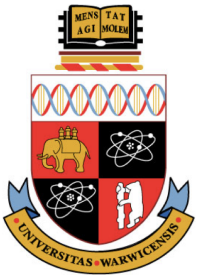


A mobile application for augmenting technical documentation using AR and AI

CS310: Third Year Project

Ani Bitri

2287990

Supervisors: Dr. Paris Giampouras, John McNamara

Department of Computer Science

University of Warwick and IBM

January 2026

Contents

1 Introduction

The discipline of software engineering is fundamentally abstract. As software systems have evolved from monolithic mainframes to distributed, microservice-based architectures, developers heavily rely on technical documentation to understand, maintain and build upon existing tools, systems and frameworks. However, traditional documentation formats, such as PDFs, static web pages and printed manuals, often fall short in conveying complicated concepts, especially when it comes to technical diagrams and schematics. These visuals are essential for understanding the systems they represent, yet they often make it more difficult and confusing for users and other developers to grasp the underlying ideas.

The core objective of this project is to offer a solution that enhances the user experience when interacting with technical documentation while still preserving the integrity and accuracy of the original content. By combining Augmented Reality (AR) and Artificial Intelligence (AI), the project aims to create an application that transforms static documentation and diagrams into interactive, immersive and context-aware experiences.

2 Research

It is essential to understand the existing landscape of applications and research that relate to the project's objectives. This section provides an overview of some of the relevant works and solutions in the field of Augmented Reality, highlighting their approaches and differences from the proposed application. Additionally, usability and accessibility concerns have been considered as part of the research process in order to deliver a solution that is user-friendly and inclusive.

2.1 Literature Review and Existing Solutions

2.1.1 DGT-AR

DGT-AR (Dependency Graph Tool in Augmented Reality) [1] is an academic project which utilises the Microsoft HoloLens and its core premise is similar, in that it aims to visualise software architecture to aid developers in understanding complex systems. This solution renders a software's dependency graph as a 3D node-link structure in the user's physical environment. The researchers on this project found that the virtual space of AR addressed the limitation that a 2D screen has when visualising large and complex graphs. By using the HoloLens, developers could walk around the graph, using motion parallax to better perceive what the graph represented.

2.1.2 Automatic Retargeting of Technical Documentation to AR

In the paper "Retargeting Technical Documentation to Augmented Reality" [2], the authors propose a system which automatically transfers printed technical documentation into an AR format. The system uses computer vision techniques to identify graphical elements and text labels in a scanned document. Given the identified metadata and a CAD model or a 3D representation of an object, the system analyses the spatial relationships to infer part identities and interactions. The approach significantly reduces the manual effort required to author AR content for technical documentation, thus making it more feasible to retarget existing 2D manuals into AR environments.

2.1.3 Industrial Standards: Vuforia and Manifest

PTC Vuforia [3] and Taqtile Manifest [4] are considered industry "gold standards" and widely used platforms in manufacturing for maintenance and training purposes. Vuforia focuses on AR content creation and deployment, excelling at object tracking and image recognition. It allows digital instructions to be overlaid on physical equipment, guiding users through complex procedures. Manifest, on the other hand, has a primary focus on "expert capture" and knowledge transfer, allowing senior technicians to document spatial workflows for trainees to follow. Both platforms function as static AR content delivery systems, requiring significant manual effort to author and maintain the AR experiences.

2.1.4 Differentiation from Existing Solutions

This project differentiates itself from existing solutions, such as DGT-AR and the work by P. Mohr et al., in several key aspects:

- This solution focuses on providing assistance to users in understanding technical documentation through AR overlays and an AI assistant, rather than visualizing software architecture.
- The project will not be reliant on specialized hardware such as the Microsoft HoloLens, but will instead be developed with the purpose of being accessible to any user who wishes to use it.

- There is no requirement for pre-existing 3D models or CAD representations of the objects being documented, nor is there a need to scan physical manuals or objects. The application will be able to process standard documentation, digital or physical, as well as 2D diagrams and images.

2.2 Usability & Accessibility

Since the solution involves creating an application, it is crucial to ensure that the app is user-friendly and accessible to any potential user. To achieve this, the design and development process, primarily for the frontend, will adhere to the Web Content Accessibility Guidelines (WCAG) 2.1. [5], which act as the international standard for web accessibility and provide an extensive set of recommendations for making digital content accessible to a wide range of users. Although not all principles may be applicable to this project, the core principles of Perceivable, Operable, Understandable and Robust (POUR) will be followed.

3 Project Evaluation and Development

This section outlines the requirements and design considerations for the application, covering both frontend and backend components. It details different aspects of the development process and the considerations taken into account to ensure a robust and user-friendly solution.

3.1 Requirements

After careful analysis of the project objectives and user needs, the following functional and non-functional requirements have been identified for the application:

3.1.1 Functional Requirements

1. **Augmented Reality System:**
   - FR-1: The system shall allow users to open the camera interface within the app to scan technical documents.
   - FR-2: The system shall allow the user to upload technical documents from their device storage.
   - FR-3: The system shall recognise diagrams, images, flowcharts, and schematics within the scanned or uploaded documents using image recognition techniques.
   - FR-4: The backend shall extract diagram elements and textual information from the recognised diagrams using IBM Granite Vision and other computer vision techniques.
   - FR-5: The system shall generate interactive AR overlays that provide explanations of components, relationships, and interactions within the diagrams.
   - FR-6: The system shall allow users to interact with the AR overlays to explore additional information about the diagram components and their relationships.
   - FR-7: The system shall provide visual cues to confirm successful recognition and overlay placement.
2. **AI Assistant:**
   - FR-8: The system shall allow users to input natural language queries related to the scanned or uploaded technical documents.
   - FR-9: The backend shall use the Granite AI model to process user queries and generate relevant responses based on the document content.
   - FR-10: The AI shall combine diagram data from Granite Vision with textual information from the documents to provide comprehensive answers.
   - FR-11: The AI assistant shall retain context from previous interactions to provide coherent and relevant responses.
3. **Data Management and Communication:**
   - FR-12: The app shall communicate with the backend over HTTPS with token-based authentication to ensure secure data transmission.
   - FR-13: The backend shall handle user requests, forward images to IBM Granite Vision, combine results with IBM Granite AI, and return structured JSON responses to the app.
   - FR-14: The system shall provide error handling mechanisms to manage failures in image recognition, AI processing, or network connectivity, and inform users appropriately.

3.1.2 Non-Functional Requirements

1. **Performance**
   - NFR-1: The system shall process scanned or uploaded documents and generate an analysis within 1 minute under normal operating conditions.
   - NFR-2: The AI assistant shall respond to user queries within 10 seconds under normal operating conditions.
   - NFR-3: The AR overlays shall render smoothly at a minimum of 30 frames per second on supported devices.
2. **Reliability and Availability**
   - NFR-4: In the event of a backend service failure or network issue, the system shall provide informative error messages to the user.
   - NFR-5: The system must ensure that uploaded documents are cryptographically hashed to prevent duplicate processing and ensure data integrity.

3.2 Design Pattern

The project adopts a layered Client-Server architecture [6], separating the frontend application from the backend services. This pattern is well-suited for separation between the resource-intensive AI computation and the user interface, addressing the hardware limitations of the user's device by offloading heavy processing tasks to the backend server. The system is structured into three layers:

1. **Presentation Layer (Client):** Its primary role is to handle user interactions, handle inputs and render the AR overlays. It relies on the backend for processing, data processing and decision-making.
2. **Service Layer (Application Interface):** By utilising RESTful APIs, this layer serves as the communication bridge between the frontend and backend. It encapsulates a service-oriented design that allows functionalities to be modular and reusable. These services handle task routing, input processing and response formatting.
3. **Computation Layer (Model Manager):** This layer manages the lifecycle of the AI models used. It is responsible for loading, maintaining and invoking the models as needed, while also ensuring efficient resource utilisation.

Figure 5 illustrates the overall system architecture, showcasing the interactions between the frontend application, backend services and AI models.

