

# AI-Integrated IT Helpdesk Web Application with an LLM-Directed Workflow

Samuel Eras

A project submitted in partial fulfilment of the requirements  
for the Wrexham University award of  
B.Sc. (Hons) in Computer Science  
undertaken within the Subject of Cyber and Computing  
Faculty for Arts, Computing and Engineering  
Wrexham University

April 2025

## Acknowledgements

I would like to express my deepest gratitude to my supervisors, Miss Birch and Mr. Perera for their guidance, support, and encouragement. Their expertise and feedback helped me to reflect on my decisions and actions, come up with solutions and additional features, and ultimately to successfully implement this project.

I am also sincerely thankful to the lecturers of the project module, Mrs. Mayers and Mrs. Jose, who provided me with the fundamental knowledge and skills necessary for conducting research and writing the documentation. Their instruction and continuous support were essential to this process.

I am truly grateful to my family and friends for their encouragement and patience. Especially during challenging times, their understanding and support helped me to maintain motivation and stay balanced.

This project has been a challenging but rewarding experience. While I improved a lot in academic writing, researching, project management, and programming, I also learned valuable personal lessons in time management, self-belief, and perseverance.

## Abstract

This bachelor's project presents the development of an AI-integrated IT Helpdesk Web Application, which leverages a Large Language Model (LLM) and Retrieval Augmented Generation (RAG) within an LLM-directed workflow. The aim was to address common inefficiencies in traditional helpdesk systems, such as the time-consuming handling of recurring issues and the lack of detail in user-written tickets. Based on an in-depth literature review, the project investigates how integrating AI automations can enhance helpdesk efficiency. A research-backed application design was created and implemented, leveraging an LLM to reduce workload, increase ticket quality and ultimately lower operational costs.

The application was developed using a hybrid project management methodology, which combined Waterfall and Agile to provide a structured approach with iterative testing and adaptive prioritisation. The AI system was developed using the Python frameworks LangChain and LangGraph to build a custom state-driven workflow which leverages a locally deployed LLM to allow for dynamic decision-making and generation of responses. To ensure a performant and user-friendly application, the project employed React on the frontend and FastAPI on the backend, while utilising Azure for secure user authentication, and storing ticket data in MySQL. Additionally, the vector database Milvus was employed to support RAG-based document retrieval and similarity search.

The developed system allows users to interact with the AI chatbot to resolve their issues. For unresolvable problems, high-quality tickets are generated, enriched with solution suggestions and linked similar cases. Thorough evaluation and testing showed that the application met its key objective of increasing helpdesk efficiency. Furthermore, it offers a secure, responsive, and user-friendly experience, while preserving high maintainability and customisability.

In conclusion, this project demonstrates the feasibility and benefits of leveraging AI in IT Service Management. It provides a scalable and customisable solution for organisations to increase helpdesk productivity, reduce workload, and increase cost-effectiveness. The open-source nature of the system offers a solid foundation for further development and future research.

## Table of Contents

Acknowledgements.....	i
Abstract.....	ii
List of Figures .....	viii
Glossary .....	ix
1 Introduction.....	1
1.1 Background and Context.....	1
1.2 Problem Statement .....	1
1.3 Objectives and Scope .....	1
1.4 Significance and Motivation .....	2
1.5 Overview of the Document.....	3
2 Literature Review.....	4
2.1 Introduction.....	4
2.2 Helpdesk .....	4
2.2.1 ITIL Framework and Its Relevance .....	4
2.2.2 Understanding Helpdesks and Their Importance.....	5
2.2.3 Service Quality & User Satisfaction .....	6
2.2.4 AI in Helpdesk Systems.....	6
2.3 Artificial Intelligence .....	7
2.3.1 AI Ethics .....	7
2.3.2 AI Technologies and Areas.....	8
2.3.3 Small LLMs .....	9
2.3.4 Increasing Accuracy and Knowledge .....	9
2.3.5 LLM directed Workflows vs Agents .....	11
2.4 Database System .....	12
2.4.1 Relational Database Systems .....	12
2.4.2 NoSQL Database Systems .....	12
2.4.3 Comparing Relational and NoSQL Systems.....	13
2.5 Backend Development .....	13
2.5.1 APIs.....	13
2.5.2 Frameworks.....	14
2.6 Frontend Development.....	14
2.6.1 Base Technologies .....	15
2.6.2 UI and UX Design .....	15

2.6.3 Single-Page vs Multi-Page Applications .....	16
2.7 Conclusion .....	16
3 Methodology .....	18
3.1 Introduction.....	18
3.2 Hybrid Approach .....	18
3.3 Conclusion .....	19
4 Investigation and Analysis .....	20
4.1 Introduction.....	20
4.2 System Investigation (The Problem) .....	20
4.3 System Analysis (The Solution) .....	21
4.4 Conclusion .....	22
5 Design .....	23
5.1 Introduction.....	23
5.2 Project Overview .....	23
5.3 Project Requirements .....	23
5.4 Design Goals and Objectives .....	24
5.5 System Architecture.....	26
5.5.1 AI System Architecture.....	26
5.5.2 Database Architecture.....	26
5.5.3 Backend Architecture.....	26
5.5.4 Frontend Architecture .....	26
5.6 User Interface Design .....	27
5.7 Technical Design .....	29
5.7.1 AI System Design .....	29
5.7.2 Database Design.....	34
5.7.3 Backend Design .....	35
5.7.4 Frontend Design.....	36
5.8 Security Design .....	38
5.9 Testing and Quality Assurance .....	38
5.10 Risk Management .....	39
5.10.1 Hardware Risks.....	39
5.10.2 Software Risks .....	40
5.10.3 Time Risks .....	40
5.10.4 Supervisor Communication Risks.....	40

5.11 Conclusion .....	40
6 Implementation and Testing .....	42
6.1 Introduction.....	42
6.2 Implementation Plan .....	42
6.3 Development Environment .....	42
6.4 Coding Standards and Practices .....	43
6.5 Module and Component Development .....	44
6.5.1 AI System Development .....	44
6.5.2 Database Development .....	48
6.5.3 Backend Development .....	48
6.5.4 Frontend Development.....	52
6.5.5 Configuration Management .....	57
6.6 Integration and System Testing .....	58
6.7 Performance Testing .....	61
6.8 Security Testing .....	64
6.9 User Acceptance Testing (UAT) .....	65
6.10 Bug Tracking and Resolution .....	66
6.11 Conclusion .....	66
7 Evaluation of the Product.....	68
7.1 Introduction.....	68
7.2 System Functionality, Achievement of Goals and Impact.....	68
7.3 Usability and User Satisfaction.....	69
7.4 Performance Evaluation.....	69
7.5 Maintainability and Customisability Evaluation .....	70
7.6 Scalability Evaluation .....	70
7.7 Security Assessment .....	71
7.8 Impact and Cost-effectiveness .....	71
7.9 Development Effectiveness, Efficiency and Sustainability .....	71
7.10 Lessons Learned.....	72
7.11 Limitations of the Evaluation.....	72
7.12 Recommendations for Future Improvements.....	72
7.13 Conclusion .....	73
8 Critical Evaluation/Reflection .....	74
8.1 Introduction.....	74

8.2 Gibbs' Reflective Cycle .....	74
8.2.1 Description .....	74
8.2.2 Feelings .....	74
8.2.3 Evaluation .....	74
8.2.4 Analysis.....	75
8.2.5 Conclusion .....	76
8.2.6 Action Plan.....	76
8.3 Conclusion .....	77
9 Conclusions.....	78
9.1 Summary of Findings.....	78
9.2 Achievement of Objectives.....	78
9.3 Significance of the Project .....	79
9.4 Recommendations for Future Work.....	79
9.5 Closing Remarks .....	80
10 References.....	81
11 Appendices.....	86
11.1 Appendix 1 – Proposal.....	86
11.2 Appendix 2 – Specification.....	88
11.3 Appendix 3 – Project Supervision Logbook.....	90
11.3.1 Logbook 1 .....	90
11.3.2 Logbook 2 .....	91
11.3.3 Logbook 3 .....	92
11.3.4 Logbook 4 .....	93
11.3.5 Logbook 5 .....	94
11.3.6 Logbook 6 .....	95
11.3.7 Logbook 7 .....	96
11.3.8 Logbook 8 .....	97
11.3.9 Logbook 9 .....	98
11.3.10 Logbook 10 .....	99
11.4 Appendix 4 – Gantt Chart of the Timeline .....	100
11.5 Appendix 5 – Raw Data and Code.....	101
11.6 Appendix 6 – Informed Consent Forms for User Acceptance Testing (UAT).....	102
11.6.1 Participant 1 .....	102
11.6.2 Participant 2 .....	103
11.6.3 Participant 3 .....	104

11.6.4 Participant 4 .....	105
11.6.5 Participant 5 .....	106
11.6.6 Participant 6 .....	107
11.6.7 Participant 7 .....	108
11.7 Appendix 7 – Questionnaire for User Acceptance Testing (UAT).....	109
11.7.1 Participant 1 .....	109
11.7.2 Participant 2 .....	111
11.7.3 Participant 3 .....	113
11.7.4 Participant 4 .....	115
11.7.5 Participant 5 .....	117
11.7.6 Participant 6 .....	119
11.7.7 Participant 7 .....	121
11.8 Appendix 8 – Lighthouse Testing Results .....	123
11.8.1 Test results for loading the Technician Portal page.....	123
11.8.2 Test results for loading the Ticket page.....	124
11.8.3 Test results for assigning a ticket.....	125
11.8.4 Test results for closing a ticket .....	126



## List of Figures

Figure 1 Infrastructure Design .....	27
Figure 2 AI-Chat Design.....	28
Figure 3 Technician Portal Design .....	28
Figure 4 Ticket Page Design.....	29
Figure 5 LLM Benchmarks.....	30
Figure 6 AI Workflow Sequence-Diagram.....	33
Figure 7 ER-Diagram.....	35
Figure 8 Code: Generate Troubleshooting Guide .....	45
Figure 9 Code: Troubleshooting Chain.....	45
Figure 10 Code: Troubleshooting Prompt .....	46
Figure 11 Code: File Embedding Insertion.....	47
Figure 12 Code: Watcher Thread.....	47
Figure 13 Code: Azure Authentication .....	49
Figure 14 Code: Role Based Access Control.....	49
Figure 15 Code: User Me Endpoint .....	50
Figure 16 Code: Get Ticket Database Function.....	51
Figure 17 Code: Main React Component .....	52
Figure 18 Code: Automatic Reauthentication.....	53
Figure 19 Code: Generic API Client.....	54
Figure 20 Code: Custom ReactQuery Hook .....	54
Figure 21 Code: ChatStore.....	55
Figure 22 Code: Close Ticket .....	56
Figure 23 Code: Display Error Message.....	56
Figure 24 Code: TicketListContainer .....	56
Figure 25 Code: Configuration File .....	57
Figure 26 Code: Environment Variables .....	58
Figure 27 Code: AI Workflow Unit Test.....	59
Figure 28 Unit Test Results .....	60
Figure 29 Cypress End to End Test .....	60
Figure 30 LangSmith Performance Results .....	61
Figure 31 Ticket Generation Workflow.....	62
Figure 32 Lighthouse Performance Test Results .....	63
Figure 33 ZAP Warnings.....	64

## Glossary

### **API (Application Programming Interface)**

A set of functions and procedures allowing the creation of applications that access the features or data of an application, or other service.

### **Artificial Neural Network (ANN)**

An ANN is a machine learning model loosely modelled after the biological neural networks, organised in layers of interconnected nodes.

### **Azure**

Azure is a cloud computing platform by Microsoft which offers several services, such as virtual machines, databases, domain and user management, and storage.

### **Deep Learning (DL)**

DL is a subfield of machine learning, using neural networks with multiple hidden layers to improve performance.

### **Embedding**

An embedding is a vector representation of data, which captures its semantic meaning. It can be used to measure the similarity of text or documents.

### **FastAPI**

FastAPI is an asynchronous, high-performance Python web framework for building backend APIs.

### **Helpdesk**

A helpdesk is the central point of contact in IT Service Management for reporting and handling of user requests and technical issues.

### **ITIL (Information Technology Infrastructure Library)**

ITIL defines best practices for IT services. It provides guidelines for selection and delivery of IT services within businesses.

### **Large Language Model (LLM)**

An LLM is a deep learning model trained on large text datasets. They can perform complex language tasks, including text generation, translation and question answering.

### **LangChain**

LangChain is a framework that allows to build applications with LLM integration by chaining calls, agents and workflows.

### **LangGraph**

LangGraph is a framework that extends LangChain to manage stateful workflows by structuring them into graphs. It is used to solve complex, multi-step problems.

### **Milvus**

Milvus is an open-source, highly scalable and performant vector database used for similarity search and retrieval.

**Natural Language Processing (NLP)**

NLP is a subfield of AI that specifically focuses on language understanding, interpretation and generation.

**Ollama**

Ollama is a tool for running and managing LLMs locally instead of deploying it in the cloud.

**React**

React is a JavaScript library developed by Facebook (Meta). It is used for creating user-interfaces for single-page applications by dynamically updating content.

**Retrieval Augmented Generation (RAG)**

RAG is a technique where external documents are retrieved and used as context for generation tasks executed by an LLM. This improves accuracy and provides extra knowledge to the LLM.

**Role-Based Access Control (RBAC)**

RBAC is a form of access restriction that only allows users with specific roles or security groups to perform certain tasks or access specific data.

**Single Page Application (SPA)**

A SPA is a type of web application that loads a single HTML page and dynamically updates content without refreshing the entire page.

**Ticket**

A ticket is a digital record in helpdesk systems, used to track the status and history of an issue or user request.

**Vector Database**

A vector database is a specialised database, designed to store, search, and compare vector embeddings, which are a numerical representation of data.

# 1 Introduction

## 1.1 Background and Context

Today's economy is driven by digital technologies, which makes their efficient use essential for remaining competitive in the market. To ensure consistent productivity, IT systems need to run seamlessly and must ensure minimal downtime. As part of IT Service Management (ITSM), IT helpdesks assist organisations to achieve these requirements by managing incidents and providing support to users efficiently. However, as a company's size increases, traditional systems encounter limitations. The sheer volume of tickets leads to repetitive tasks, inefficient ticket handling and poor use of available resources. According to the current trend, these inefficiencies could be addressed by employing Artificial Intelligence (AI). It has become a popular technology for automating routine tasks and enhancing decision-making processes, and already positively impacted various industries. Combining AI with IT helpdesk systems presents a promising opportunity to address their shortcomings and to improve efficiency and effectiveness.

## 1.2 Problem Statement

Traditional IT helpdesk systems encounter several operational issues. The primary limitation is that technicians spend a lot of their time resolving recurring issues that, due to their simplicity, could be resolved automatically. Additionally, users often provide insufficient detail in their tickets, which results in time-consuming follow up inquiries for gathering the information needed to address the issue. Furthermore, handling a lot of simple problems instead of focusing on complex tasks prevents technicians from developing their skills and causes hard or unfamiliar problems to cost even more time and resources.

These issues significantly lower helpdesk efficiency and negatively impact service quality and operational costs. Addressing these shortcomings of traditional systems by leveraging AI for intelligent automation has the potential to substantially enhance helpdesk responsiveness, improve cost-effectiveness, and elevate user satisfaction.

## 1.3 Objectives and Scope

The main objective of this project is to develop an AI-driven IT helpdesk web application that combines a Large Language Model (LLM) and Retrieval Augmented Generation (RAG) in a workflow to automate ticket handling and enhance technician productivity. Decisions in

this AI-directed workflow will be partially made by the LLM itself, presenting a flexible solution that focuses on automated troubleshooting and generation of high-quality tickets.

To ensure the successful development of this application, the following objectives were specified in the project's specification:

- Conduct a literature review on existing helpdesk applications and assess tools and technologies for AI and web integration.
- Build an AI system using RAG and web search to resolve issues or generate support tickets.
- Design a database schema for storing user and ticket data.
- Develop a backend API to manage authentication, database operations, and AI interactions.
- Create a responsive, user-friendly frontend interface.
- Test each component individually and as part of the overall system.
- Document the entire project.

The scope is focused on developing an open-source, customisable, and performant application, with attention to data privacy, user experience, and system maintainability.

## 1.4 Significance and Motivation

The project's significance lies in improving and simplifying IT Service Management by leveraging AI to improve helpdesk efficiency. The aim is to develop a helpdesk application that streamlines service operations by applying intelligent ticket handling. Automated resolution of user issues and the generation of tickets that include detailed information and solution proposals will be leveraged to reduce ticket volume and improve ticket quality. This will result in workload reduction and improved productivity, ultimately leading to lower operational costs. This project provides significant value, particularly to companies that prioritise data sovereignty and privacy. By promoting local AI deployment and providing an open-source codebase, companies can use this project as a template to build customised helpdesk solutions, tailored to their requirements.

The motivation for this project stems from four years of personal experience working in the IT support departments of two companies. Using diverse helpdesk systems at companies of different sizes allowed me to observe the differences and witness the previously discussed

problems. Experiencing these inefficiencies has driven my motivation to create a new, performance-enhanced system.

## 1.5 Overview of the Document

This report provides extensive documentation of the project lifecycle including the conducted research, development, testing, and evaluation of the product.

Following this introduction, Chapter 2 comprises a detailed literature review, discussing relevant research and information about helpdesk systems, technological advances in AI, database systems, and web development.

Chapter 3 outlines the methodology. It discusses the chosen hybrid project management approach, combining Waterfall with Agile techniques for balancing structure and flexibility.

Chapter 4 investigates and analyses traditional helpdesk systems. It points out the flaws and disadvantages of existing systems and introduces AI as a possible solution.

Chapter 5 extensively outlines the application's technical design. This involves creating sub-objectives based on the project's requirements, planning user interface and experience (UI/UX), defining data structures, outlining the AI workflow, and selecting the technology stack for the project.

Chapter 6 discusses the tools and practices used for development, describes implementation and testing phases, and explains code excerpts for key components.

Chapter 7 evaluates the developed system against initial objectives and user expectations, and examines performance metrics, user satisfaction, scalability, and security considerations.

Chapter 8 provides a critical reflection based on Gibbs' reflective cycle. It analyses encountered challenges, lessons learned, and improvements made throughout the project.

Finally, Chapter 9 summarizes the project's findings. It evaluates objective achievement, discusses the project's significance, and offers recommendations for future research and development.

## 2 Literature Review

### 2.1 Introduction

This chapter reviews existing literature to gather information, evaluate the core functionality of helpdesk applications, identify shortcomings of current systems, explore possible improvements, and ensure a fundamental understanding of the technologies used in this project. To enhance clarity and comprehensiveness, this analysis will be split into five subcategories. It will cover Helpdesk Applications, evaluate the utilisation of AI, the usage of databases for data storage, and the development of the application's front and backend.

### 2.2 Helpdesk

In IT Service Management (ITSM), Helpdesk Applications play a key role in managing technical incidents. ITSM is an approach to managing IT structure to ensure well-organised and dependable systems, satisfying the company's criteria. This is important to avoid further consequences as businesses are highly dependent on IT services [1].

This section explores the concept, utility, and evolution of helpdesk systems, as well as the determinants for user satisfaction and the integration of AI technologies.

#### 2.2.1 ITIL Framework and Its Relevance

ITIL is a framework of ITSM best practices, guiding companies in managing and improving their IT services [2], [3].

ITIL is not prescriptive, which means its methods are suggestions based on confirmed and established practices. This makes it flexible and adaptable to different organisations. While it is not a fixed standard itself, companies utilise this framework as a basis to achieve certifications like ISO 20000 compliance [4].

Axelos, the publisher of ITIL, claims that it has led the ITSM industry for 30 years [2].

A recent study [5] confirms that by stating that, even though adoption rate has plateaued, 72% of all businesses utilise ITIL. Another less recent study [6] also acknowledges it as the most relevant framework in previous years.

Due to the high relevance and adoption of this framework, this literature review will examine Helpdesk applications as ITSM components, using ITIL definitions and resources as references.

### 2.2.2 Understanding Helpdesks and Their Importance

A service desk is a single point of contact (SPOC), serving as the primary and only way of communication between the IT department and the users [1]. Users can report incidents, ask questions, and make requests, which are then logged, categorized and assigned to technicians [1], [2].

While ITIL refers to Service Desks as this component can incorporate many different services, this paper will use the term Helpdesk, as the project revolves around incident management only. Nevertheless, these two terms can and will be used interchangeably in this document.

An incident refers to any unexpected disruption or decline in the performance of an IT service. If something has failed, that hasn't yet caused an interruption but could if left unresolved, it is also considered an incident [1], [2], [4].

To reduce negative impact on the business, incident management focuses on the quick resolution of issues to return services to normal [1]. Aligning with the objective of this project, ITIL defines quick and efficient resolution as one of the key success factors in incident management [7].

These incidents are reported by users through the Helpdesk application in the form of a ticket. A ticket serves as a documented record of work. It plays a key role in setting priorities, tracking the progress of tasks, and supporting effective actions. Additionally, the collection of past tickets becomes an excellent resource for generating reports and performing data analysis [7].

Enhanced user satisfaction, faster response times, stronger business reputation, improved employee productivity, and quicker IT issue resolution are some of the key benefits offered by a service desk. For users, it simplifies the process as they do not need to reach out to multiple contacts for support. For IT staff, it allows them to focus on their technical expertise. Organisations risk wasting valuable time and resources trying to address issues and to find assistance without a centralised helpdesk [1]. Neglecting incident management can impact the core business, damage the company's reputation and result in financial losses [1].

The importance of this service, as described earlier, is reflected in its usage. In a cross-national study [6] that investigates the adoption of ITSM processes of companies using ITIL, incident management was the most widely used process, implemented by 94.86% of companies. Furthermore, it is recommended to be the first process implemented, as it delivers quick results and enhances user satisfaction [8].



### 2.2.3 Service Quality & User Satisfaction

Helpdesks improve user satisfaction by offering personalised, interactive support. They save users from searching for scattered information and allow them to describe problems in simple terms. Helpdesks can solve issues not found in manuals, give tailored solutions, and provide a more engaging, social experience than written resources [9].

User experience and satisfaction play a key role in the acceptance and adoption of a system by users. Customer satisfaction refers to how a consumer feels about a service after using it, whether positive or negative [10].

This feeling is directly impacted by service quality. Service quality is a concept perceived by comparing one's expectations with the service actually received. It depends not just on the final results but also on how the service was provided [10].

A study [10] has identified ten key factors that affect the overall customer experience when it comes to service quality, including reliability, responsiveness, competence, and communication. While these are just a few examples, all the identified factors are crucial for maintaining the Helpdesk as the single point of contact.

The book *Service Desk and Incident Manager: Careers in IT Service Management* [4] highlights that dissatisfaction may lead users to go against this concept by avoiding the Helpdesk and contacting IT staff directly. The author claims that, based on his experience, 10% to 20% of incidents are not being processed through the Helpdesk.

Another study [11] conducted a survey, questioning 97 users of various Helpdesk systems, and discovered that the primary cause of dissatisfaction was the slow pace of service delivery, closely followed by a lack of efficiency.

These insights align with this project's objectives, as the development of an AI integration aims to reduce workload, as a simulation study [12] discovered to be the most effective approach in decreasing response time.

### 2.2.4 AI in Helpdesk Systems

Advanced Robotic Process Automation (RPA) utilising AI helps businesses to improve efficiency by automating tasks, reducing costs, and minimising mistakes [7].

Previous studies [13], [14], [15] on AI integrated Helpdesk applications, primarily revolve around ticket classification, as this is proven to increase efficiency and therefore user satisfaction.

Another less recent paper [16] proposed a system that generates possible solutions to

incidents, based on similar, previously solved cases, that are clustered using a mixture language model. A reference solution is then created for the customer, which has proven to be effective and aligns with the broader goals of the current project.

Modern Helpdesks will be examined and compared to the scope of this project in Chapter 4, *Investigation and Analysis*.

## 2.3 Artificial Intelligence

Artificial Intelligence (AI) has been around for many years, dating back to the 1950s [17]. It is a branch of Computer Science studying and exploring ways of enabling computer systems to have human-like intelligence. AI applies various techniques to achieve classification, pattern-, image- and face-recognition, language understanding and mathematical reasoning, empowering machines to perform tasks that previously required human intervention [17], [18].

Due to advancements in computer hardware, algorithms, and the availability of large labelled training datasets, AI has made major progress in recent years [17], [19]. As a result, AI's popularity has grown significantly and is becoming an essential part of modern society, integrating into daily life, ranging from chatbots to large-scale business tools [19].

This section examines various fields and subfields of AI, highlights the importance of ethics, explores the rise of Large Language Models (LLMs), compares techniques to enhance their accuracy and knowledge, and investigates the best strategies for solving complex, multi-step tasks.

### 2.3.1 AI Ethics

While AI has gained significance in various sectors, benefiting economics, medical advances, and social development, it is not free of risks. There are several ethical and moral implications that have to be taken into account [20].

Since AI is building its knowledge on human-created data, it is not neutral, as it can inherit negative traits, like discrimination based on gender or race bias of the provided dataset [17], [20]. Additionally, AI services gather a huge amount of information from their users for training or continual learning. For personal or business data, misuse can cause security and data privacy issues for organisations as well as individuals [20]. Another potential issue is job replacement due to AI automation. While it is not evident whether AI will increase the unemployment rate there is a chance of AI disrupting the labour market, causing job losses in certain industries [20]. Malicious use of AI is another rising threat. Voice and video

impersonation tools are used for scams and fraud, creating serious risks for people and society. Additionally, AI could be abused, to spread false information, deploy uncontrollable AI agents, or even develop potentially new deadly chemical and biological weapons [21]. To leverage AI's advantages despite its risks, the UK government is currently developing a regulatory framework designed to encourage innovation while mitigating AI-related threats [22].

### 2.3.2 AI Technologies and Areas

The field of AI includes many different overlapping areas such as Machine Learning (ML) or Natural Language Processing (NLP). To clarify and distinguish those areas, common AI-terms will be discussed.

Machine Learning is a technique that enables systems to make predictions without explicitly programming the system for the specific task. The system learns from data and improves their performance over time by learning from past computations and finding patterns [17], [23].

There are three main types of machine learning, based on the problem and data: supervised learning, unsupervised learning, and reinforcement learning [23].

Artificial Neural Networks (ANNs) are inspired by the human brain and nervous system. They mimic biological neural networks in a simplified manner [24]. Consisting of interconnected units called neurodes or perceptrons, each unit processes input data and passes the result to others [17], [24].

A more powerful subfield of ML is Deep Learning (DL) [17], [25]. While a normal Neural Network has an input, an output, and a single hidden layer, Deep Neural Networks (DNNs) used for DL contain multiple hidden layers [25]. Deep learning is great at working with complex data to build advanced models, which explains why it is widely used across various fields [17].

Another direct subcategory of AI is Natural Language Processing (NLP) which enables machines to understand and generate written text in order to fulfil tasks. This is done by applying different Machine Learning and Deep Learning Algorithms [17].

NLP consists of two primary parts, Natural Language Understanding (NLU) and Natural Language Generation (NLG). NLU allows machines to interpret human language and break it down by identifying key elements like ideas, entities, feelings, and important terms. NLG is the generation of meaningful text following a set of defined steps to achieve that [26].

In recent times the field of NLP is dominated by Large Language Models (LLMs), a form of

Deep Learning [27]. LLMs are enhanced Language Models that excel at learning due to their vast number of parameters. They are designed to comprehend written text, and to generate text from given prompts based on predictions [28]. However, LLMs are capable of much more than just holding conversations. These modern models demonstrate exceptional reasoning, planning, decision-making, and in-context learning capabilities, resulting in their widespread use across various environments [27].

### 2.3.3 Small LLMs

Since an LLM's capabilities and accuracy are linked to its size the industry is developing bigger and bigger models with hundreds of billions of parameters. This introduces significant overhead, leading to costs that rise exponentially due to hardware requirements and power consumption of these systems [29].

Small LMs or SLMs are not fixed in size but are instead relatively smaller than the largest current models. Smaller models are underestimated due to the popularity of LLMs. However, they offer significant advantages for research or businesses with limited resources [29].

First of all, SLMs have a faster inference speed, making them generally more responsive and preferable for real-time tasks that require low latency. Moreover, basic models are generic and need fine-tuning to adapt to specific domains. For LLMs this is a very resource intensive procedure and not always practical. Fine-tuned SLMs for specific tasks may perform better than generic LLMs in certain domains [29]. Additionally, when up-to-date data is needed and to minimise hallucinations, Retrieval Augmented Generation (RAG) can be used to provide the model with context from documents retrieved from a database [27], [29], [30].

With these adjustments, smaller models can compete with LLMs in specific scenarios, reducing costs and latency [27], [29].

While both have their strengths, the key is balancing power, efficiency, and costs to meet the individual requirements [29].

### 2.3.4 Increasing Accuracy and Knowledge

There are various techniques to enhance a Language Model's performance, each serving a different purpose and offering unique advantages and disadvantages.

Full Retraining yields perfect results, as the training data can be amended and expanded with up-to-date information, resulting in enhanced accuracy and updated knowledge of the model. However, the computational power and time needed are immensely expensive [31].

In contrast, Continual Learning, also known as Incremental Learning, only updates the model with the main goal of improving language usage and reasoning, but can also be used to add new knowledge [32]. An important concern with this approach, is the occurrence of catastrophic forgetting. This is a phenomenon where the model loses prior capabilities or knowledge as it incorporates new data [32], [33].

As a subset of Continual Learning, Fine Tuning uses new data to amend the model [31], adjusting the model weights using data of one specific domain, enhancing the model's knowledge or skills in this area [34], [35]. For ongoing improvements and up-to-date knowledge this procedure has to be repeated frequently, resulting in high costs of hardware and energy for computationally expensive retraining [30]. Furthermore, there are contradictory findings regarding the effects of fine tuning on hallucinations, which are an LLM's not data based incorrect assumptions. While it is believed that fine tuning reduces hallucinations [30] another study [34] discovered an increasing amount of hallucinations after tuning the GPT 3.5 Turbo model.

