing test cases based on their similarity, the researchers sought to identify representative test cases that span various code segments, thereby enhancing the efficiency of regression testing. General classification techniques, such as Support Vector Machines (SVM) and others, were used in software fault prediction to identify potential software faults, these algorithms were used to analyze code and identify patterns that could indicate the presence of faults, thereby facilitating the proactive detection and mitigation of faults. Also, Support Vector Machines (SVM) and induced grammars were used to identify GUI test cases that are impossible

to execute, using these techniques, (Hourani et al., 2019) intended to identify GUI test cases that were impractical or impossible, thereby enabling more targeted testing effort. In system-level testing without code access, Support Vector Machine (SVM) RANK was used to prioritize test cases, without access to the underlying code, this method intended to optimize testing efforts by prioritizing test cases based on their potential impact on the system. In Logistic regression, random forest, Adaboost, and bagging algorithms also focused on optimizing testing efforts based on change propensity.

The generation of test data was investigated (Hourani et al., 2019) using genetic algorithms, this strategy intended to generate test data automatically in order to achieve maximum coverage and increase the diversity of test cases, thereby enhancing the effectiveness of testing. Also, the K-nearest neighbor algorithm was utilized in the identification of coincidentally correct test cases, this strategy intended to identify instances in which test results passed coincidentally, highlighting potential flaws or limitations in the testing procedure. In numerous investigations, NLP (Natural Language Processing) techniques were widely employed in test case prioritization, manual test case failure prediction, and test case generation from software requirements. AI has demonstrated its ability to deliver superior results in software testing, and it will likely lead a new era of QA work in the future. They anticipated that the influence of AI will improve software quality and positively impact customer experience by delivering reliable, error-free applications and solutions (Hourani et al., 2019).

The same has been confirmed by (Hossain et al., 2022), they emphasized the significance of machine learning in software testing, highlighting the application of both early and contemporary techniques in this domain. They have discussed various ML techniques, including neural networks for pruning test case sets, two-layered artificial neural networks as automated

test oracles, the use of machine learning to analyze software code and identify problems early in the project lifecycle, and the use of SVM for predicting defect-prone software modules, Bayesian reasoning methods for software testing and program analysis, semi-supervised clustering methods to enhance test selection, and the application of machine learning techniques to predict aging-related bugs. They have confirmed that ML can effectively aid software testing processes, assisting in defect prediction, fault localization, specific bug identification, and overall testing efficacy and efficiency improvement. They found that the Neural Network is widely used ML technique, with applications including fault-proneness prediction, automatic test oracles generation, execution effort estimation, metamorphic testing, software defect prediction, and identifying code fragments for manual review. Similarly, Support Vector Machines (SVM) have been extensively utilized for generating reliable test oracles, fault-proneness prediction, execution effort estimation, GUI testing, and mutation testing. Bayesian Based Methods and Random Forest were utilized for the combination of testing techniques, software defect prediction, and software quality prediction, these techniques were used for software defect prediction, software fault localization, and software quality prediction (Hossain et al., 2022).

On the other side, (Battina,2019) examined the role of artificial intelligence (AI) in the automation of software testing. He stressed that software testing is essential for the success of software and applications, and that as technology advances rapidly, manual testing becomes time-consuming and expensive, making test automation essential. (Battina,2019) discussed in his paper how AI is still in its infancy in software development, but has the potential to automate mundane duties and simplify the software development lifecycle. In addition, he identified a significant issue with test automation such as test maintenance, which is estimated to consume

40% of testers' time. He also focused on the current issues in testing caused by a lack of intelligence and excessive human involvement, which leads to ineffective test sessions. According to (Battina,2019), Model-Inference driven testing (MINTest), AutoBlackTest, AimDroid, Vista, Sikuli Test, Testilizer, and SwiftHand are AI-enhanced software tools automate distinct facets of software testing, such as GUI testing, self-diagnosis and repair, and automated compilation of test scenarios. The researcher concluded that the significance of these tools in the software testing process can help automate repetitive tasks, thereby reducing software testing's time, cost, and resources requirements (Battina,2019).

Another researcher, (Chen et al., 2022) focused on the testing and the automation as well. they claimed that machine learning plays a crucial role in software testing and automation by reducing effort and increasing effectiveness and efficiency. They provided a comprehensive analysis of the current landscape of machine learning techniques in software testing, encompassing the aforementioned testing approaches in a variety of testing scenarios. These methods included neural networks, support vector machines (SVM), clustering, decision trees, grammar induction, Bayesian methods, and random forest. Neural networks have applications in fault-proneness prediction, automated test oracle generation, software testing effort estimation, metamorphic testing, software defect prediction, and code fragment identification. SVMs are frequently used for generating trustworthy test oracles, predicting fault-proneness, estimating testing execution effort, GUI testing, and mutation testing. Clustering has been used to enhance regression test selection, generate automated test oracles, and classify test outcomes. Decision trees have been applied to defect prediction, software quality enhancement, and regression testing improvement. GUI testing has found applicability for grammar induction. In order to combine testing techniques, predict software defects, and forecast software quality,

Bayesian methods have been utilized. Last but not least, random forest has been applied to defect forecasting, fault localization, and software quality prediction.

(Geliştirme, 2021) demonstrated the disruptive potential of sophisticated machine learning and artificial intelligence (AI) applications on the field of cybersecurity and code security. He focused on dynamic code analysis, which plays a crucial role in determining the level of source code security. He proposed a framework employed ML algorithms, specifically artificial neural networks and their variants, this methodology permits a thorough evaluation of the security characteristics of C# code. The importance of securing code during the software development process in preventing vulnerabilities is emphasized. The researcher investigated the applications of Artificial Narrow Intelligence (ANI) in software source codes, these applications included calculating code complexity and quality, estimating software effort and time, determining code authorship, detecting similarities between source codes for security purposes, and even automating source code generation.

Overall, the incorporation of AI into software testing and quality assurance is proving revolutionary, expediting processes, boosting productivity, and enhancing precision. The use of machine learning techniques and AI tools has demonstrated significant promise, automated numerous testing aspects and reducing overall resource requirements. In addition, the impact of AI on code security highlights the need for secure development practices to prevent vulnerabilities. In conclusion, AI is considerably influencing the future of software testing and quality assurance, presenting promising innovations that could transform the software development life cycle.

2.5 ChatGPT in SDLC

(Paik et al., 2021) explored the application of GPT-2, a transformer-based language model, for automated code generation. The authors aim to enhance the efficiency and performance of code generation by integrating code graphs with varying features into GPT-2. They highlighted the challenges and limitations associated with using BERT for code generation, including the lack of convenience and the need for a seq2seq architecture. In contrast, they observed that GPT-2, specifically its decoder portion, demonstrated exceptional context capturing ability and semantic information management, making it a promising tool for code generation. For that (Paik et al., 2021) utilized the GPT-2 model, which consists of encoding and decoding components, the encoder model is based on BERT, while the decoder model is derived from OpenAI's GPT-2. Their study consisted of two experimental phases to evaluate the proposed method. In the pre-training phase, a model without a code graph is employed, while the fine-tuning phase introduced the code graph and assessed its impact on code generation (Paik et al., 2021). The efficacy of both the pre-trained and fine-tuned models is evaluated, (Paik et al., 2021) noticed that the inclusion of large datasets during model training significantly improved the performance of the transformer model and its derivatives, particularly in text generation. The CodeSearchNet dataset provided by CodeXBLUE is employed for training and testing purposes. This dataset comprises various code-related items, including text-to-text, text-to-code, code-to-text, and code-to-code examples. They described their data preprocessing procedures, which involve limiting the duration of comments and codes and removing the code's structure. Data organization is adjusted to accommodate comment and code duration restrictions, resulting in a reduced training dataset. In addition, They presented the model architecture of GraphCodeGPT, which builds upon GPT-2 as the base model, this

architecture utilizes code as natural language, employing a pre-trained language model with a data flow graph that resembles the Abstract Syntax Tree (AST) structure, the model leverages source code and its corresponding data flow to generate textual remarks and sequences of variables (Paik et al., 2021).

(Perez et al., 2021) worked on GPT-2 too, they conducted research on automating software development in response to the rising demand for software professionals. They proposed a machine learning model for code generation, with a focus on assisting developers in the creation of discrete units of functionality known as "functions.". Using a pre-trained language model on a corpus of Python code from GitHub, they intended to generate the function body based on the initial code and doc-string. CodeSearchNet dataset has been used, which consists of millions of (comment, code) pairings from open-source libraries written in a variety of programming languages. Python was selected as the target language due to its ease of use for transfer learning and large dataset size, thereby guaranteeing a single-language domain. Several models were evaluated, including CodeSearchNet baselines and a sequence-to-sequence RNN model. The latter encoded each character using one-hot encoding and employed a transformer architecture suitable for managing long-range dependencies, using the transformer architecture, they fine-tuned a pre-trained GPT-2 model, a large language model, for code generation, the model demonstrated the generation of plausible Python code using syntax knowledge, analysis confirmed the originality of the generated code relative to the training dataset, and the Byte Pair encoding used in GPT-2 outperformed character-level encoding. The modified GPT-2 model trained on a large dataset performed the best, according to the findings of the study. It regarded programming language as a separate domain, obtaining a BLEU score of 0.22, a 40%

improvement over the baseline. The authors emphasized the model's capacity to generate native text in programming language (Perez et al., 2021).

From database design perspective, (Trummer, 2022) introduced CodexDB as database management system that employs OpenAI's GPT-3 Codex model to facilitate customization via natural language instructions. CodexDB intended to revolutionize the design of database systems by enabling users to modify system behavior in ways that traditional systems do not. According to Immanuel Trummer, using CodexDB, developers can benchmark data processing frameworks against SQL workloads and generate customized output at each processing stage by utilizing natural language instructions in their queries, such as specifying the use of a specific library like pandas. The system operated by accepting and transforming intricate SQL queries into natural language processing sequences. These stages are then submitted to GPT-3 Codex, which generates code that can be further customized based on user-supplied instructions. CodexDB democratized database system design through the use of natural language interfaces and machine learning techniques according to Immanuel Trummer. In the majority of cases, preliminary experiments demonstrated that CodexDB can generate correct code for a popular text-to-SQL benchmark. Experiments indicated that CodexDB's natural language query plan-enhanced code generation prompts effectively generate correct code. Although the Da Vinci model had higher success rates than the Cushman model, its generation periods were lengthier. While CodexDB obtained a 22% success rate without prior training, performance was enhanced by incorporating example solutions into the prompt. Given sufficient attempts and results for comparison, CodexDB was able to resolve 80% of test cases without additional guidance (Trummer, 2022).

In their research, (Poldrack et al., 2023) investigated the implications of AI-assisted coding, focusing particularly on the capabilities of GPT-4, in relation to various aspects of code development and the Software Development Life Cycle (SDLC). The study evaluated GPT-4's efficacy in interactive coding, debugging, code refactoring, and test writing, with a focus on the generated code's quality, legibility, maintainability, and conformance to coding standards. The researchers conducted a series of experiments to simulate data scientists' actual coding scenarios. ChatGPT, fueled by GPT-4, was used to generate code for data science problems in the first experiment. While GPT-4 was successful in 72% of cases, it occasionally relied on obsolete Python libraries and functions and struggled to generate code that was compatible with modern software frameworks such as PyTorch. This demonstrated the significance of maintaining current training data for the model's efficacy. In the second experiment, they used GPT-4 to refactored existing code in an effort to improve its quality without modifying its behavior. The results demonstrated that code refactored by GPT-4 was more compliant with coding standards and more readable than the average code on GitHub. This indicated that GPT-4 could improve the maintainability of code. The third experiment investigated GPT-4's capacity to create tests for its own code. Despite the fact that GPT-4 generated tests with outstanding coverage, these tests frequently failed, necessitating human intervention to debug and identify the root causes. This emphasized the importance of human expertise in the process of code testing, highlighting the complementary roles of AI-assisted coding and human participation (Poldrack et al., 2023).

In conclusion, the application of artificial intelligence (AI), especially language models such as GPT-2, GPT-3, and GPT-4, has demonstrated enormous potential for code generation and software development. These models have been utilized to automate code generation,

improve code quality, and boost developer efficiency. These studies demonstrate the efficacy of AI models such as GPT-2 and GPT-3 in generating code fragments, functions, and even entire programs. These models have proven their capacity to capture context, comprehend programming languages, and generate code that adheres to coding standards. AI-assisted coding has also proven useful in code refactoring, test generation, and debugging, resulting in code that is more legible, maintainable, and reliable. Despite their remarkable capabilities, the studies acknowledge the need for ongoing model training, current training data, and the continued incorporation of human expertise in software development and testing processes. AI in code generation and development holds tremendous promise for accelerating software development, enhancing code quality, and facilitating innovative Software Development Life Cycle (SDLC) approaches.

2.6 Future Trends and Challenges in AI for SDLC

The incorporation of artificial intelligence (AI) into the Software Development Life Cycle (SDLC) has led to significant automation and efficiency improvements. Nonetheless, as AI continues to develop, it introduces new trends and challenges that must be addressed. This topic investigates the future trends and challenges in AI for SDLC, such as code generation, automated testing, program maintenance, and database management. This theme provides insights into the prospective benefits of AI and the areas that require further development through a comprehensive analysis of recent research. Organizations and developers can prepare for the changing landscape of AI in the SDLC by comprehending these future trends and obstacles.

According to (Dong et al., 2023), it has been observed that current code generation tools struggle to comprehensively address the design and requirements of complex software systems.

The limitations of their proposed framework have been discussed, including the simplicity of user requirements in existing code generation datasets, the team's composition, and the system's autonomy. They proposed directions for future research, including the creation of project-level code generation datasets, the modification of team composition based on practical requirements, the incorporation of human oversight, and the exploration of the incorporation of software development tools via role instructions in the self-collaboration framework.

(Priya et al., 2017) concluded with a discussion of language models that enable computers to generate text and emphasizes the efficacy of RNNs in language modeling due to their high-dimensional concealed state but they found that the magnitude of the dataset affected the complexity and patterns learned by the model, but had little effect on the quality of the generated code, this could be resolved by extended training periods to enhance code generation. Future research directions may include fine-tuning the model, incorporating reinforcement learning into code generation, and integrating additional Natural Language Processing (NLP) deep learning networks to improve the code quality (Priya et al., 2017).

According to (Paik et al., 2021), Despite the prospective advantages of GPT-2 for code generation, they uncovered a number of obstacles. In the initial experiment, GPT-2 occasionally relied on out-of-date libraries and struggled to produce code compatible with modern software, demonstrating the significance of using current training data. Also, GPT-2 required a substantial amount of training data for effective fine-tuning.

(Feng et al., 2023) evaluated the efficacy and difficulties of ChatGPT. To analyze ChatGPT's code generation capabilities, they utilized a data-driven framework comprised of social media data and gleaned insights from tweets and Reddit posts pertaining to ChatGPT's code generation. The research revealed both the advantages and disadvantages of ChatGPT

when it comes to producing high-quality code. The researchers noticed that Python and JavaScript were identified as the most frequently discussed programming languages in relation to ChatGPT's code generation. However, obstacles were identified, particularly in the handling of sophisticated code logic and the potential for generated code to be biased. Concerns regarding code quality and the potential impact on jobs were prevalent among the expressed opinions. Using Flake8, a tool for comparing Python code to style guides, they also evaluated the quality of the generated code. The analysis uncovered frequent errors, such as pycodestyle violations and excessively long lines of code. Despite the difficulties and concerns, ChatGPT's code generation capabilities were discovered to be popular and extensively used across a variety of domains. (Feng et al., 2023) emphasized the need for additional research to improve precision and resolve limitations, especially when dealing with complex code logic.

(Perez et al., 2021) claimed that utilizing machine learning and pre-trained language models have demonstrated promise for assisting developers and enhancing the efficiency of code generation. However, difficulties may arise in managing diverse programming language syntax and ensuring the generated code's accuracy and maintainability, which require further investigation in the future.

(Trummer, 2022) concluded that CodexDB is a significant advancement in database management systems, offering a novel approach to system customization via natural language instructions and the power of AI models such as GPT-3 Codex. Despite CodexDB's promising performance, additional enhancements are required to enhance precision, establish specific interpretations of natural language input, and incorporate optimization techniques to optimize query plans for specific workloads (Trummer, 2022).

In their research, (Poldrack et al., 2023) concluded that despite the substantial advancements in AI-assisted coding, human intervention is still necessary to ensure the validity and accuracy of the resulting code. The limitations identified in GPT-4 are not conclusive, and additional research is required to investigate enhanced prompting techniques that can better the performance of language models on complex coding problems. This highlights the potential for continued advancements in AI-assisted coding to improve the efficacy and effectiveness of the SDLC, while recognizing the continued importance of human expertise in code development and quality assurance (Poldrack et al., 2023).

In conclusion, the research conducted by multiple authors casts light on the advancements, challenges, and future directions in artificial intelligence (AI) for the Software Development Life Cycle (SDLC). Code generation has played a significant role in automating software development, but current tools are limited in their ability to address complex software systems. Language Learning Models (LLMs) have the potential to improve code generation, automated testing, and program maintenance. However, gaps persist in the integration of these responsibilities across the various SDLC phases. Future research directions include the creation of datasets at the project level, the modification of team composition, the incorporation of human supervision, and the use of software development tools to enhance collaboration. The effectiveness of language models, such as RNNs and GPT-2, has been evaluated, revealing the need for protracted training periods and the management of complex code logic. The code generation capabilities of ChatGPT have demonstrated both advantages and challenges, highlighting the need for improved precision and addressing the limitations of complex code logic. Despite significant advancements in AI-assisted coding, human intervention is still required to ensure code validity and accuracy. The potential for continued advancements in AI-

assisted coding offers opportunities to improve the SDLC, while acknowledging the continued significance of human expertise in code development and quality assurance. AI's future in the SDLC is promising, but additional research and development are necessary to overcome obstacles and realize its maximum potential.

Chapter 3: Reserach Model and Hypotheses Development

3.1 Research Model

The research paradigm and hypotheses for this study are grounded in two prominent theories: The Unified Theory of Acceptance and Use of Technology 2 (UTAUT2) and the Task-Technology Fit (TTF) theory. These theories were chosen due to their relevance and applicability in gaining a comprehension of the adoption and acceptance of new technologies in a variety of contexts. This section provides an overview of the UTAUT2 and TTF theories, emphasizing why they were chosen for this study and providing references to other studies that utilized these theories to support the selection.

The Unified Theory of Acceptance and Use of Technology 2 (UTAUT2) is an expansion of the original Unified Theory of Acceptance and Use of Technology (UTAUT), which was devised to examine the acceptance and adoption of new technologies by individuals. UTAUT2 includes several factors that influence technology adoption, such as performance expectations, effort expectations, social influence, facilitating conditions, hedonic motivation, price value, and habitual behavior. It provides a comprehensive framework for understanding the determinants of technology adoption and usage behavior (Venkatesh et al., 2012).

The UTAUT2 theory has been extensively utilized in studies examining the adoption and acceptance of diverse technologies in various domains. (Venkatesh et al., 2012) investigated the adoption of healthcare information systems using UTAUT2. By incorporating UTAUT2 into this study, we hope to obtain a better understanding of the factors that influence the acceptance and use of ChatGPT in the context of software development.