3.3 Frontend Design

3.3.1 User Interface and User Experience

The application will feature a user-friendly interface that allows users to easily navigate through the app and access its features. The UI will be designed to be intuitive and responsive. When the user opens the app, they will be presented with a simple home screen with options to scan a document, upload a document or access the history of previously scanned documents. Besides the home screen, the app will also include a tab for the chatbot, where users can interact with the AI assistant, and a settings tab for configuring app preferences. The image below illustrates the proposed flow and layout of the app's user interface:

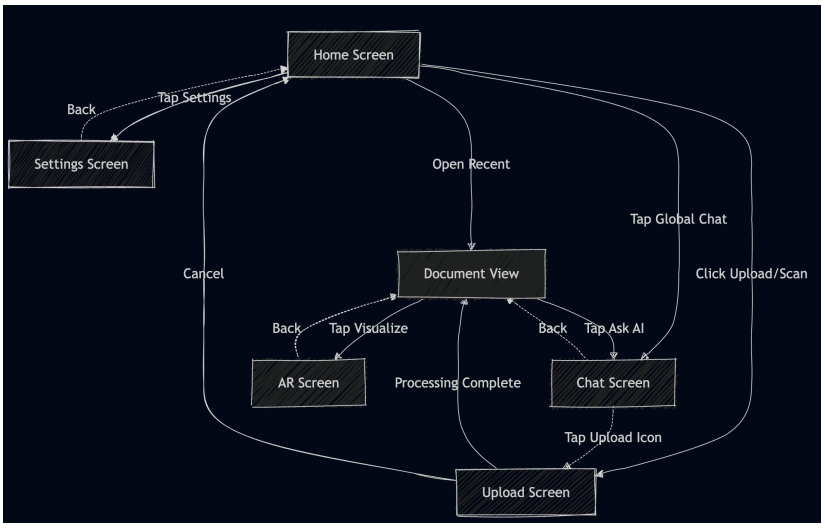


Figure 2: Proposed User Interface Flow

3.3.2 Error Handling and Feedback

To ensure a smooth user experience, the application will include robust error handling mechanisms. Internal errors will be handled gracefully and automatically, with appropriate messages displayed to the user upon interruptions. For example, if the app fails to recognise a diagram or if the AI assistant cannot process a query, the user will be informed of the issue and provided with suggestions for resolution. External errors will be handled by providing informative error messages to guide users in case of issues such as failed scans or uploads, network connectivity problems or AR tracking errors. Furthermore, error prevention strategies will be implemented to minimise their occurrence and impact on the user experience. Such strategies include input validation, network status checks and AR tracking optimisations. This approach will ensure that users can effectively utilise the app's features without being hindered by technical issues.

3.3.3 Integration with Backend and Data Flow

The frontend of the application will communicate with the backend services to process the scanned or uploaded documents and retrieve relevant information. Figure 6 illustrates the data flow between the frontend and backend components.

3.4 Backend Design

3.4.1 Overview

The backend of the application will act as the intelligence and coordination layer that connects the React Native frontend to IBM's AI services. Its primary function is to process the scanned or uploaded documents and other user inputs, understand their content, and generate appropriate AR overlays and AI responses. The backend ensures that heavy computational tasks are offloaded from the mobile device, providing a seamless user experience.

3.4.2 Workflow

The backend workflow consists of several key steps:

1. **Request Reception:** The backend receives the scanned images or uploaded files from the frontend via RESTful API calls.
2. **Image and Data Processing:**
   - If an image is received, it is processed using IBM Granite Vision to extract text and identify diagrams, returning structured data.
   - If text data is received, it is analysed to identify key components, relationships and interactions.
3. **AI Reasoning and Response Generation:** The backend constructs a prompt that combines:
   - Extracted diagram information from IBM Granite Vision.
   - Relevant documentation snippets.
   - The user's query or interaction context.

   The composite prompt is sent to IBM Granite, which then generates a natural language response.
4. **Response Delivery:** The backend merges the AI-generated response with structured metadata and sends a single JSON payload back to the app. The frontend then displays the AR overlays and AI responses to the user.

Figure 7 illustrates the backend architecture and data flow between the different components.

3.5 Technology Stack

The project utilises the following technologies:

- **Frontend:** React Native for cross-platform mobile development, utilising ViroReact (bridging ARKit and ARCore) for rendering augmented reality overlays and camera interactions.
- **Backend:** Python with Flask for the server-side API architecture. The AI pipeline integrates the Segment Anything Model (SAM 2) for spatial segmentation, IBM Granite Vision (3.1) for semantic image analysis, and IBM Granite Chat for context-aware queries
- **Development Tools:** Visual Studio Code for code editing, Git and GitHub for version control, and RESTful APIs for secure, asynchronous communication between the mobile client and backend services.

3.6 Testing Strategy

App development is an iterative process that requires continuous and thorough testing to ensure the quality and reliability of the application. The testing strategy for this project has not changed from the initial outline in the project specification. However, it has been refined to better align with the development process. The testing strategy encompasses the following key areas:

3.6.1 Backend Testing (Unit and Integration)

Given the modular nature of the backend, unit tests will be implemented for individual services and components. Each service, such as image processing or AI response generation, will have its own set of tests to verify its functionality in different scenarios. Integration tests will also be conducted to ensure that the services work together as expected and that data flows correctly between them. To simulate external dependencies, such as file uploads and user queries, mock objects and stubs will be used. These tests will confirm that the backend can handle various inputs and edge cases, simultaneously confirming that the system returns appropriate HTTP status codes and error messages when necessary.

3.6.2 Frontend Testing

The frontend will undergo testing to verify UI stability, responsiveness and correct state management during user interactions. Crucial user flows, such as document scanning, uploading and AR interactions, will be tested to ensure they render correctly under different state conditions. This involves verifying that the app responds appropriately to user inputs, displays correct feedback and handles errors gracefully, making sure that the user is never left in an uncertain state. Additionally, navigation logic will be tested to verify the integrity of the workflow between different screens and components.

3.6.3 System Integration Testing

Once both the frontend and backend components have been individually tested, system end-to-end testing will be conducted to validate the complete application workflow. This involves simulating the full user journey, from uploading or scanning a file to receiving AR overlays and AI responses. This phase of testing will validate the system's performance under realistic conditions.

3.6.4 User Acceptance Testing

Finally, user acceptance testing will be performed with a select group of target users to gather feedback on the overall user experience. This phase will have a main focus on usability, accessibility and satisfaction, verifying that the application meets the needs and expectations of its intended audience. The feedback collected during this phase will be used to make final adjustments and improvements.

4 Project Overview and Current Progress

4.1 Management

Given the agile nature of the project, the Scrum methodology has continued to be adopted for project management, as stated in the project specification. This approach has allowed for flexibility and adaptability throughout the development process, making deviations from the initial plan easier to manage and reduce the risk of falling behind schedule. Regular sprint meetings have been held to review progress, discuss challenges and plan the next steps.

Moreover, the use of version control with Git and GitHub has been beneficial for tracking changes, documenting progress and maintaining a clear history of the codebase. As seen in the picture below, the commit history shows a consistent pattern of development activity, with appropriate commit messages that describe the changes made.