Retrieval Augmented Generation (RAG) is a completely different approach. It utilises external data, stored in a Vector Database, to provide the LLM with additional knowledge that is likely outside its existing knowledge [30], [36], [37]. This is especially beneficial for information dependent tasks relying on the latest data, as we still face challenges in direct knowledge manipulation of LLMs [30], [38]. While this widely adopted concept provides knowledge and minimises hallucinations, it increases latency by introducing additional computational overhead during generation [30].

Documents of various types can be formatted into plain text, split into chunks and stored in a Vector Database in the form of Embeddings, which are numerical vector representations of the text [30], [36]. The retrieval process is conducted by converting a user query into an embedding. Using this vector representation of the query, the relevant text chunks are determined by calculating the similarity. The best matches are selected and passed to the LLM, providing context for the generation [30], [39].

Since simple RAG systems may obtain irrelevant or unrelated data, providing too little context or false information, there are measures to mitigate these issues [30]. The Advanced RAG System [40] evaluates the relevance of each retrieved document for the queried purpose by employing an LLM for decision making. If any documents seem irrelevant, further data can be retrieved from the database, or alternatively, a web search is performed to gather additional information using a web scraper. Additionally, the final generated answer could be checked for hallucinations by comparing the answer with the initial question. This series of

tasks can be realised using a graph, chaining multiple actions and LLM calls together. The Advanced RAG increases retrieval accuracy and relevance of documents, boosting the system's performance [40].

In conclusion, RAG stands out as the most efficient solution in improving a model's accuracy and knowledge [30]. A study [34] comparing GPT 3.5 Turbo variations, confirms that, while the fine-tuned model outperformed the base model, the RAG system yielded significantly better results. Nevertheless, the suitable measure depends on the requirements, as language or behaviour cannot be amended using RAG. In some cases a combination of several techniques is key for peak performance [30].

### 2.3.5 LLM directed Workflows vs Agents

There are various approaches to solving complex multi-step tasks using an LLM.

AI Agents can be employed to solve such issues. An agent is a LLM-based unit, coordinating LLM and tool calls based on its own decisions [39], [41]. These decisions are based on its perception and initial state, which depend on its defined role within the environment, making the agent an autonomous worker for specific tasks [39]. The issue with this approach is that LLMs, despite recent advancements, are still unpredictable due to their blackbox nature [23]. Therefore, logical issues in their decision making, especially for smaller models, cannot be fully prevented.

A different approach is ReAct, which is the procedure of repeatedly calling an LLM for each sub task of a multi-step problem, using the same prompt but adding information on already completed actions. This approach also requires the LLM to understand the progress made and choose the next subtask based on already completed actions, which is still thought to be unrealistic for it to work reliably [42].

A better solution is to utilise a human-designed workflow. This hands back some control over the process, increasing accuracy and transparency and debuggability [42], [43]. The workflow is defined independent of the LLMs by creating a graph structure describing the processes states, possible transitions between states and a defined start and end point [42]. Within the workflow, chains can be utilised to combine several actions for one subtask, where one action is dependent on the output of the previous [39], [43], like passing a prompt to an LLM and its output to a parser function. Several papers highlight LangChain in combination with LangGraph as the most popular framework for creating and managing state based workflows [39], [40].

## 2.4 Database System

A Database System (DBS) is a construct consisting of a Database (DB), which is an extensive collection of data, and a Database Management System (DBMS) for its organisation, storage and retrieval. This controlled form of data-storage provides several advantages, allowing simultaneous manipulation of multiple records, performing calculations, improving security, integrity, concurrency and recovery capabilities [44], [45].

There are various types of Database Systems, broadly split into two main categories: Relation Databases and NoSQL Databases [44], [46], [47].

### 2.4.1 Relational Database Systems

The Relational Database, introduced in 1970, is the most mature and remains the most prevalent system on the market [44], [46]. The data is stored in tables, with each row representing one entry and the columns representing the defined attributes of each dataset. Correlated data is linked by connecting tables through relations [44]. Using Structured Query Language (SQL) is the common way of interacting with these systems, allowing for querying and manipulating the data efficiently [44], [45].

By ensuring atomicity, consistency, isolation and durability, also known as the ACID properties, relational systems provide a reliable platform, safeguarding against potential issues like data inconsistencies due to partly executed transactions, simultaneous manipulations or system crashes [44].

However, limited scalability, increased complexity for unstructured data, and difficulties in handling vast amounts of data are notable limitations of this concept [44].

### 2.4.2 NoSQL Database Systems

NoSQL, or Non-Relational Database Systems are various other types of databases that do not use tables for data organisation [44]. These can be categorised into Key-Value databases, Document databases, Column Family stores and Graph databases [46], [47]. Especially MongoDB, being a document database that stores data in document formats such as XML, JSON or BSON [44], [46], has gained popularity and is now established as one of the most used systems on the market [46].

The major advantages of NoSQL systems are their exceptional performance when managing vast amounts of data [46], horizontal scalability, allowing it to be distributed across inexpensive hardware [44], the ability to handle unstructured data [46], [47], and their

freedom from fixed schemas [46].

On the other hand, these systems generally offer lower consistency as they often renounce the ACID properties for performance gains.

#### 2.4.3 Comparing Relational and NoSQL Systems

Performance-wise, studies [46], [47] highlight overall increase in throughput for CRUD operations when comparing MongoDB to MSSQL and MySQL. For reading data, some contradictory findings exist, as [46] determined that the relational database is faster.

However, this can likely be assumed to be the normal behaviour, as a different source [44] confirms that NoSQL databases are typically used for write-heavy systems with fewer read operations.

The differing performance of a few milliseconds in execution speed can be neglected for smaller systems, performing operations on fewer than 1000 records at a time, but for 50,000 updated records, MongoDB was measured to be 29 seconds faster than MSSQL [46].

Comparing both systems' advantages and disadvantages, it can be observed that NoSQL addresses some of Relational Database's shortcomings in terms of performance, scalability, and flexibility but may sacrifice consistency and reliability. Therefore, there is no definitive better or worse, instead the decision on which technology to use is highly dependent on the requirements. While NoSQL is the choice for unstructured data and has an edge in performance, especially for big data, relational databases are fast enough for smaller systems and provide higher integrity and reliability.

## 2.5 Backend Development

Web Applications consist of two separate parts. The front-end, running on the client, which serves for user interaction [48], [49], while the back-end on the server manages the application logic and interaction with data [50].

In a functional web application, these two components communicate with each other via Application Programming Interfaces (APIs) [51] using the TCP/IP Stack and the Hypertext Transfer Protocol (HTTP) [50].

### 2.5.1 APIs

Application Programming Interfaces (APIs) allow for communication between applications [51], [52]. The two most popular approaches are Simple Object Access Protocol (SOAP) and



Representational State Transfer (REST) [52].

SOAP is a messaging protocol that uses XML for communication [51].

It is more complex but also more reliable and secure than REST, making it a suitable choice for critical systems that have to be very robust, like banking or financial applications [51].

REST is far more flexible and lightweight. It supports not only XML, but also JSON, plain text and binary files like images or videos [51]. A REST API provides endpoints that can be called by the front-end client to access data or functionality [53]. It is a stateless protocol, meaning it does not remember sessions, instead the client has to provide the information about what data it needs by sending additional attributes to the API [51], [52]. This stateless approach promotes scalability [51], [52], [53] while also ensuring communication visibility and reliability [51], [53].

Additionally, REST is faster, uses less memory, is easier to implement and update, and can work with multiple file formats [51]. These qualities make it the preferred choice over SOAP for most web applications that don't need extreme levels of robustness and security.

### 2.5.2 Frameworks

Frameworks are structured platforms consisting of libraries and classes designed to support software development [48]. They are used for improving quality and standardisation of web applications [49]. Several criteria should be considered when choosing a framework suitable for the project's requirements. Ease of use and technical attributes like performance and compatibility are the most important factors. Popularity and community engagement are also very important, as they make it easier to find information or receive help online.

Additionally, criteria like size, support, licensing, documentation, and resource availability must be considered [49].

The most popular frameworks on the market are Node.js/Express, Django, Ruby On Rails, Laravel and Spring [48], [49]. Besides the mentioned criteria, these frameworks are built on a variety of different programming languages, which also must be considered when making a choice.

## 2.6 Frontend Development

The front-end is the part of an application the user directly interacts with. It runs on the client side, displaying information and interactive elements in the form of text, buttons, images, graphs, and tables. Important attributes are responsiveness, i.e., the ability to adjust to

different screen sizes, and performance [54]. The front-end is particularly important, as its features, look and feel are crucial for attracting users [55].

### 2.6.1 Base Technologies

The basic front-end is a combination of three technologies. Hypertext Markup Language (HTML) is used to provide the content and structure of the web page. For design capabilities, Cascading Style Sheets (CSS) are utilised to alter the look of the HTML elements. JavaScript (JS) adds logic and enables dynamic manipulation of the web page's content and style in response to user interactions [56], [57]. Despite its high popularity, JavaScript has its flaws. Since it lacks static typing, which means it does not check variable types on compilation, it has a reputation for leading to complications on large-scale projects due to poor software quality [58], [59]. TypeScript (TS) was developed to address this issue. Built on top of JavaScript, it is an extension, adding a module system, classes, interfaces, and a static type system [59].

While it is generally acknowledged and supported by research [58], [60] that TypeScript provides benefits for development, and enhancement of code quality and understandability, there are conflicting studies as to whether its static typing actually reduces the amount of bugs in a project. A study [58] analysed 604 GitHub projects and found the opposite: TypeScript projects had 60% more commits for bug fixes compared to regular JavaScript projects. Moreover, the mean time required to fix these bugs was approximately one day higher when using TypeScript (31.86 vs. 33.04 days).

A different study [60], on the other hand, evaluated public projects on GitHub and claims that TypeScript could have prevented the occurrence of 15% of the bugs that later needed fixing. Reasons for discrepancies in these studies could be that TypeScript is used on more complex applications, making the application more prone to bugs due to its complexity itself, or because TypeScript developers are more likely to document bugs thoroughly [58].

### 2.6.2 UI and UX Design

A well planned and detailed implementation of UI and UX design is essential for user satisfaction and loyalty. To provide excellent user experience, the product's look, feel, functionality and ease of use are very important [61].

The UI (User Interface) is the visual representation of the application. It is about designing the structure and look of the application to make it aesthetically pleasing and easy to use.

The UX (User Experience), on the other hand, ensures a positive user experience by

providing logical, intuitive navigation, a clear content strategy, and overall ease of use [61]. For guidance about creating UI/UX designs, this paper [61] provides an overview of concepts. It highlights five key principles for each, UI and UX, which can be used as a practical foundation for creating user-centered designs.

### 2.6.3 Single-Page vs Multi-Page Applications

Multi-Page Applications (MPAs) are traditional web applications that reload when navigating to a subpage or when content changes. Single-Page Applications (SPAs) on the other hand avoid reloading to reduce wait times and therefore improve the user experience [56], [62], [63]. While SPAs are more common in modern web development, they have some drawbacks like longer initial loading times and disadvantages in search engine optimisation (SEO) [63]. SPAs are built using front-end frameworks or libraries [56], the most popular ones being React, Angular and Vue [62]. Choosing the right framework has major importance, as it significantly impacts the application's scalability, maintainability, modularity, and interactivity [55].

## 2.7 Conclusion

This literature review highlights the importance of helpdesk applications in IT Service Management. Following frameworks like ITIL, these systems help businesses maintain smooth operations by documenting and resolving issues and improving user satisfaction.

AI has become a powerful tool in modern helpdesk systems. Automating repetitive tasks with robotic process automation (RPA) makes processes faster and more efficient. While it is currently primarily used for ticket classification, Large Language Models can also be used for ticket resolution and personalised conversational support. These AI features improve user experience and give IT teams more time to focus on complex problems.

Databases are necessary for managing helpdesk data. Relational databases are reliable and consistent. NoSQL databases are highly scalable and are used for large and unstructured data. Choosing the right database depends on the project's needs.

For developing web applications, extensions like TypeScript and frameworks for front- and back-end development help to create user-friendly, interactive platforms. Good UI/UX design ensures easy navigation and an aesthetic look, which is a necessity to ensure the user's adoption of the system.

Overall, this literature review supports the development of an AI-Integrated web application for this bachelor project. It provides a strong understanding of current technologies, the needs that exist in helpdesk systems, and how AI can be leveraged to improve helpdesk performance. These insights will guide the creation of a helpdesk that is more effective, improves user satisfaction, reduces the workload of the technicians, and therefore leads to cost-savings for companies.

## 3 Methodology

### 3.1 Introduction

Choosing the right methodology is crucial for streamlined development, ensuring an efficient workflow and minimising risks. This chapter discusses the methodology used for this project's development. It outlines a structured approach by utilising well-known project management strategies for efficient planning, implementation, and testing. The chosen approach must ensure functional and technical requirements are met within the deadline.

### 3.2 Hybrid Approach

To guarantee efficiency, quality, and timely delivery, the project requires a structured but flexible approach. Since the project is fully preplanned and objectives are clearly defined and unlikely to change, a sequential approach like Waterfall seems favourable [64]. Waterfall structures the development into clearly defined phases which enable predictable time planning [65]. Furthermore, it allows one to manage complexity and set expectations upfront. By focusing on each phase at a time and completing each section before starting the next, it ensures to maintain scope and stay on schedule by tracking progress against the predefined timeline.

However, Waterfall also has its limitations when it comes to software development. A major disadvantage is that testing occurs after the development phase, which can lead to delayed issue detection. Issues could be overlooked during development, leading to unexpected delays if major issues need debugging or rewriting of code in the testing phase [64], [65]. This occurrence is unacceptable when working within strict deadlines. Furthermore, misjudged time estimates or unforeseen circumstances in the development process endanger the timeline when using Waterfall.

To mitigate these risks, this project adopts a hybrid approach, combining Waterfall with Agile methodologies. While the structure remains the same, the principles of continuous iterative testing and adaptive planning are incorporated to improve flexibility. Instead of testing after the implementation is finished, each component will be tested continuously during and after development as it is practiced in Agile methodologies such as Extreme Programming and SCRUM [66], [67]. This ensures that bugs and performance issues are detected early, reducing the likelihood of major setbacks later.

Additionally, prioritisation will be leveraged as a risk management strategy, inspired by SCRUM's priority management in the product backlog [66]. This makes it possible to push back the implementation of less important features and shift focus on the critical components but will only be used if the deadline is at risk due to unforeseen circumstances.

Based on this approach, a Gantt Chart was defined, which clearly outlines all tasks and estimates the time required for each. It features sufficient time allocated for testing to ensure application quality, and one additional week for improvements and fixes, which also serves as a buffer in case any issues arise. This Gantt chart is appended to this document.

### 3.3 Conclusion

The discussed hybrid approach provides a clear structure, proactive risk management, and improved efficiency. Combining Waterfall's strict sequential phases with Agile's adaptability ensures well-organised and goal-focused development while remaining adaptable to issues and challenges. Following this approach helps meet deadlines and ultimately leads to a more robust and reliable final product.

## 4 Investigation and Analysis

### 4.1 Introduction

This chapter provides a comprehensive investigation and analysis of current helpdesk systems and evaluates the need, benefits, and feasibility of a new or enhanced system to address existing challenges and problems by integrating Artificial Intelligence (AI). The objective is to identify the main problems, derive potential solutions, and check their viability to justify the development and implementation of this AI-driven helpdesk system.

### 4.2 System Investigation (The Problem)

The problem identification was based on personal experience of working with different helpdesk systems in two different organisations over a period of four years. Three substantial issues were discovered and will be discussed in the following.

1. The core problem relates to the vast amount of workload that is generated by simple repetitive or recurring tickets. Technicians spend a lot of their time handling similar tickets, consuming a lot of time that could otherwise be devoted to more important and complex problems.
2. Moreover, tickets are often poorly written and do not provide a lot of information for the technician to work with. This results in time-intensive enquiries to gather more details from the user, which often introduces big delays in the resolution process.
3. Lastly, technicians are not always familiar with the more complex problems and often need a lot of time to eventually solve these issues. Manual research and understanding the problem take time and could be cut short by automatically offering solutions in the ticket or referencing similar, previously solved tickets.

Introducing AI has the potential to address these issues by significantly reducing repetitive workload, generating detailed tickets, proactively generating solutions and linking to previously solved tickets to guide technicians effectively.

Such an AI-enhanced system offers considerable opportunities for organisations. Primarily, it will introduce massive cost savings by reducing the overall workload while also increasing technician efficiency. By automating these repetitive tasks, the technicians can allocate more of their time to complex issues that require deeper analysis and innovation. This not only leads to faster resolution times of such issues but also creates a more engaging and less

routine-dependent work environment that will provide greater employee satisfaction and fulfilment.

Despite these advantages, leveraging AI introduces risks. The primary concern is potential job reductions since the automation and increased technician efficiency will ultimately lead to decreased demand for support roles. Therefore, staff retraining, redeployment and role expansion must be taken into consideration when employing such an AI-driven system.

The feasibility of integrating AI to enhance helpdesk operations is evident, supported by the recent advancements in AI and the availability of tools and frameworks, as discussed in the Literature Review. The technology required to implement such a system is mature and open-source solutions make it widely accessible.

### 4.3 System Analysis (The Solution)

The requirement for solving the identified problems is integrating an AI system which is capable of automated ticket resolution, intelligent ticket generation, and solution recommendation by providing solution guidance and previously resolved similar tickets. These demands will be broken down further into functional and non-functional requirements and translated into specific goals and objectives in Chapter 5, *Design*.

When analysing existing systems, such as basic helpdesks and commercial platforms like Zendesk that already employ AI-driven functionalities, it becomes evident that these applications typically utilise cloud-based solutions. Using AI cloud services, however, raises concerns related to data privacy and security. To avoid this issue and to create a decommercialised product, the proposed system will leverage open-source and locally hosted AI models to ensure enhanced privacy and security. This approach benefits organisations that prioritise data sovereignty and privacy and enables this project to contribute to broader research and improvements in AI applications by proposing a publicly accessible solution that can be used as a basis to improve upon, used to create custom solutions, or to come up with new designs.

The design of the proposed solution will build upon previous research [68], which demonstrated the efficacy of a simpler approach where e-mail requests were automatically answered based on predefined FAQs using a matching engine. A similar approach will be taken by applying a Large Language Model instead of the matching engine, and substituting



the FAQ with the model's knowledge, retrieval augmented generation (RAG), and web search. While the e-mail system already showed promising results in its limited scope, Large Language Models (LLMs) and workflow automation offer far superior capabilities for a fully functional helpdesk. Adapting the study's approach into a modern, real-time helpdesk system by employing an LLM promises significantly enhanced performance, flexibility, and user satisfaction.

#### 4.4 Conclusion

This chapter conducted a detailed investigation and analysis of the existing helpdesk system and identified its issues with repetitive workload and inefficiencies. Furthermore, it highlighted the opportunities and potential risks of integrating AI to solve the discovered issues and confirmed the solution's feasibility and the advantages of developing an open-source, locally hosted application.

This investigation underscored the potential for substantial improvements in helpdesk efficiency to improve cost-effectiveness and formulated a rough idea of the system's design. The next chapter will go in-depth, break down the requirements and ultimately forge a complete and comprehensive design, ready for implementation.

## 5 Design

### 5.1 Introduction

This chapter explains the technical design of the AI-Integrated IT Helpdesk. Its focus lies on planning the web application's structure and use of technologies. It discusses the main design decisions, including the system's architecture, user interface, technical design, data structure, and security features. Based on the facts that were discussed in the literature review, this chapter evaluates which technologies and techniques will be used for developing a user-friendly, secure, and efficient application.

### 5.2 Project Overview

The project's objective is to develop an IT Helpdesk Web Application capable of directly resolving user inquiries or generating detailed tickets using a pretrained Large Language Model. The project focuses on providing a fully functional web application, offering a good user experience with an appealing and intuitive UI. Users will be able to access the AI and their tickets, while technicians can filter, assign, handle and close tickets.

This AI-integrated IT Helpdesk solution's purpose is to reduce workload and costs for companies by reducing ticket volume and improving ticket quality. A significant portion of tickets relate to user errors instead of actual technological issues, which can often be resolved by a Large Language Model guiding users through the troubleshooting process. If the issue cannot be solved due to the LLM's limitations or due to factors like required privileges, the AI can create a ticket for the user, assuring high ticket quality. Many users often forget to mention important details when submitting tickets, which costs technicians a lot of time in follow-up inquiries. An automatically generated ticket will resolve this issue by using a predefined structure that includes all necessary information. Additionally, the ticket can be complemented with extra context and suggested solutions to assist the technician.

### 5.3 Project Requirements

There are four main stakeholders that have different needs and expectations:

- Users demand a visually appealing, easy-to-use application which offers efficiency and accuracy in solving tickets whether through AI or technicians.
- Technicians' expectations include reduced workload resulting from automated inquiry resolutions and high ticket-quality, which reduces follow-up efforts.

- Administrators expect a customisable, performant, and secure system that can be easily deployed and maintained.
- Companies' management are mostly concerned with overall improved helpdesk efficiency to reduce operational costs.

To fulfil these expectations, the project must meet several functional requirements essential for its successful completion:

1. The system must improve helpdesk efficiency by leveraging AI.
2. Secure registration and login functionality for users and technicians is needed.
3. The system requires role-based interfaces for users and technicians, allowing them to access role-specific features, such as managing tickets and interacting with the AI.
4. Secure storage, management, and retrieval of user credentials and ticket data must be provided.

Furthermore, there are several non-functional requirements to assure smooth operation and maintenance of this application:

5. Positive user satisfaction driven by great performance and UX/UI design is needed to ensure adoption of the system.
6. Security is essential when building a system like this, that handles a lot of personal and business data.
7. Maintainability and customisability of an open-source project are crucial as companies might want to adjust the system to their needs.

## 5.4 Design Goals and Objectives

In order to successfully complete this project, specific design goals were set to meet the discussed requirements:

1. Enhance helpdesk efficiency by leveraging AI to streamline the ticketing process.
  - Objectives:
    - Leverage a Large Language Model to solve inquiries or generate high-quality tickets.
    - Extend the model's knowledge utilising Retrieval Augmented Generation.
2. Provide registration and login functionality to ensure authentication.

- Objectives:
  - Implement backend authentication and authorisation workflows.
  - Develop the frontend for the registration and login process.
- 3. Design intuitive and responsive interfaces that allow users and technicians to perform their respective tasks.
  - Objectives:
    - Design an intuitive and responsive user dashboard that enables users to access the AI and to view and interact with submitted tickets.
    - Develop a technician portal that allows technicians to filter tickets by status and assignee. It provides tools for assigning, handling, and closing inquiries and access to archived tickets.
- 4. Ensure secure and efficient data storage and retrieval.
  - Objectives:
    - Leverage a database to securely store user credentials and ticket details.
    - Use encryption to store passwords.
- 5. Deliver a performant and UI/UX-focused system to ensure user adoption.
  - Objectives:
    - Optimise context changes and eliminate page refreshes by implementing a Single Page Application (SPA) architecture to minimise latency and ensure positive user experience.
    - Build an intuitive and visually appealing UI.
    - Use caching mechanisms for frequently accessed data to reduce latency.
    - Optimise AI response times without neglecting accuracy for performance to provide a well-balanced system.
- 6. Protect the application from cyber-attacks by implementing security measures.
  - Objectives:
    - Restrict resources and functionality using role-based access control.
    - Enforce strong password policy.
- 7. Ensure the system is easy to maintain and customise for different needs.
  - Objectives:

- Write clean, modular, and commented code and perform frequent code reviews to maintain quality.
- Allow the selection of the AI model without code modifications by configuring environment variables.

## 5.5 System Architecture

The system architecture is made up of four main parts that work together as a complete system:

### 5.5.1 AI System Architecture

Integrated with a pretrained Large Language Model, this module handles communication with the users, trying to resolve the issue or otherwise generating a detailed ticket. Included as a subsystem is a vector database which stores vector representations of documentation and business documents that serve the LLM as additional knowledge.

### 5.5.2 Database Architecture

A database is used to securely and efficiently store and retrieve ticket and user data.

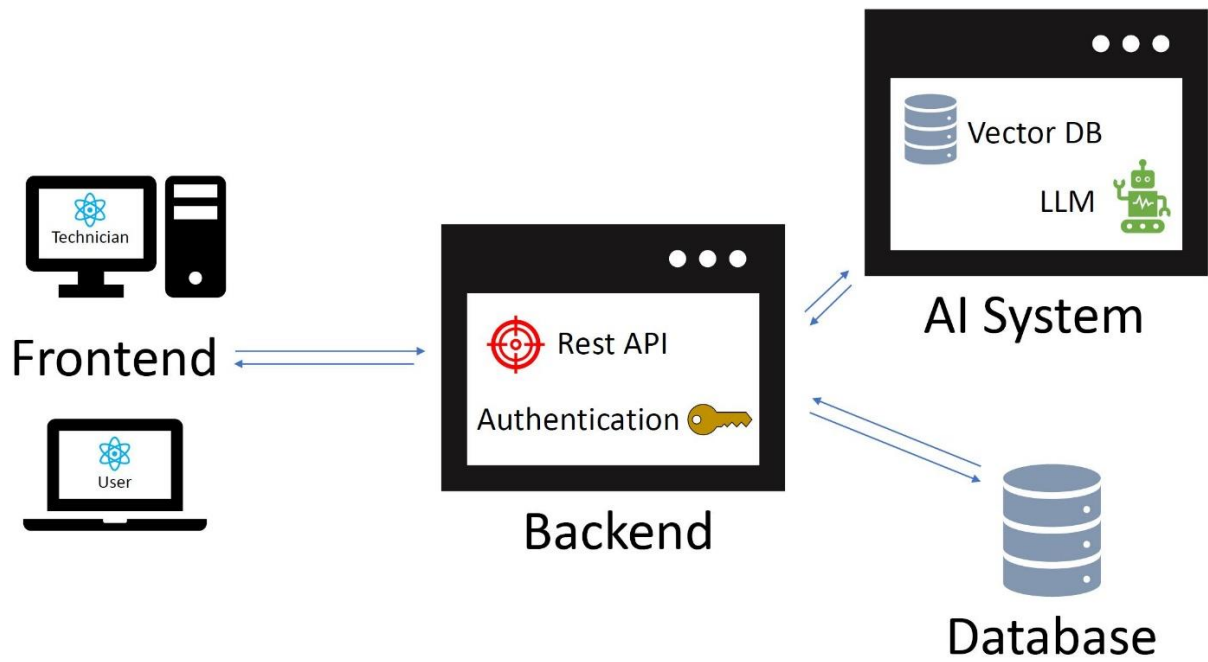
### 5.5.3 Backend Architecture

The backend handles the business logic, request processing, and interaction with the database. It provides API endpoints for user authentication, AI calls, and ticket operations. The app will integrate with Azure for authentication since most companies already use Azure to manage their users and groups. This removes the need to create separate accounts for the app.

### 5.5.4 Frontend Architecture

The frontend offers the User Interface (UI) for users and technicians. It directly communicates with the backend, calling the APIs to use the application's functionalities.

These components are represented in this system architecture diagram:



*Figure 1 Infrastructure Design*

It visually presents the high-level structure of the system, showing how the main components interact and communicate with each other.