The Task-Technology Fit (TTF) theory emphasizes on the alignment between technology characteristics and the tasks performed by users. It emphasizes the significance of

relating the capabilities of a technology to the needs and challenges of a specific task in order to improve user performance and satisfaction (Goodhue et al., 1995). The Task-Technology Fit (TTF) theory is predicated on the concept of how well technology assists individuals in completing their respective duties (Goodhue et al., 1995). Its primary purpose is to evaluate whether technology facilitates task completion adequately during the implementation of new technology. TTF theory has been extensively utilized in studies examining the compatibility between technology and tasks in various domains. For instance, (Igbaria et al., 1995; Wu et al., 2006) have used TTF to examine the fit between enterprise resource planning systems and organizational tasks, and information technology and job performance in healthcare. By incorporating TTF theory into this study, we intend to determine the compatibility and alignment between ChatGPT's capabilities and the tasks conducted by software developers throughout the software development life cycle. The TTF theory will provide a valuable framework for evaluating ChatGPT's compatibility and suitability for supporting SDLC process duties and goals. By examining the compatibility between ChatGPT's technological capabilities and the specific tasks performed by software developers, we can obtain insight into how ChatGPT can integrate into the SDLC and improve the software development process.

By combining the UTAUT2 and TTF theories, we can obtain valuable insights into the adoption and use of ChatGPT in the context of software development. The UTAUT2 theory enables us to examine individual-level factors that influence the adoption of new technologies, whereas the TTF theory concentrates on the compatibility of technology characteristics and task requirements. This integrated approach provides a comprehensive comprehension of the factors driving ChatGPT's adoption in the software development life cycle (SDLC). Previous studies have effectively employed the UTAUT2 and TTF theories in various contexts of technology

adoption, validating their applicability and efficacy (Sharif et al., 2019; Beier et al., 2020; Faqih et al., 2021).

The UTAUT2 theory, with its comprehensive set of constructs that includes performance expectancy (PE), effort expectancy (EE), social influence (SI), facilitating conditions (FC), hedonic motivation (HM), habit behavior (HB), and experience provides a robust framework for comprehending individual behaviors associated with technology adoption (Faqih et al., 2021). In contrast, the TTF theory emphasizes the need to ensure compatibility and alignment between technology characteristics (TC), task technology fit (TTF), and task characteristics (TaskC) (Sharif et al., 2019; Beier et al., 2020). This is especially pertinent in the case of ChatGPT, as its successful integration into the SDLC necessitates a close match between the technology's capabilities and the tasks performed by software developers.

Nevertheless, we will exclude price value, age, gender, and experience from our analysis in this study. The decision to exclude these factors is based on their limited relevance and impact on ChatGPT adoption and use in the SDLC. Regarding price value, it will be excluded because the workplace is responsible for supplying the necessary equipment for employees to use ChatGPT. Since cost considerations are managed by the organization, even though the cost is affordable by individuals either by using the free version of ChatGPT. Based on that, the price-to-value ratio does not directly influence the adoption decisions of users. Likewise, age, and gender will not be considered in our analysis due to the fact that ChatGPT has attracted users of all ages and genders so these factors influence on adoption decisions has diminished as ChatGPT adoption is not restricted by age or gender, as it appeals to a wide variety of software development users. Also, experience will not be considered in our analysis due to the fact that we are targeting software developers, they are anticipated to have prior experience with similar

tools, whereas ChatGPT is designed to assist users with varying levels of experience. Therefore, the effect of experience on the intent to implement ChatGPT in the SDLC may be negligible.

By focusing on the previously mentioned relevant factors, such as performance expectation, effort expectation, social influence, facilitating conditions, hedonic motivation, habit behavior, task characteristics, technology characteristics, and task-technology fit, we can gain a comprehensive understanding of the factors that truly influence the adoption and use of ChatGPT in the SDLC. This strategy enables us to concentrate on the factors that are most pertinent and have the greatest impact in this particular context. As shown in Figure 2, the model structure diagram depicts the conceptual framework utilized in this investigation.

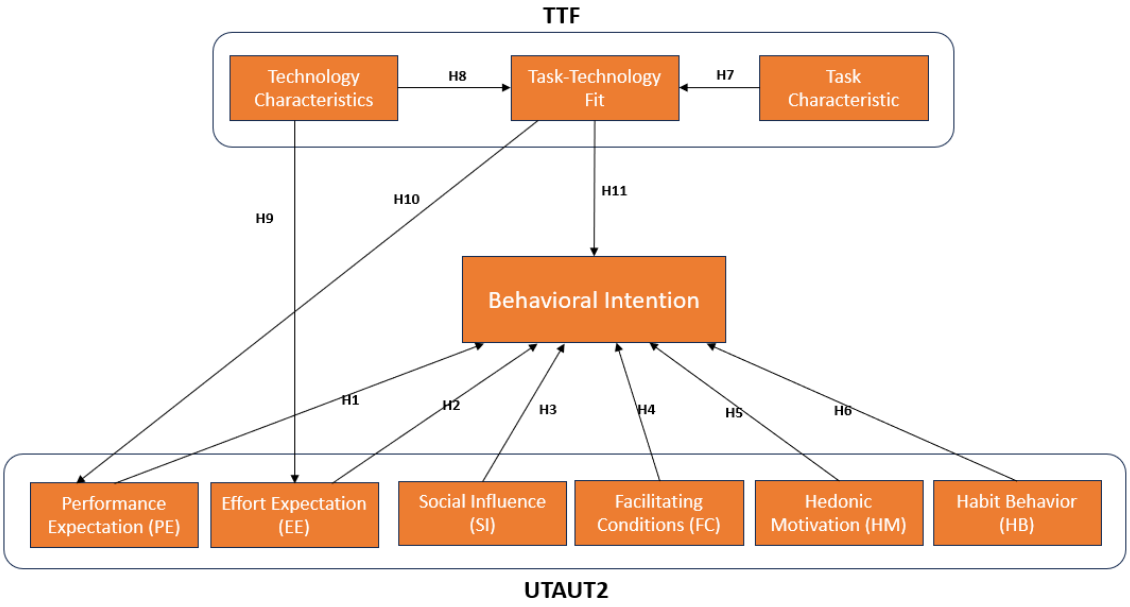

Figure 2: Conceptual Framework

3.2 Research Hypothesis

3.2.1 UTAUT2 Hypothesis

In this section, we will examine the relevance of the Unified Theory of Acceptance and Use of Technology 2 (UTAUT2) to comprehending the adoption and use of ChatGPT in the software development life cycle (SDLC). By investigating the selected factors within the context of ChatGPT in the SDLC, we can obtain valuable insight into its adoption and use among software developers.

3.2.1.1 Performance Expectancy (PE)

Performance expectancy (PE) is defined as "The degree to which an individual perceives that using a system will help him or her to attain again in job performance" (Venkatesh et al., 2003). Performance Expectation (PE) is a crucial component of the Unified Theory of Acceptance and Use of Technology 2 (UTAUT2), which centers on the perceived advantages and utility of adopting a new technology. (Venkatesh et al., 2003) revealed for the first time this essential element for the development of UTAUT theory. Since then, PE has garnered wide acceptance as a crucial technological characteristic for analyzing the adoption of new technologies. As a result, performance expectations have gained prominence in studies of technology adoption.

Multiple studies have emphasized the significance of performance expectations in technology adoption. For instance, (Venkatesh et al., 2012) found that users' behavioral intent to implement and utilize technology is significantly influenced by their performance expectations. Similarly, (Alraja et al., 2016; Moghavvemi et al., 2017; (Faqih et al., 2021)

studies have shown that performance expectancy is a key variable influencing people's behavioral intention to adopt new technologies.

In the context of utilizing ChatGPT in the software development life cycle (SDLC), performance expectation refers to the degree to which software developers assume ChatGPT will improve their performance and productivity in development tasks. On the basis of the construct of performance expectancy, the following hypothesis is proposed:

H1: Performance Expectancy (PE) positively influences the Behavioral Intention (BI) to adopt ChatGPT in software development.

3.2.1.2 Effort Expectancy (EE)

Effort expectancy (EE) is defined as “the degree of ease of use associated with the use of technology” (Venkatesh et al., 2003). It refers to the perceived simplicity of use and level of effort required to utilize a technology by individuals, it represents users' perceptions of the difficulty of implementing and utilizing the technology (Faqih et al., 2021). It is crucial to comprehend the users' perception of the effort required to use the tool in order to predict their adoption intent (Faqih et al., 2015; Paulo et al., 2018). Numerous studies examining the adoption of information technology products and services (Rahi et al., 2018; Raza et al., 2019) have consistently emphasized the significance of effort expectations as a major factor in the adoption process. This highlights the importance of assessing users' behavioral intent to implement and utilize new technological innovations based on their anticipated effort.

In the case of ChatGPT, developers' perceptions of how simple it is to incorporate ChatGPT into their workflow and complete development tasks with the aid of the tool. A higher Effort Expectancy indicates that users perceive the technology to be more difficult to use, whereas a lower Effort Expectancy indicates that users perceive the technology to be more

intuitive and user-friendly. On the basis of the construct of Effort Expectancy, the following hypothesis is proposed:

H2: Effort Expectancy (EE) positively influences the Behavioral Intention (BI) of people working in software development to adopt ChatGPT.

3.2.1.3 Social Influence (SI)

Social Influence (SI) is defined as "the degree to which an individual perceives that important others believe he or she should use the new system" (Venkatesh et al., 2003). Social Influence is a crucial determinant of users' intent to adopt and utilize new technologies. It refers to the extent to which individuals perceive that influential individuals believe they should use a specific technology. Several studies have demonstrated that social influence has a significant effect on the adoption of information technology products and services (Jaradat et al., 2014; Faqih et al., 2015; Rauschnabel et al., 2016; Rauschnabel et al., 2017; Sharif et al., 2019; Jaradat et al., 2020; Faqih, 2020; Faqih et al., 2021).

Social influence has become a powerful factor that significantly influences consumers' intentions to embrace and adopt innovation. As a result, it has been proposed that the social dimension has the ability to influence people's perceptions and behaviors in order to conform to the expectations of a social perspective (Venkatesh et al., 2012).

By examining the role of social influence in the adoption of ChatGPT in the context of software development, we can gain insight into how Friends, coworkers, and experts can shape users' attitudes and beliefs, ultimately influencing their intent to adopt and utilize ChatGPT in software development. On the basis of the construct of Social Influence, the following hypothesis is proposed:

H3: Social Influence (SI) positively influences the Behavioral Intention (BI) to adopt ChatGPT in software development.

3.2.1.4 Facilitating Conditions (FC)

Facilitating conditions (FC) is defined as "the degree to which an individual believes that organizational and technical infrastructure exists to support use of the system" (Venkatesh et al., 2003). Numerous studies have highlighted the importance of facilitating technology adoption conditions. The availability of technical support, adequate training, and infrastructure can have a significant impact on users' perceptions of a technology's usability (Venkatesh et al., 2012; Gupta et al., 2017; Tavares et al., 2018; Moorthy et al., 2019).

In the context of ChatGPT in software development, facilitating conditions refer to the availability of resources, support, infrastructure, access that allow users to adopt and utilize ChatGPT effectively in software development. Facilitating conditions may include access to dependable Internet connectivity, adequate hardware and software requirements, technical support, training programs, and organizational policies that promote the integration of ChatGPT into the software development workflow. Consideration of facilitating conditions is crucial because the successful adoption and utilization of ChatGPT in SDLC depends on the presence of a supportive environment that provides the conditions necessary for its effective implementation. Without the appropriate enabling conditions, users may encounter obstacles and difficulties that impede their ability to adopt and utilize ChatGPT effectively for software development tasks. On the basis of the construct of facilitating conditions, the following hypothesis is proposed:

H4: Facilitating Conditions (FC) positively influence the Behavioral Intention (BI) to adopt ChatGPT in software development.

3.2.1.5 Hedonic Motivation (HM)

Hedonic motivation (HM) is a term used in the study of technology adoption to describe how an individual feels and acts emotionally as a consequence of adopting innovation (Venkatesh et al., 2012). Hedonic motivation refers to the enjoyment, delight, or intrinsic satisfaction derived from the use of a specific technology. Prior research has demonstrated that hedonic motivation plays a significant role in the adoption of new technologies. When users perceive a technology as pleasant, entertaining, or providing a pleasurable experience, they are more likely to adopt and utilize it (Salimon et al., 2017; Moorthy et al., 2019; Kim et al., 2019). Hedonic motivation can increase users' engagement, satisfaction, and overall user experience, which influences their intent to adopt and use the technology positively. Users will accept and employ technology, according to recent research, if their fundamental motives are satisfied (Lowry et al., 2012). Several researchers have demonstrated that novel technologies, such as augmented reality, generate more hedonic value than utilitarian value (Javornik et al., 2016; Kim et al., 2019; Faqih et al., 2021). It appears that the hedonic motivation component is gaining prominence and gaining significance in the adoption dynamics of AR technology.

Understanding the role of hedonic motivation in the context of ChatGPT is crucial, as it can influence users' behavioral intention to implement and use the tool beyond its functional benefits. By examining the effect of hedonic motivation on ChatGPT adoption in the context of software development, we can gain insight into how the gratifying and pleasurable aspects of using ChatGPT influence users' adoption decisions. This knowledge can enlighten strategies for enhancing the ChatGPT user experience and promoting its adoption and use throughout the software development life cycle. On the basis of the construct of hedonic motivation, the following hypothesis is proposed:

H5: Hedonic Motivation (HM) positively influences the Behavioral Intention (BI) to adopt ChatGPT in software development.

3.2.1.6 Habit Behavior (HB)

Habit may be defined as "a tendency of individuals to exhibit automatic behavior depending on what they learn" (Kaur et al., 2020). It is underlined that the usage of technology in the past strongly predicts how it will be used in the future (Gerup et al., 2020). Previous research has demonstrated the effect of habitual behavior on the adoption and utilization of technology. When users develop a habit of using a technology, they are more likely to continue using it in the future, regardless of perceived utility or ease of use (Davis et al., 1989). Habit behavior can provide a sense of familiarity, comfort, and efficacy, which can positively influence the intention of users to adopt and integrate ChatGPT into their routine software development tasks. By investigating the role of habit behavior in the adoption of ChatGPT in the context of software development, we can gain insight into how the established habits and routines of users affect their continued use and integration of the tool. This knowledge can enlighten strategies to promote the formation of positive habits and routines surrounding the use of ChatGPT, resulting in its sustained adoption and utilization throughout the software development life cycle.

H6: Habit Behavior (HB) positively influences the Behavioral Intention (BI) to adopt ChatGPT in software development.

3.2.2 Task Technology Fit (TFF) Hypothesis

The Task-Technology Fit (TTF) theory stresses the significance of aligning technology characteristics with task requirements in order to improve user performance and satisfaction. It

focuses on evaluating how effectively and efficiently technology assists individuals in completing their respective duties. In the context of adopting ChatGPT in the software development life cycle (SDLC), it is crucial to comprehend the compatibility and alignment of ChatGPT's capabilities with the tasks conducted by software developers.

Recognizing the potential of UTAUT2 to explain intentional adoption behavior, we've improved its efficacy by incorporating the Task-Technology Fit (TTF) theory. This integration seeks to develop a more robust theoretical framework that accounts for the various factors that influence behavioral intention to adopt ChatGPT in software development.

The integration of UTAUT2 and TTF permits us to examine the relationships between the TTF antecedents, namely "task characteristics", "technology characteristics", and "task-technology fit" and the three primary UTAUT2 constructs, namely "performance expectancy," "effort expectancy," and "behavioral intention." Several studies have demonstrated the effectiveness of this integration in expanding the variance in behavioral intention to employ new technologies. (Sharif et al., 2019; Beier et al., 2020; Faqih et al., 2021) have effectively implemented the integration concept between UTAUT2 and the TTF model, demonstrating its utility and applicability in a variety of technology adoption contexts.

By combining UTAUT2 and TTF, we intend to develop a comprehensive theoretical framework that captures the complex interaction between individual-level factors, technology characteristics, and task requirements in the adoption of ChatGPT in software development. This integrated framework will improve our comprehension of the factors that influence behavioral intention and adoption behavior, shedding light on the successful integration of ChatGPT into the software development process.

3.2.2.1 Task Characteristics (TaskC)

In the context of the software development life cycle (SDLC), task characteristics are the specific attributes and requirements of the tasks executed. Understanding the influence of task characteristics on the adoption and use of ChatGPT is essential for determining its suitability and efficacy in supporting software development tasks. To investigate the effect of task characteristics on the adoption of ChatGPT, we will evaluate the complexity, interdependence, novelty, and routineness of tasks. Prior research has highlighted the significance of task characteristics in shaping the perceptions and attitudes of technology adopters. (Goodhue et al., 1995; Wu et al., 2006) have demonstrated the effect of task characteristics on technology adoption, underscoring the importance of a close match between technology capabilities and task requirements.

By investigating the role of task characteristics in the adoption of ChatGPT in the SDLC, we hope to gain insight into how specific task attributes influence users' perceptions of ChatGPT's usefulness and ease of use, thereby influencing their behavioral intention to adopt the technology. On the basis of the construct of task characteristics, the following hypothesis is proposed:

H7: Task Characteristics (TaskC) of software development positively influence the Task-Technology Fit (TTF).

3.2.2.2 Technology Characteristics (TC)

Technology Characteristic refer to technology specific attributes and capabilities. Understanding the effect of technology characteristics on the adoption and utilization of ChatGPT is essential for determining its suitability and efficacy in supporting software development tasks. To evaluate the technological characteristics of ChatGPT, we will consider

its precision, response time, reliability, adaptability, and user interface. These characteristics play a crucial role in determining how consumers perceive and interact with technology. Prior research has demonstrated the relationship between technology characteristics and technology adoption. Study by (Venkatesh et al., 2012; Faqih et al., 2021) indicate that technology characteristics substantially influence users' perceptions of usefulness, usability, and intent to adopt.

By investigating the role of technology characteristics in the adoption of ChatGPT in the SDLC, we hope to gain insight into how specific technological attributes affect users' perceptions of the usefulness and usability of ChatGPT, thereby influencing their behavioral intention to adopt the technology. On the basis of the construct of technology characteristics, the following hypotheses are proposed:

H8: Technology Characteristics (TC) of ChatGPT positively influence the Task-Technology Fit (TTF).

H9: Technology Characteristics (TC) positively influences the effort expectancy of ChatGPT.

3.2.2.3 Task-Technology Fit (TTF)

TTF refers to the degree to which a technology aligns with and supports the duties and requirements of users. In the context of ChatGPT and its implementation in the software development life cycle (SDLC), it is crucial to comprehend the compatibility between the technology and the tasks conducted by software developers. To evaluate the task-technology fit of ChatGPT, we will evaluate the alignment between the technology and the specific tasks executed throughout the various SDLC phases. This includes evaluating the compatibility between ChatGPT's capabilities and the needs for requirements elicitation, design, coding,

documentation, and testing an. By analyzing the compatibility between the technology and the tasks, we can determine the extent to which ChatGPT facilitates and improves the software development process. Prior research has demonstrated the importance of task-technology compatibility in technology adoption. (Goodhue et al., 1995 ;Wu et al., 2006) found that when there is a strong fit between the technology and the tasks, users perceive the technology as more useful and simpler to use, resulting in a greater adoption intent.