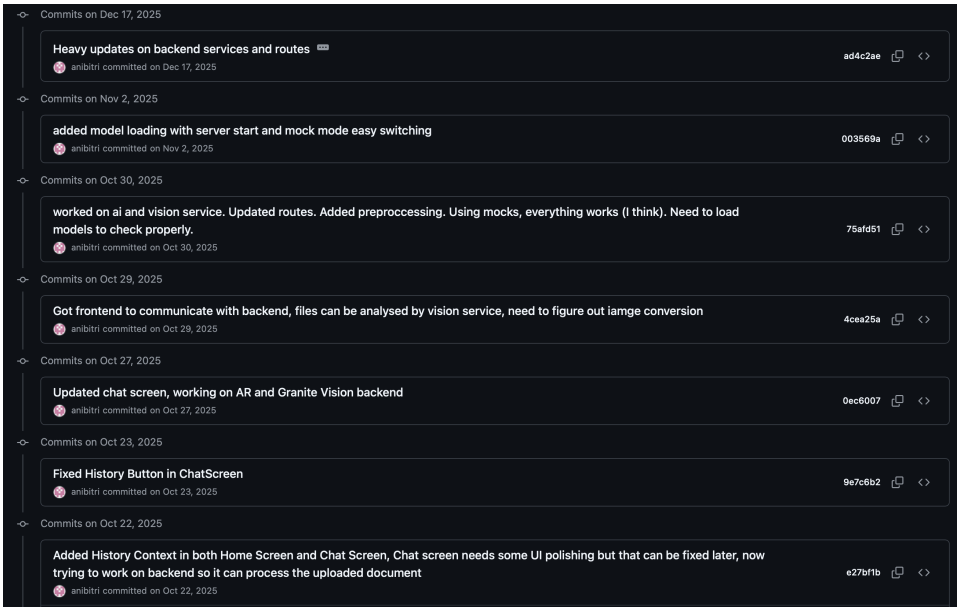
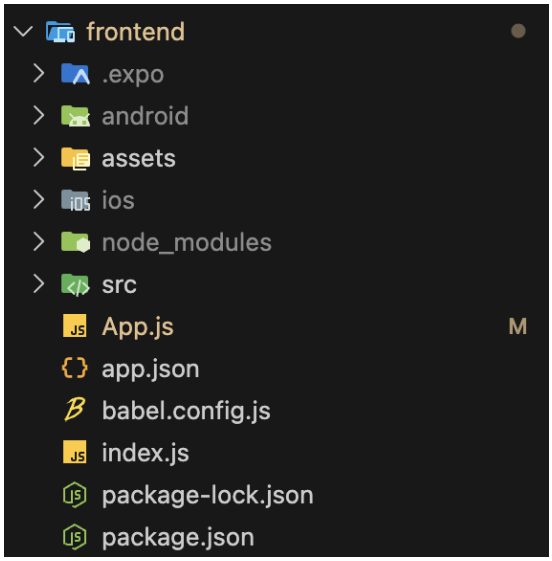
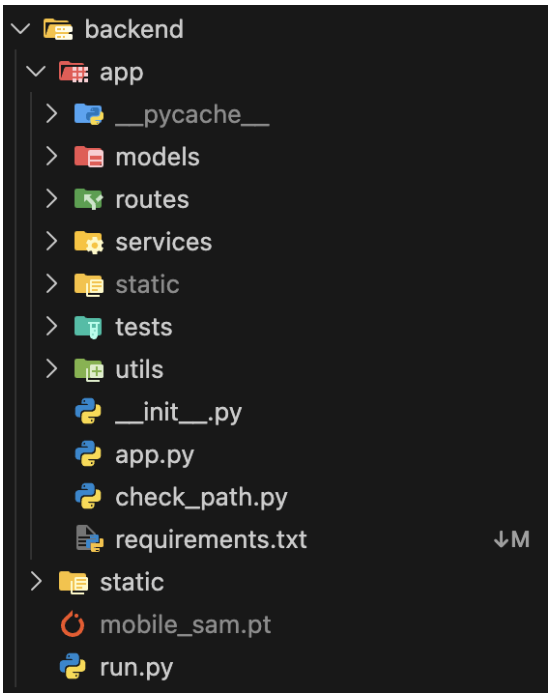


Figure 3: Git commit history showcasing development activity

The modular codebase has been organised into separate repositories for the frontend and backend components, allowing for independent development and testing. The repository structures can be seen in the images below, showcasing the organisation of files and directories for both the React Native app and the Flask backend.



(a) Frontend repository structure



(b) Backend repository structure

Figure 4: Codebase organisation for frontend and backend components

4.2 Comparison to Initial Timeline

The initial timeline for the project focused on working on the backend and the frontend simultaneously. For each four week sprint, two weeks were allocated to backend feature development, followed by two weeks of frontend integration of the newly developed backend feature. While developing the backend, several anticipated challenges were encountered, such as AR unfamiliarity, technical limitations of devices and integration between multiple services. Therefore, the timeline was adjusted to prioritise backend development and testing before moving on to frontend integration. This adjustment allowed for a more focused approach to addressing backend challenges and ensured that the core functionalities were developed and tested before integrating them into the frontend.

4.3 Completed Features

The core backend infrastructure has been established and is fully operational. It is built upon a modular Flask API architecture that serves as the system's backbone. Endpoints have been implemented to handle the complete data flow, from uploading files to processing them with different services, and finally returning structured JSON responses.

The set of services encapsulates the specific functionalities required for the application and are exposed through dedicated API routes. The `Preprocess Service` normalises image resolution and sanitises file inputs to ensure compatibility with downstream models. This prepares the data for the `AR Service` and `AI Service`, both of which invoke the loaded AI models to perform their respective tasks. The `AR Service` first invokes SAM 2 to spatially define diagram components and then chains the `Vision Service` to use Granite Vision for identifying, classifying and extracting metadata from the diagram elements. In parallel, the `AI Service` manages the conversational logic and interfaces with Granite to generate context-aware responses based on user queries and the textual content of the document. These functions are accessible through their respective API endpoints, with `/api/upload` handling secure file uploads and cryptographic hashing; `/api/ar/generate` triggering the visual analysis pipeline and returning structured JSON responses; and `/api/ai/chat` facilitating the retrieval-augmented generation process for user queries.

A `ModelManager` service has also been developed using the Singleton [7] design pattern to control the lifecycle of the different AI models used in the backend, preventing redundant initialisations and ensuring memory efficiency. To address the challenge imposed by having multiple models running simultaneously, the vision-based models, Granite Vision and SAM 2, were loaded on the GPU, while the language-based model, Granite, was loaded on the CPU. This separation allowed for optimal resource utilisation and ensured smooth operation of all models without overloading the system or causing memory issues.

On the client side, the essential React Native screens have been developed, including the Home Screen, Upload Screen, non-functional Chat Interface and a Settings Screen. The UI components have been designed to be intuitive and user-friendly, complying with WCAG guidelines. The frontend now has basic capabilities for a user to upload a document, send it to the backend for processing and receive a response from the Flask API.

4.4 Ongoing Work

With the backend infrastructure and a data flow in place, the current development focus has shifted towards the frontend implementation and the integration of the user interface using React Native. A primary area of focus is the Document View screen, which allows users to view the processed document and transition between different modes, such as AR visualisation and AI assistance.

Furthermore, the functional implementation of the Chat interface is in progress. While the backend can already process user queries and generate responses, the frontend chat interface is being developed to provide a responsive and persistent chat experience. This includes features such as preserving history context, loading states and input handling, ensuring that the dialogue between the user and the AI assistant feels natural and engaging. The goal is to have a chat interface which can dynamically reference the active document context without losing track of previous interactions.

Finally, efforts are being made towards the AR view rendering and interaction capabilities. Although the backend can generate segmentation masks and component metadata, actively rendering these overlays in an AR environment and allowing users to interact with them is a complex task. Figure9 illustrates the current progress on the visual analysis pipeline, showcasing the segmentation masks generated by SAM 2.

4.5 Testing Results

Initial backend testing has been conducted using unit tests and integration tests according to the testing strategy. The unit tests for individual services have confirmed that each component functions correctly in isolation, handling various input scenarios, such as PDF files or singular images, and edge cases, such as multiple concurrent requests. Integration tests have verified that the services work together as expected, with the data flowing as intended between the different services and endpoints. For instance, the code below shows an example of a unit test for the file upload endpoint. This test simulates the upload of a dummy image file to the `/api/upload/` route, verifies that the response status code is `200 OK` and checks that the JSON response contains the expected 'status' key.