## 5.6 User Interface Design

To ensure a well-designed and intuitive frontend, its UI was initially created using Canvas. This tool allows to easily create website design mock-ups which can then be used as a basis when implementing the frontend. Following designs were created:

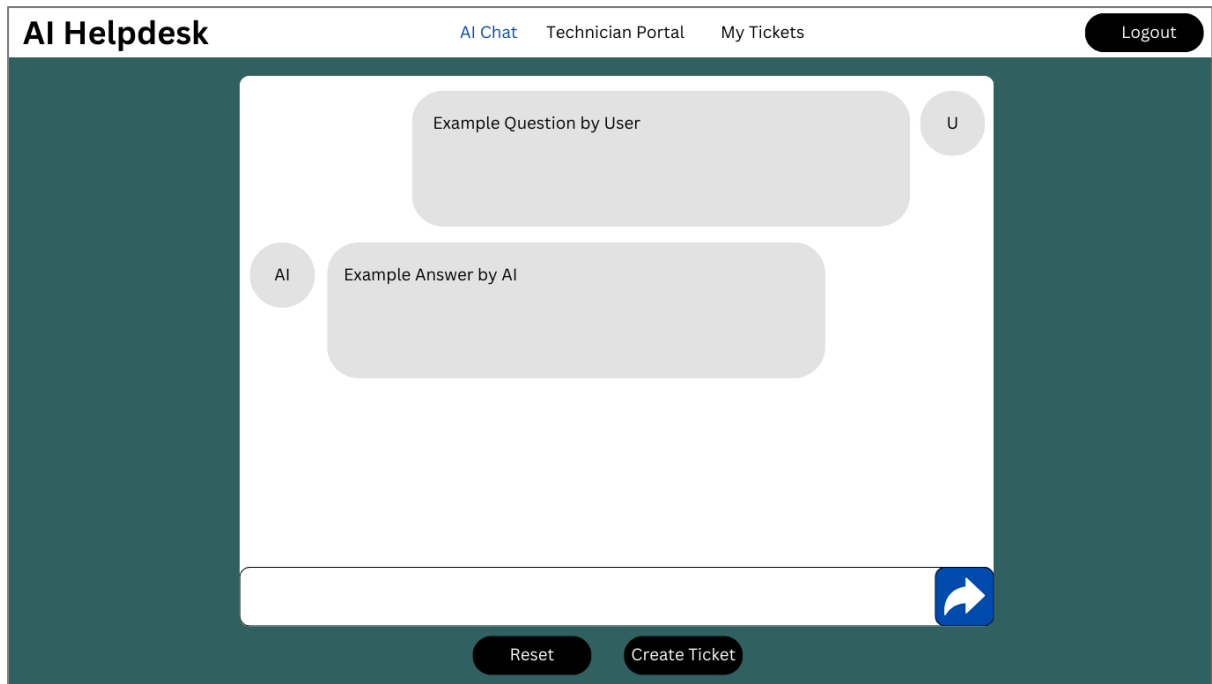


Figure 2 AI-Chat Design

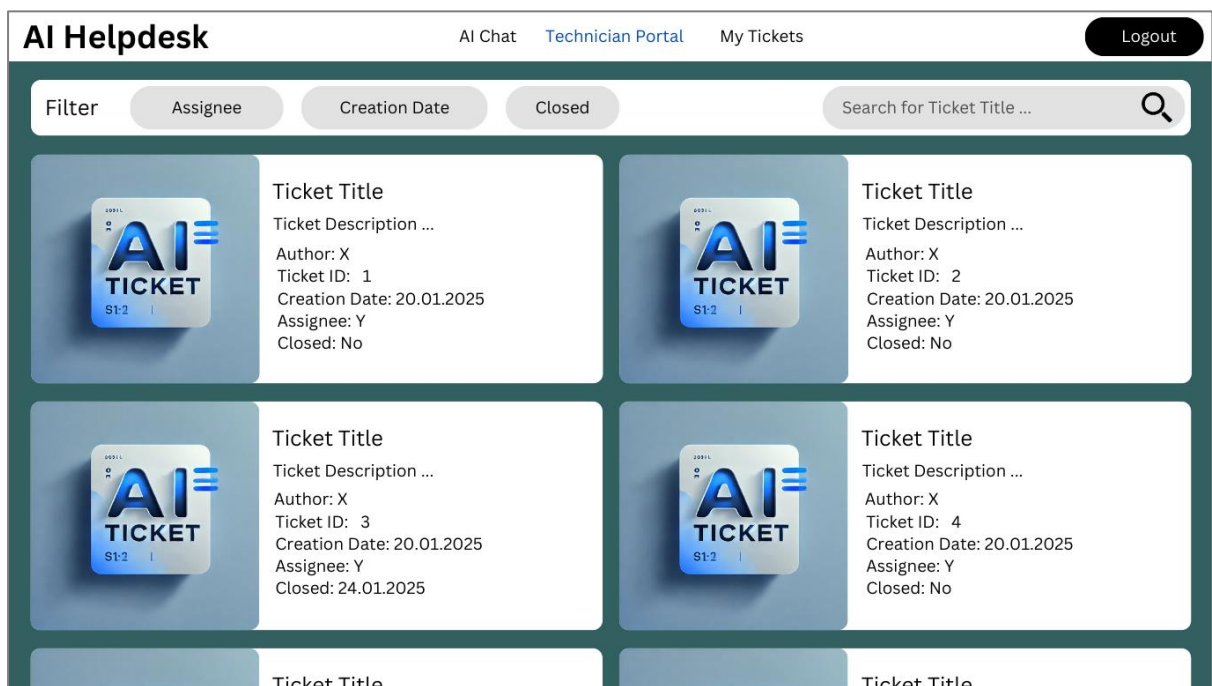


Figure 3 Technician Portal Design

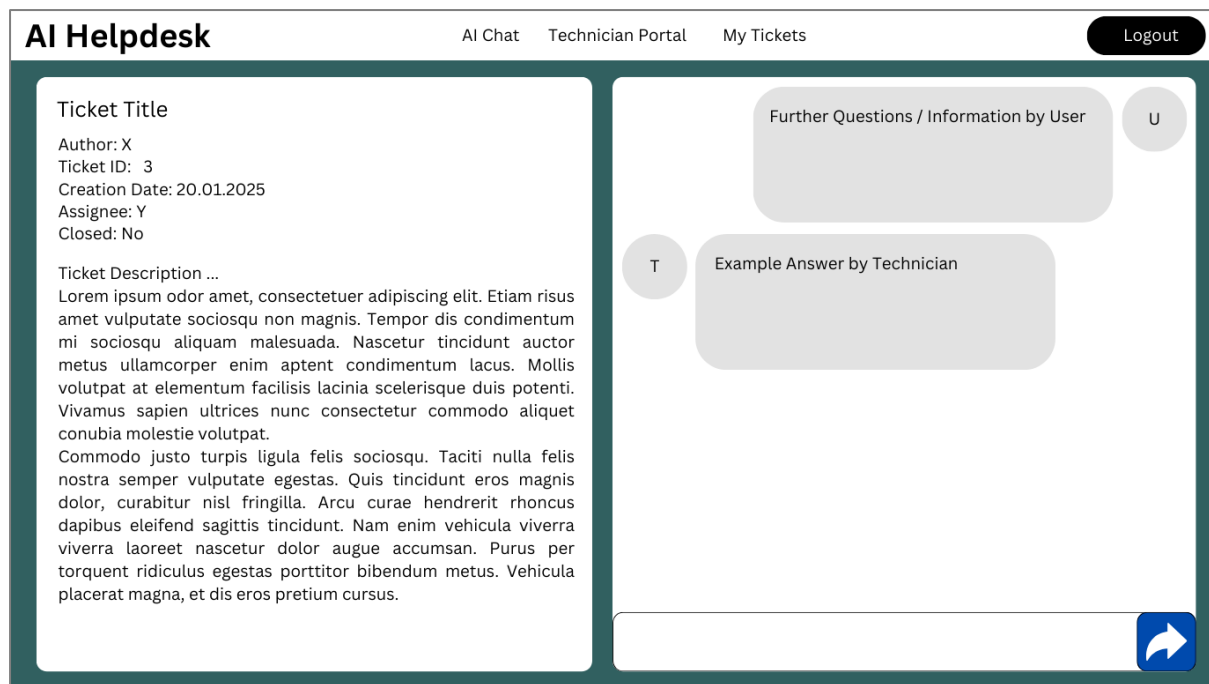


Figure 4 Ticket Page Design

For the design, it was decided to opt for a modern but minimalistic and straightforward approach. This makes it easy for the user to navigate the page and access all necessary information and features with ease, without adding unnecessary or distracting design elements.

## 5.7 Technical Design

This subchapter will discuss which languages, tools, platforms, and frameworks the application will be composed of. However, it will not consider technologies and conventions that are used for the development process itself, this will be discussed in Chapter 6, *Implementation and Testing*.

All components will be discussed individually, as they each consist of several different technologies. The decisions in this section that are not supported by citations, are made under consideration of the insights that were discussed in Chapter 2, *Literature Review*.

### 5.7.1 AI System Design

As modern Large Language Models possess exceptional problem-solving and reasoning capabilities, they are well suited as a solution to automate ticket resolution and generation. For this project the LLM would have to handle vast amounts of personal and business-related data. As previously discussed, when using LLMs as a service via an API, offered by



companies like OpenAI, potential security and data privacy issues arise. To avoid these, it is beneficial to self-host an LLM on the company's infrastructure. When choosing a local model, it is essential to balance power, efficiency, and costs to meet the project's requirements. Since this project focuses on the development of this system without intent to deploy this application in production and has limited computational power, a smaller model will be utilised. In fact, it is beneficial to leverage a less powerful model in the development process to set a base level with which the system can be expected to perform efficiently. Companies using this application may decide to leverage a different, possibly custom fine-tuned model. Using a stronger model is always possible and will increase accuracy, but when utilising models that are smaller than the one the system was developed on, it could cause unexpected results and insufficient performance.

When selecting a model for this project, only models up to 10 Billion parameters were considered due to hardware limitations. Models with less than 7 Billion parameters were tested, but were found to be insufficient in reasoning as they were not able to construct simple troubleshooting processes. Utilising the open-LLM leaderboard, published by huggingface [69], the top ten models with 7 to 10 Billion parameters, ordered by average score, were examined.

Rank	Type	Model	Average	IFEval	BBH	GPQA	MUSR	MMLU...
265		<a href="#">tiiuae/Falcon3-10B-Instruct</a>	● 35.19 %	78.17 %	44.82 %	10.51 %	13.61 %	38.10 %
277		<a href="#">tiiuae/Falcon3-7B-Instruct</a>	● 34.91 %	76.12 %	37.92 %	8.05 %	21.17 %	34.30 %
481		<a href="#">CohereForAI/c4ai-command-r7b-12-2024</a>	● 31.08 %	77.13 %	36.02 %	7.83 %	10.23 %	28.58 %
520		<a href="#">internlm/internlm2_5-7b-chat</a>	● 30.58 %	61.40 %	57.67 %	10.63 %	14.35 %	30.42 %
522		<a href="#">ibm-granite/granite-3.1-8b-instruct</a>	● 30.55 %	72.08 %	34.09 %	8.28 %	19.01 %	28.19 %
605		<a href="#">microsoft/Phi-3-small-8k-instruct</a>	● 29.67 %	64.97 %	46.21 %	8.28 %	16.77 %	38.96 %
688		<a href="#">google/gemma-2-9b-it</a>	● 28.86 %	74.36 %	42.14 %	14.77 %	9.74 %	31.95 %
721		<a href="#">microsoft/Phi-3-small-128k-instruct</a>	● 28.59 %	63.68 %	45.63 %	8.95 %	14.50 %	38.78 %
757		<a href="#">meta-llama/Meta-Llama-3.1-8B-Instruct</a>	● 28.20 %	78.56 %	29.89 %	2.35 %	8.41 %	30.68 %
786		<a href="#">01-ai/Yi-1.5-9B-Chat</a>	● 27.89 %	60.46 %	36.95 %	11.30 %	12.84 %	33.06 %

Figure 5 LLM Benchmarks

While benchmarks do give a good estimate on a model's performance, it is essential to find a model that suits the specific deployment scenario. Therefore, the decision on which model to choose was pushed back until after the completion of the AI system. All of the listed models were tested, concluding in Google's gemma-2-9b-it to provide the best and most consistent answers by showing good understanding of processes and utilising the RAG system

effectively. Other models, like the highest-rated Falcon3-10B-Instruct, sometimes struggled at following simple instructions, such as solely providing a yes or no answer.

Since the helpdesk optimised for and integrated with the gemma-2-9b-it model, it can easily be deployed with it, without the need for major customisations. But the AI system is built modularly, allowing for the exchange of the LLM using the configuration file. To allow prompt optimisation for specific models, the prompts are stored and loaded separately in an editable JSON file.

As discussed in the literature review, employing retrieval augmented generation (RAG) is key to increasing accuracy and knowledge, especially for small models. This project will use an advanced RAG approach, evaluating retrieved data using the LLM and possibly performing a web search for additional information under certain conditions. The vector representations of the business documents that support the LLM's decision-making are stored in a vector database, enabling document selection based on the similarity to the user request. Due to the same security and privacy concerns mentioned for deploying LLMs, a locally deployed vector database is the preferred solution. When comparing vector databases, Milvus stands out as an established and highly scalable database [70]. In contrast to the popular database Pinecone, it is locally deployable [70]. And while Chroma is another promising solution, it is not as scalable when used outside of the cloud, since it offers single-node deployment while relying on a lightweight SQLite database for storing the vectors [71]. While it is probably strong enough for most implementation sizes of this helpdesk, Milvus was chosen as the preferred solution since it assures higher scalability by offering multi-node deployment [72].

In addition to employing Milvus for storing documents that support issue resolution and ticket creation, it is used to link similar, already solved tickets to the current one. This is done by storing summaries of all tickets and comparing the embedded summaries with the current one's vector representation. This allows the technician to check whether this problem was solved before or might have similar troubleshooting steps to related tickets.

To further enhance the accuracy of the system, it is crucial to support multi-step task solving. As discovered in Chapter 2, *Literature Review*, this can be achieved by creating a human-designed workflow that guides the LLM through different states, instructing it what to do. This will be done by leveraging the LangChain and LangGraph frameworks. These streamline the development of AI processes [73], allow for building state graphs [74] and

integrate well with other technologies like Ollama, which will be used to run the model locally [75]. Additionally, it can be used with LangSmith, which allows monitoring every action and output the LLM performs when traversing the workflow, providing exceptional debugging capabilities for development [76].

While LangChain is available as a JavaScript framework, it was originally designed for Python. As a result, there are more resources, a larger community, and better support for the Python version, which makes it easier to find help and troubleshoot issues. The popularity can easily be assessed by comparing the monthly downloads. The Python package currently has 30 million downloads per month [77], while the JavaScript package lies at 2 million [78]. For these reasons, Python was chosen for the development of the AI system.

The LangGraph workflow is designed to break this problem into smaller and simpler sub-problems. It includes several conditional edges that choose the next step based on variables or the LLM's own decisions. This workflow will be outlined as a flow chart below. It starts with generating a query prompt based on the chat history provided. This is needed to initiate the RAG system, needed for retrieving documents to provide extra context. Next, they are graded by the LLM and their relevance to the problem is verified.

While from here, there are multiple variations of traversing this graph, there are two main paths that depend on the variable “ticket” being true or false, indicating whether the current objective is to find a solution or to generate a ticket.

1. The first sub-path focuses on issue resolution. A user asks for help by prompting an issue, which in the chart is indicated with the boolean “ticket” being false. If not enough relevant content was found utilising RAG, the variable “search” will be set to true. In this case, the next conditional edge will forward to the web-search and continue with checking the solvability of the problem. Otherwise, the web-search would be skipped. If the issue can be solved without elevated privileges, the LLM will generate a guide to lead the user through the troubleshooting process. This path might be chosen multiple times after each new user prompt, but can be limited to a certain number of iterations using the config file. This continues until the issue is resolved. If it cannot be solved, or the maximum number of iterations is reached, the LLM will offer to generate a ticket on behalf of the user. If the user accepts this offer, the second sub-path is initiated.
2. The second sub-path aims to generate a ticket for the user. It is indicated in the flow chart with the boolean “ticket” being true. Documents are retrieved, graded, and possibly

extended with web-search in the same manner as before. The next step determines whether enough information was provided by the user to generate a detailed ticket. On the first iteration of this sub-path, the LLM will ask further questions to gather details needed for the ticket, such as laptop model, software versions and error codes. If these questions are answered, it will proceed with generating the ticket.

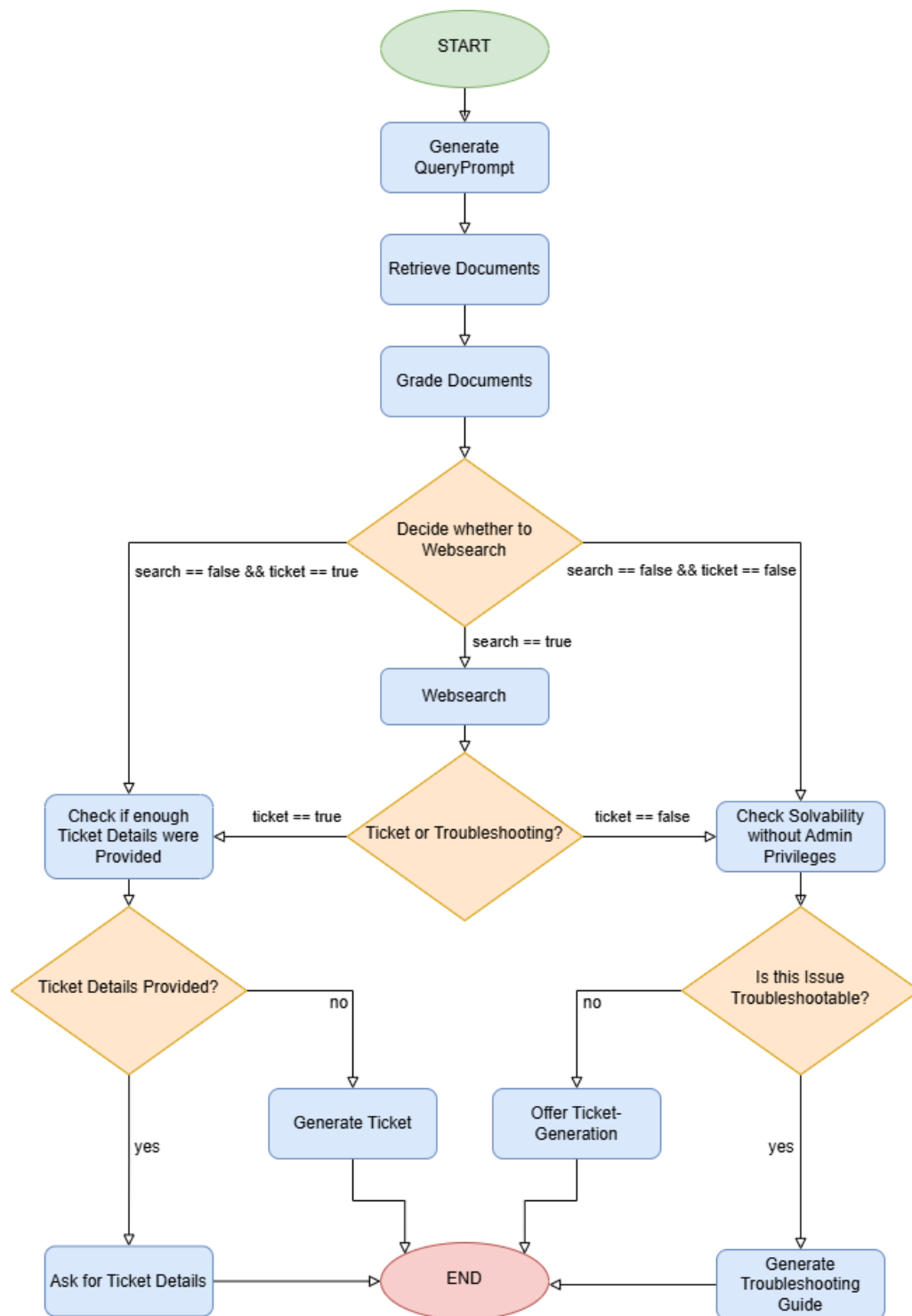


Figure 6 AI Workflow Sequence-Diagram

### 5.7.2 Database Design

A well-structured database system which offers reliability and data integrity, has major importance for this helpdesk system. It will be used to securely store and manage the AI-generated tickets.

As discovered in the Literature Review, there is no right or wrong when deciding between relational and NoSQL databases. Instead, it depends on the application's performance and consistency needs, the data structure, the required scalability, and the desired execution speed of CRUD operations.

Since this database will be solely used for storing clearly structured ticket data, a relational database would be sufficient, even for very vast deployment scenarios. The tickets are created once and might be read and updated multiple times by the user and the technician. Since typically only a small percentage of employees use a company's helpdesk system at the same time, even companies with tens of thousands of users would never outgrow a relational database's scalability capabilities, since they can rapidly handle and process thousands of simultaneous requests for read and update operations [46], [47].

MySQL, being one of the most popular relational database systems for web applications [44], was chosen for this application. There are many well-established options that all provide similar base functionality. While there are some advanced features that differ, these functionalities are not required for the scope of this project, where the sole objective is to effectively store ticket data. Therefore, a well-supported and documented open-source solution like MySQL is preferred over commercial systems like Oracle Database or IBM DB2.

Since it has been decided to integrate with Azure for user authentication in order to avoid the need to create separate accounts for the sole purpose of this application, the database has to store user data mappings from Azure instead of actual user credentials. By storing user names and groups in the local database, there is no need to repeatedly fetch these details from Azure to display ticket author and technician names on the frontend. This significantly reduces network traffic and improves the performance of the application.

The desired database structure that allows storing tickets along with the conversation between technicians and users is outlined in the following ER-diagram:

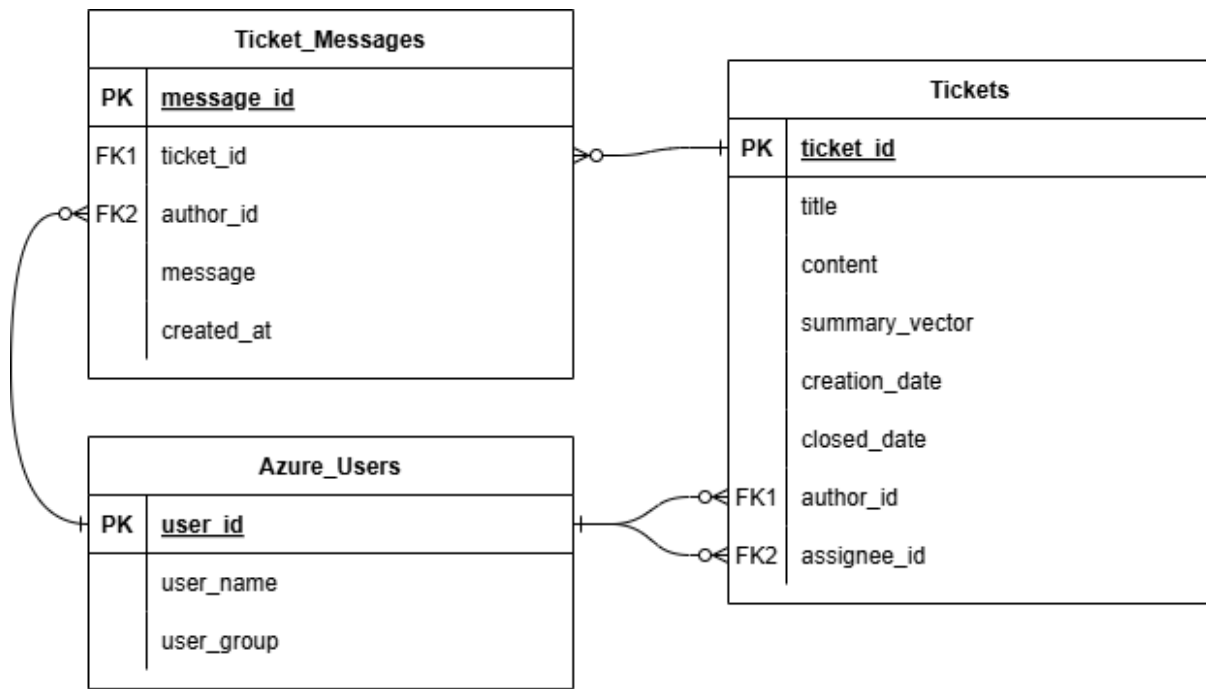


Figure 7 ER-Diagram

### 5.7.3 Backend Design

The backend is the interconnecting layer that manages interaction between all the subsystems and the frontend. It is the key element to verify authentication and translate the user's requests into actions by calling database and AI system functionalities.

Since it was decided to use Python for the AI system, it is appropriate to leverage the same technology for the backend to benefit from better integration and seamless interactions. Python is a popular language for building backends and offers a wide range of frameworks, the most popular ones being Flask, Django, and FastAPI [74]. When choosing a framework, the choice must be made regarding the project's scope and requirements.

Flask is a lightweight, flexible and modular framework which is easy to implement. But since the AI Helpdesk aims to build a scalable platform, Flask does not stand out as the optimal choice for this project due to it being synchronous and limited in scalability. It is most suited for smaller APIs or static websites [79], [80].

Django is quite the opposite. It is famous for being a feature-packed platform that provides all the tools and components needed for a fully functional backend, including an admin interface, an ORM, authentication handling, security features, and a plugin system. While

being highly flexible, powerful and scalable, it also adds boilerplate, complexity, and potentially unnecessary features. This leads to worse performance compared to lightweight frameworks and makes it difficult for beginners to build applications with this framework [74], [75].

FastAPI, on the other hand, features a more modern approach and focuses on performance and developer productivity [79], [81]. Despite being released in 2018 - 13 years after Django - FastAPI already caught up in terms of popularity [74]. Its asynchronous design allows FastAPI to handle concurrent requests, which improves performance and reduces response times. Furthermore, it is compatible with most frameworks and libraries, and its functionality can be extended using middleware. But despite being modular and lightweight, it incorporates essential features for security, authentication and type checking [81], [82].

The aim for the AI Helpdesk's backend is to provide a straightforward but scalable, performant, and secure system. Since the API will be kept simple and authentication will be handled via Azure, there is no need for extensive tooling. This is why Django's feature-richness is not needed. Instead, FastAPI was chosen since it packs all necessary features and provides superior performance.

The Azure authentication will be implemented using Microsoft's recommended authentication flow for single page applications [83]. On user login, the frontend will fetch an access token from Azure. On every API request, this token will be sent to the backend, which will verify the token and fetch the user's id, username, and security groups from Azure Entra ID. Depending on the fetched groups, the backend can enforce role-based access control (RBAC). However, if the token cannot be verified, the request will be rejected.

#### 5.7.4 Frontend Design

The frontend has major significance since it manages the user interaction and visually represents the product.

Chapter 2, *Literature Review*, discovered the current trend of developing Single-Page Applications instead of Multi-Page Applications to avoid reloads, enable seamless content changes and improved user experience. To benefit from these advantages, it was decided to adopt this technology in this project.

The choice of which frontend framework or library to use was made between the three most popular ones - React, Angular, and Vue.

When comparing startup performance, research [62] claims that Angular web apps take the longest time to load for small applications. This is due to its slightly bigger code bundle size. While React offers quicker load times, Vue's bundle is smaller and loads even faster [62].

The runtime performance describes the speed of re-renders, UI updates, and processing of user interactions. In this area, once again, React and Vue have an edge over Angular since they manipulate the HTML components that are represented by objects in the Document Object Model (DOM) in a different way. While Angular manipulates the DOM directly, React and Vue create and maintain a Virtual DOM, apply all the necessary changes, and then manipulate and re-render the actual DOM with all the accumulated changes at once. Since manipulating the actual DOM takes way more time, gathering changes first increases overall performance [62].

React and Vue emerge as the best options, and while both are performant and versatile, React was chosen as the frontend library for this project. This choice was made because React is not only more established in the industry, but also because the author has previous experience with this technology. This experience will speed up the development process and help build a more robust and well-designed application.

The UI design was already covered in the previous section, where the initial design mock-ups were presented. To implement these designs in the actual code it was decided to use an UI library instead of relying on vanilla CSS, to accelerate development.

While there are many libraries out there, it is mainly a matter of preference which one to choose. For this application, Chakra UI was chosen. It offers a lot of prebuilt but customisable components that simplify and speed up designing the webpage [84]. For styling, a dedicated CSS file is not necessary anymore, since the styles can be applied inside the React JSX components using inline style props. Since React applications are built highly modular with splitting pages into several components and sub-components, this helps to keep the code organised by placing the styling inside each component instead of maintaining one big CSS file.



## 5.8 Security Design

Security design has major importance to safeguard the application from potential threats. Several practices were applied to avert vulnerabilities or breaches. The objective is to maintain alignment with the CIA triad to ensure confidentiality, integrity, and availability.

To secure logins across several subsystems of the application it is important to follow best practices for credential management.

API keys and login credentials for databases must not be stored in code files that are accessible by frontend clients or pushed to GitHub. To prevent leakage of these details, they should be stored securely in environment variables. Furthermore, each accessed subsystem should have a dedicated user account for every application that accesses it. The permissions must be set to be as restrictive as possible to only allow necessary operations.

Another security threat is SQL injections. It is a technique of passing malicious SQL statements into input fields to break the query logic on the backend. Attackers use this method to access unauthorised data or delete and manipulate database records. This should be prevented by using parameterised queries instead of string concatenation when constructing SQL statements.

Authentication on front and backend ensures to allow access to webpages and API endpoints to authorised users only. The frontend will authenticate with Azure to receive an access token, which is used to authenticate with the backend. Additionally, role-based access control (RBAC) must be enforced to display or hide certain components on the frontend and allow or deny access to specific API endpoints based on the user's group.

To find and eliminate further security issues, tooling can be used to scan the application for vulnerabilities. These vulnerabilities must be fixed to ensure a secure application. Further information about this tooling will be discussed in the security testing section.

## 5.9 Testing and Quality Assurance

To ensure a smooth, bug-free application, testing is a necessity to detect and resolve errors. As part of a high-quality assurance process, the hybrid development methodology used is leveraging continuous testing for early bug detection to prevent more complex issues.

After each implemented feature, the component is tested manually to verify its functionality. Additionally, unit tests must be written to automatically assess the component to detect breaking changes while development continues.

To test the application as a whole, an end-to-end test must be set up to automatically work through a predefined testing scenario. This will iterate a variety of different scenarios and determine whether the components work together correctly to create the expected frontend output.

To make sure the application performs efficiently, with fast load times, smooth rendering and quick response times, performance testing must be conducted. This will involve testing the AI workflow's execution times as well as testing the frontend's responsiveness.

Furthermore, security testing will be performed using Zed Attack Proxy (ZAP). This program is a dynamic security testing tool which scans the application for vulnerabilities and potential threats.

Lastly, to determine the application's usability, intuitiveness, and accessibility, User Acceptance Testing (UAT) will be conducted at the university's testing day. Since user acceptance has major importance for the adoption of an application, it is important to match expectations and to allow for UI or UX readjustments if necessary.

## 5.10 Risk Management

This section focuses on the risk assessment conducted for this project. It was done to create awareness of potential issues and to design strategies for mitigation and handling of these risks to ensure the project remains on track and achieves its objectives.

### 5.10.1 Hardware Risks

1. **Risk:** Failure of the development machine could cause development to stall, leading to schedule delays.  
**Solution:**
  - A backup laptop is available to continue development in case of a desktop failure.
2. **Risk:** A defective hard drive could result in loss of project files and data.  
**Solution:**
  - All data is synchronized between the desktop, notebook, and a Network Attached Storage (NAS).
  - The NAS is backed up daily to decrease the likelihood of data loss.
  - GitHub is used for version control to keep a secure copy of all project files.

### 5.10.2 Software Risks

1. **Risk:** Updates to the development environment, frameworks, tools, or dependencies may cause compatibility issues with the written code.

**Solution:**

- Use Long-Term Supported (LTS) versions of software.
- Save package versions in requirement files to be able to revert to older versions if updates cause problems.

### 5.10.3 Time Risks

1. **Risk:** Underestimation of time required for each task or delays caused by illness.

**Solution:**

- This problem is addressed in the Methodology. The Waterfall method is combined with an Agile approach to allow for prioritisation in case the project deadline is at risk.

2. **Risk:** Unforeseen lack of knowledge or technical difficulties could result in delays.

**Solution:**

- For bigger issues, the supervisor will be asked for guidance if research and testing could not solve the problem.

One week is allocated at the end of the timeline for further improvements. This week can also be used as a time buffer for unforeseen delays.

### 5.10.4 Supervisor Communication Risks

1. **Risk:** Unclear and infrequent communication with the supervisor can impact the project's quality, as the supervisor provides direction and helps prevent mistakes.