By analyzing the task-technology fit of ChatGPT in the SDLC, we hope to determine how well the technology correlates with the tasks performed by software developers. On the basis of the construct of task-technology fit, the following hypotheses are proposed:

H10: Task-Technology Fit (TTF) positively influences the performance expectancy of ChatGPT.

H11: Task-Technology Fit (TTF) positively influences the behavioral intention (BI) to adopt ChatGPT in software development.

Chapter 4: Reserach Methodology

This study's research methodology consists of three essential components: data collection and sampling method, research instrument development, and data analysis. This section provides an overview of the methodology used to collect and analyze data to meet the research objectives and verify the hypotheses.

4.1 Data Collection and Method

In this study, an online survey used to capture data from people working in software development. The survey method provides a practicable and efficient way to collect data from a large number of participants, allowing for a thorough examination of the research topic. In this instance, software developers and professionals involved in the software development process have been surveyed. The sample size has been determined using the rule of thumb proposed by (Hair et al., 2011; Hair et al., 2017), which implies that the sample size should be at least 10 times the number of paths for each latent variable in the proposed model. We had 11 paths which indicate that we required 110 participants at least. However, we set our target for 300 participants as it is crucial to ensure a sufficient sample size because it improves the generalizability and reliability of the study's results.

The online survey has been sent through social media platforms like LinkedIn and Facebook to participants from a variety of organizations to assure a broad range of perspectives and experiences. The survey is designed to collect information pertinent to the adoption of ChatGPT in the software development, including the perceptions, attitudes, and intentions of participants regarding the use of ChatGPT in software development. The survey is administered via Google form, one of the popular online survey platforms to facilitate response collection and facilitate participant access. Participants contacted via invitation post on professional networks

like LinkedIn, Whatsup broadcast to specific people working on software development, and invitation post in software development communities on Facebook to partake in the survey. The invitations provided a concise summary of the objectives of the study, emphasize the confidentiality and anonymity of responses, and contain a link to the online survey. The participants were able to complete the survey at their leisure and submit their responses within a given timeframe. To increase the response rate, the post has been reshared multiple times to encourage more people to participate.

Throughout the data collection procedure, ethical considerations were strictly adhered to. Before participants commence the survey, informed consent will be obtained to ensure that they are aware of the purpose of the study, their rights as participants, and the confidentiality and anonymity of their responses. The collected information has been kept strictly confidential and used for research purposes only.

4.2 Research Instrument Development

This study's research instrument was a questionnaire designed to collect information from the participants. Electronic administration of the survey questionnaire via Google online form which considered as one of the popular survey platforms to facilitate data collection and distribution. Participants were assured of the confidentiality and anonymity of their responses, and the collected data were managed with the utmost care and used solely for research. The responses were measured using a five-point Likert scale, where 1 indicated "strongly disagree" and 5 indicated "strongly agree."

The decision to use a five-point Likert scale was influenced by several factors. First, a five-point Likert scale was selected in order to provide respondents with a manageable number of options. Using a seven-point Likert scale has the potential to cause confusion and

discouragement (Leong et al., 2020). This method intended to mitigate these potential outcomes. By limiting the number of response options, participants were able to provide responses with greater ease and precision. Secondly, the five-point Likert scale was considered to be more practical and user-friendly when choosing alternatives (Dawes, 2008). Participants could easily comprehend and use the scale, making it a practical choice for assessing their attitudes, opinions, and perceptions. The simplicity of the scale facilitated efficient data capture and analysis, allowing participants to accurately express their opinions.

The survey questionnaire included a variety of pertinent dimensions and constructs. As shown in Table 1, Each questionnaire item was crafted with care to evoke meaningful responses and capture the participants' perspectives on the adoption and use of ChatGPT in the software development. Utilizing a Likert scale in the survey allowed for quantifiable data that could be statistically analyzed. Participants' responses were designated numerical values, allowing for the examination of relationships, patterns, and trends in the data.

Table 1: Research Instrument's Constructs and Related Items

Factor	Code	Items	Reference
Performance Expectancy (PE)	PE1	I think ChatGPT will enhance my productivity in software development tasks.	(Faqih et al., 2021)
	PE2	I think ChatGPT to be useful in assisting me with writing user stories based on business requirement.	(Al Sarayreh, 2023)
	PE3	I think ChatGPT to be useful in assisting me with coding and debugging.	(Al Sarayreh, 2023)
	PE4	I think ChatGPT to be useful in assisting me with writing test cases or test scenarios	(Al Sarayreh, 2023)
	PE5	I think ChatGPT to be useful in assisting me database design and generate database scripts based on business requirement.	(Al Sarayreh, 2023)
	PE6	I think the use of ChatGPT increases my chances of achieving things that are important to me in my career in software development industry	(Faqih et al., 2021)
	PE7	I think ChatGPT can expedite my software development tasks.	(Faqih et al., 2021)
	EE1	I think using ChatGPT in the software development will be easy for me.	(Faqih et al., 2021)

Effort Expectancy (EE)	EE2	I think it is easy for me to become skillful at using ChatGPT in software development.	(Faqih et al., 2021)
	EE3	My interaction with ChatGPT in work in software development is clear and understandable.	(Faqih et al., 2021)
Social Influence (SI)	SI1	My colleagues believe that using ChatGPT in the software development is a good idea.	(Faqih et al., 2021)
	SI2	I think my senior management will support the use of ChatGPT in software development at work.	(Faqih et al., 2021)
	SI3	People whose opinions I value prefer using a ChatGPT in software development.	(Faqih et al., 2021)
Facilitating Conditions (FC)	FC1	I have access to the necessary resources and infrastructure to use ChatGPT in software development effectively.	(Faqih et al., 2021)
	FC2	I think I have the necessary knowledge to use ChatGPT in software development.	(Faqih et al., 2021)
	FC3	I think my organization supports the integration of ChatGPT into the software development.	(Faqih et al., 2021)
	FC4	I think that the use of ChatGPT in software development is	(Faqih et al., 2021)

		compatible with other technologies I use	
	FC5	I think that I can get help from others when I have difficulties using ChatGPT in software development	(Faqih et al., 2021)
Hedonic Motivation (HM)	HM1	I enjoy using ChatGPT in software development tasks.	(Faqih et al., 2021)
	HM2	Using ChatGPT adds enjoyment to my software development tasks.	(Faqih et al., 2021)
	HM3	Using ChatGPT, in software development activities are highly entertaining.	(Faqih et al., 2021)
Habit Behavior (HB)	HB1	I regularly use ChatGPT in my software development activities.	(Al Sarayreh, 2023)
	HB2	Using ChatGPT has become a habit for me during the software developments.	(Al Sarayreh, 2023)
	HB3	I think that I must use ChatGPT in software development	(Al Sarayreh, 2023)
Task Characteristics (TaskC)	TaskC1	The tasks in the software development process are well-defined and structured.	(Faqih et al., 2021)
	TaskC2	Software development tasks require creativity and problem-solving.	(Faqih et al., 2021)
	TaskC3	I think that I need to use ChatGPT to work in software development anywhere.	(Faqih et al., 2021)

	TaskC4	I think that I need to use ChatGPT to work in software development anytime.	(Faqih et al., 2021)
Technology Characteristics (TC)	TC1	ChatGPT has advanced natural language processing capabilities.	(Faqih et al., 2021)
	TC2	ChatGPT provides accurate and reliable suggestions for code improvement.	(Faqih et al., 2021)
	TC3	ChatGPT provides accurate and reliable suggestions for user stories based on given requirements.	(Faqih et al., 2021)
	TC4	ChatGPT provides accurate and reliable suggestions for test cases based on given user stories.	(Faqih et al., 2021)
	TC5	ChatGPT provides accurate and reliable suggestions for database schema based on given requirement	(Faqih et al., 2021)
	TC6	I think the use of ChatGPT in software development will be secure.	(Faqih et al., 2021)
	TC7	I think that using ChatGPT will allow me to complete my tasks in software development without interruption in service.	(Faqih et al., 2021)
	TTF1	I think the functions/features of the ChatGPT are sufficient to	(Faqih et al., 2021)

		help me complete my work in software development	
Task-Technology Fit (TTF)	TTF2	I think the functions/features of the ChatGPT are appropriate to help me complete my work in software development	(Faqih et al., 2021)
	TTF3	In general, I think the functions/features of the ChatGPT meets my work needs in software development	(Faqih et al., 2021)
	TTF4	I think using ChatGPT enhances the efficiency and effectiveness of software development tasks.	(Faqih et al., 2021)
Behavioral Intention (BI)	BI1	I intend to adopt and use ChatGPT in the software development.	(Faqih et al., 2021)
	BI2	I will always try to use ChatGPT in software development	(Faqih et al., 2021)
	BI3	I plan to use ChatGPT frequently in software development tasks	(Faqih et al., 2021)
	BI4	I am likely to recommend ChatGPT to my colleagues for software development.	-

4.3 Data Analysis

To investigate and validate the experimental data in this study, the data acquired from the valid questionnaires were subjected to rigorous statistical analysis. In order to accomplish this, the researcher employed a two-stage analytical technique involving the application of

Structural Equation Modeling-Artificial Neural Network (SEM-ANN) and Partial Least Squares-Structural Equation Modeling (PLS-SEM) techniques.

Initial evaluation of the measurement model ensured the dependability and validity of the study's indicators and constructs. This entailed evaluating the psychometric properties of the measurement model, such as its reliability, convergent and discriminant validity, and its discriminant validity. This analysis was conducted using the PLS-SEM method, which enabled a comprehensive examination of the relationships between observed variables and latent constructs.

Subsequently, the structural model was validated in order to analyze the linkages (hypotheses) between the proposed model's constructs. The SEM-ANN methodology was utilized for this objective. SEM-ANN is a combination of structural equation modeling, which is appropriate for testing complex relationships between constructs, and artificial neural network analysis, which is useful for identifying nonlinear interactions.

Utilizing the SEM-ANN method allowed for the identification of both linear and nonlinear interactions among the proposed model's components. This method is particularly useful for examining intricate relationships and documenting the complex dynamics involved in the adoption and utilization of ChatGPT during the software development. The results of the SEM-ANN analysis cast light on the significant factors influencing the use of ChatGPT in software development, revealing important insights into the interrelationships between variables.

Importantly, the implementation of the ANN method in this research allowed for more accurate predictions than many traditional regression methods (Lee et al., 2020; Qasem et al., 2020). Using the power of artificial neural networks, the study was able to identify and

comprehend intricate patterns and nonlinear relationships that may not have been readily discernible using conventional statistical methods.

In the data analysis phase, the combination of SEM-ANN and PLS-SEM methods ensured a thorough evaluation of the research model, hypotheses, and the reliability and validity of the constructs and indicators. This approach to analysis provides a solid foundation for deriving meaningful conclusions and making informed recommendations based on the findings of the study.

Chapter 5: Results

5.1 Descriptive Analysis

This section aims to present a thorough descriptive analysis of the survey data pertaining to the utilization of ChatGPT in the context of software development. Out of the 608 responses, 461 were included for analysis, while 147 responses were excluded because they belonged to irrelevant roles in software development or contained outlier responses. The present analysis aims to categorize the participant information into four primary classifications, namely Gender, Age Group, Education Level, and Current Role as shown in Table 2. This will facilitate a more comprehensive comprehension of the various cohorts of individuals exhibiting interest in ChatGPT within the software development community, and could potentially provide valuable insights into the wider patterns of ChatGPT adoption within this domain.

Table 2: Participants Demographics

Gender		
Category	**Count**	**Percentage**
Female	23	5%
Male	438	95%
Age Group		
Category	**Count**	**Percentage**
20 to 24	11	2%
25 to 30	164	36%
Above 30	280	61%
Under 20	6	1%
Education level		

Category	Count	Percentage
High School	9	2%
Student	19	4%
University	433	94%
Current Role		
Category	**Count**	**Percentage**
Analyst	61	13%
DBA	6	1%
Manager	13	3%
Programmer	254	55%
Project Manager	9	2%
QA	89	19%
Security	29	6%
Grand Total	**461**	

5.1.1 Distribution of Males and Females

An examination of the gender dynamics reveals a predominately male-dominated participation as shown in Figure 3. An astounding 438 of the total 461 respondents identified as male. This male predominance, which accounted for 95% of all participants, provides an initial indication of the gender disparity in software development utilization or interest in ChatGPT. Females, on the other hand, represent a negligible proportion of respondents, with only 23 females representing 5% of the total sample. This striking gender disparity demonstrates the

potential need for initiatives to encourage more female software developers and professionals to employ and utilize ChatGPT.

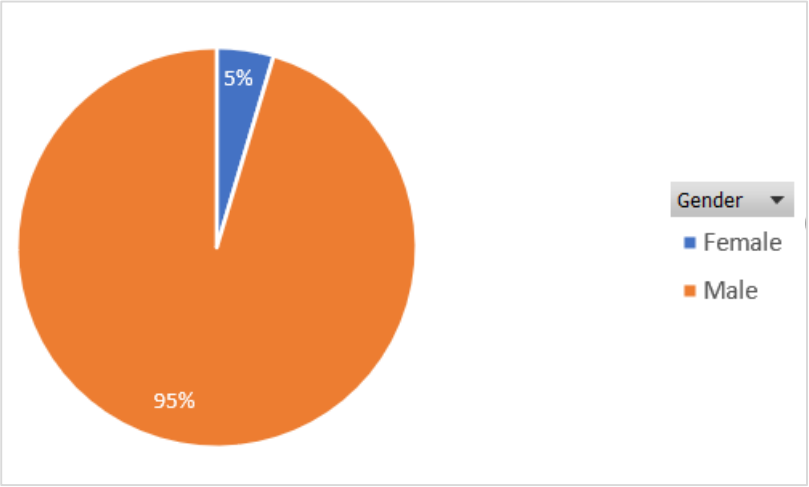

Figure 3: Participants Gender

5.1.2 Age Distribution Analysis

Changing the perspective to age distribution adds another layer of comprehension. The group of participants over the age of 30 is the largest as shown in Figure 4, comprising 280 individuals and constituting more than half of the total participants. This indicates that senior and more experienced software development professionals have a strong preference for ChatGPT. With 164 participants, the age range of 25 to 30 also represents a significant portion of ChatGPT users, indicating that young and mid-career professionals are also actively using the platform. The younger demographics, namely the under 20 and 20 to 24 age categories, represent a smaller portion of the ChatGPT ecosystem, with 6 and 11 participants, respectively, indicating a lower level of participation from novices and early-career professionals.

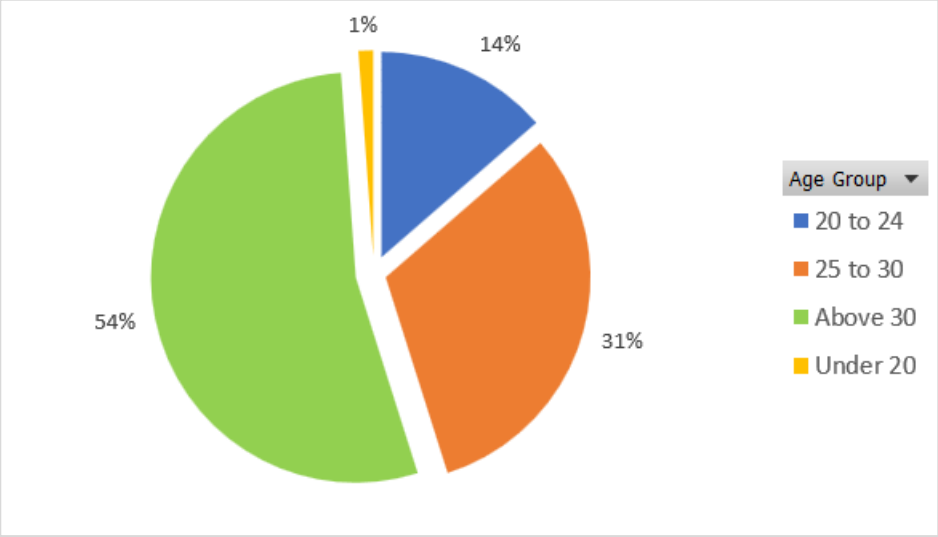

Figure 4: Age Groups

5.1.3 Educational Perspectives

An examination of the educational background of the participants enhances the demographic picture. The overwhelming majority of 433 individuals hold university-level education as shown in Figure 5. This staggering 94% representation demonstrates the increased engagement of highly educated individuals in the ChatGPT adoption journey. A total of 28 high school graduates and students are present, highlighting their willingness to use ChatGPT in their software development tasks.

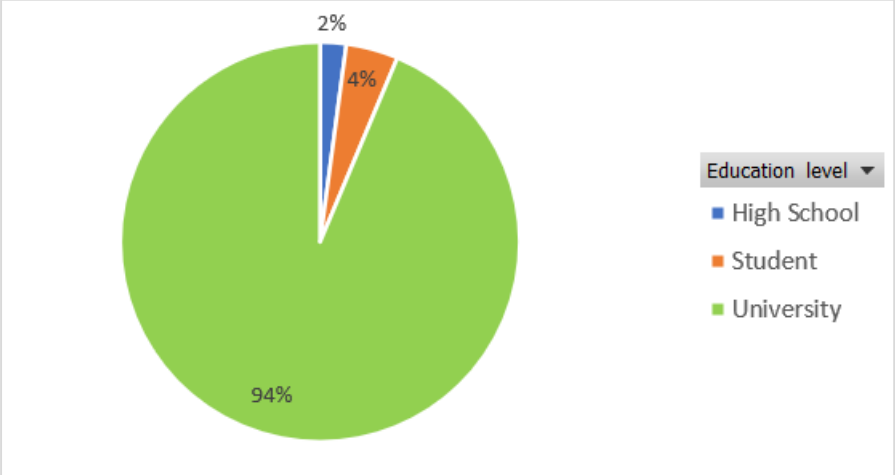

Figure 5: Education Level

5.1.4 Professional Function Assignment

In addition, the survey shed light on the participants' professional responsibilities. Programmers are the most numerous categories as shown in Figure 6, with 254 individuals indicating their presence. This significant proportion of programmers, 55%, demonstrates the broad applicability and relevance of ChatGPT within the programming community. Other positions such as analysts, database administrators, managers, project managers, quality assurance professionals, and security personnel have emerged, albeit in relatively lesser numbers. This diversified role representation exemplifies the breadth of ChatGPT's professional appeal, which transcends roles within the software development industry.