Listing 1: Test for upload endpoint

```
    def test_01_upload_endpoint(self):
        print("\n--- TEST 1: Uploading File ---")

        data = {'file': (img_byte_arr, 'test_image.png')}
        response = self.client.post(
            '/api/upload/',
            data=data,
            content_type='multipart/form-data'
        )

        self.assertEqual(response.status_code, 200)
        self.assertIn('status', response.get_json())
        print("SUCCESS: Upload returned 200 OK.")
```

Additionally, the visual analysis pipeline has been tested with the same dummy image file to verify that the system can correctly process and extract diagram components. Figure 9 illustrates the result of the visual analysis pipeline when processing a sample technical diagram, showcasing that it recognises and extracts relevant components and relationships. Similarly, the AI response generation has been tested by sending sample queries related to the processed document and verifying that the responses are contextually relevant and accurate, while still returning structured JSON data and appropriate status codes.

Furthermore, a queue stress test was conducted to simulate multiple concurrent requests to the backend, ensuring that the system can handle high loads without performance degradation or failures. The results indicated that the backend maintained stability and responsiveness under stress, with all requests being processed successfully and within acceptable time frames. However, improvements can still be made to further optimise multiple request handling and reduce server response times.

Finally, preliminary frontend testing has been performed to verify the basic functionality of the UI components and navigation flow. Figure 8 illustrates different screens of the React Native app, showcasing the Home Screen, Upload Screen and Chat Interface. Initial tests have confirmed that users can navigate between screens and receive appropriate feedback for actions such as file uploads.

4.6 Challenges Faced

The main challenge faced was handling the heavy model loads on the backend, which required careful management of system resources to prevent memory overloads and ensure fast response times. Initially, advanced models were considered to provide the user with high-quality responses. However, due to hardware limitations, it was necessary to switch to lighter models that could run efficiently on the available infrastructure. For instance, the AI LLM model was changed from IBM Granite 4.0 to Granite 3.1, which provided a good balance between performance and resource consumption. However, memory management remained a challenge. One potential solution to this issue was to implement a "lazy-loading" mechanism, where models are only loaded into memory when needed and unloaded when not in use. This approach could help free up resources and improve system performance. Unfortunately, this was not viable since it would mean a significant increase in response times, which would negatively impact the user experience. For that reason, the decision was made to keep all models running simultaneously, where computationally intensive models, such as Granite Vision and SAM 2, are running on the GPU, while lighter models, such as Granite 3.1, running on the CPU. This strategy allowed for better resource allocation and ensured that models could operate without overloading the system with every request.

5 Outlining Future Steps

5.1 Updated Timeline

Due to the adjustments made between the start of the project and the current progress, the timeline has been updated to reflect the new priorities and focus areas. An updated Gantt chart is provided in Figure 10, displaying the revised timeline for the remaining development phases, including frontend development and integration, AR rendering and interaction, optimisations and additional features, and extensive testing and user feedback incorporation.

5.2 Optimisations

As the project progresses, there will be opportunities to further optimise both the frontend and backend components. For instance, on the backend side, further improvements could be made to the model loading or request handling mechanisms to enhance performance and reduce latency. One strategy could be to adopt a multithreading approach, where requests are processed in parallel, allowing for better resource utilisation and faster response times.

On the client side, optimisations could focus on improving the AR rendering performance and interaction responsiveness. This could involve refining the AR overlay placement algorithms, adjusting rendering settings for better performance and enhancing user interactions to be more fluid and intuitive.

6 Legal, Social, Ethical and Professional Issues & Considerations

6.1 Legal Considerations

As outlined in the project specification, legal considerations primarily revolve around data privacy and intellectual property rights. The application will handle user-uploaded documents, which may contain sensitive or proprietary information. To address this, the user has full control over their data, with the option of deleting stored information available at all times. Additionally, IBM has ownership over the Intellectual Property (IP) rights of this project.

6.2 Social and Ethical Issues

There have been no identified social or ethical issues related to the project.

6.3 Professional Considerations

Throughout the development process, professional standards and best practices have been upheld. This will continue to be the case, ensuring that the project adheres to industry standards for software development, data security and user experience.

References

[1] A. Author and B. Author, "Dependency Graph Tool in Augmented Reality," in *Proceedings of Example Conference*, 2019.

[2] P. Mohr, F. Funk, and R. Malaka, "Retargeting technical documentation to augmented reality," in *2015 IEEE Virtual Reality (VR)*, pp. 83–90.

[3] PTC, "Vuforia Expert Capture," Product brief, 2023.

[4] Taqtile, "Manifest platform for industrial maintenance," Whitepaper, 2023.

[5] W3C, "Web Content Accessibility Guidelines (WCAG) 2.1," 2018.

[6] Red Hat, "What is a layered client-server architecture?," Red Hat Reference Guide, 2024. Available: `https://www.redhat.com/`.

[7] Patterns.dev, "Singleton pattern," 2024. Available: `https://www.patterns.dev/`.