**Solution:**

- Meetings are scheduled every 2–3 weeks to ensure regular feedback.
- The next appointment is scheduled at the end of each meeting to ensure consistency.

2. **Risk:** Meetings might be cancelled due to illness or unavailability of either party.

**Solution:**

- In case of cancellation, the meeting has to be rescheduled.
- If the supervisor is not available for a long or unknown period, meetings have to be scheduled with the second supervisor.

## 5.11 Conclusion

This chapter provided a comprehensive overview of the technical design behind the AI-Integrated IT Helpdesk application. It outlined the requirements, emphasizing performance, user satisfaction, security, and maintainability, and stated the corresponding goals and objectives that were set to achieve them. Furthermore, the design of the system's architecture and user interface was described in detail, explaining the core technologies and providing the rationale for each design decision.

The key design outcome for creating the performance-enhanced helpdesk was leveraging a self-hosted Large Language Model with an advanced RAG system using modern frameworks, including LangGraph and LangChain for modelling the AI workflow. Additionally, FastAPI will be leveraged as a performant backend framework, interacting with a MySQL database to store helpdesk data and the Milvus vector database for storing RAG document and ticket embeddings. React was selected as the library to construct the frontend in the form of Single Page Application (SPA).

Furthermore, security measures were discussed, focusing on threat prevention by enforcing credential management, authentication, and vulnerability scanning. Additionally, further testing procedures were discussed to ensure functionality, performance, and security.

This research-backed design provides a strong foundation and serves as a blueprint for the successful implementation of a robust and intelligent helpdesk solution.

## 6 Implementation and Testing

### 6.1 Introduction

This chapter discusses tools, strategies and practices used for development of the application. Furthermore, it provides clear insight into the implementation process, the development of each component, explains key parts of the codebase, and explicates applied testing procedures.

### 6.2 Implementation Plan

As highlighted in Chapter 3, *Methodology*, this project follows a hybrid management approach. Despite focusing on the Waterfall methodology to ensure fulfilling predefined requirements on schedule, it incorporates some agile features, such as continuous testing for early issue detection and task prioritisation to avoid missing deadlines.

Following this scheme, the implementation schedule is completely predefined and outlined in a Gantt chart, which can be found in the appendix. The development is divided into two main sections.

The first section focuses on implementing the four components of the main application, including the AI system, the database, the backend, and the frontend. These are developed in the order mentioned, since building the backend requires access to the AI system and the database, and the frontend logic is easier to implement with a running backend that is already capable of providing necessary data. Finally, these are combined into one seamlessly running system.

While this first section already includes testing during and after the implementation of each component, the second section focuses on testing the application as a whole and the interaction between components. Additionally, it schedules another week for further improvement or debugging of the system.

### 6.3 Development Environment

The environment is based on Windows, using Visual Studio Code as an Integrated Development Environment (IDE), which offers support for various languages, can handle diverse file types, integrates well with Git, and is extendable with several development plugins.

Python 3.12.7 is used for development and installed in the development environment. This version was chosen since some of the required packages were not yet compatible with the latest version of Python.

Since the frontend is built on React with TypeScript, Vite is used as a build tool for creating the React app and compiling into JavaScript.

Version control is utilised by leveraging GitHub to track and manage the project. This also serves as a backup, since the code is stored on GitHub and can be retrieved at any time. Additionally, daily backups of the codebase and the documentation are created using a Network Attached Storage (NAS) to prevent data loss.

The Microsoft Azure Portal is used to implement and test the integration and authentication with sample users.

MySQL Workbench is leveraged to run and configure the relational database used for storing the tickets and Azur user data mappings.

Since Milvus was chosen as a vector database and does not work natively with Windows, Docker is used to run this service in a container for development.

Additionally, Chrome's DevTools and supplementary browser plugins are utilised to simplify development. These include React Dev Tools for frontend debugging and Lighthouse for performance testing.

## 6.4 Coding Standards and Practices

To ensure consistency, readability, and maintainability of the codebase, it is necessary to predefine standards and practices that have to be followed during development.

Since the waterfall methodology is applied and the application is planned upfront, a well-defined architecture ensures smooth step-by-step development and prevents rewrites. This includes a modular approach, breaking code down into different components for separation of concerns (SoC) to keep it organised and maintainable. This can be achieved by choosing a layered architecture, and separating presentation, business logic, and data access.

Using TypeScript for frontend, Pydantic for API input validation, and Python type annotations help ensure type safety. This prevents errors, improves code clarity, and makes development easier with better autocompletion and debugging.

Additionally, code formatting has major importance in keeping a consistent layout to ensure readability. This is achieved by using the tool Prettier to autoformat TypeScript code on the frontend, and Black for Python. Furthermore, naming conventions have to be applied to ensure consistent variable, function, and class names. TypeScript commonly uses camelCase for variables and functions, and PascalCase for classes and interfaces. For Python, snake\_case is used.

To ensure smooth operation of the application and improve debuggability, errors need to be thrown, handled, and logged correctly. While fundamental, critical errors should terminate the application, less serious errors have to be caught, and measures for retrying or continuing despite these errors must be implemented. They have to be logged at the location they arise, and either be handled or passed up the hierarchy in order to either decide how to proceed, pass the error to the frontend, or terminate the application. For clarity, debugging should be split into separate levels, assigning messages to the categories debug, info, warning, error, and critical. These messages can be printed on the console but should also be stored in a log file to retrace recent actions and errors.

Lastly, inline comments are used to improve maintainability. They provide brief explanations of logic, parameters, or code behaviour to ensure understandability when revisiting written code.

## 6.5 Module and Component Development

This section discusses the development of each component as outlined in Chapter 5, *Design*. It explains the implementation of key features, while focusing on more complex sections of the code.

### 6.5.1 AI System Development

The first step of the AI system development was building the LangGraph workflow, strictly following the previously designed flow chart. It involved defining a function for each subproblem, adding state variables, and assembling the structured graph by adding the nodes and conditional edges. Each function accesses LangChain-defined chains that pass a prompt

with its respective task to the LLM. Since the LLM is exchangeable and can be chosen using the application's config file, the prompts were placed in a JSON file to allow manual prompt optimisation for the specific model.

The following excerpt of the code shows the node to create the troubleshooting guide for the user. It retrieves the necessary attributes from the current state, parses the RAG documents into an object of LangChain's SystemMessage class, and invokes the troubleshooting chain, providing these documents and the chat history as context. The generated output is returned, updating the state graph's variable.

```
def generate_troubleshooting_guide(state: GraphState):
    input = state["input"]
    documents = state["documents"]
    chat_history = state["chat_history"]
    generation = troubleshooting_chain.invoke(
        {
            "documents": documents,
            "input": input,
            "chat_history": chat_history,
        }
    ).content
    generation = remove_think_tags(generation)
    return {"generation": generation}
```

Figure 8 Code: Generate Troubleshooting Guide

The invoked chain initiates the prompt's assembly and passes it to the LLM.

```
troubleshooting_chain = troubleshooting_prompt() | llm
```

Figure 9 Code: Troubleshooting Chain

The prompt is imported from a separate script that handles the prompt assembly by inserting the custom prompts from the JSON file.



```
def troubleshooting_prompt():
    return ChatPromptTemplate.from_messages(
        [
            (
                "system",
                prompts["troubleshooting_prompt"]["default_prompt"],
            ),
            MessagesPlaceholder("documents"),
            MessagesPlaceholder("chat_history"),
            MessagesPlaceholder("input"),
            (
                "system",
                prompts["troubleshooting_prompt"]["followup_prompt"],
            ),
        ]
    )
```

Figure 10 Code: Troubleshooting Prompt

All other nodes are implemented in a similar fashion.

Wrapped into a data class, a method to initiate this workflow was made accessible to the backend, enabling troubleshooting or ticket generation.

For accessing the documents, RAG was implemented using Milvus as the vector database. Since this database does not work natively on Windows, Docker was used to set it up for development. For modularity and maintainability, the connection, initialisation, and methods to access the vector store were implemented in a separate file. This file automatically handles the creation of user credentials and required schemas and collections, which simplifies the deployment of this application. It fully configures the database using the standard root password, which can be changed after initial setup is done.

Two collections are created, one for storing documents for the RAG system to provide extra context for the LLM, and one that stores ticket summaries in order to find similar tickets. Similar tickets will be linked to each new ticket, allowing the technician to look at previous solutions to related problems.

The following code handles the insertion of PDF files into the vector store.

It tries to read the PDF file's data and appends the text of each page to a string. This string is split into chunks and embedded into vector representation using the embedding model.

The embeddings, the origin text chunk, and its metadata are added to a list that follows the collections schema. Finally, the data is inserted into the vector store.

```

def process_file(self, file_path):
    file_name = os.path.basename(file_path)
    try:
        reader = PdfReader(file_path)
    except:
        print("Error: File not readable. Only pdf files are supported.")
        return
    text = ""
    for page in reader.pages:
        text += page.extract_text()
    text_splitter = RecursiveCharacterTextSplitter()
    text_chunks = []
    metadata_list = []

    for text_chunk in text_splitter.split_text(text):
        text_chunks.append(text_chunk)
        metadata_list.append(file_name)

    embeddings = embedding_model.embed_documents(text_chunks)

    data = [
        {"embedding": embedding, "text": text, "metadata": metadata}
        for embedding, text, metadata in zip(embeddings, text_chunks, metadata_list)
    ]

    milvus_client.insert(collection_name="collection_rag", data=data)

```

Figure 11 Code: File Embedding Insertion

For keeping the vector store's data up to date, the library watchdog is used to observe the folder in which the RAG documents will be placed. On insertion or deletion, it will pass the file's name to the application, which will either embed these documents and store them or delete the associated vectors from the database.

As shown below, watchdog is executed in a separate thread to prevent it from stalling the main application since it is running an infinite loop to check for file changes.

```

watcher_thread = threading.Thread(
    target=_watch_directory,
    args=(
        milvus_client,
        config["milvus"]["rag_documents_folder_absolute_path"],
    ),
    daemon=True,
)
watcher_thread.start()

```

Figure 12 Code: Watcher Thread

### 6.5.2 Database Development

The first step of implementing the MySQL database was installing and setting up MySQL Workbench. While it is used to run the MySQL server for development, this is not suitable for production, where a dedicated MySQL server instance must be installed on a separate server.

The database, along with its respective tables and constraints, was implemented as outlined in the ER-diagram from Chapter 5, *Design*. This was achieved by writing an SQL script to create the structure, which is included in the appendix.

For connecting and authenticating from the backend, a separate database user was created. The privileges were set to allow only reading and writing to this application-specific database, without permission for data definition operations like altering or deleting tables.

### 6.5.3 Backend Development

The backend implementation started with initialising FastAPI, setting up the basic structure, and creating the route for serving static files, which will later serve the frontend.

Furthermore, the config file and environment variables were loaded to specify the database hostname and credentials, and Azure URLs, and IDs.

A rotating file and console logger was set up to enable debugging and event tracking. Events are categorized by severity. When running the application, it can be specified which level of severity is logged. To prevent log files from becoming too big, they are set to a maximum size of 5MB. Once this size is reached, a new file is created, which will override the oldest existing log once a maximum of 5 log files is reached.

Authentication was implemented using Azure. For development, a free Azure account was created, and users with security groups “users” and “technicians” were created for testing authentication and role-based access control. As described in Chapter 5, *Design*, the frontend acquires an access token from Azure and sends it along with the API request to the backend. FastAPI’s OAuth2AuthorizationCodeBearer, a security scheme to authenticate via an OAuth provider, was used to set up Azure’s OAuth Authorization Code Flow. On every endpoint that requires authentication, the following function is called to verify the token with Azure.

```

async def verify_token(token: Annotated[str, Depends(oauth2_scheme)]) -> User:
    headers = {"Authorization": f"Bearer {token}"}
    response = requests.get(GRAPH_API_URL, headers=headers)
    user_resp = response.json()

    if response.status_code != 200:
        raise HTTPException(status_code=401, detail="Invalid token")

    groups_response = requests.get(f"{GRAPH_API_URL}/memberOf", headers=headers)
    groups_data = groups_response.json()

    group_ids = []
    if "value" in groups_data:
        group_ids = [group["id"] for group in groups_data["value"]]

    group = ""
    if USERS_GROUP_ID in group_ids:
        group = "users"
    if TECHNICIANS_GROUP_ID in group_ids:
        group = "technicians"

    return User(
        user_id=user_resp.get("id"),
        user_name=user_resp.get("displayName"),
        group=group,
    )

```

Figure 13 Code: Azure Authentication

It passes the token to the OAuth scheme, authenticates the user, and fetches their ID and name. Additionally, groups for the specific user are retrieved from Azure. The gathered data is now returned to the endpoint function that initiated the authentication in the form of a defined Pydantic model called User, which type checks the user information to ensure the right format.

When an endpoint not only enforces authentication but requires the user to belong to the “users” or “technicians” security group, the following function is called instead.

```

def check_user_group(required_group_id: str) -> Callable[[User], User]:
    def role_checker(user: Annotated[User, Depends(verify_token)]) -> User:
        user_group = user.group
        if required_group_id != user_group:
            raise HTTPException(status_code=403, detail="Unauthorized access")
        return user

    return role_checker

```

Figure 14 Code: Role Based Access Control

It uses the previous function to authenticate the user, but additionally checks their group against the group name defined in the endpoint to restrict access to specific roles.

When a user logs in, the frontend will have to fetch the user's information from the backend, since the elements rendered on the frontend depend on the user's role. The following endpoint will serve this information.

```
@app.get("/users/me")
async def read_users_me(user: Annotated[User, Depends(verify_token)]):
    try:
        if not is_azure_user_in_db(user.user_id, config):
            insert_azure_user(user.user_id, user.user_name, user.group, config)
    except Exception:
        raise HTTPException(status_code=500, detail="Storing user data failed")
    print(user)
    return user
```

*Figure 15 Code: User Me Endpoint*

Additionally, this function will initially store the user's attributes in the relational database if an entry does not already exist. This is needed to render ticket author and assignee names without fetching this information from Azure every time tickets are queried.

After authentication was set up, an endpoint for interacting with the AI system was created. Every time a user prompts a message on the frontend, this endpoint is called, which starts the AI workflow with the provided chat history. The inputs are type checked with a custom Pydantic model that defines the expected data types to ensure the parameters are valid. The endpoint will either respond with an answer to the user's issue or generate a ticket and return its ID in order to display it on the frontend. Additionally, an embedding of the ticket summary is created and inserted into the MySQL database along with the ticket title, content, and author. If a ticket gets closed, its ticket summary embedding is inserted into the vector database. This way similar tickets can be determined when querying a ticket by performing a similarity check with the retrieved ticket's summary vector to all other stored summary vectors of resolved tickets. This way, a technician will get resolved tickets similar to the one they have queried and may receive useful insights from how these tickets were solved.

Besides the endpoint for accessing the AI workflow, several other endpoints were defined for querying a single ticket by ID, or a list of tickets by assignee or author, fetching a list of

technicians for assignee filtering, closing or assigning tickets, and adding messages to the ticket conversation.

Since these endpoints need to interact with the MySQL database, a separate file was created to handle database connection and operations using the Python library MySQL Connector. The connection is established based on the hostname and credentials provided using the configuration file and environment variables. The module offers several functions to query for tickets and technicians, handle ticket operations, and insert Azure user information.

Following function fetches a single ticket from the database, based on its ID, in order to display it on the frontend.

```
def get_ticket(ticket_id: int, user: User, config: AppConfig) -> Ticket:
    cnx = connect_to_mysql(config)
    try:
        cursor = cnx.cursor()
        query = """SELECT ticket_id, title, content, summary_vector, creation_date,
                        closed_date, u.user_name as author_name, a.user_name as assignee_name
                        FROM tickets t INNER JOIN azure_users u ON t.author_id = u.user_ID
                        LEFT JOIN azure_users a ON t.assignee_id = a.user_id WHERE ticket_id = %s"""
        cursor.execute(query, (ticket_id,))
        ticket = cursor.fetchone()

        ticket["similar_tickets"] = []
        if "summary_vector" in ticket and user.group == "technician":
            summary_vector = json.loads(ticket["summary_vector"])
            similar_tickets = retrieve_similar_tickets_milvus(summary_vector)
            ticket["similar_tickets"] = similar_tickets

        if "summary_vector" in ticket:
            ticket.pop("summary_vector")

        ticket["ticket_messages"] = get_ticket_messages(ticket_id, config)

        return ticket

    except Exception as e: ...

    finally:
        cursor.close()
        cnx.close()
```

Figure 16 Code: Get Ticket Database Function

It fetches the ticket from the database and uses two joins to fetch the author's and assignee's name from the database. A left join is used to fetch the assignee, since the assignee might be null on an unassigned ticket. Parameters, such as the ID, are inserted with parameterised queries instead of string concatenations to prevent SQL injections.

When fetching a ticket, the user object is provided, which includes information about the user's role. If a technician is requesting the ticket, the summary vector of this ticket is

compared to those in the vector database in order to retrieve the IDs of resolved similar tickets. These similar tickets will be presented to the technician on the frontend. Since the summary vector is not needed on the frontend and adds unnecessary payload, it is removed from the ticket object.

Finally, another function is called which fetches and adds the messages written within the ticket conversation to the ticket object and returns it to the API endpoint which will send it as a response to the client.

#### 6.5.4 Frontend Development

Implementing the frontend started with setting up the React app and installing necessary packages and dependencies.

The main component uses multiple providers and implements routing functionality.

```
const queryClient = new QueryClient();

createRoot(document.getElementById("root")!).render(
  <StrictMode>
    <MsalProvider instance={msalInstance}>
      <Provider>
        <QueryClientProvider client={queryClient}>
          <BrowserRouter>
            <Routes>
              <Route element={<App />} />
              <Route index element={<LoginPage />} />
              <Route element={<Layout />} />
              <Route path="ai-chat" element={<AIChatPage />} />
              <Route
                path="technician-portal"
                element={<TechnicianPortalPage />}
              />
              <Route path="my-tickets" element={<MyTicketsPage />} />
              <Route path="ticket/:id" element={<TicketPage />} />
              <Route path="*" element={<ErrorPage />} />
            </Routes>
          </BrowserRouter>
          <ReactQueryDevtools />
        </QueryClientProvider>
      </Provider>
    </MsalProvider>
  </StrictMode>
);
```

Figure 17 Code: Main React Component

The MsalProvider wraps the application to integrate the Microsoft Authentication Library (MSAL) to set up the Azure authentication. The QueryClientProvider provides React Query's functionality, which is used to create advanced API queries that support caching, refetching

and error handling with automatic retries. ReactQueryDevtools provide a debugging tool for React Query that displays current fetches and cached queries. Routing between pages is implemented using BrowserRouter. This router sets the App component as the main entry point which enforces authentication and redirects to the LoginPage component if the user is not authenticated. Inside the App, the LoginPage or the Layout, which contains the navigation bar, and the respective subpage are rendered.

The StrictMode enables additional checks and warnings to catch potential issues in development.

In the LoginPage component, the MSAL instance is used to create the Microsoft Login window. The login with Microsoft authenticates and fetches an access token from Azure that will be saved in a React state using the library Zustand, which allows saving global states. This way, the token does not get lost when navigating between pages. This is necessary since this token must be sent to the backend on every API call to ensure authorisation.

The useAuthToken custom hook handles the automatic reauthentication to acquire a new token once the old one expires.

```
useEffect(() => {
  if (expiresOn) {
    const timeUntilExpiry = new Date(expiresOn).getTime() - Date.now();
    const refreshTime = timeUntilExpiry - 60000; //the time of 1 minute before the access token expires
    if (refreshTime > 0) {
      const timeout = setTimeout(getToken, refreshTime); //Call auth refresh 1 minute before expiration
      return () => clearTimeout(timeout); //Clear timeout on useeffect rerun to avoid multiple timeouts
    }
  }
}, [expiresOn]);
```

Figure 18 Code: Automatic Reauthentication

This is done by fetching the time until the token expires, subtracting one minute, and setting a timeout to call the getToken function, which initiates reauthentication before the old token expires. This way, the user remains logged in and it avoids setting long token expiry times which would impact security negatively.

On successful authentication, the /api/users/me endpoint of the API is called, which returns the user object. Based on this object, conditional renders are performed to only show the components the user is authorised for based on their group.



For API interaction, a generic API client was created that accepts a generic type T to specify different return types.

```
const BACKEND_URL = import.meta.env.VITE_BACKEND_URL;

const axiosInstance = axios.create({
  baseURL: BACKEND_URL,
});

You, last week | 1 author (You)
class APIClient<T> {
  endpoint: string;

  constructor(endpoint: string) {
    this.endpoint = endpoint;
  }

  get = async (accessToken: string, config: AxiosRequestConfig = {}) => {
    if (!accessToken) {
      return Promise.reject(new Error("User not authenticated."));
    }
    config.headers = {
      ...config.headers,
      Authorization: `Bearer ${accessToken}`,
    };
    return axiosInstance.get<T>(this.endpoint, config).then((res) => res.data);
  };
};
```

Figure 19 Code: Generic API Client

This API client offers several functions to execute operations, including get, getByID, getAll, getAllFiltered, initAiWorkflow, and mutate.

These are called in custom react query hooks that create an API client with the desired return type and endpoint, and then call the necessary function with React Query.

```
const ticketClient = new APIClient<TicketList>("/api/tickets");

const useFilteredTickets = (accessToken: string, filter?: TicketFilter) => {
  return useQuery<TicketList>({
    queryKey: ["tickets", filter],
    queryFn: () => ticketClient.getAllFiltered(filter || {}, accessToken),
    retry: 2,
    placeholderData: keepPreviousData,
  });
};

export default useFilteredTickets;
```

Figure 20 Code: Custom ReactQuery Hook

In this example, tickets are fetched from the /api/tickets endpoint. Since it requires filtering, the getAllFiltered function is called and the filter and the accessToken are passed. The filter variable as queryKey ensures the query is rerun every time a filter attribute changes to fetch the relevant data. Additionally, on error, the query will retry two more times. The placeholder data attribute allows the previous data to be displayed while refetching to ensure a smooth content update without a blank screen in between.

For the AI-Chat, the chat state is stored in a global Zustand store to preserve the chat while changing pages.

```
interface ChatStore {
  conversation: [string, string][];
  executionCount: number;
  ticket: boolean;
  ticketButton: boolean;
  workflowRequest: WorkflowRequest;
  setConversation: (conversation: [string, string][[]]) => void;
  setExecutionCount: (count: number) => void;
  setTicket: (ticket: boolean) => void;
  setTicketButton: (ticketButton: boolean) => void;
  setWorkflowRequest: (request: WorkflowRequest) => void;
  resetChat: () => void;
}
```

Figure 21 Code: ChatStore

This store offers several functions that can be called inside the chat component to set the states. Additionally, this state will be stored in the browser's local storage since states get lost on page reload or closing the browser. This state will be reinserted from the local storage into the application's Zustand store once the user logs back in. This is necessary since users might be advised by the AI to restart their computer to solve their issue. Therefore, the chat must be preserved, and the user must be able to continue with the conversation.

To enable a seamless experience, all update and insert operations were implemented as optimistic operations. This means, when writing a message, assigning or closing a ticket, the change will be set instantly within the frontend application to enable real-time feedback. Only in the unexpected case that the operation fails, these changes get rolled back and an error notification is displayed.

In the following example, when closing a ticket, the closeTicket function calls the React Query custom mutation hook to send this change to the backend. The setClosedDate sets a

local state that forces the component to re-render and display the date when the ticket was closed.

```
const handleCloseTicket = () => {
  closeTicket({ ticket_id: ticket.ticket_id, accessToken });
  setClosedDate(new Date());
};
```

Figure 22 Code: Close Ticket

Only when React Query returns the `closeTicketError`, it triggers this `useEffect`.

```
useEffect(() => {
  closeTicketError &&
  toaster.create({
    title: "Error",
    description: "Failed to close Ticket. Please try again.",
  });
  setClosedDate(ticket.closed_date);
}, [closeTicketError]);
```

Figure 23 Code: Display Error Message

It displays an error message to the user and updates the local state with the previous state of the ticket object. This again causes a re-render and displays the ticket as opened.

To provide structure, maintainability, and reusability, all components were built in a modular fashion.

```
interface TicketListProps {
  ticketList?: TicketList;
}

const TicketListContainer = ({ ticketList }: TicketListProps) => {
  return (
    <Grid
      gridTemplateColumns={{ base: "1fr", xl: "1fr 1fr" }}
      gap="2rem"
      mb="3rem"
    >
      {ticketList?.tickets?.map((ticket) => (
        <TicketListItem key={ticket.ticket_id} ticket={ticket} />
      ))}
    </Grid>
  );
};

export default TicketListContainer;
```

Figure 24 Code: TicketListContainer

This component for example, displays a list of tickets and is used on the Technician Portal, as well as on the MyTickets page. The TicketListItem is another subcomponent, which represents the individual tickets that are dynamically rendered in a Grid within the TicketListContainer.

#### 6.5.5 Configuration Management

The configuration of this application is managed in a JSON file, and its values are read on startup and runtime. It not only enables to change the LLM but allows for setting the logging level, choosing the embedding model, configuring the Milvus database, and setting the connection attributes for MySQL. The max\_count\_of\_troubleshootings sets a fixed number of iterations the workflow can perform to troubleshoot an issue. This is necessary since smaller models like Gemma 9b, which was used for development, sometimes fail to recognise when an issue is not troubleshootable. If more capable models are applied that reliably detect when a user is stuck, and is most likely not able to solve the problem based on the LLM's guidance, this value can be set to a higher number.

```
{
  "logging": {
    "logging_level": "INFO"
  },
  "ollama": {
    "llm": "gemma2:9b-instruct-q4_K_M",
    "embedding_model": "mxbai-embed-large"
  },
  "workflow": {
    "max_count_of_troubleshootings": 3
  },
  "milvus": {
    "host": "http://localhost:19530",
    "user": "aihelpdesk",
    "metric_type_rag": "COSINE",
    "index_type_rag": "AUTOINDEX",
    "metric_type_tickets": "COSINE",
    "index_type_tickets": "AUTOINDEX",
    "number_of_retrieved_documents": 3,
    "rag_documents_folder_absolute_path": "rag_documents"
  },
  "mysql": {
    "user": "aihelpdesk",
    "host": "127.0.0.1",
    "database": "db_ai_helpdesk"
  }
}
```

Figure 25 Code: Configuration File

Sensitive data like API keys and database credentials are stored in the .env file, and for deployment, they will be imported from the environment variables of the system.

```
# LangChain
LANGCHAIN_TRACING_V2=true
LANGCHAIN_ENDPOINT="https://api.smith.langchain.com"
LANGCHAIN_API_KEY="lsv2_pt_XXXXXXXXXXXXXXXXXX"
LANGCHAIN_PROJECT="langchain_test"
# Tavily Websearch
TAVILY_API_KEY="tvly-dev-XXXXXXXXXXXXXXXXXX"
#Milvus
MILVUS_PASSWORD="XXXXXXXXXXXXXXXXXX"
#Azure AD
TENANT_ID="7ff5cf6e-XXXXXXXXXXXXXXXXXX"
CLIENT_ID="07066dcd-XXXXXXXXXXXXXXXXXX"
USERS_GROUP_ID="be4c15ee-XXXXXXXXXXXXXXXXXX"
TECHNICIANS_GROUP_ID="909d8909-XXXXXXXXXXXXXXXXXX"
AZURE_CLIENT_SECRET="R8J8Q~4Fe-XXXXXXXXXXXXXXXXXX"
#MySQL
MYSQL_PASSWORD="XXXXXXXXXXXXXXXXXX"
#PyTest Testing Modus
TESTING=false
VALID_TICKET_ID=41
AZURE_USER_NAME="User.Foo@samueleras123gmail.onmicrosoft.com"
AZURE_USER_PASSWORD="XXXXXXXXXXXXXXXXXX"
AZURE_TECHNICIAN_NAME="Technician.Foo@samueleras123gmail.onmicrosoft.com"
AZURE_TECHNICIAN_PASSWORD="XXXXXXXXXXXXXXXXXX"
```

Figure 26 Code: Environment Variables

The environment variables contain keys and attributes to connect with LangSmith and Tavily for tracing the LLM's execution and executing the web search. Azure IDs are provided to enable authentication and fetching user details. Additionally, passwords for Milvus and MySQL are provided.

The testing variables are used to set the application into testing mode and to provide necessary information and credentials for testing. The testing mode allows to run unit tests without impacting production data since the application will roll back all database operations when set to true.

## 6.6 Integration and System Testing

Integration and system testing is necessary to ensure that the components are functional and work together as expected. Besides manual testing, a combination of unit tests and end-to-end testing was chosen to thoroughly investigate the system.

Testing the API endpoints and the associated backend logic was done using pytest, a test framework, which allows to write unit tests in Python.