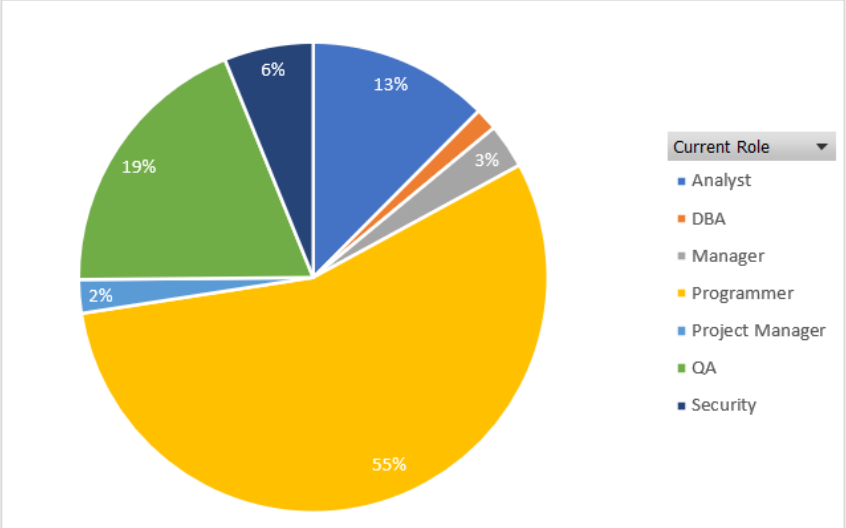

Figure 6: Participants Roles

In essence, the descriptive analysis of the survey on ChatGPT adoption in software development paints a picture of a landscape dominated by university-educated males over the age of thirty. The significant proportion of programmers demonstrates ChatGPT's robust relevance and prospective impact in the programming community. However, low female participation and limited engagement from younger and less educated demographics highlight potential areas for enhanced outreach and engagement efforts. Insights garnered from this analysis serve as the basis for strategies designed to promote the widespread adoption and utilization of ChatGPT within the diverse software development landscape.

5.2 Measurement Model Assessment

The assessment of the measurement model is an essential initial stage in examining the integration of ChatGPT into software development. This process aims to establish a dependable and valid evaluation of the connections between indicators and latent variables. The present

portion commences with an analysis of Construct Reliability or Internal Consistency, which is assessed through the utilization of Cronbach's Alpha (α) and Composite Reliability (CR). These metrics provide evidence of the consistency and reliability of the constructs employed in the study. Subsequently, the attention turns towards the evaluation of Convergent Validity. The analysis of Indicator Reliability and Average Variance Extracted (AVE) is conducted to assess the extent to which the indicators accurately represent the underlying variables, therefore ensuring the conceptual validity of the constructs (Hair et al., 2016; Hair et al., 2017).

Finally, the assessment of Discriminant Validity involves the utilization of many methodologies such as the Fornell & Larcker Criterion, Cross-loadings analysis, and the Heterotrait-Monotrait Ratio of Correlations (HTMT). This assessment provides confirmation that the structures under investigation are clearly differentiated, hence reinforcing the general validity and reliability of the research findings pertaining to the implementation of ChatGPT in the context of software development (Hair et al., 2016; Hair et al., 2017).

5.2.1 Construct Reliability (Internal Consistency)

When evaluating the measurement model, the first stage entails examining Composite Reliability (rho_a), Cronbach's alpha. Composite Reliability (rho_a) is a measure that, similar to Cronbach's alpha, assesses the internal consistency of a construct. According to (Hair et al., 2016; Ramayah et al., 2018), a value of 0.7 is deemed acceptable. To exclude the items less below the accepted threshold, we examined the loadings of items on their corresponding constructs or latent variables. Ideally, it is desirable for the loadings to be 0.7 or higher (Ramayah et al., 2018). During the preliminary assessment, the loading of FC5 was seen to be 0.407, a value that was found to be much lower than the intended amount. As a result, FC5 was

excluded from the model. PE5, which had a loading of 0.668, was omitted from the analysis due to theoretical rationales and its potential negative effect on the Average Variance Extracted (AVE). Three other items, namely TaskC1 (loading: 0.246), TC1 (loading: 0.543), and TC6 (loading: 0.624), were excluded from the analysis due to their loadings being below the predetermined threshold.

Following the aforementioned modifications as shown in Table 3, the reassessed model exhibits a comprehensive and acceptable array of loadings for each item within the various structures. One notable observation is to the loading of TaskC2, which, while marginally below the established threshold of 0.7, is considered to be within an acceptable range. The result is based on the compensatory high loadings of TaskC3 and TaskC4, which contribute to the overall dependability of the "Task Characteristics" construct. According to (Byrne, 2010), loadings that are equal to or more than 0.6 are considered acceptable, as long as they contribute to a high overall loading that results in an average variance extracted (AVE) score over the conventional threshold of 0.6. After eliminating items with loadings below the accepted threshold, the study revealed that all the constructs under consideration show an internal consistency surpassing the set threshold. The evidence for this strong internal consistency is further supported by the Cronbach's alpha values, which range from 0.728 to 0.953, and the composite reliability values, which range from 0.733 to 0.962 as shown in Table 3.

5.2.2 Convergent Validity

From Convergent validity perspective, which refers to the extent to which different measures of the same construct are positively related to each other. In order to establish convergent validity, it is recommended that the Average Variance Extracted (AVE) values

should be equal to or greater than 0.5 (Hair et al., 2016). As shown in Table 3, AVE ranged from 0.534 to 0.876 which support strong convergent validity in the present investigation.

Table 3: Convergent Validity, Internal Consistency, and Reliability results

		Convergent Validity		Construct Reliability (Internal Consistency)	
Constructs	**Items**	**Factor Loading**	**Average variance extracted (AVE)**	**Cronbach's alpha**	**Composite reliability**
Behavioral Intention	BI1	0.828	0.702	0.858	0.858
	BI2	0.850			
	BI3	0.867			
	BI4	0.805			
Effort Expectancy	EE1	0.787	0.647	0.728	0.733
	EE2	0.811			
	EE3	0.813			
Facilitating Conditions	FC1	0.936	0.876	0.953	0.962
	FC2	0.921			
	FC3	0.951			
	FC4	0.936			
Habit Behavior	HB1	0.843	0.693	0.776	0.781
	HB2	0.888			
	HB3	0.761			
Hedonic Motivation	HM1	0.850	0.832	0.898	0.900
	HM2	0.944			
	HM3	0.939			
Performance Expectancy	PE1	0.779	0.534	0.825	0.828
	PE2	0.738			
	PE3	0.702			
	PE4	0.713			

	PE6	0.726			
	PE7	0.721			
Social Influence	SI1	0.707	0.745	0.819	0.824
	SI2	0.930			
	SI3	0.933			
Task Characteristics	TaskC2	0.677	0.763	0.841	0.928
	TaskC3	0.959			
	TaskC4	0.954			
Task-Technology Fit	TTF1	0.796	0.645	0.816	0.817
	TTF2	0.837			
	TTF3	0.820			
	TTF4	0.758			
Technology Characteristics	TC2	0.759	0.572	0.813	0.813
	TC3	0.772			
	TC4	0.760			
	TC5	0.777			
	TC7	0.712			

5.2.3 Discriminant Validity

The present analysis places significant emphasis on the assessment of discriminant validity to measure the model, employing the Fornell-Larcker Criterion, Cross Loading, and HTMT Ratio as evaluation measures. The comprehensive examination guarantees that every construct inside the model is a distinct entity, rather than simply a reflection of other variables within the model (Hair et al., 2016).

5.2.3.1 Fornell-Larcker Criterion

In the context of conducting a rigorous assessment of discriminant validity using the Fornell-Larcker criterion. The average variance extracted (AVE) of a latent variable should

consistently exhibit greater values than the squared correlations between said latent variable and all other variables inside the model according to (Henseler et al., 2015). The fulfillment of this criterion serves as evidence that each variable in the model effectively represents a distinct phenomenon that is not duplicated by other variables. As shown in Table 4, the discriminant validity of various constructs has been established with impressive results. Behavioral Intention, with an AVE square root of 0.838, surpasses its highest correlation with other constructs. Effort Expectancy achieves an AVE square root of 0.804, exceeding its most significant correlation with Performance Expectancy at 0.635. Facilitating Conditions demonstrates its validity with an AVE square root of 0.936, which is notably higher than its correlations with other components. Similarly, Habit Behavior, with an AVE square root of 0.832, outdoes its peak correlation with other constructs. Hedonic Motivation possesses an AVE square root of 0.912, emphasizing its distinctiveness from other constructs. Performance Expectancy, with its AVE square root of 0.730, surpasses its correlations with other constructs. Social Influence, with an AVE square root of 0.863, surpasses its correlations with other components. Task Characteristics shows a prominent AVE square root of 0.874, which is higher than its correlation with Habit Behavior at 0.597. Task-Technology Fit's AVE square root is 0.803, exceeding correlations with other constructs. Lastly, Technology Characteristics, with an AVE square root of 0.757, surpasses its associations with other components, cementing the discriminant validity of each construct mentioned.

Table 4: Discriminant Validity (Fornell-Larcker Criterion Test Results)

	BI	E	C	B	M	E	I	Task C	TF	C
BI	0.838									
EE	0.508	.804								
FC	0.321	.323	.936							
HB	0.671	.515	.388	.832						
HM	0.573	.51	.315	.646	.912					
PE	0.626	.635	.293	.541	.603	.73				
SI	0.424	.448	.497	.386	.348	.424	.863			
TaskC	0.547	.44	.348	.597	.504	.474	.42	0.874		
TTF	0.679	.479	.277	.582	.544	.535	.334	0.503	.803	
TC	0.578	.454	.305	.578	.52	.521	.321	0.495	.629	.757

In conclusion, the thorough validation process enhances the dependability and distinctiveness of each construct, guaranteeing that the model effectively encompasses diverse and separate phenomena.

5.2.3.2 Cross-Loading

In conjunction with the Fornell-Larcker criterion, the assessment of discriminant validity is further enhanced through the examination of cross-loadings of indicators. In essence, it is

expected that the loadings of an indicator on its designated latent variable exhibit a much greater magnitude compared to its loadings on any other latent variables (Gansser et al., 2021). As shown in Table 5, the cross-loadings highlight that elements related to Behavioral Intention (BI) predominantly load on their intended construct, with values between 0.805 and 0.867, emphasizing its distinctiveness. Similarly, Effort Expectancy (EE) components primarily load on their construct, with values from 0.787 to 0.813. Facilitating Conditions (FC) items show consistent high loadings above 0.900, while Habit Behavior (HB) items, like HB1 and HB2, load at 0.843 and 0.888 respectively. Hedonic Motivation (HM) items all load above 0.850, distinctly higher than other constructs. Performance Expectancy (PE) items, like PE1 at 0.779, clearly support their construct's distinctiveness. Social Influence (SI) items, such as SI2 and SI3, have loadings of 0.930 and 0.933, reinforcing their discriminant validity. Most TaskC items associate mainly with the Task Characteristics (TaskC) construct, even if some values, like 0.677 for TaskC2, are relatively lower, they still outperform loadings on other constructs. Task-Technology Fit (TTF) items all load over 0.750, indicating strong discriminant validity. Lastly, Technology Characteristics (TC) items all load higher than other constructs. The finding supports the notion that each construct in the model possesses distinct and distinctive characteristics.

Table 5: Discriminant Validity (Cross-Loading)

	I	E	C	B	M	E	I	askC	TF	C
BI1	**.828**	.433	.252	.552	.476	.547	.321	0.466	.554	.488
BI2	**.85**	.4	.254	.576	.488	.528	.369	0.416	.542	.506
BI3	**.867**	.427	.278	.586	.49	.519	.396	0.465	.568	.485
BI4	**.805**	.441	.293	.533	.465	.505	.334	0.487	.612	.46
EE1	.381	**.787**	.25	.349	.41	.558	.379	0.358	.361	.354
EE2	.402	**.811**	.279	.44	.411	.519	.355	0.345	.388	.307
EE3	.437	**.813**	.253	.449	.41	.462	.349	0.358	.405	.423
FC1	.308	.318	**.936**	.384	.335	.294	.449	0.336	.265	.302
FC2	.264	.266	**.921**	.318	.238	.23	.444	0.284	.238	.234
FC3	.339	.339	**.951**	.415	.348	.318	.497	0.371	.284	.334
FC4	.282	.277	**.936**	.322	.24	.244	.466	0.3	.245	.259
HB1	.59	.483	.354	**.843**	.596	.472	.317	0.51	.514	.471
HB2	.56	.454	.295	**.888**	.551	.439	.332	0.496	.487	.495
HB3	.521	.339	.32	**.761**	.458	.438	.315	0.484	.447	.478
HM1	.571	.502	.302	.585	**.85**	.546	.344	0.486	.52	.464
HM2	.485	.427	.273	.583	**.944**	.546	.294	0.436	.472	.47
HM3	.494	.455	.279	.591	**.939**	.552	.305	0.445	.487	.483
PE1	.527	.453	.179	.459	.452	**.779**	.329	0.376	.433	.37
PE2	.41	.453	.204	.41	.424	**.738**	.307	0.35	.367	.34
PE3	.462	.369	.256	.345	.4	**.702**	.314	0.321	.378	.401

PE4	.45	.447	.206	.333	.439	**.713**	.289	0 .317	.386	.405
PE6	.461	.502	.206	.421	.447	**.726**	.308	0 .361	.415	.4
PE7	.42	.567	.243	.396	.485	**.721**	.312	0 .347	.358	.367
SI1	.349	.433	.273	.333	.344	.418	**.707**	0 .372	.321	.266
SI2	.372	.357	.505	.333	.277	.346	**.93**	0 .363	.272	.275
SI3	.371	.368	.495	.33	.28	.334	**.933**	0 .349	.27	.286
Task C2	.253	.219	.24	.282	.268	.266	.238	**0 .677**	.27	.303
Task C3	.568	.441	.335	.611	.526	.473	.418	**0 .959**	.512	.495
Task C4	.542	.444	.329	.598	.476	.463	.41	**0 .954**	.488	.47
TTF1	.467	.333	.219	.462	.409	.356	.213	0 .393	**.796**	.509
TTF2	.543	.389	.208	.456	.468	.428	.253	0 .394	**.837**	.522
TTF3	.562	.382	.228	.494	.425	.407	.288	0 .447	**.82**	.515
TTF4	.597	.426	.234	.454	.442	.515	.31	0 .38	**.758**	.475
TC2	.468	.339	.261	.468	.449	.384	.267	0 .437	.501	**.759**
TC3	.406	.361	.266	.425	.423	.421	.256	0 .356	.443	**.772**
TC4	.412	.336	.229	.428	.407	.384	.2	0 .347	.427	**.76**
TC5	.414	.366	.24	.407	.363	.404	.271	0 .374	.471	**.777**
TC7	.479	.314	.16	.453	.327	.377	.215	0 .354	.527	**.712**

5.2.3.3 Heterotrait-Monotrait Ratio of Correlations (HTMT)

The Heterotrait-Monotrait Ratio (HTMT) is a robust and validated metric used also to evaluate the extent of discriminant validity. HTMT values close to 1 indicate a lack of discriminant validity, discriminant validity is often established when the heterotrait-monotrait

ratio of correlations (HTMT) value is found to be less than 0.85 (Henseler et al., 2015). As shown in Table 6, the HTMT value of 0.639 demonstrates that there is strong discriminant validity between Behavioral Intention and Effort Expectancy, as it is safely below the threshold of 0.85. The discriminant validity of Behavioral Intention and Facilitating Conditions is further supported by the low HTMT value of 0.352, emphasizing their differentiation. The HTMT result of 0.822, albeit closer to the threshold, indicates that there is still discriminant validity between the conceptions of behavioral intention and habit behavior. The HTMT score of 0.646 provides evidence for the discriminant validity between Behavioral Intention and Hedonic Motivation. The discriminant validity between the notions of behavioral intention and performance expectancy is strengthened by the HTMT value of 0.741. The discriminant validity between Behavioral Intention and Social Influence is supported by a notably low HTMT value of 0.507. The HTMT score of 0.614 supports the notion of discriminant validity between the notions of behavioral intention and task characteristics. The HTMT score of 0.807 is rather high, but it still falls below the threshold of 0.85, which confirms the presence of discriminant validity. The discriminant validity between the dimensions of behavioral intention and technology characteristics is further supported by an HTMT score of 0.690. The remaining pairs have HTMT values below 0.85, with a significant number falling below 0.7.

Table 6: Discriminant Validity (HTMT)

	I	E	C	B	M	E	I	askC	TF	C
BI										
EF	.639									
FC	.352	.385								
HB	.822	.678	.447							
HM	.646	.626	.333	.768						
PE	.741	.826	.329	.674	.699					
SI	.507	.584	.559	.486	.405	.519				
Task	.614	.54	.383	.707	.555	.552	.494			
TTF	.807	.616	.312	.729	.63	.645	.407	0.586		
TC	.69	.583	.343	.728	.607	.636	.393	0.585	.769	

In summary, the tests consistently demonstrate the discriminant validity of the constructs, emphasizing their distinction and uniqueness within the model. This also confirms the reliability of the measuring model for research purposes. The HTMT values for all constructs in the study are found to be below the threshold of 0.85, suggesting a distinct separation between the constructs. The Fornell-Larcker criterion is met when the square root of each construct's average variance extracted (AVE) is greater than its correlations with other constructs. The examination of cross-loadings provides additional confirmation that items primarily load most strongly on their target constructs, hence strengthening the discriminant validity.

5.3 Structural Model Assessment

After confirming the measurement model, the subsequent step involves assessing the structural model. This involves analyzing the model's predictive prowess and the relationships between constructs, as outlined by (Hair et al., 2016; Hair et al., 2017; Al-Sharafi et al., 2021). Accordingly, the structural model's evaluation was carried out using the bootstrap resampling method with 5,000 re-samples to determine the coefficient of determination (R^2), path coefficients, and their associated t-values.

5.3.1 Assessment of the Path Coefficient

This stage emphasizes understanding the depth and significance of relationships among latent variables. To enhance the robustness of the analysis, inferential statistics, primarily t-values, are derived using the bootstrap standard error. Established benchmarks, such as a t-value of 1.96 for a two-tailed test (Cohen et al., 1948), serve as thresholds to determine significance. Though our hypotheses were directional, a two-tailed test approach was chosen over a one-tailed test to ensure a conservative stance, allowing room for unanticipated findings especially when exploring new technologies like ChatGPT. Additionally, based on prior studies, the path coefficient is considered significant if its confidence interval does not encompass zero (Hair et al., 2012; Hair et al., 2016; Al-Sharafi et al., 2021).

Eight hypotheses out of eleven we supported as shown in Table 7, Performance Expectancy (PE) demonstrated a positive influence on Behavioral Intention, reflected by a coefficient (β) of 0.235 and a significant t-value of 4.641, thus supporting H1. Similarly, while Effort Expectancy revealed a slight negative correlation with Behavioral Intention, with β at -0.014 and a t-value of 0.265, H2 wasn't supported. Social Influence positively affected

Behavioral Intention (β = 0.096, t = 2.057), affirming H3. Yet, Facilitating Conditions (β = -0.012, t = 0.289) and Hedonic Motivation (β = 0.027, t = 0.595) didn't present substantial impacts on Behavioral Intention, not giving credence to H4 and H5 respectively. Habit Behavior exerted a strong positive influence on Behavioral Intention (β = 0.303, t = 6.072), endorsing H6. The relationship between Task Characteristics and Task-Technology Fit was also positive and significant (β = 0.254, t = 5.154), validating H7. Likewise, Technology Characteristics showcased a strong positive relation with Task-Technology Fit (β = 0.503, t = 11.712), underpinning H8. Task-Technology Fit's influence on Performance Expectancy (β = 0.535, t = 13.084) and Behavioral Intention (β = 0.34, t = 6.371) was both substantial and positive, bolstering H9 and H10. Lastly, the bond between Technology Characteristics and Effort Expectancy was positive and highly significant (β = 0.454, t = 9.861), reinforcing H11.