Appendix

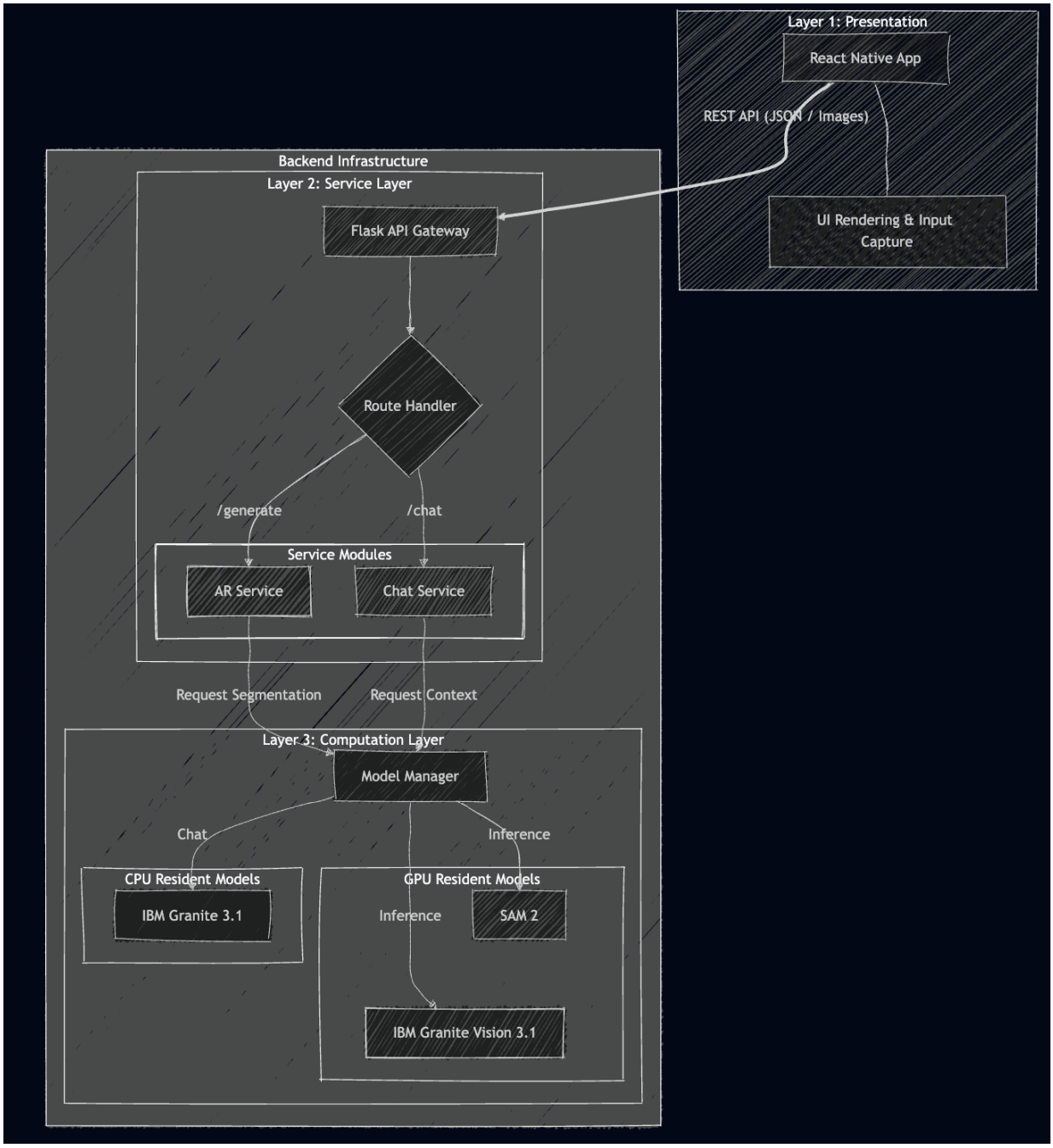


Figure 5: System architecture overview

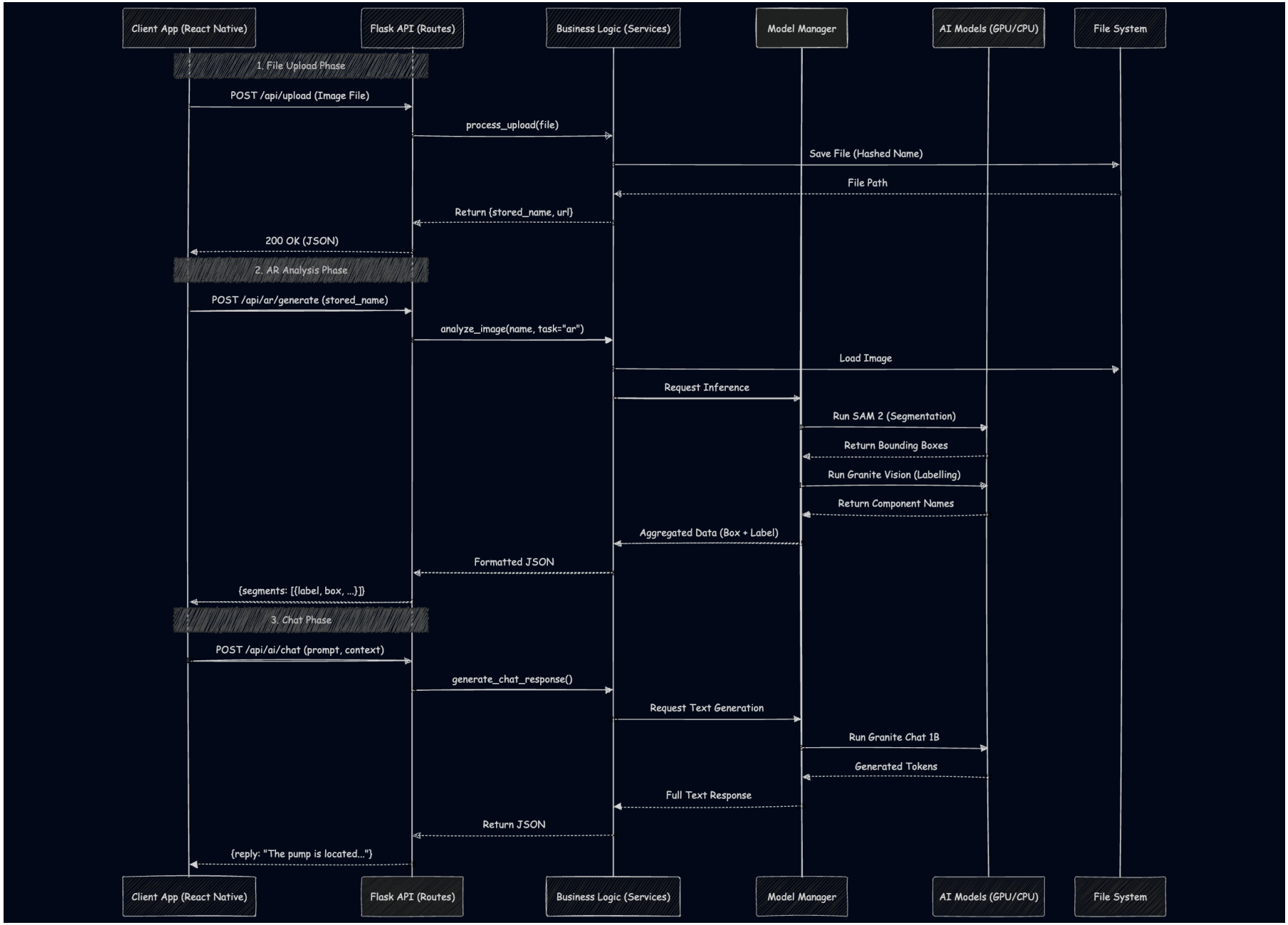


Figure 6: Frontend-backend data flow

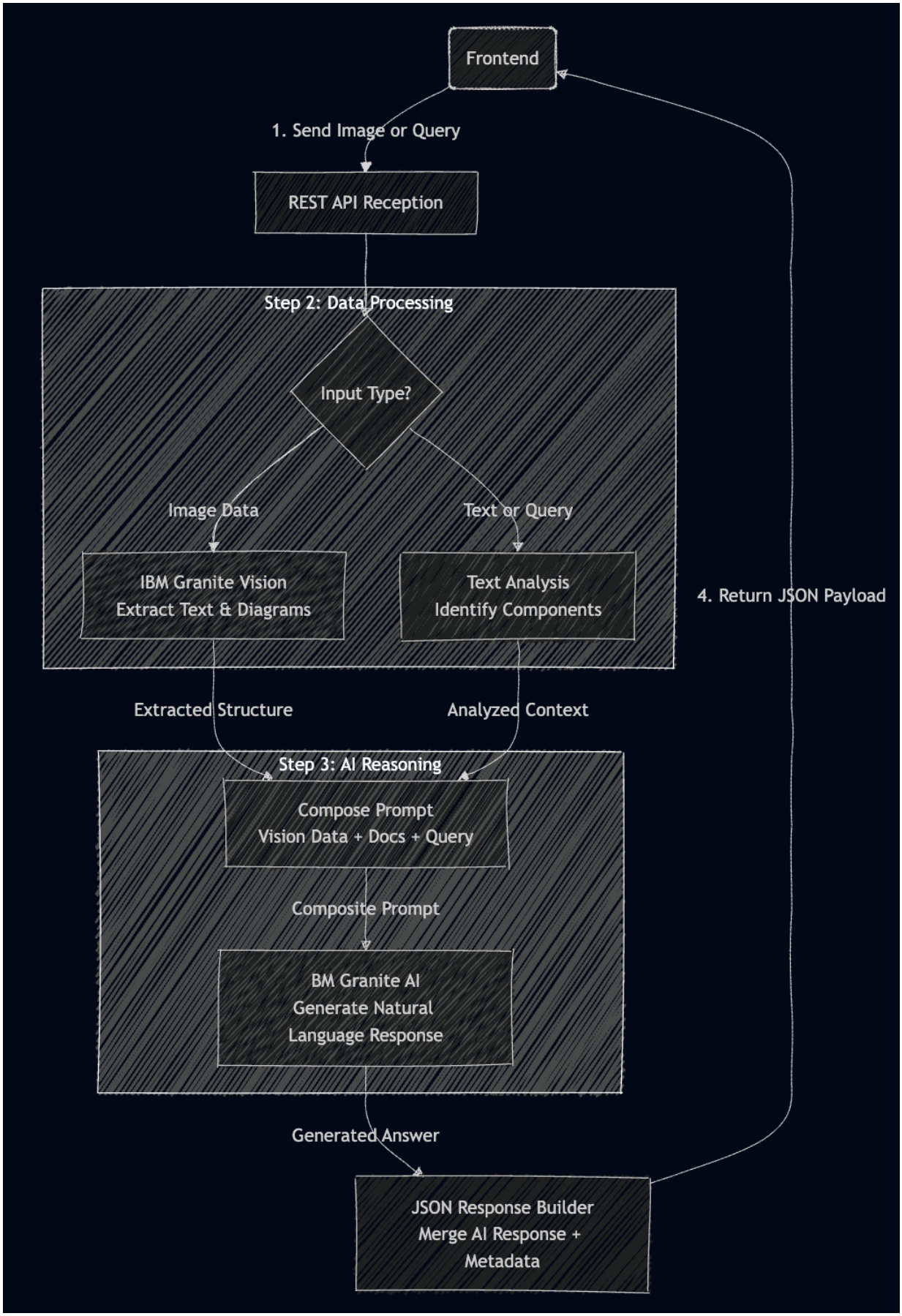
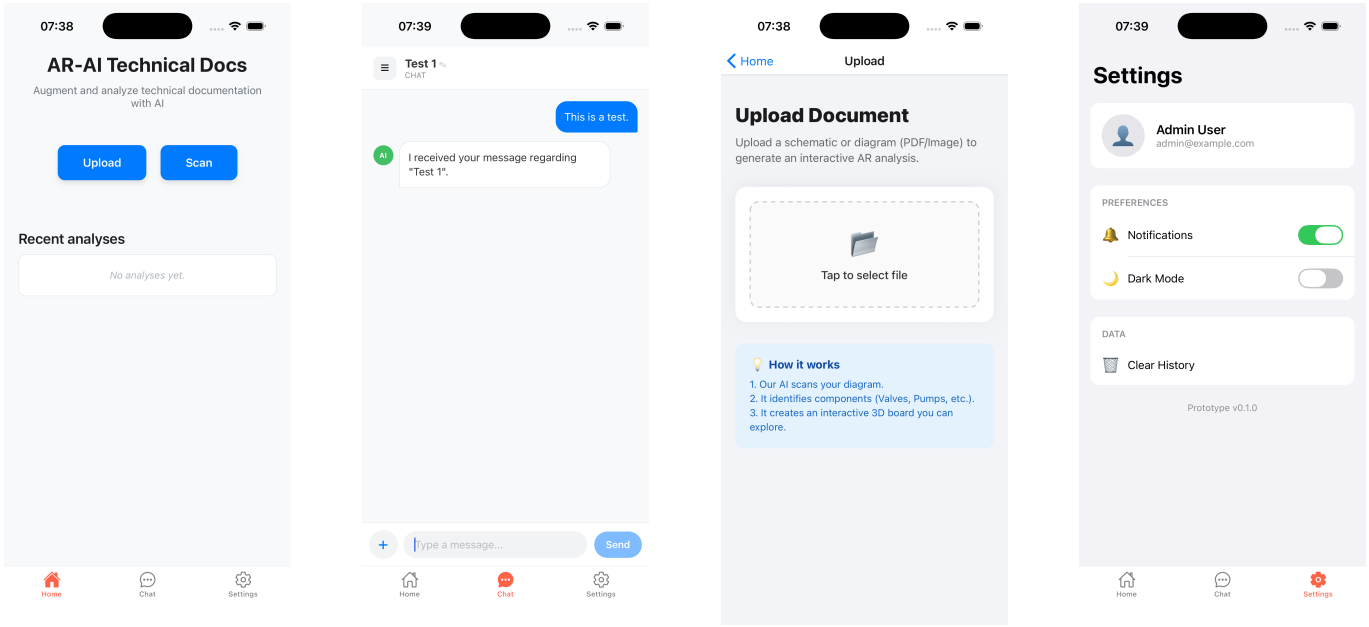
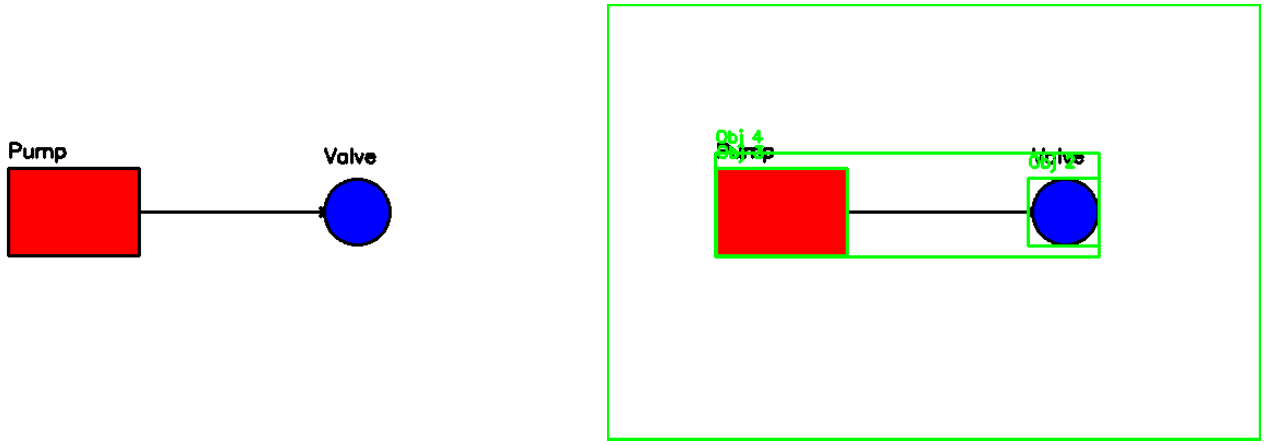