As described in the previous section, the configuration of testing attributes must be set in the environment variables. These provide information such as user and technician credentials to automatically authenticate with Azure to carry out the test. The following code snippet is a generic test function for the AI workflow, which is used by multiple tests that enforce the testing of different parts of the workflow using different input variables and different user groups. It will call the workflow and verify that the status code and returned variables match the expected types.

```
def init_ai_workflow(request_data):  
  
    response = client.post(  
        "/api/init_ai_workflow",  
        headers={"Authorization": f"Bearer {AZURE_USER_TESTING_ACCESS_TOKEN}"},  
        json=request_data,  
    )  
  
    assert response.status_code == 200, f"Unexpected response: {response.json()}"  
  
    response_data = response.json()  
    assert isinstance(response_data, dict)  
  
    expected_keys = APIWorkflowResponse.__annotations__.keys()  
    assert all(  
        key in response_data for key in expected_keys  
    ), f"Missing keys in response: {response_data.keys()}"  
  
    assert isinstance(response_data["llm_output"], str | None)  
    assert isinstance(response_data["ticket"], bool | None)  
    assert isinstance(response_data["ticket_id"], int | None)
```

Figure 27 Code: AI Workflow Unit Test

To verify all functionalities, a total of 23 unit tests were written and executed, as shown in the output below.

```

tests/test_ai_workflow.py::test_troubleshooting_ai_workflow PASSED [ 4%]
tests/test_ai_workflow.py::test_offer_ticket_ai_workflow PASSED [ 8%]
tests/test_ai_workflow.py::test_further_questions_ai_workflow PASSED [ 13%]
tests/test_ai_workflow.py::test_ticketcreation_ai_workflow PASSED [ 17%]
tests/test_ai_workflow.py::test_ai_workflow_invalid_input PASSED [ 21%]
tests/test_assign_ticket.py::test_assign_ticket PASSED [ 26%]
tests/test_assign_ticket.py::test_assign_ticket_unauthorized PASSED [ 30%]
tests/test_assign_ticket.py::test_assign_ticket_invalid_input PASSED [ 34%]
tests/test_close_ticket.py::test_close_ticket_authorized PASSED [ 39%]
tests/test_close_ticket.py::test_close_ticket_unauthorized PASSED [ 43%]
tests/test_get_mytickets.py::test_get_mytickets PASSED [ 47%]
tests/test_get_mytickets.py::test_get_mytickets_invalid_input PASSED [ 52%]
tests/test_get_ticket.py::test_get_ticket_by_id PASSED [ 56%]
tests/test_get_tickets.py::test_get_tickets PASSED [ 60%]
tests/test_get_tickets.py::test_get_tickets_invalid_input PASSED [ 65%]
tests/test_get_tickets.py::test_get_tickets_unauthorized PASSED [ 69%]
tests/test_insert_ticket_message.py::test_insert_ticket_message PASSED [ 73%]
tests/test_insert_ticket_message.py::test_insert_ticket_message_invalid_data PASSED [ 78%]
tests/test_reopen_ticket.py::test_reopen_ticket_authorized PASSED [ 82%]
tests/test_reopen_ticket.py::test_reopen_ticket_unauthorized PASSED [ 86%]
tests/test_technicians.py::test_get_technicians PASSED [ 91%]
tests/test_technicians.py::test_get_technicians_unauthorized PASSED [ 95%]
tests/test_users_me.py::test_read_users_me PASSED [100%]

===== 23 passed in 65.09s (0:01:05) =====

```

Figure 28 Unit Test Results

To check whether the whole application works as expected, an end-to-end test with Cypress was written. Cypress is a test automation tool that interacts with the frontend based on a predefined sequence. This way, a test scenario was created that works through all features and verifies the received outcome. The image below shows an execution of the defined Cypress test.

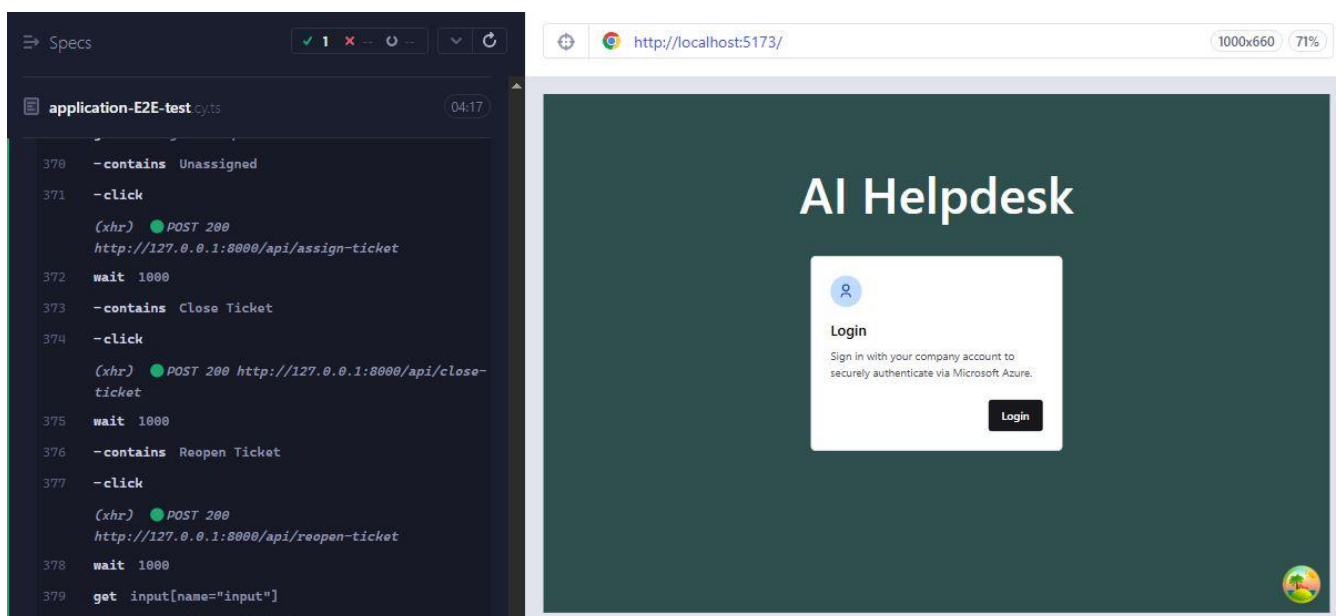


Figure 29 Cypress End to End Test



## 6.7 Performance Testing

A seamless, smoothly running frontend without loading delays as well as fast API interactions are crucial for a positive user experience. To assess these metrics, LangSmith was used to evaluate the AI workflow's execution speed, while the Chrome extension Lighthouse measured frontend responsiveness and API call timings.

The following screenshot shows the latency of a full execution of the AI workflow. From bottom to top, it shows the execution of three troubleshooting guide generations, followed by a ticket offer, asking further questions, and the final ticket generation.

✓	Name	Input	Output	Error	Start Time	Latency	Tokens
✓	LangGraph	human: Monitor: BENQ Se...	lenovo yoga 3...		20.3.2025, 17:23:45	⌚ 33.56s	8,616
✓	LangGraph	human: Please create a Ti...	monitor not w...		20.3.2025, 17:23:31	⌚ 10.76s	6,061
✓	LangGraph	human: None of these hel...	monitor not w...		20.3.2025, 17:23:26	⌚ 3.38s	5,056
✓	LangGraph	human: None of these hel...	monitor not w...		20.3.2025, 17:23:17	⌚ 7.42s	5,470
✓	LangGraph	human: None of these hel...	monitor not w...		20.3.2025, 17:23:09	⌚ 6.32s	4,715
✓	LangGraph	human: My monitor does ...	monitor not w...		20.3.2025, 17:22:59	⌚ 8.28s	3,710

Figure 30 LangSmith Performance Results

While for normal requests the execution times average below 10 seconds, the ticket generation can take 30 to 50 seconds. A detailed execution of the ticket generation is depicted below. Since the generation is split into four subparts including generating the issue description, possible solutions, ticket summary, and ticket title to improve generation quality, this takes a substantial amount of time.



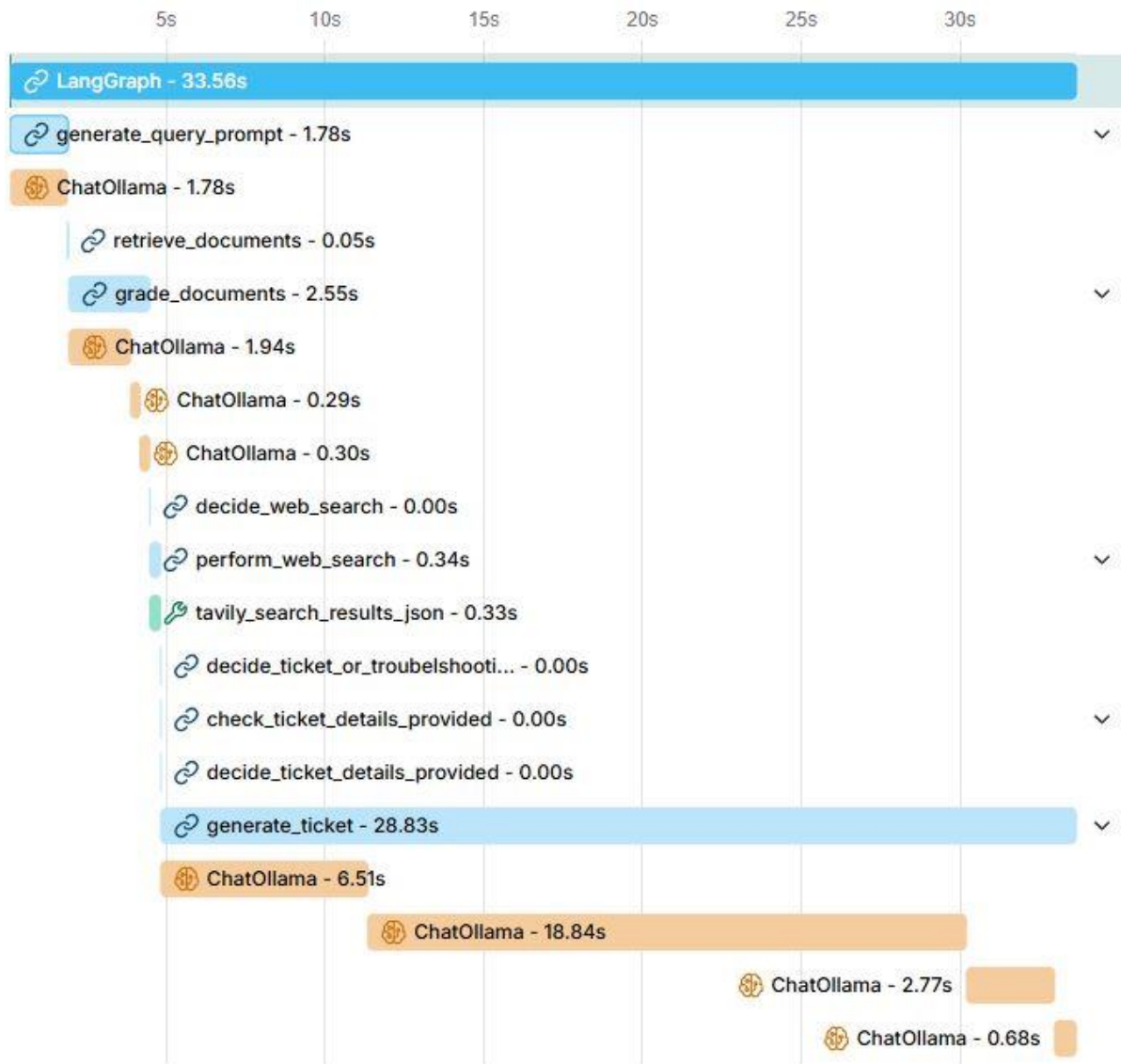


Figure 31 Ticket Generation Workflow

To address this on the frontend, users are shown a message informing them about the expected ticket generation time.

However, these times should be interpreted qualitatively, since they are completely dependent on the chosen model and the utilised hardware. The presented results were obtained by executing the gemma-2-9b-it model on an NVIDIA RTX 3060 GPU with 12 GB of VRAM.

The frontend performance was verified using Lighthouse. The following report shows the evaluated metrics including performance, accessibility, best practices, and search engine optimisation.

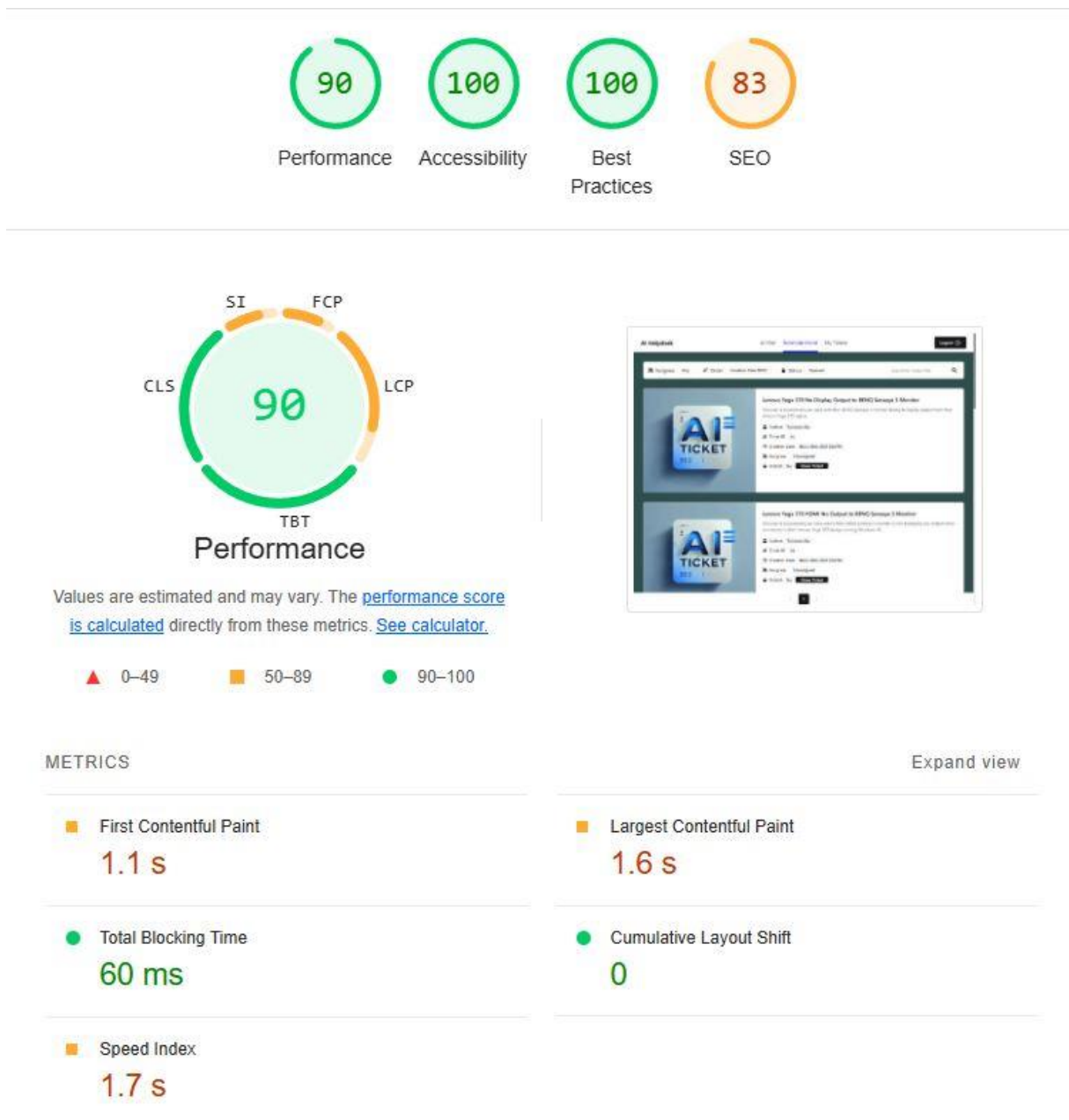


Figure 32 Lighthouse Performance Test Results

The report indicates a positive outcome, demonstrating fast performance and minimal blocking time. The initial load times are not fast, but this was expected and already covered as a drawback of single page applications in the Literature Review. However, once loaded, the application offers rapid page and content switches since React dynamically handles content changes without requiring a page reload.

Additional results analysing the loading times of the technician portal, ticket page, and ticket assignment or closure can be found in the appendix.

## 6.8 Security Testing

An application needs to be secured properly to provide confidentiality, integrity, and availability in order to mitigate potential threats or weaknesses. To ensure a high level of security, a variety of different tests were run to find and resolve vulnerabilities.

Authentication and role-based access control are essential to keep unauthorized users from accessing the system. The correct behaviour of the authentication was already checked within the previously discussed unit tests. Different users with different security groups were used to check whether they could access each API endpoint.

Another potential threat is deprecated libraries, packages, or dependencies that contain known vulnerabilities that could be exploited to gain unauthorized access to the application or its host server. To avoid this risk, all Python packages were checked using pip-audit. The check returned one issue, which was then eliminated by upgrading the dependency. For the frontend, all packages were scanned using npm audit. This check returned no vulnerabilities.

Finally, the application was scanned using Zed Attack Proxy (ZAP). This testing tool scanned the application for security gaps and returned multiple warnings in the output below.

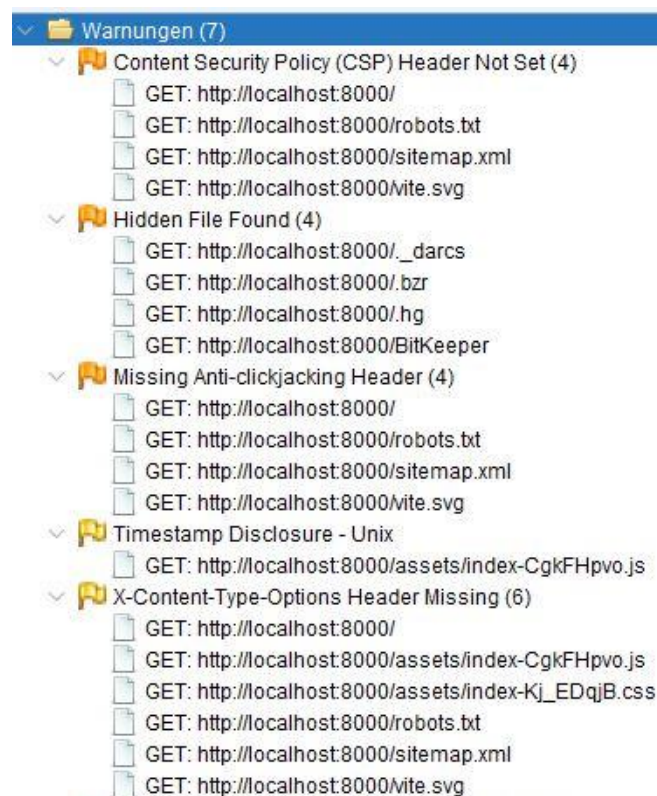


Figure 33 ZAP Warnings

These issues were resolved by adding middleware in FastAPI to add the Content Security Policy (CSP) and X-Content-Type-Options to the response header, and by blocking access to hidden files that are not supposed to be accessed.

## 6.9 User Acceptance Testing (UAT)

The User Acceptance Testing was conducted on the university's testing day. To evaluate the user's perspective and perception of the application's usability, accessibility, functionality, durability, and scalability, a testing scenario was created for the participants to go through.

This testing scenario begins with logging in and engaging in a conversation with the AI to ultimately generate a ticket. Thereafter, users are encouraged to add messages to the created ticket and try the assigning, closing, and reopening functionalities. Lastly, the My Tickets page and the Technician Portal are opened and checked for querying and filtering capabilities. The test run ends with logging out of the application.

After completing the testing scenario, participants were asked to fill out a questionnaire to record their impressions. These questionnaires, along with the participants' consent forms are included in the appendix.

While the number of testers was relatively small (7 participants), the evaluation of the questionnaires supported the outcomes of earlier tests. The users were satisfied with the application's usability, giving it an average score of 4.86 out of 5. Overall, the testers rated the troubleshooting as useful but mentioned its limitations when it comes to more complex issues, which aligns with expectations. All participants praised the ticket quality of generated tickets and confirmed that they believe this would help technicians to resolve issues more effectively. Accessibility, functionality, durability, and scalability also received high ratings, and no major issues were reported. One minor bug that occurred was incorrect ordering of tickets when filtered in ascending order by creation date. This bug was resolved after the testing session by adjusting the corresponding SQL query. Furthermore, participants criticised that the chat window does not automatically scroll to the latest message. Consequently, this feature was added to improve the application's intuitiveness.

Overall, the User Acceptance Testing was successful, confirming the application's effectiveness and quality, and helping identify and resolve small issues.

## 6.10 Bug Tracking and Resolution

Bug tracking has major importance in large-scale projects with multiple developers. Since this project is comparatively small in scope, with only one developer working on it, there was no need to apply a formal bug tracking tool like Jira. Instead, bugs were either fixed as soon as they were detected or written down in a personal bug log using a simple Word file including a description of the bug, along with the steps to reproduce it.

When a bug was identified, the first step was looking for error messages on the console to check for potential issues. When the bug occurred without throwing errors because of faulty logic in the code, the application logs were checked for more detailed information on the latest operations that could have led to the issue. The cause was further investigated by looking into the related components that were most likely responsible for the bug. If no obvious logic errors were found, further debugging was conducted by inserting temporary debug outputs. Additionally, for debugging on the frontend, Chrome DevTools were utilised for displaying console logs, inspecting cookies, local storage and session storage, and monitoring the network traffic between the frontend and backend.

When multiple bugs occurred at the same time, they were prioritised based on their impact. Critical issues affecting core features were addressed immediately, while design issues or edge cases were scheduled for later.

After implementing a fix, the system was retested to verify the bug was resolved successfully and that no new issues were introduced.

## 6.11 Conclusion

This chapter outlined the end-to-end implementation and testing process of the application based on a hybrid project management approach.

Each component of the system was implemented as specified in Chapter 5, *Design*, with respect to best practices. The use of modular, layered architecture, consistent coding standards, and modern tools including Visual Studio Code, Docker, MySQL Workbench, GitHub, and frameworks such as LangGraph, LangChain, FastAPI, and React with TypeScript ensured maintainability, scalability, and cross-platform compatibility. Integrating a dynamic AI workflow, secure backend, and a responsive frontend resulted in a complete and efficient system.

To ensure full functionality and to prevent errors or bugs, testing had major focus throughout and after development. Besides manual testing, unit testing with Pytest, end-to-end testing with Cypress, and performance and security assessments using tools like Lighthouse and ZAP helped verify the system's stability, speed, and safety. Additionally, User Acceptance Testing was conducted, and validated the application's quality with highly positive feedback.

Overall, the application was successfully implemented and thoroughly tested, meeting expectations in terms of functionality, usability, performance, and security.

## 7 Evaluation of the Product

### 7.1 Introduction

This chapter will evaluate the developed AI-Integrated IT Helpdesk Web Application, focusing on the defined requirements and quality metrics, including the product's effectiveness, usability, performance, maintainability, scalability, security, and cost-effectiveness. This evaluation will assess whether the set objectives have been achieved and will reflect whether the project outcomes comply with stakeholder expectations. Insights and feedback gained from various testing phases, including integration, performance, security, and user acceptance testing (UAT), will be discussed to investigate the application's strengths, weaknesses, and opportunities for future improvement.

### 7.2 System Functionality, Achievement of Goals and Impact

The primary goal of this project was to create an AI-powered solution that effectively enhances helpdesk efficiency to significantly reduce the technicians' workload and ultimately decreases costs. This goal was broken down into two objectives, which demanded utilising a Large Language Model and its extension with retrieval augmented generation. While both these objectives were fulfilled, the achievement of the goal to actually improve helpdesk efficiency cannot be directly proven, since the helpdesk has not been run in production yet. However, it can be assumed to be fulfilled since manual testing and the participants of the user acceptance test confirmed that the application is capable of solving simple issues and drastically improves ticket quality, which would ultimately conclude in improved efficiency and reduced workload.

All core functional requirements and their associated goals and objectives were met. Secure user registration was implemented using Azure, role-based access control (RBAC) was successfully implemented for each API endpoint, and user and ticket data was securely stored and managed leveraging a combination of Azure and MySQL.

Additionally, all non-functional requirements, including UI/UX and performance-driven user satisfaction, secure handling of user and business data, and the application's maintainability and customisability were successfully achieved and confirmed by rigorous integration, system, performance, security and user acceptance testing.

These tests validated the system's functionality and achievement of objectives and helped to further evaluate the application's performance and quality metrics which will now be discussed in detail.

### 7.3 Usability and User Satisfaction

The application delivered excellent usability and user satisfaction. It was developed as a Single Page Application (SPA) with React, which contributed to a seamless user experience by avoiding reloads and therefore limiting delays on navigation and context changes. Additionally, its minimalistic but modern design provided a pleasing UI that is easy to understand and navigate.

As a result, and as discussed in Chapter 6, *Implementation and Testing*, the user acceptance test (UAT) resulted in highly positive feedback, with usability achieving an average score of 4.86 out of 5. The participants highlighted the intuitive user interface, responsiveness, and ease of navigating through the AI troubleshooting and ticketing functionalities.

This confirms that the stakeholder expectation of users, who demanded a visually appealing but easy-to-use application, was successfully met.

### 7.4 Performance Evaluation

Performance testing demonstrated strong results in backend API calls as well as frontend interactions.

LangSmith was used to analyse the AI workflow's execution times and revealed that most requests were averaging a response time of 10 seconds, except for ticket generation tasks, which took 30 to 50 seconds due to their composition of multiple subprocesses for increased generation quality. Of course, AI tasks are resource intensive and time-consuming, which makes it essential to balance quality and speed. With this taken into account, the execution times cannot be objectively assessed, as there is insufficient comparative data to evaluate the complex relationship between quality and time. Subjectively, however, the observed timeframe appears reasonable and sufficiently fast.

Frontend performance was measured using the Chrome extension Lighthouse, which showed commendable results in performance, best practices and SEO. The slow initial load times, which are inherent to SPA frameworks, were the only noted performance issue, but were



within acceptable limits. After the initial load up, the interactions were smooth, with little to no blocking time or delays, and rapid page and content switching.

These results meet the expectations and verify the commendable performance of the developed application.

## 7.5 Maintainability and Customisability Evaluation

To ensure maintainability and customisability, several best practices were applied in the development process.

A well-defined layered architecture, which separates presentation, business logic, and data, ensured separation of concerns. The use of TypeScript, Python type annotations, and type-checking with Pydantic significantly enhanced code clarity and type safety, reducing potential runtime errors and increasing maintainability.

Additionally, tools for code formatting, naming conventions, inline comments, and logging and error handling mechanisms were leveraged to improve readability, understandability, and debuggability.

Customisability is given since this project is developed as an open-source project, which can be updated and extended to the deployers liking. However, even in its original form, the application can be amended by editing its configuration file to switch the LLM or embedding model, change logging behaviour, alter the AI workflow, or change the vector database's comparison metric.

Based on these discussed practices and configuration options, this project successfully meets the stakeholder expectations of administrators, who demanded a customisable and maintainable system.

## 7.6 Scalability Evaluation

The application's scalability is ensured by utilising FastAPI's asynchronous API endpoints to enable handling of simultaneous requests efficiently. By choosing this design, the application is capable of scaling horizontally by handling increased traffic without performance degradation. Additionally, support for future enhancements or scalability modifications without extensive codebase changes is ensured due to the modular approach chosen in the development process.

## 7.7 Security Assessment

To ensure the application's security, various tools and techniques were applied to detect and eliminate vulnerabilities. Authentication functionality was verified using Pytest, and dependency vulnerabilities were assessed utilising pip-audit for Python packages, and npm audit for JavaScript packages. Additionally, Zed Attack Proxy (ZAP) was used to identify security gaps, which were closed by leveraging middleware actions to enhance security by appending security headers to the API responses and restricting unauthorised file access.

These actions and test results validate the application's security and confirm the stakeholder requirements for administrators of a secure system have been met.

## 7.8 Impact and Cost-effectiveness

Similarly to the evaluation of the AI Helpdesk's ability to increase efficiency and reducing workload, the project's impact cannot be proven by direct numerical evaluation due to it not being run in production. However, when it is assumed that helpdesk efficiency is increased as discussed earlier, this would ultimately impact companies by obtaining immediate productivity gains, creating a more dynamic and satisfying work environment. Cost-effectiveness is directly influenced by cost savings on staff reductions, or potential increased revenue by reallocating employees towards innovation and value-creating activities.

Consequently, under these assumptions, the application yields high impact and the stakeholder expectation of company management to reduce operational costs can be considered met.

## 7.9 Development Effectiveness, Efficiency and Sustainability

Using a hybrid approach by combining Waterfall with Agile contributed significantly to the development efficiency and effectiveness. A well-defined architecture, detailed system design, and consistent use of best practices including formatting, naming conventions, and typing tools, streamlined the development process. These practices directly impacted the project's sustainability by providing modular, readable, and maintainable code, enabling the project to adapt and evolve according to future needs. Choosing modern and scalable technologies like FastAPI further supported sustainable long-term growth and integration of additional functionalities.

Despite careful planning, some unforeseen challenges arose during the development process. These are listed in the following section as lessons learned.

## 7.10 Lessons Learned

Several valuable lessons emerged during the project:

- The use of unfamiliar technology can impact both the project timeline and the quality of the final product.
- Design concepts may not work as expected, leading to delays and rework.
- Complex features can introduce challenges that impact the entire project workflow.

These lessons will be discussed in more detail as part of the critical reflection in the next chapter.

## 7.11 Limitations of the Evaluation

While comprehensive, this evaluation faced some constraints, including:

- The user acceptance testing (UAT) featured a relatively small sample (7 participants), which limits the generalizability of findings. For broader and more reliable insights, tests should involve larger groups of participants.
- Performance results are presented in absolute numbers. However, the results depend heavily on hardware capabilities and the parameter size of the leveraged Large Language Model (LLM). While this project employed an NVIDIA RTX 3060 GPU to generate outputs with Google's gemma-2-9b-it model, using different compositions would affect execution times and output quality, leading to major variability in results.

## 7.12 Recommendations for Future Improvements

The application would benefit from the addition of several new features that were out of scope due to the limited timeline of this project. These include:

- Automated categorisation and severity assignment for tickets using AI
- LLM training or fine-tuning based on resolved tickets to improve accuracy and relevance over time
- Analytics to present helpdesk performance through numbers and charts

- E-mail notifications for ticket updates
- Editing of tickets and messages
- Manual ticket creation functionality for technicians without relying on the AI workflow
- File upload functionality in tickets to share images or videos

### 7.13 Conclusion

The evaluation confirmed that the AI Helpdesk is an effective, usable, performant, maintainable, scalable, secure, cost-effective, and sustainable application. The project satisfied stakeholder requirements by meeting its core objectives, improving efficiency and reducing operational costs, while ensuring high user satisfaction. Future improvements were outlined, focusing on further AI integrations, enhancing user interaction and flexibility within the ticketing system.

## 8 Critical Evaluation/Reflection

### 8.1 Introduction

In this chapter, I critically reflect and evaluate my experience during development of the AI-Integrated IT Helpdesk Web Application, using Gibbs' Reflective Cycle. In this reflection, I will provide a detailed analysis of my thoughts, actions, and assumptions throughout the project. Additionally, my experiences and outcomes will be linked back to my initial specification to assess whether the initially declared objectives were achieved.