Table 7: Path Coefficient result

	Structural Path	Original sample(O)	Sample mean(M)	Standard deviation (STDEV)	T statistics (|O/STDEV|)	P values	Empirical Evidence
1	Performance Expectancy -> Behavioral Intention	0.235	0.234	0.051	4.641	0	Supported
2	Effort Expectancy -> Behavioral Intention	-0.014	-0.01	0.051	0.265	0.791	Not Supported
3	Social Influence -> Behavioral Intention	0.096	0.096	0.047	2.057	0.04	Supported
4	Facilitating Conditions -> Behavioral Intention	-0.012	-0.009	0.04	0.289	0.772	Not Supported
5	Hedonic Motivation -> Behavioral Intention	0.027	0.026	0.045	0.595	0.552	Not Supported
6	Habit Behavior -> Behavioral Intention	0.303	0.304	0.05	6.072	0	Supported

7	Task Characteristics -> Task-Technology Fit	0.254	0.253	0.049	5.154	0	Supported
8	Technology Characteristics -> Task-Technology Fit	0.503	0.506	0.043	11.712	0	Supported
9	Task-Technology Fit -> Performance Expectancy	0.535	0.539	0.041	13.084	0	Supported
10	Task-Technology Fit -> Behavioral Intention	0.34	0.337	0.053	6.371	0	Supported
11	Technology Characteristics -> Effort Expectancy	0.454	0.457	0.046	9.861	0	Supported

5.3.2 Assessment of the Coefficient of determination (R2)

The Coefficient of Determination (R^2) assesses a model's explanatory capacity, reflecting how well it explains variability in dependent variables. (Falk et al., 1992) suggest a minimum R^2 value of 0.10 for appropriateness. This study evaluates R^2 values for Behavioral Intention, Effort Expectancy, Performance Expectancy, and Task-Technology Fit, ensuring the model's robustness and its relevance in academia and practice. From Table 8, Behavioral Intention has an R^2 of 0.627, explaining 62.7% of its variance, significantly surpassing (Falk et al., 1992) benchmark. Effort Expectancy's R^2 is 0.206, indicating 20.6% explained variance, which is notable and above the set criterion. Performance Expectancy has an R^2 of 0.287, accounting for 28.7% variance, demonstrating moderate model effectiveness. Task-Technology Fit's R^2 is 0.444, accounting for 44.4% variance, showcasing the model's strong predictive capability.

Table 8: Coefficient of determination (R2) result

Constructs	R-square	R-square adjusted
Behavioral Intention	0.627	0.621
Effort Expectancy	0.206	0.204
Performance Expectancy	0.287	0.285
Task-Technology Fit	0.444	0.442

In conclusion, it can be observed that all the R-squared values exceed the threshold of 0.10, so confirming the model's strength and dependability in clarifying the connections among the constructs investigated in this study.

5.3.3 Assessment of the Effect Size (f2)

The effect size (f2) evaluation is essential for our model, quantifying the influence of exogenous constructs on endogenous ones. According to (Cohen, 1988), an f2 beyond 0.02 indicates a small effect, above 0.15 suggests medium, and over 0.35 denotes a large effect.

As detailed in Table 9, Performance expectancy has a small effect on behavioral intention (f2 = 0.068). It signifies that developers' expectations about ChatGPT's performance moderately influence their adoption decision. Effort expectancy's non-significant effect (f2 = 0.000) indicates that the anticipated effort for ChatGPT doesn't notably impact adoption intention. Social influence's effect is negligible (f2 = 0.016), hinting at limited peer influence on developers' intent for ChatGPT. Hedonic motivation has an almost negligible effect (f2 = 0.001), suggesting minimal impact of enjoyment on the adoption decision. Habit behavior has a small effect (f2 = 0.114), showing limited habitual influence on ChatGPT adoption intention. Task characteristics and task-technology fit both show small effects (f2 = 0.087). Technology

characteristics, especially task-technology fit and effort expectancy, have medium effects with f2 values of 0.344 and 0.259, indicating a moderate impact on technology-task fit and effort expectancy. Task-Technology Fit has a large effect on Performance Expectancy (f2 = 0.402) and a medium effect on behavioral intention (f2 = 0.176), emphasizing the significance of task-technology alignment.

Table 9: F2 Results

Structural Path	f-square	Effect
Performance Expectancy -> Behavioral Intention	0.068	Small
Effort Expectancy -> Behavioral Intention	0	No Effect
Social Influence -> Behavioral Intention	0.016	Negligible
Facilitating Conditions -> Behavioral Intention	0	No Effect
Hedonic Motivation -> Behavioral Intention	0.001	Negligible
Habit Behavior -> Behavioral Intention	0.114	Small
Task Characteristics -> Task-Technology Fit	0.087	Small
Technology Characteristics -> Task-Technology Fit	0.344	Medium
Task-Technology Fit -> Performance Expectancy	0.402	Large
Task-Technology Fit -> Behavioral Intention	0.176	Medium
Technology Characteristics -> Effort Expectancy	0.259	Medium

5.3.4 Assessment of the Predictive Relevance (Q2)

The study evaluates the predictive accuracy of the ChatGPT adoption model using the PLSpredict methodology, which involves k-fold cross-validation. Key metrics include Q2 for

predictive relevance, RMSE for prediction accuracy, and MAE for average error magnitude. A Q2 value above zero indicates good predictive relevance, as noted by (Hair et al., 2017). Lower RMSE and MAE values suggest superior predictive accuracy. The dataset is segmented into ten parts for a thorough assessment. Table 10 reveals, Behavioral Intention (BI) constructs (BI1 to BI4) have Q^2predict values between 0.332 and 0.383, indicating strong predictive importance. Their RMSE and MAE are also within acceptable bounds. Effort Expectancy (EE) constructs (EE1 to EE3) have Q^2predict values from 0.085 to 0.17, meeting the minimum Q2 criterion. RMSE for EE1 and EE3 suggests the PLS-SEM model outperforms linear modeling in accuracy. Performance Expectancy constructs (PE1 to PE6) have Q^2predict values from 0.133 to 0.157, indicating moderate predictive importance. RMSE and MAE values are within acceptable limits.

Lastly, Task-Technology Fit (TTF) constructs (TTF1 to TTF4) show Q^2predict values between 0.245 and 0.305, satisfying the threshold for predictive relevance. Their RMSE and MAE values confirm this compliance.

Table 10: Prediction Summary Result

	Q^2predict	**PLS-SEM_RMSE**	**PLS-SEM_MAE**	**LM_RMSE**	**LM_MAE**
BI1	0.35	0.759	0.589	0.757	0.572
BI2	0.372	0.81	0.645	0.82	0.628
BI3	0.383	0.81	0.631	0.807	0.614
BI4	0.332	0.808	0.603	0.819	0.597
EE1	0.121	0.878	0.674	0.828	0.624
EE2	0.085	0.961	0.739	0.907	0.692
EE3	0.17	0.82	0.629	0.784	0.579
PE1	0.154	0.756	0.575	0.726	0.555
PE2	0.133	0.826	0.631	0.803	0.624
PE3	0.148	0.843	0.627	0.832	0.623
PE4	0.15	0.844	0.65	0.824	0.623

PE6	0.157	0.854	0.66	0.837	0.634
PE7	0.142	0.895	0.672	0.86	0.653
TTF1	0.28	0.839	0.628	0.849	0.632
TTF2	0.288	0.771	0.579	0.781	0.58
TTF3	0.305	0.781	0.596	0.797	0.604
TTF4	0.245	0.793	0.585	0.781	0.574

In summary, the PLS-SEM model demonstrates superior predictive relevance and accuracy when evaluating the adoption of ChatGPT in software development.

5.4 Importance-Performance Map Analysis (IPMA)

In this research, the Importance-Performance Map Analysis (IPMA) is used as an advanced technique within PLS-SEM to scrutinize the behavioral intention surrounding the adoption of ChatGPT in software development. Implementing the IPMA approach offers enhanced insights and a more nuanced understanding of the PLS-SEM findings (Ringle et al., 2016). The methodology integrates both the average values of latent constructs and their indicators, delivering an insightful analysis of importance and performance. As outlined by (Ringle et al., 2016), IPMA presumes that total effects delineate the predecessor factors' importance in shaping the target factor, while the average values of latent constructs reflect their performance.

The results of the IPMA are illustrated as shown in Table 11 and Figure 7, capturing the importance and performance metrics of various independent factors including Effort Expectancy, Facilitating Conditions, Habit Behavior, Hedonic Motivation, Performance Expectancy, Social Influence, Task Characteristics, Task-Technology Fit, and Technology Characteristics. These factors are analyzed for their effect on behavioral intention in adopting ChatGPT in software development.

In relation to the importance measure, Task-Technology Fit emerges as the most pivotal factor in predicting behavioral intention with a total effect of 0.466, followed by Habit Behavior (0.303) and Performance Expectancy (0.235). Despite Effort Expectancy and Facilitating Conditions registering low total effects (-0.014 and -0.012 respectively), they showcase significant performance values (70.092 and 63.959 respectively).

Focusing on the performance measure, Performance Expectancy is observed to have the highest value (73.026), suggesting that among all the constructs, individuals consider Performance Expectancy as the most satisfying, followed by Effort Expectancy (70.092) and Hedonic Motivation (68.047).

Table 11: Importance-Performance Map Analysis (IPMA) Assessment

	Behavioral Intention (Total Effects)	**Performance**
Effort Expectancy	-0.014	70.092
Facilitating Conditions	-0.012	63.959
Habit Behavior	0.303	61.986
Hedonic Motivation	0.027	68.047
Performance Expectancy	0.235	73.026
Social Influence	0.096	65.419
Task Characteristics	0.118	64.232
Task-Technology Fit	0.466	65.797
Technology Characteristics	0.228	62.366

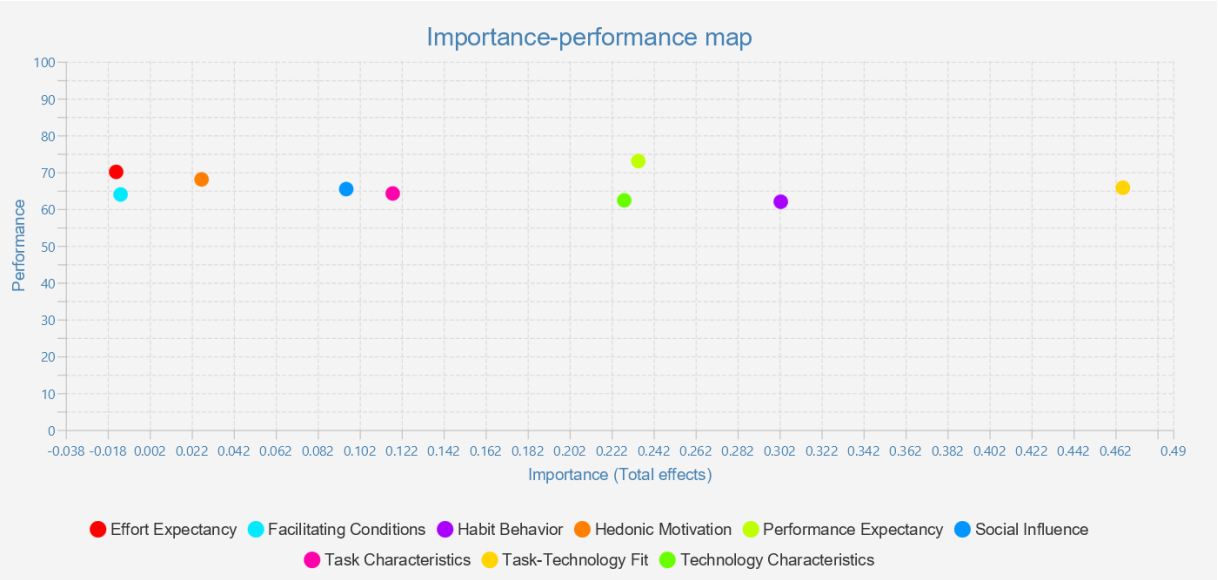

Figure 7: Importance-Performance Map Analysis (IPMA) Assessment

In summary, the IPMA results align with and bolster the structural model outcomes in Table 7. Task-Technology Fit stands out as the most potent predictor variable for the behavioral intention in the context of ChatGPT adoption in software development, in terms of both importance and performance measures. The alignment between the high importance and performance of Task-Technology Fit underscores its crucial role in influencing the behavioral intention towards ChatGPT adoption in software development.

5.5 Artificial Neural Network (ANN)

PLS-SEM was employed initially in this investigation to scrutinize the posited relationships and discern the factors that influence software developers' intention to adopt ChatGPT. In the subsequent phase, the ANN analytical technique was harnessed to rank the driving elements behind the adoption. In line with prior studies (Ooi et al., 2018; Lee et al., 2020; Qasem et al., 2020; Alkawsi et al., 2021; Kalinić et al., 2021; Sharma et al., 2021), this research opted for ANN as it surpasses traditional statistical instruments like multiple linear regression, binary logistics regression, and SEM in pinpointing both linear and intricate non-

linear associations rather than simply gauging linear connections. A standard neural network consists of an input layer, one or more hidden layers, and an output layer. The activation function employed for the hidden and output neurons was the sigmoid function. The spectrum for both input and output neurons were consistently kept within (0, 1) to optimize the model's efficacy (Liu et al., 2012; Kalinić et al., 2021; Sharma et al., 2021). To circumvent the predicament of overfitting, a ten-fold cross-validation strategy was integrated into the ANN models. Accordingly, 90% of the data was dedicated to the training segment, with the remaining 10% allocated for testing (Alshamaila et al., 2013; Kalinić et al., 2021).

The focus was primarily on Behavioral Intention (BI), Task-Technology Fit (TTF), and effort and performance expectancies, leading to the development of four distinct ANN models as shown in Table 12. The first, as shown in Figure 8, considered various exogenous constructs as input layers, including Task-Technology Fit (TTF), Task Performance Expectancy (PE), and Effort Expectancy (EE), culminating in an output layer representing BI.

Table 12: RMSE values of ANN models

	Model 1		Model 2		Model 3		Model 4	
Network	RMSE (Training)	RMSE (Testing)	RMSE (Training)	RMSE (Testing)	RMSE (Training)	RMSE (Testing)	RMSE (Training)	RMSE (Testing)
1	0.092	0.094	0.098	0.102	0.120	0.105	0.103	0.076
2	0.092	0.084	0.109	0.095	0.119	0.121	0.110	0.111
3	0.085	0.070	0.106	0.075	0.121	0.104	0.098	0.097
4	0.090	0.098	0.107	0.087	0.120	0.112	0.102	0.105
5	0.094	0.096	0.101	0.093	0.118	0.127	0.098	0.095
6	0.091	0.088	0.100	0.125	0.121	0.112	0.097	0.105
7	0.094	0.082	0.103	0.116	0.119	0.149	0.109	0.095

8	0.093	0.096	0.109	0.109	0.119	0.114	0.109	0.129
9	0.092	0.071	0.101	0.081	0.120	0.107	0.105	0.086
10	0.091	0.111	0.098	0.097	0.135	0.166	0.099	0.098
Mean	**0.091**	**0.089**	**0.103**	**0.098**	**0.121**	**0.122**	**0.103**	**0.100**
Standard deviation	**0.002547**	**0.012820382**	**0.004136905**	**0.015615615**	**0.004988925**	**0.020655217**	**0.005143663**	**0.014240844**

The accuracy assessment involved computing the RMSE, a trusted metric (Alkawsi et al., 2021; (Abbasi et al., 2021; Al-Shihi et al., 2018). Model 1's RMSE values (0.091 for training and 0.089 for testing) highlighted its significant predictive accuracy.

Expanding this, Model 2 (Figure 9) centered on Technology Characteristics (TC), Task Characteristics (TaskC) and their effect on TTF, recording average RMSE values of 0.103 and 0.098 for training and testing respectively. Model 3 (Figure 10) focused on the impact of Technology Characteristics (TC) on the effort expectancy of ChatGPT, with RMSE values of 0.121 and 0.122 for training and testing, respectively. Lastly, Model 4 (Figure 11) delved into the connections between TTF and PE demonstrating RMSE values of 0.103 and 0.100 for training and testing, respectively. Each model's low RMSE values underscore their predictive accuracy and reliability in diverse facets of ChatGPT adoption in software development.