Figure 7: Backend architecture and data flow



(a) Home (b) Chat (c) Upload (d) Settings

Figure 8: Frontend screen samples (Home, Chat, Upload, Settings).



(a) Input schematic (b) AR overlay debug result

Figure 9: Visual analysis pipeline example.

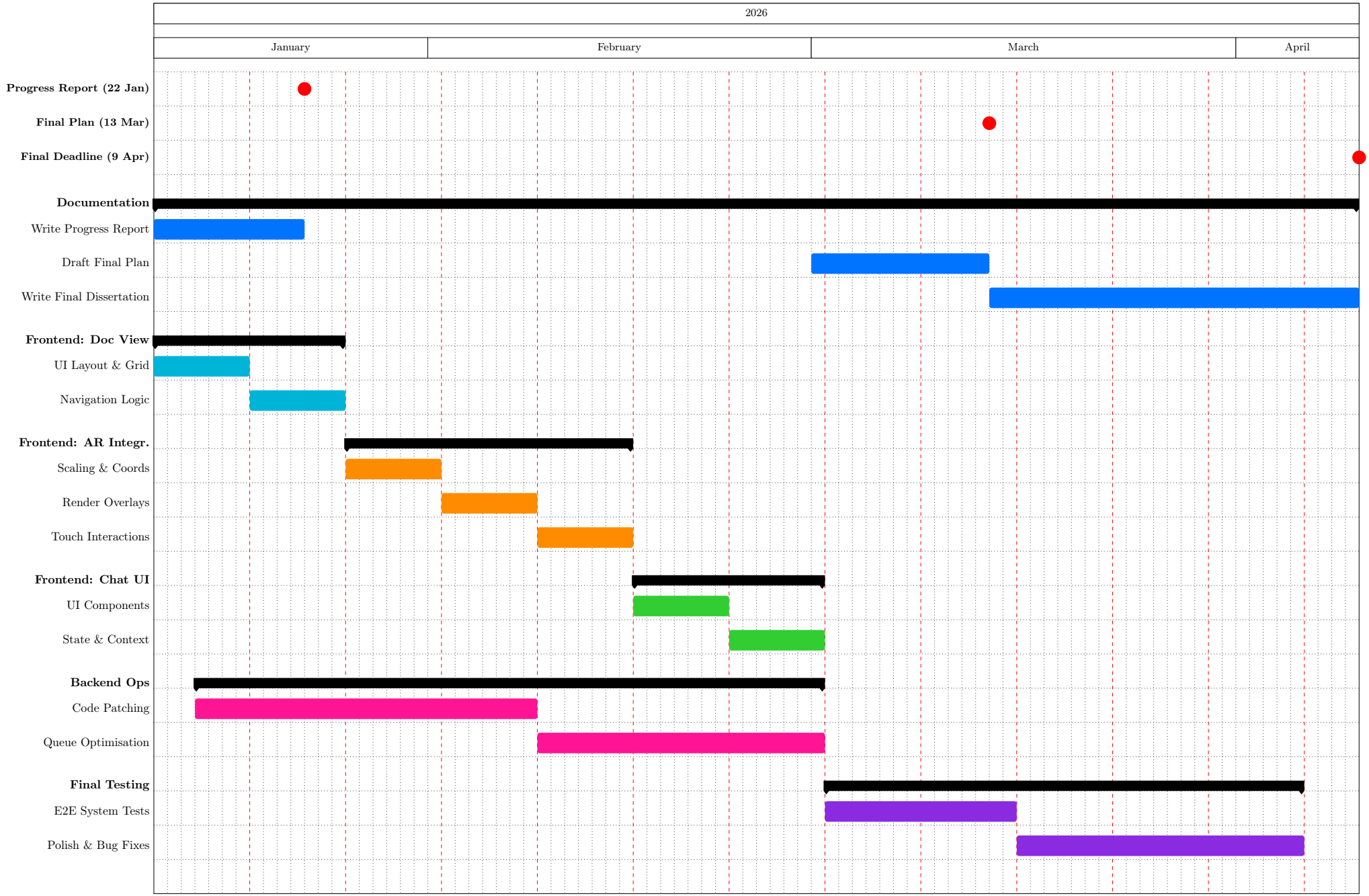
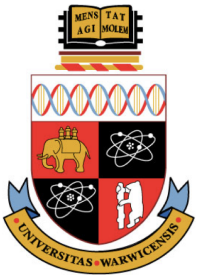