### 8.2 Gibbs' Reflective Cycle

#### 8.2.1 Description

In this project, I adopted a hybrid methodology combining Waterfall and Agile. I chose this approach to benefit from the Waterfall's structure and predictability while maintaining flexibility and applying iterative testing to mitigate risks and enhance efficiency. However, when I applied this method in development, several scenarios occurred that challenged my initial assumptions and plans.

#### 8.2.2 Feelings

At the beginning, I felt really confident about the choice of my methodology. I believed the hybrid approach would streamline development and prevent significant setbacks. However, my confidence was soon tested when unexpected complications arose. Particularly, when integrating Milvus as vector database, Azure for authentication, and implementing the AI workflow using LangChain and LangGraph, I encountered challenges that made me concerned about meeting the deadline. Although I managed to stay on schedule, I remained worried that additional unforeseen issues could still emerge.

#### 8.2.3 Evaluation

Integrating Milvus took significantly longer than expected, which I compensated for by extending working hours to remain on schedule. This made me realise that my initial research was insufficient because I massively underestimated the technology's complexity.

Similarly, using Azure for authentication introduced major challenges in unexpected ways. It not only impacted the implementation of the authentication process itself but complicated the testing phase due to unit tests having to authenticate with Azure to perform API requests.

Additionally, my initial design of the AI workflow fell massively short of expectations in terms of troubleshooting capabilities and generation quality. This came down to the fact that I significantly overestimated the performance of smaller Large Language Models. Due to this insufficient performance, my workflow required multiple redesigns and extensive adjustments. To address the performance limitations, I upgraded my hardware from an NVIDIA GTX 1060 to a more powerful NVIDIA RTX 3060 GPU. This upgrade provided additional computational power and VRAM, which ultimately allowed me to run Google's gemma-2-9b-it model. While this LLM outperformed previously tested models, it still proved insufficient for my current design and required further workflow improvements to achieve acceptable results. These implications drastically impacted the project timeline.

Despite these challenges, the hybrid model also proved its strengths. The iterative testing allowed for early identification of issues, which significantly reduced delays in later stages. While this testing proved valuable for reaching product quality goals, it also added considerable workload, especially during frequent design adjustments.

#### 8.2.4 Analysis

To learn from these challenges, the experience must be reflected on and boiled down to the underlying issue. These challenges occurred in very specific scenarios with specific technologies that are unlikely to reoccur in the exact same way. Therefore, the generic underlying issue must be derived in order to discover a solution that is applicable to similar situations in the future.

Reflecting deeply on these events, I discovered that the main issues were:

- The use of unfamiliar technologies. This results in significant risks to the project timeline and can compromise the quality of the product if complexity and the required learning curve are underestimated. Early prototyping and even more in-depth research on unfamiliar technologies could be employed to detect potential implementation difficulties early, to prevent delays and integration challenges.
- Integrating complex features may impact the whole project. Features may not only impact the implementation complexity of the specific component but may also affect the whole project workflow and testing phases, introducing unforeseen challenges and delays into the development process. This could be addressed by checking for possible implications of new features and allocating time buffers accordingly.

- Initial design concepts may not always perform as expected and may need multiple design iterations. This can be caused by several factors, including insufficient research, flawed assumptions, or other limitations like hardware restrictions, and can introduce massive timeline delays. Especially difficult to accurately estimate is the time required for tasks with quality-dependent requirements. Early prototyping and allocating additional time buffers for potential redesign iterations could mitigate this issue in the future.

Overall, despite the outlined complications, all objectives set out in the project specification were successfully achieved within the defined timeframe. A well-informed technology stack, effective AI integration, robust backend and database design, and a responsive, user-friendly frontend helped to meet requirements and integrated into a fully functional web application. The comprehensive testing phase confirmed the project's success.

#### 8.2.5 Conclusion

From this reflection, I learned that relying on unfamiliar technologies, integrating complex features without checking for potential implications, and being overly optimistic about early technical designs can significantly impact project timelines and workload. Managing these risks at an early stage, by conducting deeper preliminary research and prototyping to allocate time buffers and specify hardware requirements based on the findings, is essential for deadline adherence.

However, the hybrid methodology itself is an effective approach, providing both structure and flexibility to adapt to unexpected situations, and it excels even more when avoiding the outlined mistakes. The methodology helped me to successfully achieve all objectives and turned this project into a success.

#### 8.2.6 Action Plan

For future projects, I plan to invest even more time into the initial research and preliminary testing, including performance benchmarks of hardware and software. While this will take up more time upfront, it will result in a more profound design and realistic time schedule, which will ultimately reduce unexpected scenarios and delays. I will adopt prototyping as standard practice before full-scale implementation to detect design flaws, and to become familiar with the technologies and their challenges, limitations, and potential influence on the rest of the system.

### 8.3 Conclusion

In conclusion, reflecting on my experiences during the project, I gained significant insight and understanding into project management practices. While some insights confirmed my assumptions, others challenged my beliefs. This chapter underlined the importance of comprehensive research, early prototyping, realistic assumptions, and correctly utilising time buffers. By reflecting and evaluating the successes and difficulties, I learned to focus on making better-informed decisions to manage future projects more effectively.



## 9 Conclusions

This conclusion chapter discusses the project's overall achievements and highlights the significant outcomes. It summarises key insights, evaluates the achievement of objectives, discusses the project's relevance, provides recommendations for future research and development, and concludes with reflective closing remarks.

### 9.1 Summary of Findings

This project developed an AI-Integrated IT Helpdesk Web Application by leveraging a Large Language Model (LLM) to automate ticket generation and resolution. The Literature Review revealed insights into the significance of IT Helpdesks for successful incident management and discussed how AI can be used to automate processes to boost efficiency and cost savings. The developed system successfully implements these findings and offers an advanced workflow to solve issues or generate tickets on behalf of the user. Furthermore, the system utilised Retrieval Augmented Generation (RAG) and web search to provide additional information to the LLM, boosting its troubleshooting and generation capabilities, to enhance helpdesk performance. The finished product offers a substantial reduction in workload, increased ticket quality, and improved user satisfaction, which was validated by User Acceptance Testing (UAT). Additionally, several automated tests demonstrated its scalability, maintainability, performance, and security.

### 9.2 Achievement of Objectives

All initially defined goals and objectives as outlined in the project specification were successfully met. The project began with a comprehensive literature review and developed the AI-driven system utilising a Large Language Model with Retrieval Augmented Generation for automated ticket handling. User and ticket data was stored securely by leveraging MySQL and Azure, while the backend was implemented with FastAPI, managing frontend requests, authentication, and database interactions. A responsive frontend was built using React and Chakra. Finally, all components were tested, and the project was documented thoroughly in this report.

To achieve all this, the specification's primary goals were broken down into more tangible and concrete sub-goals and objectives in the design part. These objectives were successfully

met, which ultimately concluded in the project's success, delivering a practical, secure, and efficient AI-driven helpdesk solution.

### 9.3 Significance of the Project

The project's significance lies in improving and simplifying IT Service Management by developing a practical approach to leveraging AI to automate helpdesk applications. The developed system demonstrates its impact by streamlining service operations using automation and intelligent ticket handling, ultimately leading to workload reductions and lower operational costs. This project provides significant value, particularly to companies that prioritise data sovereignty and privacy. By promoting local AI deployment and providing an open-source codebase, companies can use this project as a template to build customised helpdesk solutions, tailored to their requirements.

This work also contributes to the academic understanding of practical AI applications. The project focused on utilising LLM-directed workflows to handle complex problems. Special focus was placed on splitting large tasks into sub-problems and leveraging techniques such as Retrieval Augmented Generation (RAG) to enable efficient use of smaller models while maintaining quality. This approach stands in contrast to current trends of relying on increasingly larger models and instead promotes resource-efficient solutions to reduce hardware requirements, energy consumption, and costs.

### 9.4 Recommendations for Future Work

As discussed in Chapter 7, *Evaluation of the Product*, several enhancements have been proposed to improve the helpdesk's efficiency, performance, and usability. These include integrating intelligent ticket categorisation and assignment, LLM fine-tuning, performance monitoring, and other recommendations to improve usability, such as notifications, message and ticket editing functionality, enabling manual ticket creation possibilities for technicians and supporting media attachments. Together, these features aim to provide a more adaptive, efficient, and user-friendly platform.

For future researchers, there is significant potential to explore advanced fine-tuning strategies that could be employed to tailor an LLM towards a company's requirements based on historical ticket data and analysis of past interactions. Furthermore, performance evaluation of AI-directed workflows in comparison to fully agentic approaches could provide insights

into their trade-offs. In this project, using autonomous agentic approaches with small models was discovered to be highly unreliable. Research could assess whether an agentic approach using larger models, where the LLM is given full control, truly offers more performance, while also accounting for potential drawbacks such as increased energy consumption and hardware requirements. It would also be valuable to identify specific use cases where one approach clearly outperforms the other.

## 9.5 Closing Remarks

In conclusion, this project significantly contributed to advancements in applying artificial intelligence within IT Service Management. It developed an open-source AI-integrated IT Helpdesk Web Application that addresses operational inefficiencies, provides substantial economic benefits, and significantly enhances user satisfaction. The outcomes highlight the potential and impact of AI-driven systems in practical, everyday business operations. This journey has been both challenging and rewarding and laid the groundwork for future improvements and research.

Finally, I want to express gratitude to my supervisors, the lecturers of the project module, and the library staff for their support, encouragement, and insightful feedback throughout this academic journey.

## 10 References

- [1] T. BS, *Practical IT Service Management: A Concise Guide for Busy Executives*. Ely, UNITED KINGDOM: IT Governance Ltd, 2014. Accessed: Nov. 13, 2024. [Online]. Available: <http://ebookcentral.proquest.com/lib/glyndwr-ebooks/detail.action?docID=1778757>
- [2] AXELOS, *ITIL foundation: ITIL 4 edition*, First edition. Norwich: TSO (The Stationery Office), 2019.
- [3] M. F. Bosu, D. Abuaiadah, P. Khanna, S. Nepia, and D. Palmer, 'Evaluation of IT Service Desk: A Case Study', 2019.
- [4] *Service Desk and Incident Manager : Careers in IT Service Management*. Accessed: Nov. 12, 2024. [Online]. Available: <https://ebookcentral.proquest.com/lib/glyndwr-ebooks/detail.action?pq-origsite=primo&docID=1713961>
- [5] 'ITIL Adoption Unlikely to Show Further Significant Growth', Avasant. Accessed: Nov. 14, 2024. [Online]. Available: <https://avasant.com/report/itil-adoption-unlikely-to-show-further-significant-growth/>
- [6] M. Marrone, F. Gacenga, A. Cater-Steel, and L. Kolbe, 'IT Service Management: A Cross-national Study of ITIL Adoption', *CAIS*, vol. 34, 2014, doi: 10.17705/1CAIS.03449.
- [7] A. Limited, *ITIL 4: Create, Deliver and Support Reference and Study Guide*. London, UNITED KINGDOM: The Stationery Office Ltd, 2021. Accessed: Nov. 14, 2024. [Online]. Available: <http://ebookcentral.proquest.com/lib/glyndwr-ebooks/detail.action?docID=6720545>
- [8] L. Lema, J. Calvo-Manzano, R. Colomo-Palacios, and M. Arcilla, 'ITIL in small to medium-sized enterprises software companies: towards an implementation sequence', *J Software Evolu Process*, vol. 27, no. 8, pp. 528–538, Aug. 2015, doi: 10.1002/smr.1727.
- [9] M. Steehouder, 'How Helpdesk Agents Help Clients', in *2007 IEEE International Professional Communication Conference*, Oct. 2007, pp. 1–9. doi: 10.1109/IPCC.2007.4464071.
- [10] A. G. Woodside, L. L. Frey, and R. T. Daly, 'Linking service quality, customer satisfaction, and behavioral intention', *J Health Care Mark*, vol. 9, no. 4, pp. 5–17, Dec. 1989.
- [11] I. Pogarcic, S. Raspor Jankovic, and R. Seturidze, 'How Does the Help Desk Quality Improve Customer Satisfaction?', *Athens Journal of Mass Media and Communications (online)*, vol. 3, no. 4, pp. 343–362, 2017, doi: 10.30958/ajmmc/3.4.4.
- [12] P. Bober, 'Simulation for IT Service Desk Improvement', *QIP Journal*, vol. 18, no. 1, Art. no. 1, Jul. 2014, doi: 10.12776/qip.v18i1.343.
- [13] A. Revina, K. Buza, and V. G. Meister, 'IT Ticket Classification: The Simpler, the Better', *IEEE Access*, vol. 8, pp. 193380–193395, 2020, doi: 10.1109/ACCESS.2020.3032840.
- [14] D. Dlodlo and K. Sibanda, 'Automated Ticket Classification for Information Technology Helpdesks using Machine Learning', in *2023 2nd Zimbabwe Conference of Information and Communication Technologies (ZCICT)*, Nov. 2023, pp. 1–7. doi: 10.1109/ZCICT59466.2023.10528578.
- [15] F. Al-Hawari and H. Barham, 'A machine learning based help desk system for IT service management', *Journal of King Saud University. Computer and information sciences*, vol. 33, no. 6, pp. 702–718, 2021, doi: 10.1016/j.jksuci.2019.04.001.

- [16] D. Wang, T. Li, S. Zhu, and Y. Gong, 'iHelp: An intelligent online helpdesk system', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 1, pp. 173–182, 2010.
- [17] S. Tuffery, *Deep Learning: From Big Data to Artificial Intelligence with R*. Newark, UNITED KINGDOM: John Wiley & Sons, Incorporated, 2022. Accessed: Dec. 01, 2024. [Online]. Available: <http://ebookcentral.proquest.com/lib/glyndwr-ebooks/detail.action?docID=7133414>
- [18] N. Gupta and R. Mangla, *Artificial Intelligence Basics: A Self-Teaching Introduction*. Bloomfield, UNITED STATES: Mercury Learning & Information, 2020. Accessed: Dec. 01, 2024. [Online]. Available: <http://ebookcentral.proquest.com/lib/glyndwr-ebooks/detail.action?docID=6128252>
- [19] Y. Qin, Z. Xu, X. Wang, and M. Skare, 'Artificial Intelligence and Economic Development: An Evolutionary Investigation and Systematic Review', *J Knowl Econ*, vol. 15, no. 1, pp. 1736–1770, Mar. 2024, doi: 10.1007/s13132-023-01183-2.
- [20] K. Siau and W. Wang, 'Artificial Intelligence (AI) Ethics: Ethics of AI and Ethical AI', *Journal of Database Management*, vol. 31, no. 2, pp. 74–87, Apr. 2020, doi: 10.4018/JDM.2020040105.
- [21] D. Hendrycks, M. Mazeika, and T. Woodside, 'An Overview of Catastrophic AI Risks', Oct. 09, 2023, *arXiv*: arXiv:2306.12001. doi: 10.48550/arXiv.2306.12001.
- [22] 'A pro-innovation approach to AI regulation', GOV.UK. Accessed: Dec. 04, 2024. [Online]. Available: <https://www.gov.uk/government/publications/ai-regulation-a-pro-innovation-approach/white-paper>
- [23] C. Janiesch, P. Zschech, and K. Heinrich, 'Machine learning and deep learning', *Electron Markets*, vol. 31, no. 3, pp. 685–695, Sep. 2021, doi: 10.1007/s12525-021-00475-2.
- [24] S. Walczak and N. Cerpa, 'Artificial Neural Networks'.
- [25] K. Sharifani and M. Amini, 'Machine Learning and Deep Learning: A Review of Methods and Applications', 2023, *Social Science Research Network, Rochester, NY*: 4458723. Accessed: Nov. 22, 2024. [Online]. Available: <https://papers.ssrn.com/abstract=4458723>
- [26] D. Khurana, A. Koli, K. Khatter, and S. Singh, 'Natural language processing: state of the art, current trends and challenges', *Multimed Tools Appl*, vol. 82, no. 3, pp. 3713–3744, Jan. 2023, doi: 10.1007/s11042-022-13428-4.
- [27] H. Naveed *et al.*, 'A Comprehensive Overview of Large Language Models', Oct. 17, 2024, *arXiv*: arXiv:2307.06435. doi: 10.48550/arXiv.2307.06435.
- [28] Y. Chang *et al.*, 'A Survey on Evaluation of Large Language Models', *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 3, pp. 1–45, Jun. 2024, doi: 10.1145/3641289.
- [29] L. Chen and G. Varoquaux, 'What is the Role of Small Models in the LLM Era: A Survey', Sep. 30, 2024, *arXiv*: arXiv:2409.06857. doi: 10.48550/arXiv.2409.06857.
- [30] Y. Gao *et al.*, 'Retrieval-Augmented Generation for Large Language Models: A Survey', Mar. 27, 2024, *arXiv*: arXiv:2312.10997. doi: 10.48550/arXiv.2312.10997.
- [31] T. Shi *et al.*, 'Preliminary Study on Incremental Learning for Large Language Model-based Recommender Systems', Jul. 30, 2024, *arXiv*: arXiv:2312.15599. doi: 10.48550/arXiv.2312.15599.
- [32] T. Wu, L. Luo, Y.-F. Li, S. Pan, T.-T. Vu, and G. Haffari, 'Continual Learning for Large Language Models: A Survey', Feb. 07, 2024, *arXiv*: arXiv:2402.01364. doi: 10.48550/arXiv.2402.01364.
- [33] H. Shi *et al.*, 'Continual Learning of Large Language Models: A Comprehensive Survey', Jun. 30, 2024, *arXiv*: arXiv:2404.16789. doi: 10.48550/arXiv.2404.16789.

- [34] J. Dodgson *et al.*, ‘Establishing Performance Baselines in Fine-Tuning, Retrieval-Augmented Generation and Soft-Prompting for Non-Specialist LLM Users’, Mar. 19, 2024, *arXiv*: arXiv:2311.05903. doi: 10.48550/arXiv.2311.05903.
- [35] M. R. J. K. VM, H. Warriar, and Y. Gupta, ‘Fine Tuning LLM for Enterprise: Practical Guidelines and Recommendations’, Mar. 23, 2024, *arXiv*: arXiv:2404.10779. doi: 10.48550/arXiv.2404.10779.
- [36] C. Jeong, ‘A Study on the Implementation Method of an Agent-Based Advanced RAG System Using Graph’, Sep. 13, 2024, *arXiv*: arXiv:2407.19994. Accessed: Nov. 01, 2024. [Online]. Available: <http://arxiv.org/abs/2407.19994>
- [37] S. Zhao, Y. Yang, Z. Wang, Z. He, L. K. Qiu, and L. Qiu, ‘Retrieval Augmented Generation (RAG) and Beyond: A Comprehensive Survey on How to Make your LLMs use External Data More Wisely’, Sep. 23, 2024, *arXiv*: arXiv:2409.14924. doi: 10.48550/arXiv.2409.14924.
- [38] P. Lewis *et al.*, ‘Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks’, in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 9459–9474. Accessed: Dec. 01, 2024. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>
- [39] ‘Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast’, *ResearchGate*, Oct. 2024, doi: 10.59287/icaens.1127.
- [40] C. Jeong, ‘A Study on the Implementation Method of an Agent-Based Advanced RAG System Using Graph’, Sep. 13, 2024, *arXiv*: arXiv:2407.19994. doi: 10.48550/arXiv.2407.19994.
- [41] X. Li, S. Wang, S. Zeng, Y. Wu, and Y. Yang, ‘A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges’, *Vicinagearth*, vol. 1, no. 1, p. 9, Oct. 2024, doi: 10.1007/s44336-024-00009-2.
- [42] Y. Wu, T. Yue, S. Zhang, C. Wang, and Q. Wu, ‘StateFlow: Enhancing LLM Task-Solving through State-Driven Workflows’, Sep. 14, 2024, *arXiv*: arXiv:2403.11322. Accessed: Nov. 01, 2024. [Online]. Available: <http://arxiv.org/abs/2403.11322>
- [43] M. Grunde-McLaughlin, M. S. Lam, R. Krishna, D. S. Weld, and J. Heer, ‘Designing LLM Chains by Adapting Techniques from Crowdsourcing Workflows’, May 06, 2024, *arXiv*: arXiv:2312.11681. doi: 10.48550/arXiv.2312.11681.
- [44] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, and D. Gosain, ‘A Survey and Comparison of Relational and Non-Relational Database’, *International Journal of Engineering Research*, vol. 1, no. 6, 2012.
- [45] T. Connolly and S. Navathe, *Database systems: a practical approach to design, implementation and management*, Global Edition., vol. 7ed Harlow. United Kingdom: Pearson, 2016.
- [46] C. Gyorödi, R. Gyorödi, and R. Sotoc, ‘A Comparative Study of Relational and Non-Relational Database Models in a Web- Based Application’, *ijacsa*, vol. 6, no. 11, 2015, doi: 10.14569/IJACSA.2015.061111.
- [47] C. Gyorodi, R. Gyorodi, G. Pecherle, and A. Olah, ‘A comparative study: MongoDB vs. MySQL’, in *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, Oradea, Romania: IEEE, Jun. 2015, pp. 1–6. doi: 10.1109/EMES.2015.7158433.
- [48] I. H. Madurapperuma, M. S. Shafana, and M. J. A. Sabani, ‘State-of-Art Frameworks for Front-end and Back-end Web Development’, 2022.
- [49] M. Kaluža, M. Kalanj, and B. Vukelić, ‘A comparison of back-end frameworks for web application development’, *Zb. Veleuč. Rij. (Online)*, vol. 7, no. 1, pp. 317–332, 2019, doi: 10.31784/zvr.7.1.10.

- [50] A. Adhikari, 'Full Stack JavaScript: Web Application Development with MEAN'.
- [51] 'API Features Individualizing of Web Services: REST and SOAP', *ResearchGate*, Oct. 2024, doi: 10.35940/ijitee.I1107.0789S19.
- [52] S. Patni, *Pro RESTful APIs*. Berkeley, CA: Apress, 2017. doi: 10.1007/978-1-4842-2665-0.
- [53] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, 'RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions', *Applied Sciences*, vol. 12, no. 9, Art. no. 9, Jan. 2022, doi: 10.3390/app12094369.
- [54] J. Shetty, D. Dash, and A. K. Joish, 'Review Paper on Web Frameworks, Databases and Web Stacks', vol. 07, no. 04, 2020.
- [55] M. F. S. Lazuardy and D. Anggraini, 'Modern Front End Web Architectures with React.Js and Next.Js', vol. 7, no. 1.
- [56] D. Dinh and Z. Wang, 'MODERN FRONT-END WEB DEVELOPMENT'.
- [57] A. Ranjan, A. Sinha, and R. Battewad, *JavaScript for Modern Web Development: Building a Web Application Using HTML, CSS, and JavaScript*. BPB Publications, 2020.
- [58] J. Bogner and M. Merkel, 'To Type or Not to Type? A Systematic Comparison of the Software Quality of JavaScript and TypeScript Applications on GitHub', in *Proceedings of the 19th International Conference on Mining Software Repositories*, May 2022, pp. 658–669. doi: 10.1145/3524842.3528454.
- [59] G. Bierman, M. Abadi, and M. Torgersen, 'Understanding TypeScript', in *ECOOP 2014 – Object-Oriented Programming*, R. Jones, Ed., Berlin, Heidelberg: Springer, 2014, pp. 257–281. doi: 10.1007/978-3-662-44202-9\_11.
- [60] Z. Gao, C. Bird, and E. T. Barr, 'To Type or Not to Type: Quantifying Detectable Bugs in JavaScript', in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, Buenos Aires: IEEE, May 2017, pp. 758–769. doi: 10.1109/ICSE.2017.75.
- [61] N. Hamidli, 'Introduction to UI/UX Design: Key Concepts and Principles'.
- [62] R. Vyas, 'Comparative Analysis on Front-End Frameworks for Web Applications', *IJRASET*, vol. 10, no. 7, pp. 298–307, Jul. 2022, doi: 10.22214/ijraset.2022.45260.
- [63] K. Phan, 'A Comprehensive Study on Single-Page Applications: Pros, Cons, and Practical Guidelines'.
- [64] S. Balaji, 'WATEERFALL Vs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC', . *Vol.*, no. 1, 2012.
- [65] K. Petersen, C. Wohlin, and D. Baca, 'The Waterfall Model in Large-Scale Development', in *Product-Focused Software Process Improvement*, vol. 32, F. Bomarius, M. Oivo, P. Jaring, and P. Abrahamsson, Eds., in *Lecture Notes in Business Information Processing*, vol. 32. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 386–400. doi: 10.1007/978-3-642-02152-7\_29.
- [66] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, 'Agile Software Development Methods: Review and Analysis', Sep. 25, 2017, *arXiv*: arXiv:1709.08439. doi: 10.48550/arXiv.1709.08439.
- [67] S. Alsaqqa, S. Sawalha, and H. Abdel-Nabi, 'Agile Software Development: Methodologies and Trends', *Int. J. Interact. Mob. Technol.*, vol. 14, no. 11, p. 246, Jul. 2020, doi: 10.3991/ijim.v14i11.13269.
- [68] K. Iwai, K. Iida, M. Akiyoshi, and N. Komoda, 'A help desk support system with filtering and reusing e-mails', in *2010 8th IEEE International Conference on Industrial Informatics*, Jul. 2010, pp. 321–325. doi: 10.1109/INDIN.2010.5549401.
- [69] 'Open LLM Leaderboard - a Hugging Face Space by open-llm-leaderboard'. Accessed: Jan. 14, 2025. [Online]. Available: [https://huggingface.co/spaces/open-llm-leaderboard/open\\_llm\\_leaderboard](https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard)

- [70] X. Xie, H. Liu, W. Hou, and H. Huang, ‘A Brief Survey of Vector Databases’, in *2023 9th International Conference on Big Data and Information Analytics (BigDIA)*, Dec. 2023, pp. 364–371. doi: 10.1109/BigDIA60676.2023.10429609.
- [71] ‘Performance - Chroma Docs’. Accessed: Jan. 28, 2025. [Online]. Available: <https://docs.trychroma.com/production/administration/performance>
- [72] J. Wang *et al.*, ‘Milvus: A Purpose-Built Vector Data Management System’, in *Proceedings of the 2021 International Conference on Management of Data*, in SIGMOD ’21. New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 2614–2627. doi: 10.1145/3448016.3457550.
- [73] ‘Introduction | LangChain’. Accessed: Jan. 15, 2025. [Online]. Available: <https://python.langchain.com/docs/introduction/>
- [74] ‘LangGraph’. Accessed: Jan. 15, 2025. [Online]. Available: <https://langchain-ai.github.io/langgraph/>
- [75] ‘ChatOllama | LangChain’. Accessed: Jan. 15, 2025. [Online]. Available: <https://python.langchain.com/docs/integrations/chat/ollama/>
- [76] ‘Get started with LangSmith | LangSmith’. Accessed: Jan. 15, 2025. [Online]. Available: <https://docs.smith.langchain.com/>
- [77] *langchain: Building applications with LLMs through composability*. Python. Accessed: Jan. 28, 2025. [Online]. Available: <https://github.com/langchain-ai/langchain>
- [78] ‘langchain npm’, npm. Accessed: Jan. 28, 2025. [Online]. Available: <https://www.npmjs.com/package/langchain>
- [79] ‘2025’s Top 10 Python Web Frameworks Compared’, DEV Community. Accessed: Feb. 05, 2025. [Online]. Available: <https://dev.to/leapcell/top-10-python-web-frameworks-compared-3o82>
- [80] ‘Top Python Web Development Frameworks in 2025 · Reflex Blog’. Accessed: Feb. 05, 2025. [Online]. Available: <https://reflex.dev/blog/2024-12-20-python-comparison/>
- [81] S. Emanuilov, ‘Django vs FastAPI in 2024’, Medium. Accessed: Feb. 05, 2025. [Online]. Available: <https://medium.com/@simeon.emanuilov/django-vs-fastapi-in-2024-f0e0b8087490>
- [82] ‘Django vs FastAPI: Which is the Best Python Web Framework? | The PyCharm Blog’, The JetBrains Blog. Accessed: Feb. 05, 2025. [Online]. Available: <https://blog.jetbrains.com/pycharm/2023/12/django-vs-fastapi-which-is-the-best-python-web-framework/>
- [83] ‘Application types for the Microsoft identity platform - Microsoft identity platform’. Accessed: Feb. 28, 2025. [Online]. Available: <https://learn.microsoft.com/en-us/entra/identity-platform/v2-app-types>
- [84] ‘Components | Chakra UI’. Accessed: Mar. 18, 2025. [Online]. Available: <https://chakra-ui.com/docs/components/concepts/overview>



## 11 Appendices

### 11.1 Appendix 1 – Proposal

Prifysgol Wrexham  
Wrexham University

#### Computing Project Proposal Form

Student Name	Samuel Eras
Student Email	<a href="mailto:S23014443@mail.glyndwr.ac.uk">S23014443@mail.glyndwr.ac.uk</a>
Course	BSc (Hons) Computer Science

**Proposed Project Title:**

Designing an AI-Integrated IT Helpdesk Web Application with an LLM-Directed Workflow and Retrieval-Augmented Generation for Automated Ticket Resolution and Improved Ticket Quality

**Requested Supervisor(s):**

First (primary) supervisor: Teri Birch

Second supervisor: Sahan Perera

**Project Outline:**

The project's objective is to develop an IT Helpdesk Web Application capable of directly resolving user inquiries or generating detailed tickets using a pretrained Large Language Model. This model will be able to utilise company-specific data provided by helpdesk technicians through retrieval-augmented generation. By chaining multiple requests within a workflow, the model can shape the process dynamically based on its own decisions. This leads to improved document retrieval, grading of those documents and determining whether to perform a web search for additional information. The project focuses on providing a fully functional web application, offering a good user experience with an appealing and intuitive UI. Authentication will distinguish between users and technicians. Users will be able to access the AI and their tickets, while technicians can filter, assign, handle and close tickets.