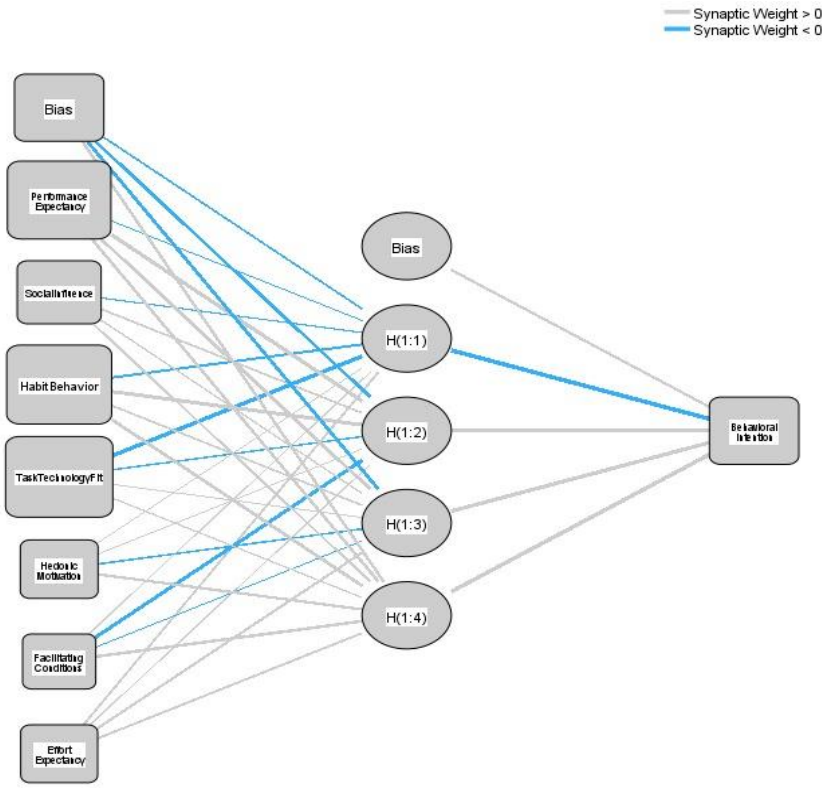

Figure 8: ANN Model 1

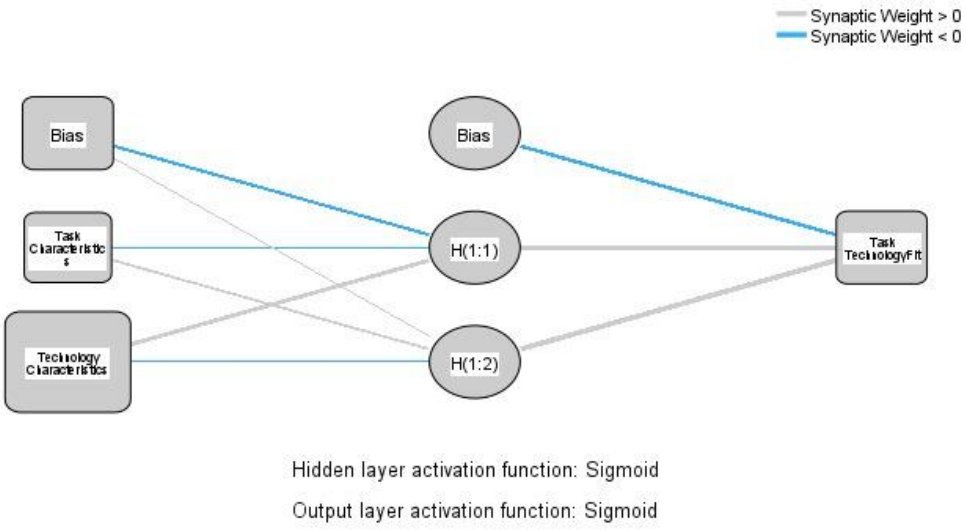

Figure 9: ANN Model 2

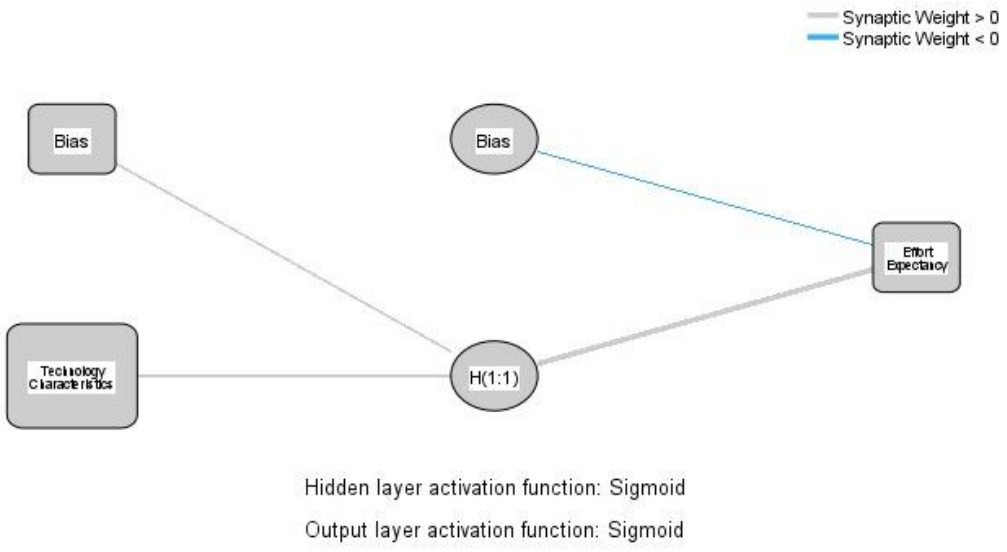

Figure 10: ANN Model 3

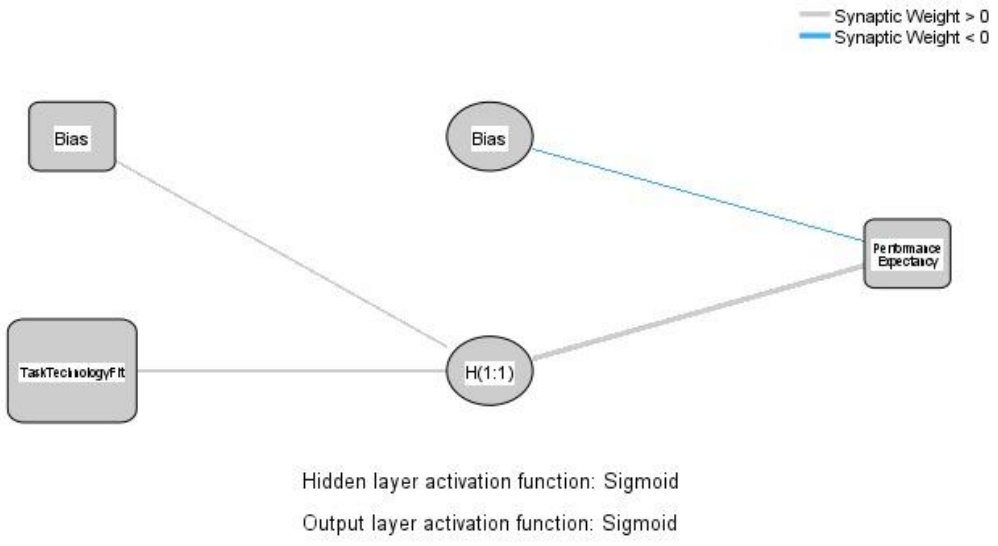

Figure 11: ANN Model 4

In conclusion, the holistic and multi-dimensional view offered by the four precise ANN models contributes substantially to understanding the factors influencing the Behavioral Intention to adopt ChatGPT in software development. The detailed and comprehensive analysis, marked by low average RMSE values, reinforces the dependability and accuracy of these models in decoding various facets of ChatGPT adoption in software development. Beyond linear associations, the study's use of ANN helps unravel the intricate influences of each variable, delivering a well-rounded, extensive understanding essential for both academic and practical spheres. The commendable results and insights from this study affirm ANN's efficacy in executing robust and insightful analyses, heralding future research and exploration in this arena.

5.6 Sensitivity Analysis

Sensitivity analysis within the model ascertains the alterations in the endogenous constructs predicated upon the shifts in the exogenous constructs aligned with the model (Chong, 2013; Al-Shihi et al., 2018; Kalinić et al., 2021). In this investigation, the contributive significance of each predictor towards the adoption of ChatGPT in software development was gauged through the sensitivity analysis for model 1 which focus on the main construct which Behavior Intention. Consequently, both the relative significance of the determinants and their normalized importance (NI) were elucidated.

The sensitivity analysis outcomes, presented in Table 13 and Figure 12, denote that Task-Technology Fit (NI = 100.0%) stands out as the predominant exogenous construct influencing the adoption of ChatGPT in software development. This is succeeded by Performance Expectancy (NI = 71.3%), Habit Behavior (NI = 72.3%), Social Influence (NI = 27.2%), Hedonic Motivation (NI = 13.8%), Effort Expectancy (NI = 11.9%), and lastly, Facilitating Conditions (NI = 8.6%).

Table 13: Sensitivity analysis for model 1

Neural Networks	PE	SI	HB	TTF	HM	FC	EE
1	0.252	0.098	0.264	0.282	0.049	0.013	0.043
2	0.232	0.063	0.238	0.350	0.060	0.033	0.023
3	0.160	0.097	0.200	0.300	0.067	0.047	0.129
4	0.260	0.116	0.248	0.282	0.056	0.012	0.027
5	0.234	0.058	0.218	0.359	0.073	0.031	0.027
6	0.295	0.063	0.246	0.301	0.031	0.039	0.025
7	0.216	0.084	0.232	0.389	0.041	0.010	0.028
8	0.216	0.100	0.271	0.342	0.031	0.033	0.007
9	0.237	0.095	0.243	0.337	0.027	0.031	0.030
10	0.224	0.111	0.201	0.379	0.011	0.034	0.040
Mean importance	0.233	0.089	0.236	0.332	0.045	0.028	0.038
Normalized importance (%)	71.3%	27.2%	72.3%	100.0%	13.8%	8.6%	11.9%
Ranking	3	4	2	1	5	7	6

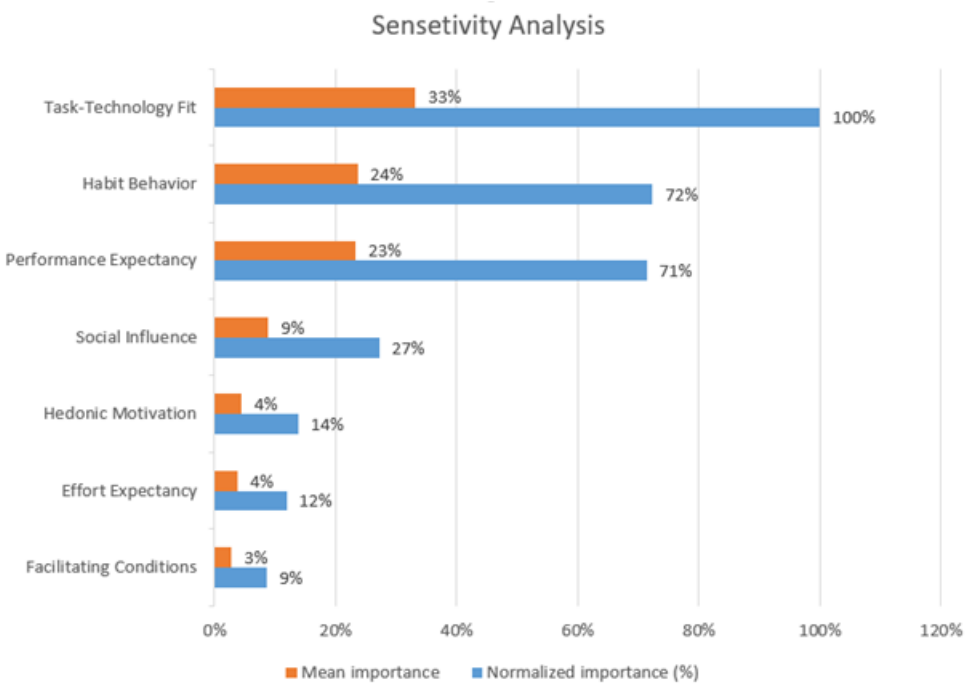

Figure 12: Sensitivity Analysis

Chapter 6: Discussion

6.1 Overview

The research is primarily centered on identifying and measuring the factors that enable the adoption of ChatGPT within software development. ChatGPT, with its capabilities in software development such as requirement elections, code generation, and test cases development, through natural language processing, offers a transformative approach to traditional software development processes. Our earlier literature review confirmed the profound capabilities of AI tools and ChatGPT in this domain, underscoring its potential significance.

For our investigative framework, we turned to the UTAUT2 model integrated with Task-Technology Fit (TTF). This integration was chosen due to its comprehensive nature and proven efficiency in understanding technology adoption scenarios. The UTAUT2 already encompasses essential factors influencing user adoption, and when combined with TTF, it allows a closer examination of how well a technology, in this instance ChatGPT, aligns with specific tasks within software development. To ground our study in real-world contexts, data was gathered from professionals actively engaged in software development. Their firsthand insights and experiences enriched our dataset, ensuring it was both relevant and representative of the current industry nuances. Our analysis of this data yielded supportive results for the proposed model. A compelling revelation most of the factors (PE, SI, HB, TaskC, TC, TTF) we identified had significant effects on the adoption of ChatGPT in software development except three factors (Effort Expectancy, Hadronic Motivation, Facilitating Conditions). This not only underscores the robustness of our model but also its efficacy in elucidating developers' intentions towards ChatGPT adoption.

6.2 UTAUT2 Factors Effects

The results from our study firmly establish a significant correlation between performance expectancy and behavioral intention with reference to ChatGPT (H1: β = 0.235, t-value = 4.641). This finding is congruent with prior research conducted in similar technological domains (Venkatesh et al., 2003; Alblooshi et al., 2019 ; Jaradat et al., 2020; Wan et al., 2020; Basak et al., 2020). Performance expectancy, in essence, evaluates the degree to which a user believes that employing a particular technology will aid in task accomplishment. In the context of software development, where efficiency and precision are paramount, a tool's perceived utility can greatly influence its adoption rate. If developers ascertain that ChatGPT can amplify their productivity or refine the quality of their outputs, it logically follows that they would be inclined to incorporate it into their working paradigm. Furthermore, this relationship between performance expectancy and behavioral intention underscores a fundamental tenet of technology adoption: the perceived benefits must outweigh the costs, be they tangible or intangible. In environments characterized by evolving challenges and rapid technological progressions, such as software development, the efficacy of a tool becomes a primary consideration. Hence, the prominence of performance expectancy in our findings aligns with the practical imperatives of the field and offers a reflection of developers' pragmatic approach to technology adoption. The same has been confirmed in sensitivity analysis, performance expectancy's normalized importance is recorded at 71.3%, placing it third in ranking. This metric reaffirms the weightage of performance expectancy in influencing developers' behavioral intentions towards ChatGPT. In essence, the higher normalized importance signifies that when evaluating the suite of UTAUT2 factors, performance expectancy consistently emerges as a major determinant, even amidst slight variances in the sample data.

Further, our study results pertaining to Effort Expectancy (EE) did not align with some commonly held views in technology adoption literature (Venkatesh et al., 2003; Basak et al., 2020). Specifically, Effort Expectancy (EE) did not present a significant positive influence on Behavioral Intention (H2: $\beta = -0.014$, t-value = 0.265). Historically, ease of use or the perceived effort associated with technology adoption has been a strong predictor of behavioral intention, especially in the earlier stages of technology adoption. However, in the context of ChatGPT's integration into software development, it appears that the effort required to learn or adopt the technology doesn't hold paramount importance. A plausible explanation could be the evolving nature of the software development domain, where professionals are accustomed to the continuous learning curve, often associated with new tools and technologies. Software developers may prioritize the functionality, performance enhancement, and alignment with tasks over the ease of initial adoption. The sensitivity analysis further attests to this finding, showing a relatively lower normalized importance for Effort Expectancy (11.9%). This places it sixth in the ranking, suggesting that other factors, such as the task-technology fit and performance expectancy, overshadow the importance of effort expectancy in the decision-making process for adopting ChatGPT. Another noteworthy aspect is that while the main analysis indicated a non-significant role of EE on behavioral intention, the sensitivity analysis still acknowledged its presence, albeit in a limited capacity. This highlights the nuanced nature of technology adoption, where even factors with reduced direct significance can play indirect roles, possibly influencing other stronger mediators or serving specific niche segments within the software development community.

Moving to Social Influence (SI) factor, the data gleaned from our study sheds light on the substantive influence of Social Influence (SI) on Behavioral Intention (H3: $\beta = 0.096$, t-

value = 2.057). This outcome, underscoring the weight of social factors in the adoption decision-making process, harmonizes well with prior research findings in diverse IT contexts (Venkatesh et al., 2012; Jaradat et al., 2014 ; Faqih et al., 2015) In many technology adoption scenarios, especially in fields as collaborative as software development, the opinions, endorsements, and practices of peers, colleagues, or thought leaders can considerably sway an individual's intention to embrace a particular technology. In this case, ChatGPT's integration into software development is not exempt from this phenomenon. It's interesting to consider that, in an era dominated by individualized, bespoke software solutions, social considerations continue to hold. Developers, while highly skilled and often autonomous in their tasks, still lean on collective wisdom, shared experiences, and the reassurance of community validation when it comes to adopting novel tools. This underscores the inherently social nature of the software development milieu, where shared challenges, solutions, and tools foster a collaborative spirit and collective mindset. The sensitivity analysis further corroborates the relevance of Social Influence, attributing it a normalized importance of 27.2%, ranking it fourth. While not the most paramount determinant, it's unmistakably evident that SI holds considerable clout in shaping behavioral intentions towards ChatGPT adoption.

Furthermore, the analysis from our investigation delineates an interesting observation regarding Facilitating Conditions (FC) and their relationship to Behavioral Intention (H4: β = -0.012, t-value = 0.289). Contrary to expectations and in divergence from previous studies in diverse IT (Venkatesh et al., 2003; Faqih et al., 2015), the influence of FC on the intention to adopt ChatGPT in software development was not statistically significant. Facilitating Conditions typically encompass the infrastructural and organizational support mechanisms that are believed to aid users in leveraging a particular technology. This can range from training

programs, accessible user manuals, technical support, to even peer assistance. Historically, the presence of robust facilitating conditions has been seen as a booster for the adoption of new technologies, as users often feel more supported and confident to transition. Yet, our findings present an anomaly. One potential reason could be that software developers, often well-versed with novel technologies, might not rely heavily on external facilitating conditions when assessing a new tool like ChatGPT. Their evaluation could be predominantly anchored on the intrinsic capabilities of the technology, rather than the external support structures available. The sensitivity analysis further elucidates this observation, as FC holds a normalized importance of just 8.6%, positioning it at the seventh rank. This suggests that while facilitating conditions may be pivotal in other settings, their influence on software developers' decisions to adopt ChatGPT is minimal.

The same was with Hedonic Motivation (HM), our dataset underscores an intriguing dimension regarding Hedonic Motivation (HM) and its association with Behavioral Intention (H5: $\beta = 0.027$, t-value = 0.595). Notably, the influence of HM on the intention to adopt ChatGPT in software development was not statistically significant, standing in contrast to prior findings in several IT research spheres (Van der Heijden, 2004; Salimon et al., 2017; Paulo et al., 2018; Kim et al., 2019). Hedonic Motivation generally pertains to the pleasure or intrinsic satisfaction a user derives from using a technology. Historically, technologies that offer an enjoyable or stimulating experience, irrespective of their practical utility, are more likely to witness higher adoption rates. Given the nature of ChatGPT and its primary role in software development, one might surmise that hedonic attributes may not be paramount in a developer's decision-making calculus. After all, software development is a task-driven field, with primary emphasis on functionality, efficiency, and reliability. While an enjoyable user experience is

always a plus, it might not be the linchpin that determines a developer's willingness to adopt a tool like ChatGPT. The sensitivity analysis corroborates this perspective, with HM showcasing a normalized importance of 13.8%, granting it the fifth rank. This suggests that while hedonic aspects could be influential in other technological contexts, their bearing on the adoption of ChatGPT among software developers appears to be limited.

Lastly, the data underscores a pronounced influence of Habitual Behavior (HB) on Behavioral Intention (H6: $\beta = 0.303$, t-value = 6.072), positioning it as a formidable determinant in the adoption of ChatGPT for software development. This observation is harmonious with the findings of many prior studies conducted in various IT domains (Limayem et al., 2007). Habitual Behavior typically refers to the automatic behavioral response developed through prolonged and consistent usage of a technology. In essence, once a user becomes accustomed to a particular tool or system, the habituated behavior can significantly influence their intention to continue using it, sometimes even superseding cognitive evaluations of its performance or usefulness. In the realm of software development, the development environment and tools play a pivotal role in defining the rhythm and flow of a developer's daily tasks. Consequently, once developers become acquainted with certain tools, like ChatGPT, and embed them into their routine workflows, the act becomes instinctual, thereby heightening the probability of its continued adoption. The sensitivity analysis lends further credence to this perspective. With HB exhibiting a normalized importance of 72.3%, it secures the second rank. This implies that the habitual use of ChatGPT can significantly bolster its long-term adoption among software developers. When developers familiarize themselves with ChatGPT, and it becomes part of their habitual repertoire, the likelihood of sustained utilization amplifies.