Figure 10: Revised Project Timeline



A mobile application for augmenting technical documentation using AR and AI

CS310: Third Year Project

Ani Bitri

Supervisors: Dr. Paris Giampouras, John McNamara

Department of Computer Science

University of Warwick and IBM

October 2025

1 Problem Statement

1.1 Introduction and Context

Technical documentation plays a critical role in helping developers understand, implement, operate, and maintain software systems. In enterprise environments, documentation is the primary interface between developers and complex systems, making clarity and accessibility essential. However, despite its importance, technical documentation remains predominantly static, text-based and difficult to navigate. As systems grow in complexity, dynamic processes such as data flows, dependencies and interactions are often poorly represented in traditional documentation formats. Consequently, many users struggle to find the information they need, leading to frustration, errors and inefficiencies. This underscores the need for innovative solutions that can enhance the user experience and improve comprehension of technical documentation.

1.2 Existing Technologies

There are several existing technologies which address parts of the problem but do not yet offer a comprehensive solution. These include:

- **Static and Web-Based Documentation:** Different Platforms like GitBook, ReadTheDocs and Markdown-based tools offer well-structured and accessible documentation but lack interactivity and visualization capabilities. Information is conveyed effectively through text and images but remains static and fails to capture the dynamic nature of modern software systems.
- **AI Assistants:** In the last few years, the rise of AI-powered assistants like ChatGPT and Gemini has been transformative. These tools can understand and interpret technical text, summarise content and answer user queries. Yet, these assistants operate in isolation from the documentation itself, requiring users to switch contexts and manually input queries, disconnecting users from visual elements and offering limited support for understanding system architecture and interactions.
- **Augmented Reality Applications:** Industrial AR tools, such as PTC Vuforia, Microsoft Dynamics 365 Guides and Scope AR, have been used to overlay digital information onto real-world objects. These solutions enhance spatial understanding and are particularly useful in maintenance and training contexts. However, they typically rely on specific and expensive hardware, such as the HoloLens, or complex 3D models, thus limiting their accessibility for general users.

1.3 Gaps in Current Solutions

Despite the advancements in AR and AI technologies, the current solutions exhibit several limitations:

- **Lack of integration between AR and AI:** Visualisation (AR) and comprehension (AI) are treated as separate domains. There is limited integration or implementation that combines the strengths of both to deliver a seamless user experience.
- **Limited Visual Comprehension:** Text and 2D diagrams are insufficient for conveying complex interactions and dynamic processes. Current documentation formats do not effectively utilise visual aids to enhance understanding.
- **Hardware and Platform Barriers:** Most AR implementations are dependent on specialized hardware, such as the HoloLens or AR glasses, which are costly or not widely available. This restricts the accessibility of AR-enhanced documentation to a broader audience.

1.4 Problem Definition

Given the limitations of existing documentation systems and the potential of emerging technologies, there is a need for an innovative solution that is easily accessible, interactive and capable of enhancing the understanding of technical documentation. To address this need, the project aims to develop an AI-powered mobile application that uses AR technology to augment technical documentation. The proposed system must allow users to point their mobile devices at a technical diagram and view interactive AR overlays that explain components, relationships and interactions. At the same time, an integrated AI assistant, IBM Watson/Granite, will be available to answer user queries and provide additional context. Therefore, this project aims to bridge the gap between visual comprehension and intelligent interaction in technical documentation, ultimately demonstrating how static documents can become interactive, explainable and user-friendly.

2 Project Overview

2.1 Purpose

This project aims to create an AI-powered mobile application which, supported by AR technology, will enhance the user experience by providing interactive and immersive documentation. Powered by IBM Watson/Granite, the app will offer several features to assist developers in navigating and comprehending complex technical documents, including text recognition, interactive AR overlays, chatbot assistance, and more.

2.2 Objectives within the scope

The primary objectives of this project are to:

1. Implement AR diagram augmentation
   - Detect and track diagrams in printed and digital forms.
   - Overlay interactive elements on diagrams to provide additional context and explanations.
2. Integrate an AI assistant
   - Employ IBM Watson/Granite to interpret the scanned documentation text and answer user queries.
   - Support natural language questions such as "What is the purpose of this diagram?" or "Explain this concept in simpler terms".
3. Preserve accessibility and compliance
   - Keep the core document unchanged and externalize enhancements through AR overlays to comply with accessibility and legal requirements.
4. Develop a functional mobile prototype
   - Deliver a working mobile application prototype that demonstrates the key features and functionalities.
   - Conduct user testing to gather feedback and refine the application.
   - Provide a short demonstration.

2.3 Boundaries and Out-of-Scope Elements

To keep the project achievable within the given timeframe, the following elements are considered out of scope:

- Full production deployment or enterprise-level integration.
- Hardware-specific AR is excluded; the focus is on mobile devices.
- Cross-platform optimisation beyond the primary target platform (e.g., iOS or Android).

2.4 Expected Deliverables

- A prototype mobile application demonstrating real-time recognition and overlay of technical documentation.
- Integrated AI assistant interface capable of answering user queries based on the documentation content.
- A comprehensive project report detailing the design, implementation, testing processes and evaluation results.
- VIVA presentation and demonstration.

2.5 Target Outcomes

- Enhanced user experience for developers interacting with technical documentation.
- Improved comprehension of complex technical concepts through interactive AR elements and AI assistance.
- A foundation for future development and potential commercialization of the application.

3 Methods and Methodologies

3.1 Development Methodology

Given the dynamic nature of the project and the need for iterative development, the Scrum framework within Agile methodologies will be adopted. The approach will allow for flexibility, continuous feedback and incremental delivery of features. Weekly stand-up meetings will be held to discuss progress and address any challenges. The work will be

organized into sprints, each lasting approximately one to two weeks, with clearly defined goals. At the end of each sprint, a review session will be held with the assessors to demonstrate progress and gather feedback.

3.2 Version Control and Documentation

Version control will be managed using Git and GitHub, ensuring all changes are tracked and documented. Regular commits will be made to maintain a history of the development process. This approach supports the Scrum framework by ensuring transparency, traceability and code integrity throughout the project lifecycle.

3.3 Requirements Gathering and Analysis

Requirements will be gathered and refined through continuous engagement with the assessors, ensuring the project remains aligned with the initial objectives. Any changes to the requirements will be documented and communicated promptly. Research into existing technologies and best practices will be conducted to inform design decisions and implementation strategies. This includes exploring AR frameworks, AI integration techniques and mobile development tools.