**Rationale for choice of Project:**

The rationale for creating an AI-integrated IT Helpdesk solution is to reduce workload and costs for companies by reducing ticket volume and improving ticket quality. A significant portion of tickets relate to user errors instead of actual technological issues, which can often be resolved by

a Large Language Model guiding users through the troubleshooting process. If the issue cannot be solved due to the LLM's limitations or due to factors like required privileges, the AI can create a ticket for the user, assuring high ticket quality. Many users often forget to mention important details when submitting tickets, which costs technicians a lot of time in follow-up inquiries. An automatically generated ticket will resolve this issue by using a predefined structure that includes all necessary information. Additionally the ticket can be complemented with extra context and suggested solutions to assist the technician.

While achieving those objectives will make the created application valuable for companies, the project itself is also in line with my interests. It will prove as an opportunity to immerse myself and acquire significant knowledge in web development and AI. As I want to work in this sector, I see this project as a foundation for my future career.

**People with whom you have discussed the project (eg, employer, members of the lecturing staff):**

Teri Birch

**Research Areas of Study:**


Web Development, AI (LLM, AI Agents, RAG), Databases

**Intended Deliverables:**

Fully functional Web Application including Front-end and Back-end.

Project Documentation.

**Resources Needed by Project:**

<b>Student Signature</b>	
<b>Date</b>	<b>30.10.2024</b>

## 11.2 Appendix 2 – Specification

### Computing Project Specification Form

<b>Student Name</b>	Samuel Eras
<b>Student Email</b>	<a href="mailto:S23014443@mail.glyndwr.ac.uk">S23014443@mail.glyndwr.ac.uk</a>
<b>Course</b>	BSc (Hons) Computer Science
<b>Primary Supervisor</b>	Teri Birch

#### Project Title:

AI-Integrated IT Helpdesk Web Application for Automated Ticket Generation and Resolution

#### Project Aims and Objectives:

The aim is to develop a fully functional Helpdesk Web Application that utilises a Large Language Model to resolve and generate Helpdesk tickets.

To achieve this, the project is set to fulfil the following objectives:

- Conduct a literature review on existing Helpdesk Applications and analyse tools and technologies to identify the most suitable solutions for AI integration and Web Application development.
- Implement an AI-driven system that resolves or generates tickets, utilising retrieval-augmented generation and web search capabilities to improve response accuracy.
- Establish a database structure for storing ticket and user data.
- Develop a backend API that provides endpoints for the frontend interface, manages authentication, handles database interactions, and interacts with the AI-driven system to resolve or generate tickets.
- Design and implement the intuitive and responsive frontend.
- Test the functionality of each developed component individually and as part of the overall system.
- Create the project documentation.

#### Research Areas of Study:

Web Development, AI (LLM, AI Workflows, RAG), Databases

**Methodology:**

It is crucial to conduct the development of an application using the most suitable methodology. This choice determines planning, structure, and organisation which depending on the approach, can affect different aspects such as quality, efficiency, adaptability, and risk management.

As this project has clearly defined objectives that are unlikely to change, a sequential development methodology like Waterfall seems favourable. It reduces complexity compared to an agile approach by defining a clear structure and deadlines.

However, Waterfall also has its downsides, as it typically involves delayed testing after the development of each component. This can delay issue detection and lead to unforeseen extensions to the timeline, which is not acceptable with a strict deadline. Moreover, the timeline can only be estimated, and some components could take more time than expected.

To mitigate these risks, a hybrid approach can be applied that combines some aspects of both methods. For this project, the Waterfall model will be complemented with continuous, iterative testing to ensure early issue detection. Prioritisation of core features will be applied if deadlines are at risk due to unforeseen circumstances or misjudged time constraints.

This approach provides a clear structure, improved quality assurance, and helps to ensure deadline compliance.

**Project Outcomes and Deliverables:**


Fully functional Web Application including Front-end and Back-end.

Project Documentation.

**Project Timetable:**

The project timetable is divided into four main phases. The project will begin with planning and research. In the second phase, the Web Application will be developed. Following development, the application will be tested and evaluated. Finally, the project documentation will be created and the presentation will be prepared.

Further details are provided in the Gantt chart in the appendices.

<b>Student Signature</b>	
<b>Date</b>	<b>11.11.2024</b>
<b>Agreed by Supervisor</b>	<i>Terí Bírch</i>

## 11.3 Appendix 3 – Project Supervision Logbook

### 11.3.1 Logbook 1

<b>BSc Project Diary/Logbook</b>	
Name	Samuel Eras
Project title	Designing an AI-Integrated IT Helpdesk Web Application with an LLM-Directed Workflow and Retrieval-Augmented Generation for Automated Ticket Resolution and Improved Ticket Quality
Date	14.11.2024
Work achieved since last meeting	<ul style="list-style-type: none"><li>- Writing the proposal</li><li>- Planning the timeline</li><li>- Researching the methodology</li><li>- Writing the specification</li><li>- Gathering literature</li><li>- Structuring the literature review (LR) draft</li></ul>
Problems encountered	<ul style="list-style-type: none"><li>- Unsure about what to include in the LR</li></ul>
Work planned for next meeting	<ul style="list-style-type: none"><li>- Reading the literature</li><li>- Starting the LR</li><li>- Conducting the risk assessment</li><li>- Creating the project poster</li></ul>
Project plan status (on, ahead, behind)	on
Supervisor comments	Student is on plan having already outlined the literature review and has most of the text for his

	<p>poster.</p> <p>Student is going to change the title of the project on the project specification as it is very long, he had already thought of this for the poster so he has an idea of what to reduce it to.</p>
Supervisor signature	<i>JABirch</i>

### 11.3.2 Logbook 2

BSc Project Diary/Logbook	
Name	Samuel Eras
Project title	AI-Integrated IT Helpdesk Web Application for Automated Ticket Resolution and Generation
Date	28.11.2024
Work achieved since last meeting	<ul style="list-style-type: none"> <li>- Reading Literature</li> <li>- Writing first part of the Literature Review about Helpdesk Systems</li> <li>- Conducting the Risk Assessment</li> <li>- Creating the Project Poster</li> </ul>
Problems encountered	
Work planned for next meeting	<ul style="list-style-type: none"> <li>- Gathering and Reading Literature about AI</li> <li>- Printing the Poster</li> <li>- Conducting Poster Presentation</li> </ul>
Project plan status (on, ahead, behind)	on

Supervisor comments	<p>Posters look ok, advised to justify the text in the text boxes to see how it looks, if doesn't look good stick to how it is.</p> <p>Student to carry on with literature review and send to supervisor before next meeting, where he is up to at that point – not expected to be finished yet but it is good to see where student is up to.</p>
Supervisor signature	<i>JA Birch</i>

### 11.3.3 Logbook 3

BSc Project Diary/Logbook	
Name	Samuel Eras
Project title	AI-Integrated IT Helpdesk Web Application for Automated Ticket Resolution and Generation
Date	12.12.2024
Work achieved since last meeting	<ul style="list-style-type: none"> <li>- Gathering and Reading Literature about AI</li> <li>- Writing the Literature Review about AI</li> <li>- Printing the Poster</li> <li>- Preparing for the Poster Presentation</li> <li>- Conducting the Poster Presentation</li> </ul>
Problems encountered	
Work planned for next meeting	<ul style="list-style-type: none"> <li>- Writing the Literature Review about Databases and Web Development</li> <li>- Writing the Conclusion and Finalising the Literature Review</li> </ul>

	- Creating an initial Design for the Implementation
Project plan status (on, ahead, behind)	on
Supervisor comments	
Supervisor signature	<i>JA Birch</i>

#### 11.3.4 Logbook 4

BSc Project Diary/Logbook	
Name	Samuel Eras
Project title	AI-Integrated IT Helpdesk Web Application for Automated Ticket Resolution and Generation
Date	15.01.2025
Work achieved since last meeting	<ul style="list-style-type: none"> <li>- Writing the Literature Review about Databases and Web Development</li> <li>- Writing the Conclusion and Finalising the Literature Review</li> <li>- Creating an initial Design for the Implementation of the AI System</li> </ul>
Problems encountered	
Work planned for next meeting	<ul style="list-style-type: none"> <li>- Finishing the Design of the AI System</li> <li>- Implementing the AI System</li> </ul>
Project plan status	on



(on, ahead, behind)	
Supervisor comments	<p>Student is slightly ahead which is good as this allows for if things happen that mean they cant do work on the project.</p> <p>Supervisor will take a look at literature review over next week to give some feedback.</p>
Supervisor signature	<i>JA Birch</i>

#### 11.3.5 Logbook 5

BSc Project Diary/Logbook	
Name	Samuel Eras
Project title	AI-Integrated IT Helpdesk Web Application for Automated Ticket Resolution and Generation
Date	30.01.2025
Work achieved since last meeting	<ul style="list-style-type: none"> <li>- Finishing the Design of the AI System</li> <li>- Implementing the AI System</li> </ul>
Problems encountered	<ul style="list-style-type: none"> <li>- The desired vector database does not run natively on Windows, so Docker was used to deploy it for development.</li> <li>- Development was disrupted due to hardware issues.</li> </ul>
Work planned for next meeting	<ul style="list-style-type: none"> <li>- Testing the AI System</li> <li>- Writing the Implementation Chapter for the AI System</li> <li>- Writing the Design and Implementation Chapter for the Database</li> <li>- Implementing the Database</li> </ul>

Project plan status (on, ahead, behind)	on
Supervisor comments	Student is on track with the project and has a good plan going forward.
Supervisor signature	<i>7 A Birch</i>

### 11.3.6 Logbook 6

BSc Project Diary/Logbook	
Name	Samuel Eras
Project title	AI-Integrated IT Helpdesk Web Application for Automated Ticket Resolution and Generation
Date	13.02.2025
Work achieved since last meeting	<ul style="list-style-type: none"> <li>- Writing the Implementation Chapter for the AI System</li> <li>- Writing the Design and Implementation Chapter for the Database</li> <li>- Implementing the Database</li> <li>- Started implementing the FastAPI Backend</li> </ul>
Problems encountered	<ul style="list-style-type: none"> <li>- AI System did not work as expected. Redesigned the workflow</li> </ul>
Work planned for next meeting	<ul style="list-style-type: none"> <li>- Writing the Design and Implementation Chapter for the Backend</li> <li>- Finish implementing the Backend</li> </ul>

Project plan status (on, ahead, behind)	On
Supervisor comments	Student is on track to complete with no issues. Student had questions as there was some confusion with literature reviews hopefully cleared that up. Student has already done it but was concerned after recent classes.
Supervisor signature	<i>JA Birch</i>

#### 11.3.7 Logbook 7

BSc Project Diary/Logbook	
Name	Samuel Eras
Project title	AI-Integrated IT Helpdesk Web Application for Automated Ticket Resolution and Generation
Date	06.03.2025
Work achieved since last meeting	<ul style="list-style-type: none"> <li>- Writing the Design and Implementation Chapter for the Backend</li> <li>- Finish implementing the Backend</li> <li>- Starting with the Frontend Implementation</li> </ul>
Problems encountered	
Work planned for next meeting	<ul style="list-style-type: none"> <li>- Writing the Design Chapter for the Frontend</li> <li>- Creating UI Design Mockups for the Frontend</li> <li>- Continue Implementing the Frontend</li> </ul>
Project plan status	On

(on, ahead, behind)	
Supervisor comments	Student is doing well, making go head way.  Meeting agreed for 2 weeks' time.
Supervisor signature	<i>7 A Birch</i>

#### 11.3.8 Logbook 8

BSc Project Diary/Logbook	
Name	Samuel Eras
Project title	AI-Integrated IT Helpdesk Web Application for Automated Ticket Resolution and Generation
Date	20.03.2025
Work achieved since last meeting	<ul style="list-style-type: none"> <li>- Writing the Frontend Design Chapter</li> <li>- Creating UI Design Mockups for the Frontend</li> <li>- Finish Implementing the Frontend</li> <li>- Writing Implementation Chapter for the Frontend</li> </ul>
Problems encountered	
Work planned for next meeting	<ul style="list-style-type: none"> <li>- Writing Design Chapter for Testing</li> <li>- Conducting Integration, Performance and Security Testing</li> <li>- Writing Implementation Chapter for Testing</li> </ul>
Project plan status (on, ahead, behind)	On

Supervisor comments	Student has made good progress with both the artifact and the report
Supervisor signature	<i>TABirch</i>

#### 11.3.9 Logbook 9

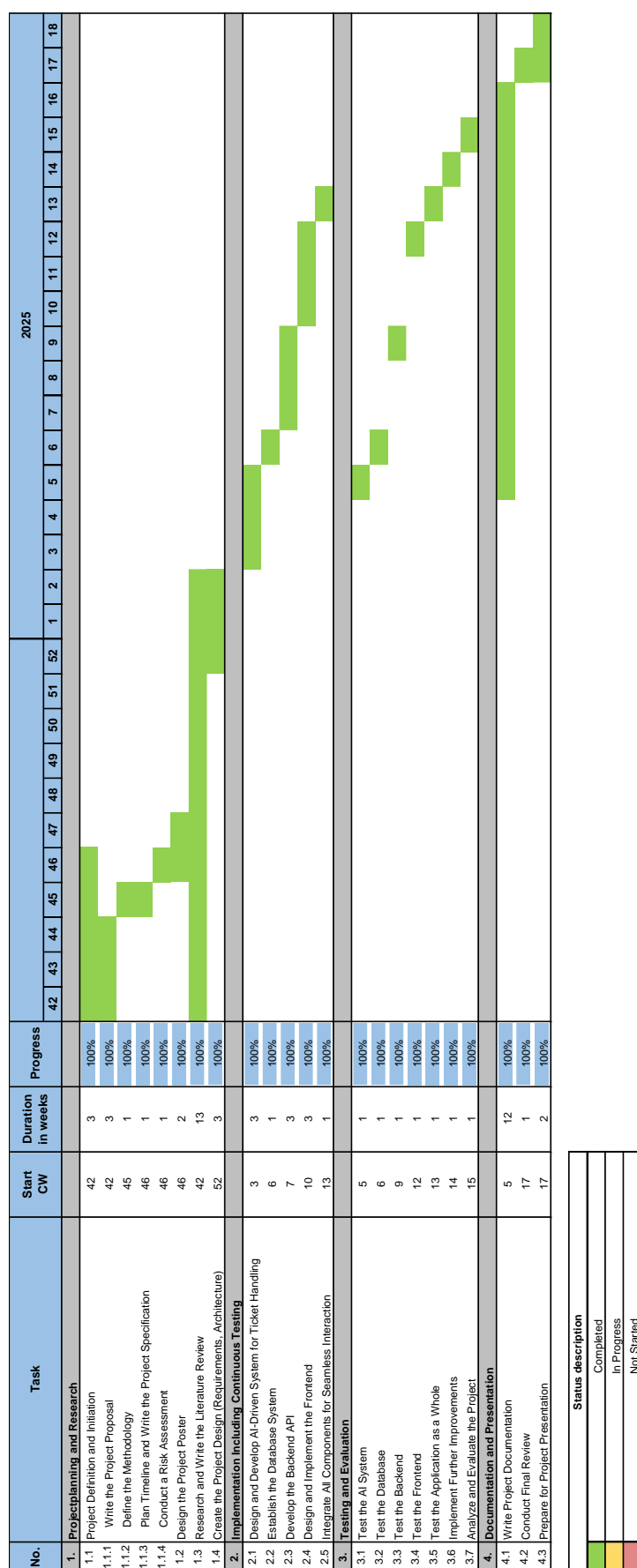
BSc Project Diary/Logbook	
Name	Samuel Eras
Project title	AI-Integrated IT Helpdesk Web Application for Automated Ticket Resolution and Generation
Date	03.04.2025
Work achieved since last meeting	<ul style="list-style-type: none"> <li>- Writing Design Chapter for Testing</li> <li>- Conducting Integration, Performance, Security and User Acceptance Testing</li> <li>- Writing Implementation Chapter for Testing</li> <li>- Writing Evaluation of the Product</li> </ul>
Problems encountered	
Work planned for next meeting	<ul style="list-style-type: none"> <li>- Writing Critical Evaluation, Conclusion, Introduction, Abstract, and Acknowledgements</li> <li>- Final Corrections and Formatting</li> </ul>
Project plan status (on, ahead, behind)	On
Supervisor comments	Student has finished the artifact and is working on the final bits of the report. Asked to send to

	supervisor so they can start reading through to give feedback.
Supervisor signature	<i>TA Birch</i>

### 11.3.10 Logbook 10

BSc Project Diary/Logbook	
Name	Samuel Eras
Project title	AI-Integrated IT Helpdesk Web Application for Automated Ticket Resolution and Generation
Date	24.04.2025
Work achieved since last meeting	<ul style="list-style-type: none"> <li>- Writing Critical Evaluation, Conclusion, Introduction, Abstract, and Acknowledgements</li> <li>- Final Corrections and Formatting</li> </ul>
Problems encountered	
Work planned for next meeting	<ul style="list-style-type: none"> <li>- Preparing for the final Presentation</li> </ul>
Project plan status (on, ahead, behind)	On
Supervisor comments	Student has provided supervisor with report to review, will review and send back though what I have seen so far is good.
Supervisor signature	<i>TA Birch</i>

#### 11.4 Appendix 4 – Gantt Chart of the Timeline

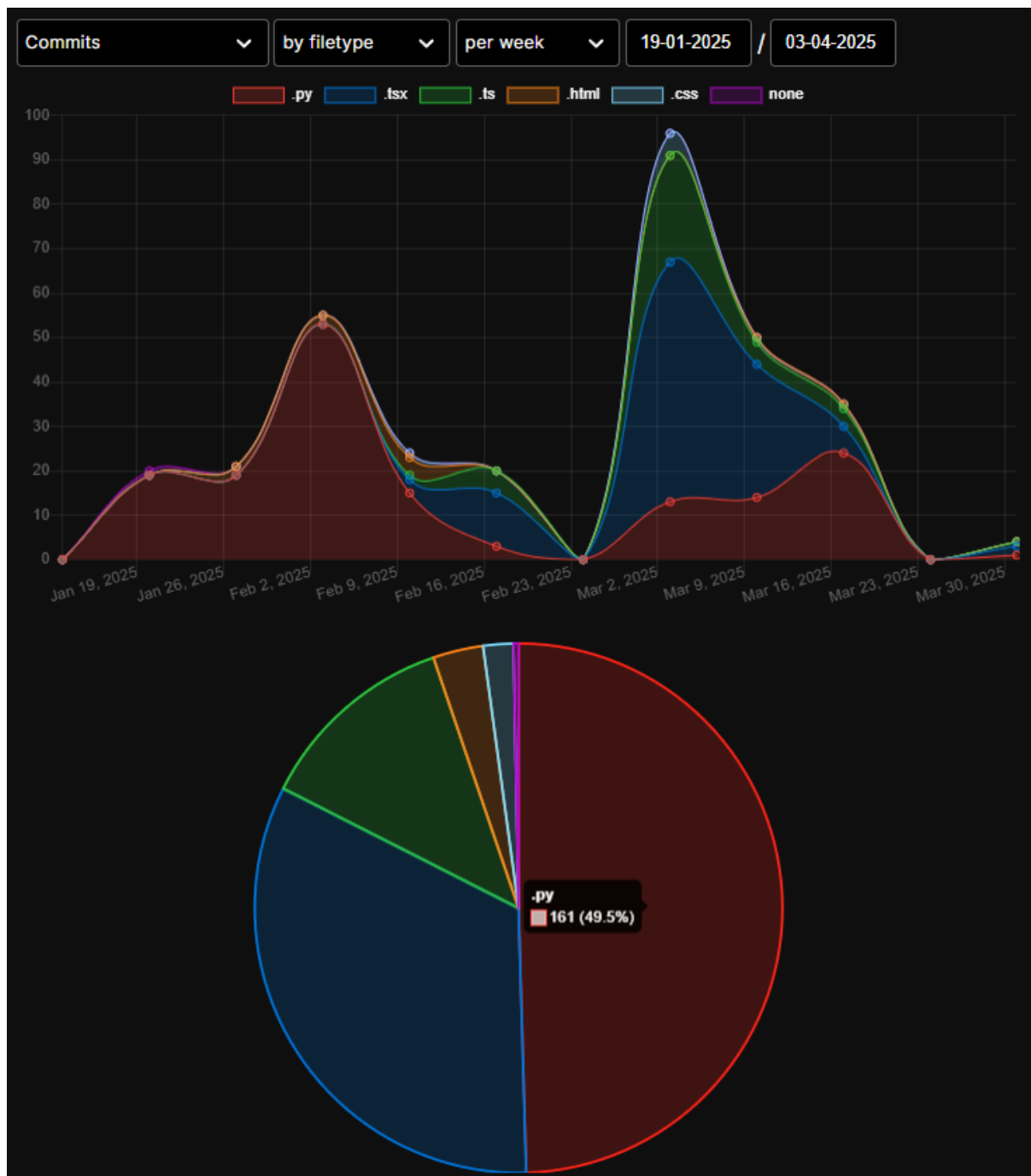


## 11.5 Appendix 5 – Raw Data and Code

The full code is available and openly accessible in the following public GitHub repository:

<https://github.com/samuelleras/ai-helpdesk-bachelors-project>

The following graph displays the weekly commit statistics by file type:





## 11.6 Appendix 6 – Informed Consent Forms for User Acceptance Testing (UAT)

The consent forms were intentionally provided in a different order than the UAT questionnaires to maintain anonymity.

### 11.6.1 Participant 1

#### INFORMED CONSENT FORM

##### BSc Computer Science, Computing, Computer Networks and Security, and Cyber Security Final Year Project Testing

###### Participant Information

You are invited to participate in the testing of undergraduate final-year projects. The purpose of this testing is to assess the functionality, durability, accessibility, usability, scalability, and overall effectiveness of student-created artifacts. Testing will take place in the university's PC Lab, and data will be collected through various methods such as test scripts, test scenarios, forums, feedback, interviews, etc..

This testing will be conducted in a **safe environment** with your **consent**, ensuring that **no sensitive information** is collected. All **Personally Identifiable Information (PII)** will be kept confidential, and all data will be anonymized where possible.

Once data has been collected, **you will not be able to withdraw your consent**, as it will be integrated into the study. This will be detailed further in the consent process.

A **data retention plan** is in place, and by signing this form, you agree that collected data will be retained for academic purposes only and data will be anonymised, stored securely for a specified period before being deleted. Data will be stored in the student's Office 365 account and will be shared with the supervisor. The data will be automatically deleted one year after graduation, upon termination of the student accounts.

###### Participant Consent

By signing this form, you confirm that:

- You understand the purpose of the testing and agree to participate.
- You understand that your participation is voluntary and that you can withdraw at any time **before data collection**.
- You agree to the collection and use of data for academic purposes.
- You understand that all collected data will remain confidential and anonymized where necessary.
- You acknowledge that once data is collected, you cannot withdraw consent.
- You agree to the data retention plan as outlined.
- You have had the opportunity to ask questions and received satisfactory answers.

Participant Name: Hasan Sarikaya

Signature: 

Date: 02.06.25

## 11.6.2 Participant 2

### INFORMED CONSENT FORM

#### BSc Computer Science, Computing, Computer Networks and Security, and Cyber Security Final Year Project Testing

##### Participant Information

You are invited to participate in the testing of undergraduate final-year projects. The purpose of this testing is to assess the functionality, durability, accessibility, usability, scalability, and overall effectiveness of student-created artifacts. Testing will take place in the university's PC Lab, and data will be collected through various methods such as test scripts, test scenarios, forums, feedback, interviews, etc..

This testing will be conducted in a **safe environment** with your **consent**, ensuring that **no sensitive information** is collected. All **Personally Identifiable Information (PII)** will be kept confidential, and all data will be anonymized where possible.

Once data has been collected, **you will not be able to withdraw your consent**, as it will be integrated into the study. This will be detailed further in the consent process.

A **data retention plan** is in place, and by signing this form, you agree that collected data will be retained for academic purposes only and data will be anonymised, stored securely for a specified period before being deleted. Data will be stored in the student's Office 365 account and will be shared with the supervisor. The data will be automatically deleted one year after graduation, upon termination of the student accounts.

##### Participant Consent

By signing this form, you confirm that:

- You understand the purpose of the testing and agree to participate.
- You understand that your participation is voluntary and that you can withdraw at any time **before data collection**.
- You agree to the collection and use of data for academic purposes.
- You understand that all collected data will remain confidential and anonymized where necessary.
- You acknowledge that once data is collected, you cannot withdraw consent.
- You agree to the data retention plan as outlined.
- You have had the opportunity to ask questions and received satisfactory answers.

Participant Name:

*Lehar Nda*

Signature:

*Lehar Nda*

Date:

*02/06/2025*

### 11.6.3 Participant 3

#### INFORMED CONSENT FORM

##### BSc Computer Science, Computing, Computer Networks and Security, and Cyber Security Final Year Project Testing

###### Participant Information

You are invited to participate in the testing of undergraduate final-year projects. The purpose of this testing is to assess the functionality, durability, accessibility, usability, scalability, and overall effectiveness of student-created artifacts. Testing will take place in the university's PC Lab, and data will be collected through various methods such as test scripts, test scenarios, forums, feedback, interviews, etc..

This testing will be conducted in a **safe environment** with your **consent**, ensuring that **no sensitive information** is collected. All **Personally Identifiable Information (PII)** will be kept confidential, and all data will be anonymized where possible.

Once data has been collected, **you will not be able to withdraw your consent**, as it will be integrated into the study. This will be detailed further in the consent process.

A **data retention plan** is in place, and by signing this form, you agree that collected data will be retained for academic purposes only and data will be anonymised, stored securely for a specified period before being deleted. Data will be stored in the student's Office 365 account and will be shared with the supervisor. The data will be automatically deleted one year after graduation, upon termination of the student accounts.

###### Participant Consent

By signing this form, you confirm that:

- You understand the purpose of the testing and agree to participate.
- You understand that your participation is voluntary and that you can withdraw at any time **before data collection**.
- You agree to the collection and use of data for academic purposes.
- You understand that all collected data will remain confidential and anonymized where necessary.
- You acknowledge that once data is collected, you cannot withdraw consent.
- You agree to the data retention plan as outlined.
- You have had the opportunity to ask questions and received satisfactory answers.

Participant Name: *Manus Zeller*

Signature: *M. Zeller*

Date: *2.4.25*

#### 11.6.4 Participant 4

##### INFORMED CONSENT FORM

###### BSc Computer Science, Computing, Computer Networks and Security, and Cyber Security Final Year Project Testing

###### Participant Information

You are invited to participate in the testing of undergraduate final-year projects. The purpose of this testing is to assess the functionality, durability, accessibility, usability, scalability, and overall effectiveness of student-created artifacts. Testing will take place in the university's PC Lab, and data will be collected through various methods such as test scripts, test scenarios, forums, feedback, interviews, etc..

This testing will be conducted in a **safe environment** with your **consent**, ensuring that **no sensitive information** is collected. All **Personally Identifiable Information (PII)** will be kept confidential, and all data will be anonymized where possible.

Once data has been collected, **you will not be able to withdraw your consent**, as it will be integrated into the study. This will be detailed further in the consent process.

A **data retention plan** is in place, and by signing this form, you agree that collected data will be retained for academic purposes only and data will be anonymised, stored securely for a specified period before being deleted. Data will be stored in the student's Office 365 account and will be shared with the supervisor. The data will be automatically deleted one year after graduation, upon termination of the student accounts.

###### Participant Consent

By signing this form, you confirm that:

- You understand the purpose of the testing and agree to participate.
- You understand that your participation is voluntary and that you can withdraw at any time **before data collection**.
- You agree to the collection and use of data for academic purposes.
- You understand that all collected data will remain confidential and anonymized where necessary.
- You acknowledge that once data is collected, you cannot withdraw consent.
- You agree to the data retention plan as outlined.
- You have had the opportunity to ask questions and received satisfactory answers.

Participant Name: Max Halbratter

Signature: Halbratter

Date: 02.04.2025

### 11.6.5 Participant 5

#### INFORMED CONSENT FORM

##### BSc Computer Science, Computing, Computer Networks and Security, and Cyber Security Final Year Project Testing

###### Participant Information

You are invited to participate in the testing of undergraduate final-year projects. The purpose of this testing is to assess the functionality, durability, accessibility, usability, scalability, and overall effectiveness of student-created artifacts. Testing will take place in the university's PC Lab, and data will be collected through various methods such as test scripts, test scenarios, forums, feedback, interviews, etc..

This testing will be conducted in a **safe environment** with your **consent**, ensuring that **no sensitive information** is collected. All **Personally Identifiable Information (PII)** will be kept confidential, and all data will be anonymized where possible.

Once data has been collected, **you will not be able to withdraw your consent**, as it will be integrated into the study. This will be detailed further in the consent process.

A **data retention plan** is in place, and by signing this form, you agree that collected data will be retained for academic purposes only and data will be anonymised, stored securely for a specified period before being deleted. Data will be stored in the student's Office 365 account and will be shared with the supervisor. The data will be automatically deleted one year after graduation, upon termination of the student accounts.

###### Participant Consent

By signing this form, you confirm that:

- You understand the purpose of the testing and agree to participate.
- You understand that your participation is voluntary and that you can withdraw at any time **before data collection**.
- You agree to the collection and use of data for academic purposes.
- You understand that all collected data will remain confidential and anonymized where necessary.
- You acknowledge that once data is collected, you cannot withdraw consent.
- You agree to the data retention plan as outlined.
- You have had the opportunity to ask questions and received satisfactory answers.

Participant Name: *Patrick Weilbech*

Signature: *Weilbech*

Date: *02.07.25*



## 11.6.6 Participant 6

### INFORMED CONSENT FORM

#### BSc Computer Science, Computing, Computer Networks and Security, and Cyber Security Final Year Project Testing

##### Participant Information

You are invited to participate in the testing of undergraduate final-year projects. The purpose of this testing is to assess the functionality, durability, accessibility, usability, scalability, and overall effectiveness of student-created artifacts. Testing will take place in the university's PC Lab, and data will be collected through various methods such as test scripts, test scenarios, forums, feedback, interviews, etc..