6.3 Joint Effects between TTF and UTAUT2

We delved into five hypotheses (H7, H8, H9, H10, H11) that emerged from the union of TTF and UTAUT2 frameworks. The primary objective was to decode the complexities surrounding the adoption of ChatGPT in the domain of software development. The empirical results not only supported but also robustly affirmed all five hypotheses. This observation resonates with previous studies (Zhou et al., 2010; D'Ambra et al., 2013; Paulo et al., 2018; Rai et al., 2019) that have attempted to integrate TTF with various adoption models in diverse IT sectors. It was particularly revealing to note the substantial positive effects both task and technology characteristics exert on TTF, as evidenced by H7: ($\beta = 0.254$, t-value = 5.154), and H8: ($\beta = 0.503$, t-value = 11.712). TTF theory has long suggested that these intertwined characteristics are crucial in molding user perceptions, particularly regarding the compatibility between a technology and its intended tasks. In the context of ChatGPT, this means that as task and technology characteristics become more refined and aligned, users perceive a heightened sense of fit between the software development tasks and ChatGPT's capabilities. An intriguing observation from our data was the pronounced influence of technology characteristics on TTF, surpassing that of task characteristics. This might hint at the evolving nature of technology, like ChatGPT, becoming more adaptable and multifaceted, thereby offering a broader range of solutions to meet varied task requirements.

Historically, TTF's influence on the propensity to embrace new technologies was predominantly assessed in the light of task and technology characteristics. But it's essential to recognize that in today's rapidly evolving technological landscape, where platforms like ChatGPT are pushing boundaries, the lines are blurring. The dynamism and adaptability of technology might be overshadowing traditional task constraints. By weaving TTF into our

analytical tapestry, we've not just offered a fresh perspective on technology adoption but also underscored the importance of ensuring that technology evolves in tandem with tasks. This symbiosis is particularly pivotal in domains like software development, where the right technology-task alignment can drive innovation, efficiency, and overall productivity.

Further, the empirical evidence gleaned from our research underscores the influential role of Task-Technology Fit (TTF) in shaping the intention to adopt ChatGPT within the realm of software development, as shown by H10: ($\beta = 0.34$, t-value = 6.371). The data suggests that the closer ChatGPT's features align with the specific needs and challenges of software development, the stronger the inclination of developers to integrate it into their workflow. This is aligned with the underlying principle of technology adoption—that tools are only truly embraced when they demonstrably support users in executing their tasks more proficiently. The same has been confirmed by the sensitivity analysis, TTF recorded a normalized importance of 100.0%, ranking it first among all the UTAUT2 factors under examination. This implies that, across various neural network assessments, TTF consistently emerged as the most pivotal determinant influencing the adoption of ChatGPT in the context of software development. The high normalized importance underscores TTF's consistent and dominant influence, even in the face of potential fluctuations or variations in the input data. This leading rank in the sensitivity analysis further corroborates TTF's paramount role in technology adoption decisions. Specifically, in an industry like software development, which often juggles rapidly evolving technologies and methodologies, a tool's fitness for the task at hand remains an uncompromisable criterion. The seamless alignment between a tool's capabilities and the task requirements not only augments productivity but also fosters innovation and drives excellence.

Thus, the sensitivity analysis accentuates the indispensable nature of TTF and reinforces its foundational significance in the technology adoption landscape within software development.

Diving deeper, we observed that TTF also indirectly steers intention through the mediating effect of performance expectancy (H9: β = 0.535, t-value = 13.084), this implies that greater TTF leads to greater performance expectancy in relation to the adoption of ChatGPT in software development. Previous studies carried out in varying IT domains provided similar findings (Venkatesh et al., 2003; Paulo et al., 2018; Wan et al., 2020). This suggests that a tool's fit isn't just about matching functions to tasks but also about influencing user expectations. If ChatGPT embodies the desired attributes tailored to the unique demands of software development, it not only motivates adoption but also elevates expectations regarding improved performance outcomes. ChatGPT uses different technical aspects to perform its tasks. Therefore, it is important to have a strong compatibility between technology characteristics and task characteristics for the effective integration of ChatGPT into software development to achieve encouraging outcomes. The nexus between TTF and performance expectancy can be traced back to the alignment between what developers seek and what ChatGPT offers, which in turn molds their performance anticipations. Emphasizing the alignment, the insights suggest that developers might be reticent to onboard ChatGPT unless they discern tangible enhancements to their performance metrics.

In a similar vein, we discerned that technology characteristics have a notable positive impact on effort expectancy (H11: β = 0.454, t-value = 9.861). This pivotal relationship in our study accentuates the significance of technology characteristics in amplifying facets of effort expectancy. It implies that when ChatGPT's features empower developers to achieve their goals more efficiently, it implicitly reduces the perceived effort as confirmed in a previous study by

(Gefen et al., 2005). Intriguingly, while technology characteristics showed a positive relationship with effort expectancy, our data (in H2) didn't exhibit a positive influence of effort expectancy on behavioral intention. This poses a thought-provoking quandary. Perhaps while ChatGPT's features might make tasks seem easier, this reduced effort alone isn't a strong enough catalyst for adoption. It underscores that while ease-of-use is pivotal, developers might prioritize other facets, like performance enhancements or compatibility, when deciding to adopt new tools like ChatGPT. This divergent insight serves as a reminder that the multifaceted nature of technology adoption cannot be distilled into singular metrics but requires a more holistic understanding.

Chapter 7: Conclusion

This chapter presents a summary of the study, highlights the primary discoveries, addresses research constraints, and elaborates on the theoretical implications and the model for evaluating acceptance. Furthermore, the chapter offers recommendations and potential avenues for future investigations.

7.1 Research Summary

In exploring the adoption dynamics of ChatGPT in software development, this research embarked on a journey to delve into the factors that might guide software developers towards embracing or steering clear of this AI-driven tool. Leveraging an online survey, the study tapped into a diverse cohort of professionals in the software development domain, reaching out to them through platforms such as LinkedIn and Facebook. Out of the 608 responses, 461 were included for analysis, while 147 irrelevant responses were excluded. The survey indicated a significant gender gap, with 95% male and 5% female respondents out of 461 total. Participants over 30 (61%) showed the strongest preference for ChatGPT, followed by the age group of 25 to 30 (36%). Most respondents (94%) had university-level education, and programmers (55%) were the largest professional category, demonstrating ChatGPT's broad appeal in software development. To sift through the gathered data, the study employed a sophisticated blend of SEM-ANN and PLS-SEM analytical methods. This approach illuminated both the straightforward and complex interrelationships within the proposed research model. Eight hypotheses have been supported out of eleven. Performance Expectancy (PE), Social Influence (SI), Habit Behavior (HB), has a positive relationship with Behavioral Intention (BI) However, Effort Expectancy (EE), Hedonic Motivation, and Facilitating Conditions (FC) do not show significant relationships with BI. Task Characteristics (TaskC) and Technology Characteristics

(TC) significantly influence Task-Technology Fit (TTF), while TTF positively impacts PE and BI. Technology Characteristics (TC) significantly affect Effort Expectancy (EE). The main model accuracy reached to 63%, Sensitivity analysis showed that TTF has the highest influence on BI then HB, and PE came on the 2nd and 3rd respectively.

Among the standout revelations, developers seemed to be swayed towards ChatGPT when they discerned direct benefits in their work, be it in terms of enhanced efficiency or smoother collaborative engagements. Moreover, endorsements from influential peers or thought leaders in the developer community also appeared to bolster ChatGPT's adoption rates. Intriguingly, while many conventional factors like perceived ease of use and hedonic motivations often determine technology uptake, in the context of ChatGPT's integration in software development, these elements didn't resonate as strongly. Instead, tangible utility and alignment with specific developmental tasks took precedence. This finding underscores the discerning nature of professionals in the software development arena and hints at the need for a shift in strategy when promoting AI-driven tools like ChatGPT. Rather than solely spotlighting the tool's inherent simplicity or the pleasure derived from its use, it would be more prudent to emphasize its tangible benefits and its ability to seamlessly mesh with the multifaceted tasks of software development.

7.2 Key Findings

This research centered around five pivotal research questions. The primary findings of our study will provide insights to address them.

7.2.1 ChatGPT and AI Tools Support SDLC

We have answered the first question (Q1) on how AI tools and ChatGPT utilized in SDLC in chapter 2, Artificial intelligence (AI), including tools like ChatGPT, is reshaping SDLC, offering transformative impacts across its various stages. (Dong et al., 2023) highlight the cooperative and collaborative nature of software development, segmented traditionally into stages such as requirement analysis, design, and deployment. (Arora et al., 2021) emphasize the potential of AI to simplify the complexities of SDLC by suggesting its integration in software requirement prioritization, cost estimation, and defect prediction, among other aspects. Echoing this, (Kumari et al., 2018) underscore the growing symbiosis between AI and Software Engineering. They detail AI's role in stages such as Software Planning, where it aids in project feasibility; during defining requirements, where AI ensures precise software requirement specification document generation; and in the constructing phase, where AI helps in producing design document specification (DDS)-compliant code. Furthermore, in the testing phase, AI aids in defect identification and rectification, aligning with SRS-defined quality metrics. Lastly, AI is instrumental in the deployment stage, facilitating user Acceptance testing and ensuring seamless maintenance for users (Kumari et al., 2018).

7.2.2 UTAUT2 & TTF Integration Introduced an Effective Adoption Model

The effective model to capture developers' intention to use ChatGPT in software development (Q2) was the integration of UTAUT2 and TTF as detailed in chapter 3, UTAUT2 provided a framework to understand the determinants of technology adoption by focusing on factors like performance expectations, effort expectations, social influence, facilitating conditions, hedonic motivation, and habitual behavior which has a track record of effectively

analyzing adoption and acceptance of diverse technologies in multiple contexts (Venkatesh et al., 2012). On the other hand, the TTF theory emphasizes the alignment of technology characteristics with user tasks, underlining the need for a technology's capabilities to match the demands and challenges of specific tasks for optimal user performance and satisfaction (Goodhue et al., 1995). The combination of these theories allowed for a holistic understanding of factors propelling ChatGPT's adoption in the software development domain. Previous research supports the effectiveness of using UTAUT2 and TTF theories together in diverse technology adoption contexts (Sharif et al., 2019; Beier et al., 2020; Faqih et al., 2021).

7.2.3 Main Factors Influence Developers Adoption

The main factors that influenced software developers' adoption for ChatGPT (Q3) was detailed in chapter 6, developers were more inclined towards ChatGPT when they perceived direct benefits in their tasks, such as heightened efficiency and enhanced collaboration. This tangible benefit perception, labeled as Performance Expectancy (PE), played a significant role in their acceptance. Another notable influence was Social Influence (SI), where endorsements or recommendations from influential figures in the software development community significantly bolstered the adoption of ChatGPT. Habit Behavior (HB), which encompasses the established habits and frequent interactions with the tool, also emerged as a guiding factor. The study further illuminated the importance of both Task Characteristics (TaskC) and Technology Characteristics (TC) in influencing the Task-Technology Fit (TTF). This fit suggests that if ChatGPT aligns well with the tasks developers are engaged in, they're more likely to adopt it. In fact, Task-Technology Fit was found to hold a profound influence on the behavioral intention to use ChatGPT.

7.2.4 Model Accuracy

The fourth question (Q4) on to what extent the model explained the determinants of ChatGPT adoption in software development was answered in chapter 6, by evaluating factors such as Performance Expectancy, Social Influence, Habit Behavior, Task Characteristics, and Technology Characteristics, the model sheds light on both the direct and intricate interrelationships that sway adoption behavior. Achieving a model accuracy of 63% showcases its considerable predictive capacity. The importance of Task-Technology Fit, which signifies the harmony between ChatGPT's capabilities and the intricacies of developmental tasks, is prominently highlighted, directly influencing both Performance Expectancy and Behavioral Intention. Yet, while the model is comprehensive, the incorporation of additional factors might further refine our understanding. Organizational culture, for instance, can play a pivotal role. Different organizations have varying degrees of receptivity to AI-driven tools, and understanding how organizational dynamics and values influence ChatGPT adoption can provide richer insights. Similarly, addressing specific concerns and reservations of developers can offer a more nuanced understanding of barriers to adoption. Lastly, security factor could be taken into consideration, the organization will be willing to expose their source code to third party tool and how this can impact their adoption strategies.

7.2.5 Organizations Adoption Strategies

To answer the last question (Q5), to foster a successful implementation of ChatGPT in software development, organizations can adopt the following strategies.

Firstly, emphasize how ChatGPT can enhance a developer's skill set, making them indispensable in roles that demand creativity, complex problem-solving, and informed decision-

making. By showcasing the integral role of ChatGPT in modern development processes, underscore its potential in aiding developers' career progression. Secondly, facilitate detailed training programs, ensuring that developers become adept at leveraging ChatGPT to its fullest potential, understanding both its capabilities and the wide range of benefits it can offer. Thirdly, spotlight the direct advantages of integrating ChatGPT, such as streamlined coding processes and minimized debugging efforts. Reinforce this by sharing success stories or case studies from within the organization, illustrating tangible outcomes. Fourthly, pinpoint specific tasks within the software development lifecycle where ChatGPT shines. By demonstrating how the tool can enhance and simplify these tasks, developers can see firsthand the tool's relevance to their daily work. Fifthly, initiate pilot projects allowing a subset of developers to actively engage with and test out ChatGPT. This hands-on approach provides a safe space for experimentation, and the feedback garnered can be invaluable for wider implementation. Sixthly, motivate developers by offering incentives or recognitions for those who effectively incorporate ChatGPT into their workflow, leading to notable gains in productivity or improvements in code quality. Lastly, maintain open channels for developers to voice their feedback and concerns regarding ChatGPT. Addressing these in real-time can foster trust and ensure that the tool aligns closely with the developers' needs.

By adopting these strategies, organizations can not only enhance the acceptance of ChatGPT among developers but also ensure that it becomes an integral, value-adding component in the software development process.

7.3 Theoretical Contributions

The current investigation into the adoption of ChatGPT within the software development environment, underpinned by the UTAUT2 model and enriched with the Task-Technology Fit (TTF), contributes to academic discourse in multiple significant ways.

First, integration of UTAUT2 and TTF Constructs help in addressing the gap in the literature, this research brings together UTA+UT2 constructs with Task-Technology Fit dimensions. The integration provides a comprehensive insight into the dynamics governing the adoption of AI-driven tools, such as ChatGPT, in the software development domain which provide a better understanding of ChatGPT's adoption mechanics and its implications. Also, by incorporating both frameworks, the study delves deeper into understanding not just the behavioral intentions of software developers but also the compatibility of the technology with the intrinsic tasks of software development.

Secondly, whereas much of the existing literature has delved into the acceptance patterns of technologies in general settings, our study pioneers an exploration into the realm of software development. This shifts the lens from broad-based technology acceptance studies to the specific nuances and exigencies of the software development industry, thereby adding depth and specificity to our theoretical understanding.

Thirdly, diverging from traditional applications that typically position Effort Expectancy (EE), Facilitating Condition (FC), Habitual Behavior (HB), within consumer domains, this research underscores its significance in professional environments like software development. This extends our understanding of how AI tools are adopted in professional contexts.

Fourthly, by leveraging sensitivity analysis, the research injects a layer of methodological rigor, enabling a dynamic evaluation of the relative import of various factors.

This approach provides a fresh perspective, adapting a method typically associated with machine learning to the field of technology acceptance, thus enhancing the methodological richness of the domain.

In essence, this research doesn't just reaffirm established theories but illuminates new intersections, particularly concerning AI-enhanced tools in specialized professional settings. The study's findings have the potential to guide future research and industry practices in harnessing the full potential of AI technologies in software development and other professional domains.

7.4 Practical Implications

The utilization of ChatGPT within the software development landscape offers myriad practical implications. At its core, ChatGPT provides the potential to revolutionize workflows in software firms. As developers engage with it across different facets of the Software Development Life Cycle—from eliciting requirements to the intricate nuances of coding and testing—it becomes clear that the pathway to development becomes more refined and time-efficient.

This is not just about efficiency; it’s a gateway to a new dimension of collaboration. Imagine teams having access to a neutral entity like ChatGPT that can readily identify and clarify ambiguities in project specifications. This doesn't just mitigate misunderstandings, but by acting as a consistent mediator, it paves the way for a smoother project lifecycle, minimizing internal disagreements. Furthermore, this research reveals potential pitfalls and areas where developers may feel they will replaceable when integrating AI-driven tools. Organizations can use these insights to tailor their training endeavors, ensuring their staff not only become

proficient with the tools but also develop trust in them as this tool came to support them not to replace them.

From a financial vantage point, ChatGPT's integration could denote significant cost savings. Reducing manual hours and curtailing post-launch glitches means organizations can enjoy a balance of monetary savings and bolstered product quality. Also, being at the nexus of this technological shift provides adopters with a competitive edge. By delivering products faster, ensuring top-tier software quality, and consistently meeting user demands, they can establish a distinct presence in a competitive marketplace.

7.5 Limitations and Future work

While our research offered a comprehensive look into the integration of ChatGPT in the software development lifecycle, there are areas where our approach could have been refined and expanded upon.

Firstly, the demographic composition of our participant pool was primarily centered around specific regions (UAE, Jordan), which could potentially restrict the wider applicability of our findings. Distinct demographics might possess different viewpoints which might not have been adequately represented in our study. Furthermore, for unknown reason, low female participation was observed which highlights potential areas for future research.

Secondly, the bulk of our feedback came from developers. While their insights are critical, given they experience the full software development cycle, we shouldn't overlook the perspectives of other integral roles like QA, DBA, and business analysts. A deeper dive into each specific area of the software development lifecycle, with tailored research, could furnish a more holistic view.

Thirdly, ChatGPT version should be taken into consideration. Feedback from users of the free version, ChatGPT 3.5, might differ from those using the more advanced 4.0. This differentiation could be pivotal, given that the earlier version might have inconsistencies that aren't present in the latter which may impact their decision to adopt. Delving deeper into these discrepancies could help shape our understanding better.

Fourthly, our primary method of data collection was through surveys. Although surveys are valuable in garnering a wide range of responses, they might not always reflect nuanced insights. The complexity arises from the reliance on self-reported data. Such data can sometimes be influenced by biases, such as the participants' desire to present themselves in a positive light or potential inaccuracies due to imperfect memory recall. Future studies might enhance validity by integrating more objective measures or directly observing behavioral trends. Interviews approach might have provided a more in-depth perspective, especially since developers might have emotional reactions to tools like ChatGPT due to fears of job displacement. A discussion-based format could help mitigate such emotional responses and provide a more grounded view.

Fifthly, while the amalgamation of UTAUT2 and TTF constructs offered a comprehensive theoretical approach, it also inherently left out some insights that alternative models or unexplored constructs might provide. Several influential factors, such as organizational culture, training initiatives, or individual career trajectories or concerns from being replaced, which could impact the adoption curve, were not within the scope of our study. These unexplored factors hint at potential avenues for further research.

Seventhly, the unexpected findings from our sensitivity analysis, particularly the 100% value for TTF, warrants a dedicated probe. Such anomalies beckon a closer scrutiny, potentially through controlled experiments or case studies, to decode underlying factors or biases that might

have influenced this outcome. Additionally, our research highlighted certain factors – Habitual Behavior, Facilitating Condition, and Hedonic Motivation – that remained unsupported which against the normal behavior of adoption model. Delving into these dimensions could be pivotal in sketching a holistic image of the adoption dynamics. Perhaps, a sector-specific or role-centric exploration might offer revelations on their relevance.