3.4 Testing and Evaluation

To enable an objective and reliable evaluation of the application, a series of tests will be conducted throughout the development process. These tests will ensure the functionality, usability and performance of the application. Moreover, user testing will be performed with a small group of target users to gather feedback and identify areas for improvement. The testing process will include:

- **Unit Testing:** Individual components and functions will be tested to ensure they work as intended.
- **Integration Testing:** The interaction between different components, such as AR overlays and AI responses, will be tested to ensure seamless functionality.
- **User Acceptance Testing (UAT):** Target users will test the application to provide feedback on usability, functionality and overall experience.
- **Performance Testing:** The application will be evaluated for responsiveness, load times and resource usage to ensure optimal performance on mobile devices.

3.5 Time Management and Milestones

The Gantt chart below outlines the planned work schedule, including key milestones and deadlines. Weekly meetings with the assessors will also be scheduled to ensure alignment and address any issues promptly.

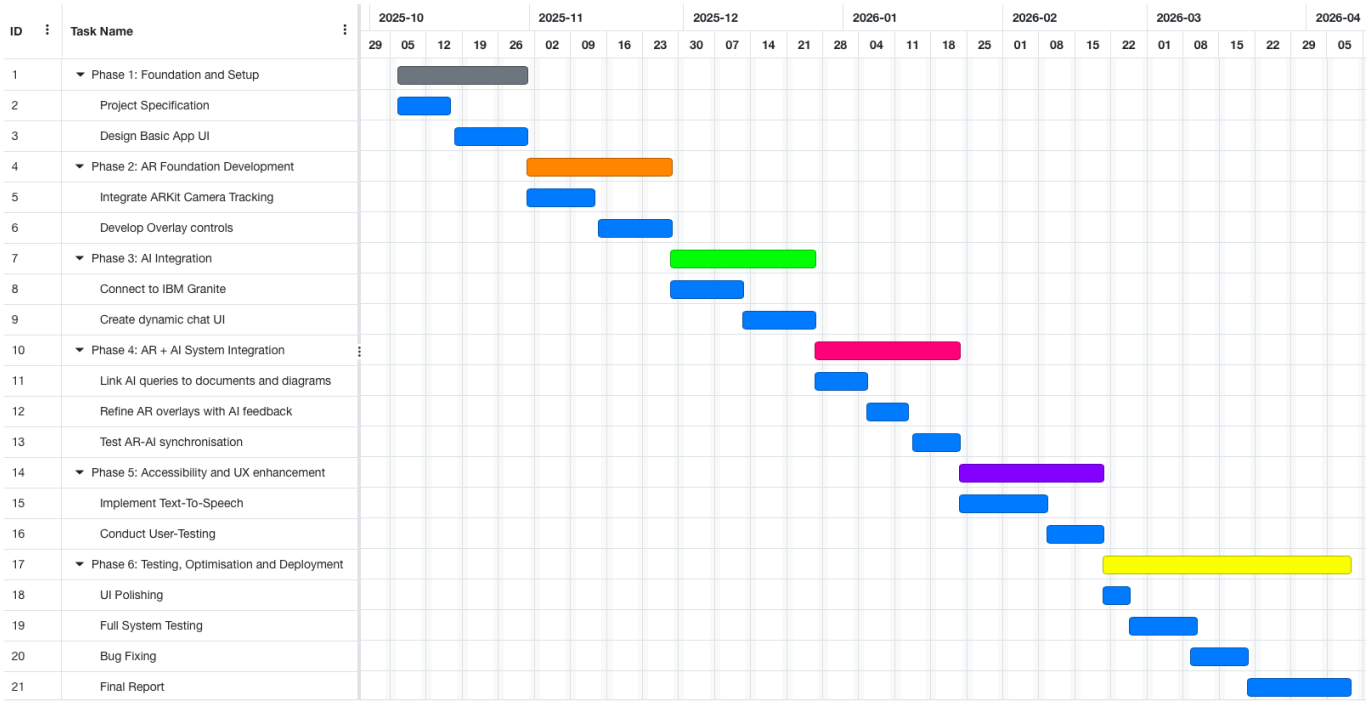


Figure 2: Gantt Chart of Planned Work Schedule

4 Resource Management

The project is primarily developed by a single developer, with support and guidance from the assessors. The developer is responsible for all aspects of the project, including design, implementation, testing and documentation. Access to the essential software tools and platforms, such as IBM Granite and IBM Cloud, is provided by IBM. Other resources, such as hardware for testing, necessary software and other materials, are handled by the developer. Except for the resources provided by IBM, the developer is responsible for acquiring any additional resources needed for the successful completion of the project. In the event of resource unavailability, such as laptop malfunction or equipment failure, the developer will rely on the university's resources provided to students.

5 Risk Management

The following risks have been identified for the project, along with their descriptions, risk levels, likelihoods and mitigation strategies.

1. Technology Limitations
    - Risk description: There is unfamiliarity with some tools, libraries, or frameworks, which may cause delays or reduced performance.
    - Risk Level: Tolerable
    - Risk Likelihood: Moderate
    - Mitigation Strategy: Conduct thorough research and allocate time for learning and experimentation with new technologies.
2. Rollback Challenges
    - Risk description: Lack of a version control system could prevent rolling back to the software's last stable state in case of errors.
    - Risk Level: Catastrophic
    - Risk Likelihood: Low
    - Mitigation Strategy: Utilise GitHub to maintain a stable main branch and update it only when confident that changes will not affect usability.
3. Testing Risks
    - Risk description: Insufficient testing may reduce confidence in the software.
    - Risk Level: Serious
    - Risk Likelihood: Moderate
    - Mitigation Strategy: Unit tests will verify functionality, and user testing will gather feedback and identify issues.
4. Time Management
    - Risk description: Underestimating task duration or improper prioritisation might result in delayed work.
    - Risk Level: Serious
    - Risk Likelihood: Low
    - Mitigation Strategy: Meetings with the assessors will be held weekly to ensure that the project is on track and any issues are addressed promptly.
5. Requirement Misalignment
    - Risk description: During development, the end product might deviate from the requirements described in the deliverables due to unforeseen circumstances.
    - Risk Level: Catastrophic
    - Risk Likelihood: Low
    - Mitigation Strategy: Regular meetings with the assessors will be held to ensure that the project is on track and any issues are addressed promptly.
6. Underestimating Sprint Workload

- Risk description: Tasks may take longer than expected due to AI and AR complexities, which may lead to incomplete sprints and delays.
- Risk Level: Serious
- Risk Likelihood: Moderate
- Mitigation Strategy: Buffer time will be allocated in each sprint to accommodate unforeseen challenges, and tasks will be prioritized to ensure critical features are completed first.

6 Legal, Social, Ethical and Professional Issues and Considerations

6.1 Legal Issues

The project will comply with all relevant legal requirements, including data protection laws (e.g., UK GDPR), intellectual property rights and software licensing agreements. The application will not store any personal data without user consent, and all third-party libraries and tools used will be properly licensed and attributed. The developer will ensure that the application does not infringe on any patents or copyrights.

6.2 Social and Ethical Issues

As of now, there are no identified social or ethical issues related to the project. However, the developer will remain vigilant and address any concerns that may arise during the development process.

6.3 Professional Issues

The project will adhere to professional standards and best practices in software development, including code quality, documentation and testing. The developer will also ensure that the application is accessible to users with disabilities, following guidelines such as the Web Content Accessibility Guidelines (WCAG).

7 Conclusion

In conclusion, this project aims to develop an application that uses Augmented Reality (AR) and Artificial Intelligence (AI) to help developers understand technical documentation better. By integrating AR overlays with an AI assistant, the application seeks to enhance the user experience, making complex technical concepts more accessible and easier to comprehend. The project will follow a structured development methodology, incorporating risk management strategies and adhering to legal, social, ethical and professional considerations. The successful completion of this project will result in a functional prototype that demonstrates the potential of combining AR and AI technologies to revolutionise the way technical documentation is presented and understood.