This testing will be conducted in a **safe environment** with your **consent**, ensuring that **no sensitive information** is collected. All **Personally Identifiable Information (PII)** will be kept confidential, and all data will be anonymized where possible.

Once data has been collected, **you will not be able to withdraw your consent**, as it will be integrated into the study. This will be detailed further in the consent process.

A **data retention plan** is in place, and by signing this form, you agree that collected data will be retained for academic purposes only and data will be anonymised, stored securely for a specified period before being deleted. Data will be stored in the student's Office 365 account and will be shared with the supervisor. The data will be automatically deleted one year after graduation, upon termination of the student accounts.

##### Participant Consent

By signing this form, you confirm that:

- You understand the purpose of the testing and agree to participate.
- You understand that your participation is voluntary and that you can withdraw at any time **before data collection**.
- You agree to the collection and use of data for academic purposes.
- You understand that all collected data will remain confidential and anonymized where necessary.
- You acknowledge that once data is collected, you cannot withdraw consent.
- You agree to the data retention plan as outlined.
- You have had the opportunity to ask questions and received satisfactory answers.

Participant Name:

Johannes Göggelmann

Signature:

J. Göggelmann

Date:

02.04.25

### 11.6.7 Participant 7

#### INFORMED CONSENT FORM

##### **BSc Computer Science, Computing, Computer Networks and Security, and Cyber Security Final Year Project Testing**

###### **Participant Information**

You are invited to participate in the testing of undergraduate final-year projects. The purpose of this testing is to assess the functionality, durability, accessibility, usability, scalability, and overall effectiveness of student-created artifacts. Testing will take place in the university's PC Lab, and data will be collected through various methods such as test scripts, test scenarios, forums, feedback, interviews, etc..

This testing will be conducted in a **safe environment** with your **consent**, ensuring that **no sensitive information** is collected. All **Personally Identifiable Information (PII)** will be kept confidential, and all data will be anonymized where possible.

Once data has been collected, **you will not be able to withdraw your consent**, as it will be integrated into the study. This will be detailed further in the consent process.

A **data retention plan** is in place, and by signing this form, you agree that collected data will be retained for academic purposes only and data will be anonymised, stored securely for a specified period before being deleted. Data will be stored in the student's Office 365 account and will be shared with the supervisor. The data will be automatically deleted one year after graduation, upon termination of the student accounts.

###### **Participant Consent**

By signing this form, you confirm that:

- You understand the purpose of the testing and agree to participate.
- You understand that your participation is voluntary and that you can withdraw at any time **before data collection**.
- You agree to the collection and use of data for academic purposes.
- You understand that all collected data will remain confidential and anonymized where necessary.
- You acknowledge that once data is collected, you cannot withdraw consent.
- You agree to the data retention plan as outlined.
- You have had the opportunity to ask questions and received satisfactory answers.

**Participant Name:** Axel Schaff

**Signature:** Axel Schaff

**Date:** 02.04.2025

## 11.7 Appendix 7 – Questionnaire for User Acceptance Testing (UAT)

### 11.7.1 Participant 1

#### Questionnaire

##### **AI-Integrated IT Helpdesk Web Application with an LLM-Directed Workflow**

##### **BSc Computer Science**

##### **Final Year Project Testing**

**Date:** 02.04.2025

#### **Usability**

1. On a scale of 1-5 (1 = very difficult, 5 = very easy), how easy was it to navigate the project?  
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5
  2. Were the instructions clear and easy to follow?  
☒ Yes ☐ No ☐ Somewhat (please specify): \_\_\_\_\_
  3. Did you encounter any errors or issues while using the project? If so, please describe them. I encountered no problems.
- 

#### **Accessibility**

4. Was the project accessible and usable for people with different abilities (e.g., colour contrast, screen reader compatibility, keyboard navigation)?  
☒ Yes ☐ No ☐ Not sure
  5. Did you face any difficulties related to accessibility while using the project?  
☐ Yes (please explain): \_\_\_\_\_ ☒ No
- 

#### **Functionality & Fit for Purpose**

6. Did all features work as expected?  
☒ Yes ☐ No (please specify which ones did not): \_\_\_\_\_
7. Does the automated troubleshooting work as expected?  
☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_



8. On a scale of 1-5 (1 = very bad, 5 = very good), how helpful/effective was the troubleshooting (for simple issues)?  
☐ 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5
9. Did the ticket generation work as expected?  
☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_
10. On a scale of 1-5 (1 = very bad, 5 = very good), how would you rate the ticket quality?  
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5
11. Are there any missing features or functionalities you would expect?  
☐ Yes (please describe): \_\_\_\_\_ ☒ No
- 

### **Durability**

9. Did the project function as expected throughout the testing session?  
☒ Yes ☐ No (please describe any issues): \_\_\_\_\_
10. How would you rate the system's stability under normal usage?  
☐ Poor ☐ Fair ☐ Good ☐ Very Good ☒ Excellent
- 

### **Scalability**

11. Do you think this project could handle a larger number of users or data without issues?  
☒ Yes ☐ No ☐ Unsure
12. Did you notice any performance lags or slowdowns when performing tasks?  
☐ Yes (please specify): \_\_\_\_\_ ☒ No
- 

### **General Feedback**

13. What did you like most about the project?

High quality tickets

---

14. What areas do you think need improvement?
- 

15. Any additional comments or suggestions?
-

## Questionnaire

### AI-Integrated IT Helpdesk Web Application with an LLM-Directed Workflow

**BSc Computer Science**

**Final Year Project Testing**

**Date:** 02.04.2025

#### Usability

4. On a scale of 1-5 (1 = very difficult, 5 = very easy), how easy was it to navigate the project?  
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5
5. Were the instructions clear and easy to follow?  
☒ Yes ☐ No ☐ Somewhat (please specify): \_\_\_\_\_
6. Did you encounter any errors or issues while using the project? If so, please describe them. The ticket filter did not work as expected.
- 

#### Accessibility

6. Was the project accessible and usable for people with different abilities (e.g., colour contrast, screen reader compatibility, keyboard navigation)?  
☐ Yes ☐ No ☒ Not sure
7. Did you face any difficulties related to accessibility while using the project?  
☐ Yes (please explain): \_\_\_\_\_ ☒ No
- 

#### Functionality & Fit for Purpose

12. Did all features work as expected?  
☐ Yes ☒ No (please specify which ones did not): The ticket filter did not work as expected.
13. Does the automated troubleshooting work as expected?  
☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_

14. On a scale of 1-5 (1 = very bad, 5 = very good), how helpful/effective was the troubleshooting (for simple issues)?  
☐ 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5
15. Did the ticket generation work as expected?  
☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_
16. On a scale of 1-5 (1 = very bad, 5 = very good), how would you rate the ticket quality?  
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5
17. Are there any missing features or functionalities you would expect?  
☒ Yes (please describe): Technicians should be able to create tickets manually ☐ No
- 

### **Durability**

11. Did the project function as expected throughout the testing session?  
☒ Yes ☐ No (please describe any issues): \_\_\_\_\_
12. How would you rate the system's stability under normal usage?  
☐ Poor ☐ Fair ☐ Good ☐ Very Good ☒ Excellent
- 

### **Scalability**

13. Do you think this project could handle a larger number of users or data without issues?  
☒ Yes ☐ No ☐ Unsure
14. Did you notice any performance lags or slowdowns when performing tasks?  
☐ Yes (please specify): \_\_\_\_\_ ☒ No
- 

### **General Feedback**

16. What did you like most about the project?  
Good idea, detailed tickets
- 
17. What areas do you think need improvement?
- 
18. Any additional comments or suggestions?
-

## Questionnaire

### AI-Integrated IT Helpdesk Web Application with an LLM-Directed Workflow

**BSc Computer Science**

**Final Year Project Testing**

**Date:** 02.04.2025

#### Usability

7. On a scale of 1-5 (1 = very difficult, 5 = very easy), how easy was it to navigate the project?  
☐ 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5
8. Were the instructions clear and easy to follow?  
☒ Yes ☐ No ☐ Somewhat (please specify): \_\_\_\_\_
9. Did you encounter any errors or issues while using the project? If so, please describe them. The chat should scroll so I can see the newest message
- 

#### Accessibility

8. Was the project accessible and usable for people with different abilities (e.g., colour contrast, screen reader compatibility, keyboard navigation)?  
☒ Yes ☐ No ☐ Not sure
9. Did you face any difficulties related to accessibility while using the project?  
☐ Yes (please explain): \_\_\_\_\_ ☒ No
- 

#### Functionality & Fit for Purpose

18. Did all features work as expected?  
☒ Yes ☐ No (please specify which ones did not): \_\_\_\_\_
19. Does the automated troubleshooting work as expected?  
☒ Yes ☐ No ☒ Somewhat (please explain): \_\_\_\_\_

20. On a scale of 1-5 (1 = very bad, 5 = very good), how helpful/effective was the troubleshooting (for simple issues)?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

21. Did the ticket generation work as expected?

☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_

22. On a scale of 1-5 (1 = very bad, 5 = very good), how would you rate the ticket quality?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

23. Are there any missing features or functionalities you would expect?

☐ Yes (please describe): \_\_\_\_\_ ☒ No

---

### **Durability**

13. Did the project function as expected throughout the testing session?

☒ Yes ☐ No (please describe any issues): \_\_\_\_\_

14. How would you rate the system's stability under normal usage?

☐ Poor ☐ Fair ☐ Good ☐ Very Good ☒ Excellent

---

### **Scalability**

15. Do you think this project could handle a larger number of users or data without issues?

☒ Yes ☐ No ☐ Unsure

16. Did you notice any performance lags or slowdowns when performing tasks?

☐ Yes (please specify): \_\_\_\_\_ ☒ No

---

### **General Feedback**

19. What did you like most about the project?

Ticket generation

---

20. What areas do you think need improvement?

The chat should scroll automatically

---

21. Any additional comments or suggestions?

---

## Questionnaire

### AI-Integrated IT Helpdesk Web Application with an LLM-Directed Workflow

**BSc Computer Science**

**Final Year Project Testing**

**Date:** 02.04.2025

#### Usability

10. On a scale of 1-5 (1 = very difficult, 5 = very easy), how easy was it to navigate the project?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

11. Were the instructions clear and easy to follow?

☒ Yes ☐ No ☐ Somewhat (please specify): \_\_\_\_\_

12. Did you encounter any errors or issues while using the project? If so, please describe them.

---

#### Accessibility

10. Was the project accessible and usable for people with different abilities (e.g., colour contrast, screen reader compatibility, keyboard navigation)?

☒ Yes ☐ No ☐ Not sure

11. Did you face any difficulties related to accessibility while using the project?

☐ Yes (please explain): \_\_\_\_\_ ☒ No

---

#### Functionality & Fit for Purpose

24. Did all features work as expected?

☒ Yes ☐ No (please specify which ones did not): \_\_\_\_\_

25. Does the automated troubleshooting work as expected?

☐ Yes ☐ No ☒ Somewhat (please explain): It is not perfect, especially if you don't provide a lot of detail, but still useful and should be able to solve small issues

26. On a scale of 1-5 (1 = very bad, 5 = very good), how helpful/effective was the troubleshooting (for simple issues)?

☐ 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5

27. Did the ticket generation work as expected?

☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_

28. On a scale of 1-5 (1 = very bad, 5 = very good), how would you rate the ticket quality?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

29. Are there any missing features or functionalities you would expect?

☐ Yes (please describe): \_\_\_\_\_ ☒ No

---

### **Durability**

15. Did the project function as expected throughout the testing session?

☒ Yes ☐ No (please describe any issues): \_\_\_\_\_

16. How would you rate the system's stability under normal usage?

☐ Poor ☐ Fair ☐ Good ☐ Very Good ☒ Excellent

---

### **Scalability**

17. Do you think this project could handle a larger number of users or data without issues?

☐ Yes ☐ No ☒ Unsure

18. Did you notice any performance lags or slowdowns when performing tasks?

☒ Yes (please specify): AI response a little slow, but not too bad ☐ No

---

### **General Feedback**

22. What did you like most about the project?

Good UI, easy to use

---

23. What areas do you think need improvement?

Maybe faster AI responses

---

24. Any additional comments or suggestions?

---

## Questionnaire

### AI-Integrated IT Helpdesk Web Application with an LLM-Directed Workflow

**BSc Computer Science**

**Final Year Project Testing**

**Date:** 02.04.2025

#### Usability

13. On a scale of 1-5 (1 = very difficult, 5 = very easy), how easy was it to navigate the project?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

14. Were the instructions clear and easy to follow?

☒ Yes ☐ No ☐ Somewhat (please specify): \_\_\_\_\_

15. Did you encounter any errors or issues while using the project? If so, please describe them.

---

#### Accessibility

12. Was the project accessible and usable for people with different abilities (e.g., colour contrast, screen reader compatibility, keyboard navigation)?

☐ Yes ☐ No ☒ Not sure

13. Did you face any difficulties related to accessibility while using the project?

☐ Yes (please explain): \_\_\_\_\_ ☒ No

---

#### Functionality & Fit for Purpose

30. Did all features work as expected?

☒ Yes ☐ No (please specify which ones did not): \_\_\_\_\_

31. Does the automated troubleshooting work as expected?

☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_



32. On a scale of 1-5 (1 = very bad, 5 = very good), how helpful/effective was the troubleshooting (for simple issues)?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

33. Did the ticket generation work as expected?

☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_

34. On a scale of 1-5 (1 = very bad, 5 = very good), how would you rate the ticket quality?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

35. Are there any missing features or functionalities you would expect?

☐ Yes (please describe): \_\_\_\_\_ ☒ No

---

### **Durability**

17. Did the project function as expected throughout the testing session?

☒ Yes ☐ No (please describe any issues): \_\_\_\_\_

18. How would you rate the system's stability under normal usage?

☐ Poor ☐ Fair ☐ Good ☐ Very Good ☒ Excellent

---

### **Scalability**

19. Do you think this project could handle a larger number of users or data without issues?

☒ Yes ☐ No ☐ Unsure

20. Did you notice any performance lags or slowdowns when performing tasks?

☐ Yes (please specify): \_\_\_\_\_ ☒ No

---

### **General Feedback**

25. What did you like most about the project?

Issue solving and tickets, Microsoft login

---

26. What areas do you think need improvement?

---

27. Any additional comments or suggestions?

---

## Questionnaire

### AI-Integrated IT Helpdesk Web Application with an LLM-Directed Workflow

**BSc Computer Science**

**Final Year Project Testing**

**Date:** 02.04.2025

#### Usability

16. On a scale of 1-5 (1 = very difficult, 5 = very easy), how easy was it to navigate the project?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

17. Were the instructions clear and easy to follow?

☒ Yes ☐ No ☐ Somewhat (please specify): \_\_\_\_\_

18. Did you encounter any errors or issues while using the project? If so, please describe them.

---

#### Accessibility

14. Was the project accessible and usable for people with different abilities (e.g., colour contrast, screen reader compatibility, keyboard navigation)?

☒ Yes ☐ No ☐ Not sure

15. Did you face any difficulties related to accessibility while using the project?

☐ Yes (please explain): \_\_\_\_\_ ☒ No

---

#### Functionality & Fit for Purpose

36. Did all features work as expected?

☒ Yes ☐ No (please specify which ones did not): \_\_\_\_\_

37. Does the automated troubleshooting work as expected?

☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_

38. On a scale of 1-5 (1 = very bad, 5 = very good), how helpful/effective was the troubleshooting (for simple issues)?

☐ 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5

39. Did the ticket generation work as expected?

☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_

40. On a scale of 1-5 (1 = very bad, 5 = very good), how would you rate the ticket quality?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

41. Are there any missing features or functionalities you would expect?

☐ Yes (please describe): \_\_\_\_\_ ☒ No

---

### **Durability**

19. Did the project function as expected throughout the testing session?

☒ Yes ☐ No (please describe any issues): \_\_\_\_\_

20. How would you rate the system's stability under normal usage?

☐ Poor ☐ Fair ☐ Good ☐ Very Good ☒ Excellent

---

### **Scalability**

21. Do you think this project could handle a larger number of users or data without issues?

☒ Yes ☐ No ☐ Unsure

22. Did you notice any performance lags or slowdowns when performing tasks?

☐ Yes (please specify): \_\_\_\_\_ ☒ No

---

### **General Feedback**

28. What did you like most about the project?

tickets

---

29. What areas do you think need improvement?

---

30. Any additional comments or suggestions?

---

## Questionnaire

### AI-Integrated IT Helpdesk Web Application with an LLM-Directed Workflow

**BSc Computer Science**

**Final Year Project Testing**

**Date:** 02.04.2025

#### Usability

19. On a scale of 1-5 (1 = very difficult, 5 = very easy), how easy was it to navigate the project?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

20. Were the instructions clear and easy to follow?

☒ Yes ☐ No ☐ Somewhat (please specify): \_\_\_\_\_

21. Did you encounter any errors or issues while using the project? If so, please describe them.

---

#### Accessibility

16. Was the project accessible and usable for people with different abilities (e.g., colour contrast, screen reader compatibility, keyboard navigation)?

☐ Yes ☐ No ☒ Not sure

17. Did you face any difficulties related to accessibility while using the project?

☐ Yes (please explain): \_\_\_\_\_ ☒ No

---

#### Functionality & Fit for Purpose

42. Did all features work as expected?

☒ Yes ☐ No (please specify which ones did not): \_\_\_\_\_

43. Does the automated troubleshooting work as expected?

☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_

44. On a scale of 1-5 (1 = very bad, 5 = very good), how helpful/effective was the troubleshooting (for simple issues)?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

45. Did the ticket generation work as expected?

☒ Yes ☐ No ☐ Somewhat (please explain): \_\_\_\_\_

46. On a scale of 1-5 (1 = very bad, 5 = very good), how would you rate the ticket quality?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5

47. Are there any missing features or functionalities you would expect?

☐ Yes (please describe): \_\_\_\_\_ ☒ No

---

### **Durability**

21. Did the project function as expected throughout the testing session?

☒ Yes ☐ No (please describe any issues): \_\_\_\_\_

22. How would you rate the system's stability under normal usage?

☐ Poor ☐ Fair ☐ Good ☐ Very Good ☒ Excellent

---

### **Scalability**

23. Do you think this project could handle a larger number of users or data without issues?

☒ Yes ☐ No ☐ Unsure

24. Did you notice any performance lags or slowdowns when performing tasks?

☐ Yes (please specify): \_\_\_\_\_ ☒ No

---

### **General Feedback**

31. What did you like most about the project?

Chatting with AI, issue resolution and ticket generation, UI/UX

---

32. What areas do you think need improvement?

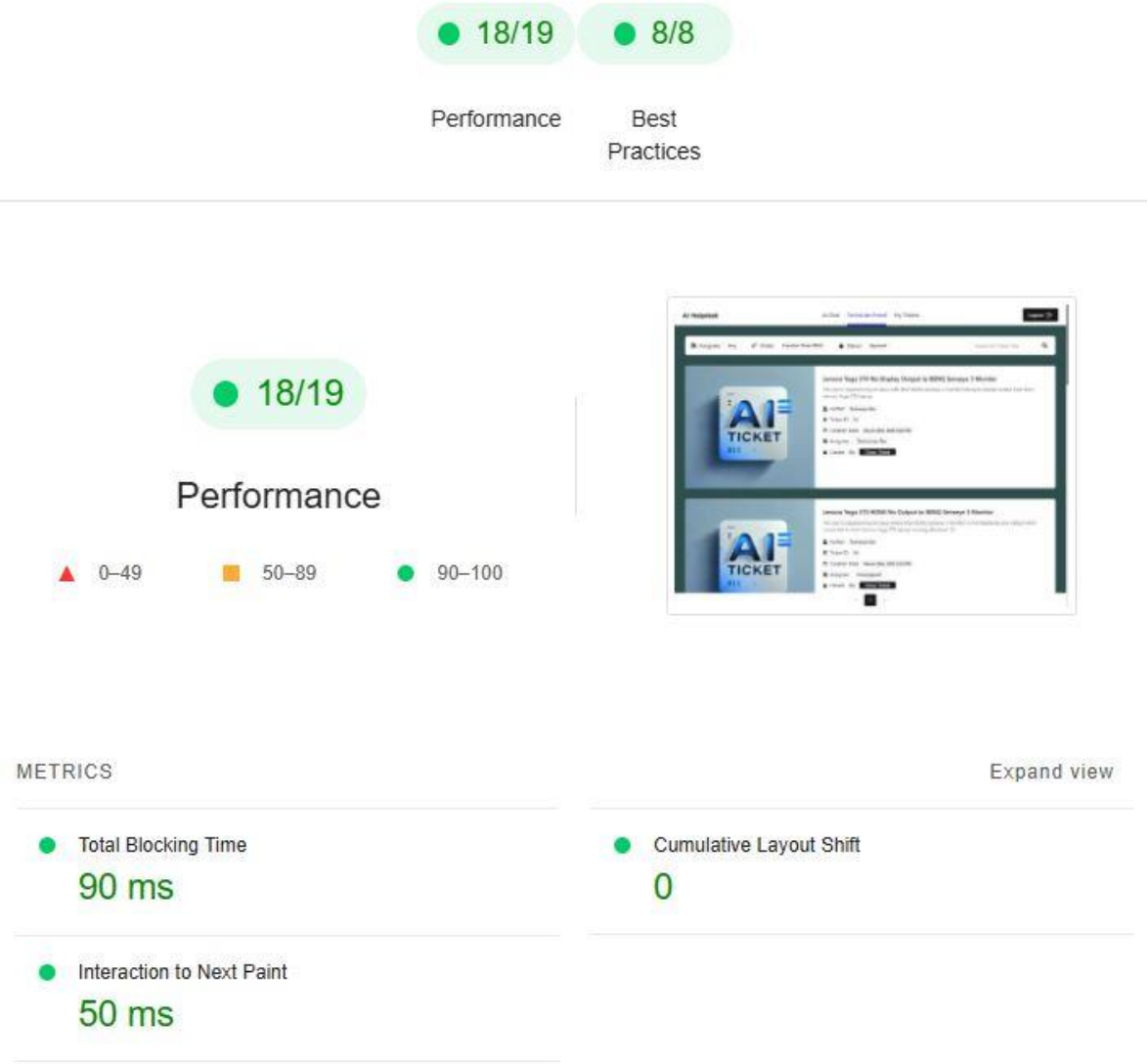
---

33. Any additional comments or suggestions?

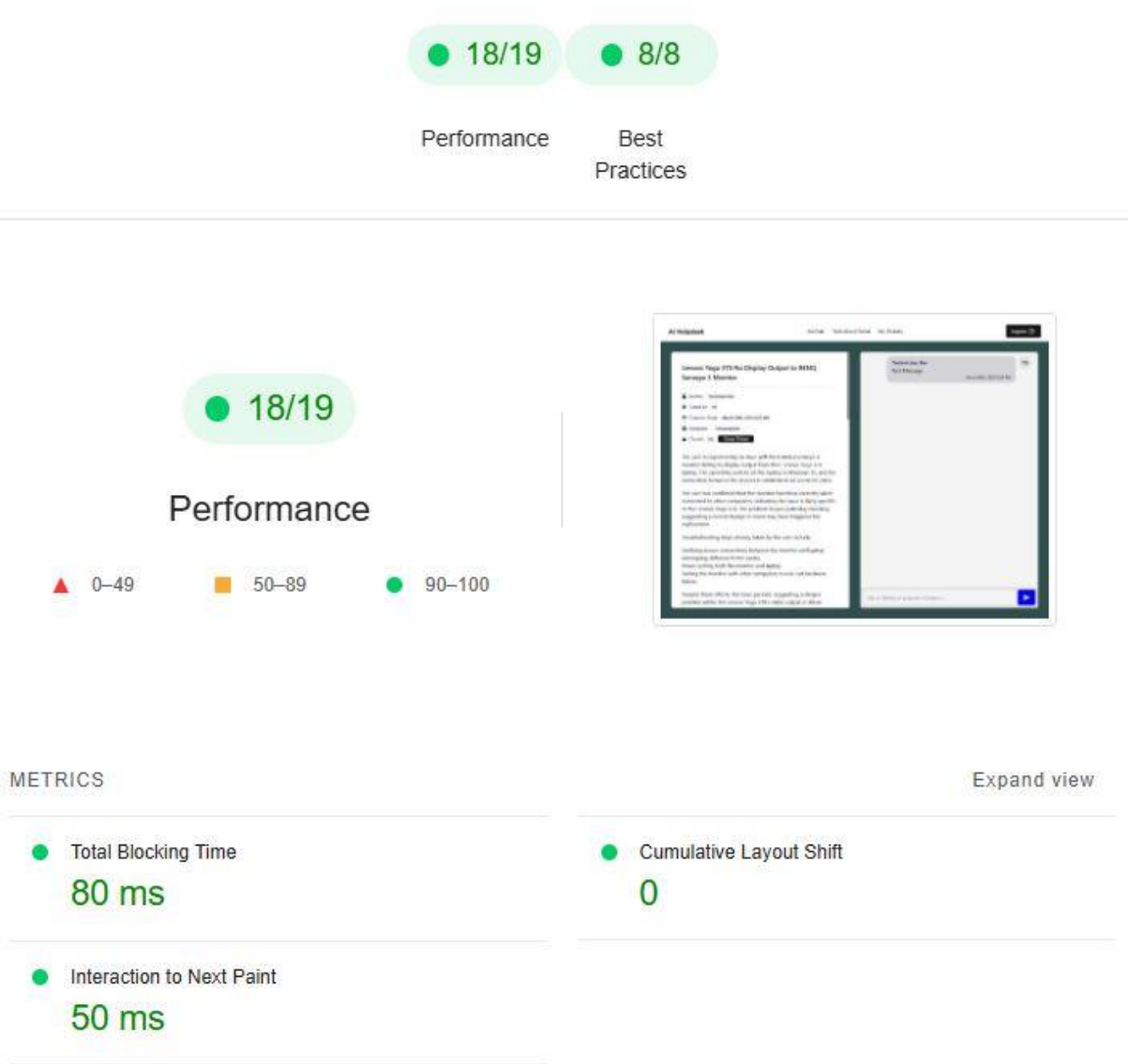
---

11.8 Appendix 8 – Lighthouse Testing Results

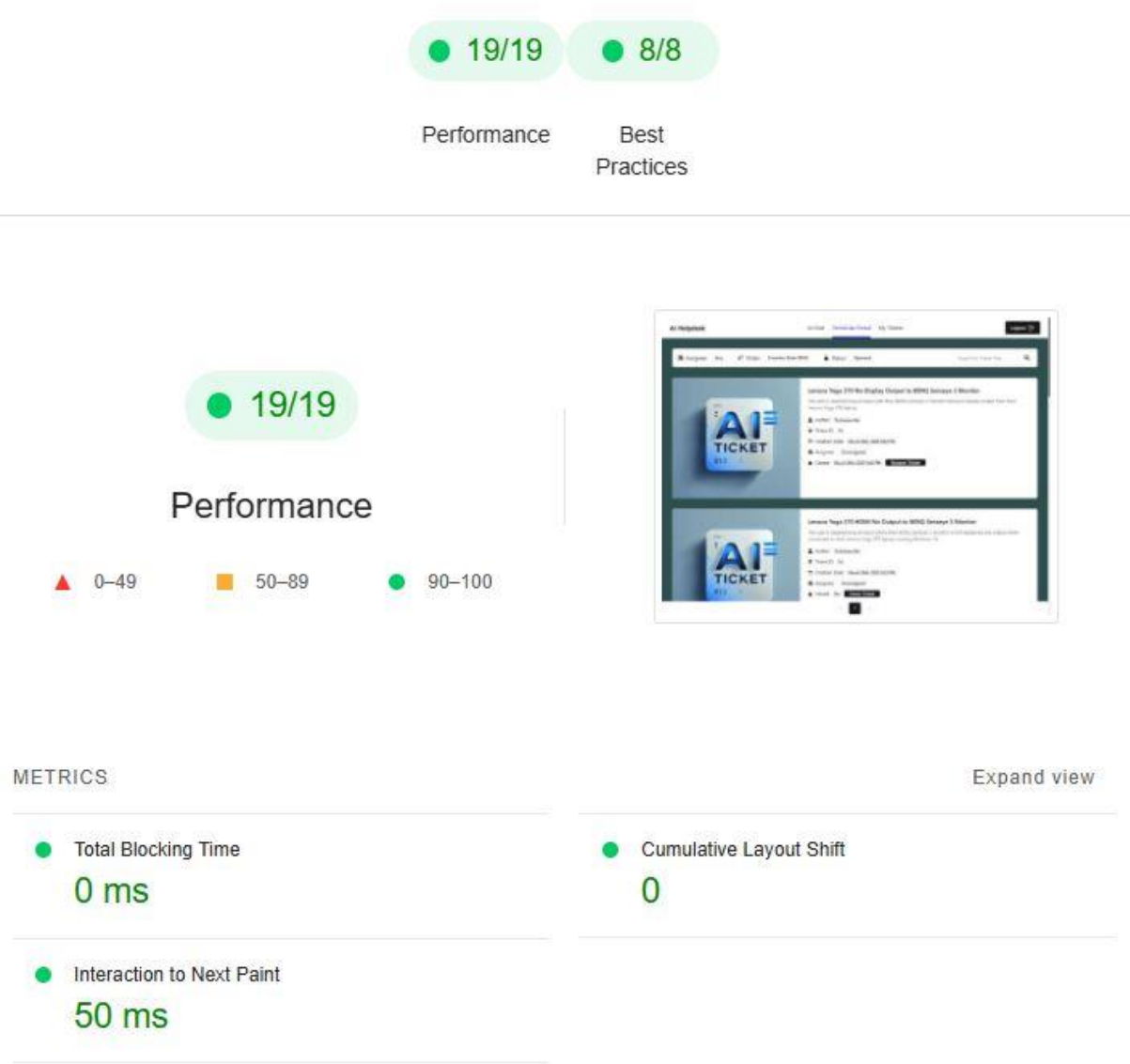
11.8.1 Test results for loading the Technician Portal page



11.8.2 Test results for loading the Ticket page



11.8.3 Test results for assigning a ticket





11.8.4 Test results for closing a ticket