Eighthly, it would be of immense value to draw comparisons with emerging tools, such as Google's Bard. As the AI landscape evolves rapidly, juxtaposing ChatGPT's capabilities with its contemporaries could offer fresh perspectives.

Lastly, future endeavors could involve a comprehensive case study where an application is developed from scratch using ChatGPT. Such a hands-on approach would not only shed light on the tool's capabilities across different stages of development but also pinpoint specific areas that demand enhancement.

References

Dong, Y., Jiang, X., Jin, Z. & Li, G., 2023. Self-collaboration Code Generation via ChatGPT.

Belani, H., Vuković, M. & Car, Ž., 2019. Requirements Engineering Challenges in Building AI-Based Complex Systems.

Naikwadi, B.H., Kharade, K.G., Yuvaraj, S. & Vengatesan, K., 2021. A Systematic Review of Blockchain Technology and Its Applications. In: Recent Trends in Intensive Computing, pp.467-473. IOS Press.

Kharade, K., Budake, R. & Bhoite, S., 2023. CHALLENGES AND FUTURE OF AI-BASED REQUIREMENT ANALYSIS: A LITERATURE REVIEW. Available at: https://www.researchgate.net/publication/368668029.

Hourani, H., Hammad, A. & Lafi, M., 2019. The Impact of Artificial Intelligence on Software Testing.

Arora, S. & Kumar, V., 2021. DESIGNING SOFTWARE WITH ARTIFICIAL INTELLIGENCE TO TEST IT AGAINST SRS.

Hossain, M. & Chen, H., 2022. Application of Machine Learning on Software Quality Assurance and Testing: A Chronological Survey.

Battina, D.S., 2019. Artificial Intelligence in Software Test Automation: A Systematic Literature Review.

Chen, H. & Hossain, M., 2022. Application of Machine Learning on Software Quality Assurance and Testing: A Chronological Survey.

Yapay Zekâ ile Güvenli Yazılım Geliştirme, 2021. Usage of Artificial Intelligence to Improve Secure Software Development.

Sharma, S. & Pandey, S.K., 2013. Integrating AI Techniques in Requirements Phase: A Literature Review.

Sharma, S. & Pandey, S.K., 2019. Integrating AI Techniques in Requirements Elicitation.

KUMARI, V. & KULKARNI, S., 2018. Use of Artificial Intelligence in Software Development Life Cycle Requirements and its Model.

Poldrack, R.A., Lu, T. & Beguš, G., 2023. AI-ASSISTED CODING: EXPERIMENTS WITH GPT-4.

Paik, I. & Wang, J.-W., 2021. Improving Text-to-Code Generation with Features of Code Graph on GPT-2. Available at: https://www.mdpi.com/2079-9292/10/21/2706.

Trummer, I., 2022. CodexDB: Generating Code for Processing SQL Queries using GPT-3 Codex.

Priya, R., Wang, X., Sun, Y. & Hu, Y., 2017. A Deep Dive into Automatic Code Generation Using Character Based Recurrent Neural Networks.

Perez, L. & Ottens, L., 2021. Automatic Code Generation using Pre-Trained Language Models.

Feng, Y., Vanam, S., Cherukupally, M., Zheng, W., Qiu, M. & Chen, H., n.d. (2023) Investigating Code Generation Performance of ChatGPT with Crowdsourcing Social Data.

Habib, M., 2019. On the Automated Entity-Relationship and Schema Design by Natural Language Processing. The International Journal of Engineering and Science, 8(11), pp.42-48.

Giray, G., 2021. A software engineering perspective on engineering machine learning systems: State of the art and challenges. Journal of Systems and Software, 180. Available at: https://doi.org/10.1016/j.jss.2021.111031.

Beier, G., Ullrich, A., Niehoff, S., Reißig, M. & Habich, M., 2020. Industry 4.0: How it is defined from a sociotechnical perspective and how much sustainability it includes—A literature review. J. Clean. Prod., 259. [CrossRef].

Paulo, M.M., Rita, P., Oliveira, T. & Moro, S., 2018. Understanding mobile augmented reality adoption in a consumer context. Journal of hospitality and tourism technology, 9(2), pp.142-157.

Rauschnabel, P.A., Rossmann, A. & tom Dieck, M.C., 2017. An adoption framework for mobile augmented reality games: the case of Pokémon Go. Comput. Hum. Behav., 76, pp.276-286.

Kaur, D.P., Mantri, A. & Horan, B., 2020. Enhancing student motivation with use of augmented reality for interactive learning in engineering education. Procedia Computer Science, 172, pp.881-885.

Venkatesh, V., Thong, J.Y. & Xu, X., 2012. Consumer acceptance and use of information technology: extending the unified theory of acceptance and use of technology. MIS Q., 36(1), pp.157-178.

Sharif, A., Afshan, S. & Qureshi, M.A., 2019. Acceptance of learning management system in university students: an integrating framework of modified UTAUT2 and TTF theories. Int. J. Technol. Enhanc. Learn. (IJTEL), 11(2), pp.201-229.

Jaradat, M.I.R.M., Ababneh, H.T., Faqih, K.M. and Nusairat, N.M., 2020. Exploring cloud computing adoption in higher educational environment: an extension of the UTAUT

model with trust. International Journal of Advanced Science and Technology, 29(5), pp.8282–8306.

Faqih, K.M., 2020. The influence of perceived usefulness, social influence, internet self-efficacy and compatibility on users' intentions to adopt e-learning: investigating the moderating effects of culture. International E-Journal of Advances in Education, 5(15), pp.300–320.

Rauschnabel, P.A. and Ro, Y.K., 2016. Augmented reality smart glasses: an investigation of technology acceptance drivers. Int. J. Technol. Market., 11(2), pp.123–148.

Gerup, J., Soerensen, C.B. and Dieckmann, P., 2020. Augmented reality and mixed reality for healthcare education beyond surgery: an integrative review. Int. J. Med. Educ., 11, p.1.

Venkatesh, V., Morris, M.G., Davis, G.B. and Davis, F.D., 2003. User acceptance of information technology: toward a unified view. MIS Q., pp.425–478.

Tavares, J., Goul˜ao, A. and Oliveira, T., 2018. Electronic health record portals adoption: empirical model based on UTAUT2. Inf. Health Soc. Care, 43(2), pp.109–125.

Goodhue, D.L. and Thompson, R.L., 1995. Task-technology fit and individual performance. MIS Q., 19(2), pp.213–236.

Alraja, M.N., Hammami, S., Chikhi, B. and Fekir, S., 2016. The influence of effort and performance expectancy on employees to adopt e-government: evidence from Oman. Int. Rev. Manag. Market., 6(4), pp.930–934.

Moghavvemi, S., Phoong, S.W. and Lee, S.T., 2017. Impact of perceived desirability, perceived feasibility and performance expectancy on use of IT innovation: technology adoption decisions and use behaviour. Vidyodaya Journal of Management, 3(1), pp.43–76.

Faqih, K.M. and Jaradat, M.I.R.M., 2015. Assessing the moderating effect of gender differences and individualism-collectivism at individual-level on the adoption of mobile commerce technology: TAM3 perspective. J. Retailing Consum. Serv., 22, pp.37–52.

Rahi, S., Ghani, M., Alnaser, F. and Ngah, A., 2018. Investigating the role of unified theory of acceptance and use of technology (UTAUT) in internet banking adoption context. Management Science Letters, 8(3), pp.173–186.

Raza, S.A., Shah, N. and Ali, M., 2019. Acceptance of mobile banking in Islamic banks: evidence from modified UTAUT model. Journal of Islamic Marketing, 10(1), pp.357–376.

Jaradat, M.I.R.M. and Faqih, K.M., 2014. Investigating the moderating effects of gender and self-efficacy in the context of mobile payment adoption: a developing country perspective. Int. J. Bus. Manag., 9(11), p.147.

Gupta, A. and Dogra, N., 2017. Tourist adoption of mapping apps: a UTAUT2 perspective of smart travellers. Tourism Hospit. Manag., 23(2), pp.145–161.

Moorthy, K., Yee, T.T., T'ing, L.C. and Kumaran, V.V., 2019. Habit and hedonic motivation are the strongest influences in mobile learning behaviours among higher education students in Malaysia. Australas. J. Educ. Technol., 35(4), pp.174–191.

Lowry, P.B., Gaskin, J., Twyman, N., Hammer, B. and Roberts, T., 2012. Taking 'fun and games' seriously: proposing the hedonic-motivation system adoption model (HMSAM). J. Assoc. Inf. Syst. Online, 14(11), pp.617–671.

Kim, M.J. and Hall, C.M., 2019. A hedonic motivation model in virtual reality tourism: comparing visitors and non-visitors. Int. J. Inf. Manag., 46, pp.236–249.

Salimon, M.G., Yusoff, R.Z.B. and Mokhtar, S.S.M., 2017. The mediating role of hedonic motivation on the relationship between adoption of e-banking and its determinants. Int. J. Bank Market., 35(4), pp.558–582.

Javornik, A., Rogers, Y., Moutinho, A.M. and Freeman, R., 2016. Revealing the shopper experience of using a" magic mirror" augmented reality make-up application. Conference on Designing Interactive Systems, 2016, pp.871–882. Association for Computing Machinery (ACM).

Faqih, K.M.S. and Jaradat, M.I.R.M., 2021. Integrating TTF and UTAUT2 theories to investigate the adoption of augmented reality technology in education: Perspective from a developing country. doi:10.1016/j.techsoc.2021.101787

Hair, J.F., Ringle, C.M. and Sarstedt, M., 2011. PLS-SEM: indeed a silver bullet. Journal of Marketing Theory and Practice, 19(2), pp.139-152.

Hair, J.F. Jr., Matthews, L.M., Matthews, R.L. and Sarstedt, M., 2017. PLS-SEM or CB-SEM: updated guidelines on which method to use. International Journal of Multivariate Data Analysis, 1(2), pp.107-123.

Leong, L.Y., Hew, T.S., Ooi, K.B. and Chong, A.Y.L., 2020. Predicting the antecedents of trust in social commerce – a hybrid structural equation modeling with neural network approach. Journal of Business Research, 110, pp.24-40.

Dawes, J., 2008. Do data characteristics change according to the number of scale points used? An experiment using 5-point, 7-point and 10-point scales. International Journal of Market Research, 50(1), pp.61-104.

Lee, V. H., Hew, J. J., Leong, L. Y., Tan, G. W. H., & Ooi, K. B., 2020. Wearable payment: A deep learning-based dual-stage SEM-ANN analysis. Expert Systems with Applications, 157, pp.1-15.

Qasem, Y. A. M., Asadi, S., Abdullah, R., Yah, Y., Atan, R., Al-Sharafi, M. A., & Yassin, A. A., 2020. A multi-analytical approach to predict the determinants of cloud computing adoption in higher education institutions. Applied Sciences (Switzerland), 10(14).

Wu, J.H. and Wang, Y.M., 2006. Measuring KMS success: A respecification of the DeLone and McLean's model. Information & Management, 43(6), pp.728-739.

Igbaria, M., Guimaraes, T. and Davis, G.B., 1995. Testing the determinants of microcomputer usage via a structural equation model. Journal of Management Information Systems, 11(4), pp.87-114.

Davis, F.D., Bagozzi, R.P. and Warshaw, P.R., 1989. User acceptance of computer technology: A comparison of two theoretical models. Management Science, 35(8), pp.982-1003.

Hair, J.F., Hult, G.T.M., Ringle, C. and Sarstedt, M., 2017. A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM). Sage Publications.

Ringle, C.M. and Sarstedt, M., 2016. Gain more insight from your PLS-SEM results: The importance-performance map analysis. Industrial management & data systems.

Alkawsi, G. A., Ali, N., Mustafa, A. S., Baashar, Y., Alhussian, H., Alkahtani, A., Tiong, S. K., & Ekanayake, J., 2021. A hybrid SEM-neural network method for identifying acceptance factors of the smart meters in Malaysia: Challenges perspective. Alexandria Engineering Journal, 60(1), pp.227–240. Available at: https://doi.org/10.1016/j.aej.2020.07.002.

Kalinić, Z., Marinković, V., Kalinić, L., & Liébana-Cabanillas, F., 2021. Neural network modeling of consumer satisfaction in mobile commerce: An empirical analysis. Expert Systems with Applications, 175, Article 114803. Available at: https://doi.org/10.1016/j.eswa.2021.114803.

Lee, V. H., Hew, J. J., Leong, L. Y., Tan, G. W. H., & Ooi, K. B, 2020. Wearable payment: A deep learning-based dual-stage SEM-ANN analysis. Expert Systems with Applications, 157, pp.1–15. Available at: https://doi.org/10.1016/j.eswa.2020.113477.

Ooi, K. B., Lee, V. H., Tan, G. W. H., Hew, T. S., & Hew, J. J., 2018. Cloud computing in manufacturing: The next industrial revolution in Malaysia? Expert Systems with Applications, 93, pp.376–394. Available at: https://doi.org/10.1016/j.eswa.2017.10.009.

Qasem, Y.A.M. et al., 2020. A multi-analytical approach to predict the determinants of cloud computing adoption in higher education institutions. Applied Sciences (Switzerland), 10(14), Article 4905. Available at: https://doi.org/10.3390/app10144905.

Sharma, A., Dwivedi, Y. K., Arya, V., & Siddiqui, M. Q., 2021. Does SMS advertising still have relevance to increase consumer purchase intention? A hybrid PLS-SEM-neural network modelling approach. Computers in Human Behavior,124, pp.1–16. Available at: https://doi.org/10.1016/J.CHB.2021.106919.

Liu, L., Hu, G., Huang, Z., & Peng, Y., 2012. White cloud or black cloud: Opportunity and challenge of spectrum sharing on cloud computing. 2012 International Conference on Frontiers of Advanced Materials and Engineering Technology, FAMET 2012, 430–432, pp.1290–1293. Available at: https://doi.org/10.4028/www.scientific.net/AMR.430-432.1290.

Alshamaila, Y., Papagiannidis, S. and Li, F., 2013. Cloud computing adoption by SMEs in the north east of England: A multi-perspective framework. Journal of Enterprise Information Management, 26(3), pp.250–275. Available at: https://doi.org/10.1108/17410391311325225.

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D., 2003. User acceptance of information technology: Toward a unified view. MIS Quarterly, pp.425-478.

Limayem, M., Hirt, S.G. and Cheung, C.M., 2007. How habit limits the predictive power of intention: The case of information systems continuance. MIS Quarterly, pp.705-737.

Goodhue, D.L. and Thompson, R.L., 1995. Task-technology fit and individual performance. MIS Quarterly, pp.213-236.

Van der Heijden, H., 2004. User acceptance of hedonic information systems. MIS Quarterly, 28(4), pp.695-704.

Abbasi, G.A., Tiew, L.Y., Tang, J., Goh, Y.-N. & Thurasamy, R. 2021, 'The adoption of cryptocurrency as a disruptive force: Deep learning-based dual stage structural equation modelling and artificial neural network analysis', PLoS ONE, vol. 16, no. 3, pp. 1–26. Available at: https://doi.org/10.1371/journal.pone.0247582.

Al-Shihi, H., Sharma, S.K. & Sarrab, M. 2018, 'Neural network approach to predict mobile learning acceptance', Education and Information Technologies, vol. 23, no. 5, pp. 1805–1824. Available at: https://doi.org/10.1007/s10639-018-9691-9.

Chong, A.Y.L. 2013, 'Predicting m-commerce adoption determinants: A neural network approach', Expert Systems with Applications, vol. 40, no. 2, pp. 523–530. Available at: https://doi.org/10.1016/j.eswa.2012.07.068.

Cohen, A. & Holiday, M. 1984, Statistical for Social Scientists, Harper and Row, London.

Falk, R.F & Miller, N.B. 1992, A primer for soft modeling, University of Akron Press, Akron, OH.

Cohen, J. 1988, Statistical power analysis for the behavioral science, Lawrence Erlbaum Associates, Hillsdale, NJ.

Hair, J.F., Jr., Hult, G.T.M., Ringle, C.M. & Sarstedt, M. 2016, A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM), Sage Publications.

Ramayah, T., Cheah, J., Chuah, F., Ting, H. & Memon, M.A. 2018, Partial Least Squares Structural Equation Modeling (PLS-SEM) Using smartPLS 3.0, Pearson.

Byrne, B.M. 2010, Structural equation modeling with Mplus.

Henseler, J., Ringle, C.M. & Sarstedt, M. 2015, 'A new criterion for assessing discriminant validity in variance-based structural equation modeling', Journal of the Academy of Marketing Science, vol. 43, no. 1, pp. 115-135. Available at: https://doi.org/10.1007/s11747-014-0403-8.

Al Sarayreh, S. 2023, Understanding the impact of using Chatbot for shipment delivery toward environmental sustainability using the SEM-ANN approach.

Gansser, O. A. & Reich, C. S. 2021, A new acceptance model for artificial intelligence with extensions to UTAUT2: An empirical study in three segments of application. Technology in Society. Elsevier Ltd, vol. 65, p. 101535.

Al-Sharafi, M. A., Al-Qaysi, N., Iahad, N. A. & Al-Emran, M. 2022, Evaluating the sustainable use of mobile payment contactless technologies within and beyond the COVID-19 pandemic using a hybrid SEM-ANN approach. International Journal of Bank Marketing, vol. 40(5), pp. 1071–1095.

Hair, J.F., Sarstedt, M., Ringle, C.M. and Mena, J.A. (2012), “An assessment of the use of partial least squares structural equation modeling in marketing research”, Journal of the Academy of Marketing Science, Vol. 40 No. 3, pp. 414-433. Available at: https://doi: 10.1007/s11747-011-0261-6.

Zhou, T., Lu, Y., & Wang, B. 2010, Integrating TTF and UTAUT to explain mobile banking user adoption, Comput. Hum. Behav.

D’Ambra, J., Wilson, C.S., & Akter, S. 2013. Application of the task-technology fit model to structure and evaluate the adoption of E-books by A academics, J. Am. Soc. Inf.Sci. Technol.

Rai, R.S., & Selnes, F. 2019. Conceptualizing task-technology fit and the effect on adoption– A case study of a digital textbook service, Inf. Manag.

Wan, L., Xie, S., & Shu, A. 2020. Toward an understanding of university students’ continued intention to use MOOCs: when UTAUT model meets TTF model, SAGE Open.

Gefen, D., & Straub, D. 2005. A practical guide to factorial validity using PLS-Graph: tutorial and annotated example, Commun. Assoc. Inf. Syst.

Alblooshi, S., & Hamid, N.A.B.A. 2019, An empirical investigation of the unified theory of acceptance and use of technology in E-learning adoption in higher education institutions in the UAE, International Journal of Research and Review.

Basak, S.K., Wotto, M., & B´elanger, P. 2020, International students’ gender impact to use mobile learning in tertiary education, in: Inventive Communication and Computational Technologies, Springer, Singapore.