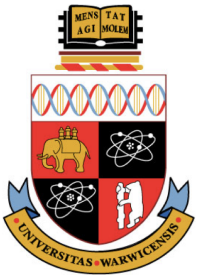


A mobile application for augmenting technical documentation using AR and AI

CS310: Third Year Project

Ani Bitri

Supervisors: Dr. Paris Giampouras, John McNamara

Department of Computer Science

University of Warwick and IBM

October 2025

1 Requirements

Each requirement will be described in the format RnC/D, where n is the requirement number and C or D indicates whether the requirement is customer (C) or developer (D) oriented. For example, R1C refers to the first customer requirement, while R2D refers to the second developer requirement. Each requirement will be detailed with its description, priority, verification method and traceability.

1.1 Functional Requirements

List the key functionalities the software system must support. Provide clear and concise descriptions of features and interactions.

1. **Augmented Reality System**

1.2 Non-Functional Requirements

Outline performance, usability, reliability, and other quality attributes expected from the system.

2 System Architecture

Provide a high-level overview of the system architecture. Include diagrams where appropriate to illustrate the system components and their interactions.

2.1 Design Pattern

The design pattern which will be adopted for this project is the Model-View-ViewModel (MVVM) pattern. This pattern is well-suited for applications that require a clear separation of concerns between the user interface and logic, making it easier to manage and test the application. Furthermore, the MVVM pattern facilitates a more modular and maintainable codebase, which is essential for the iterative development process planned for this project. The image below illustrates the MVVM architecture which will be used in the project.

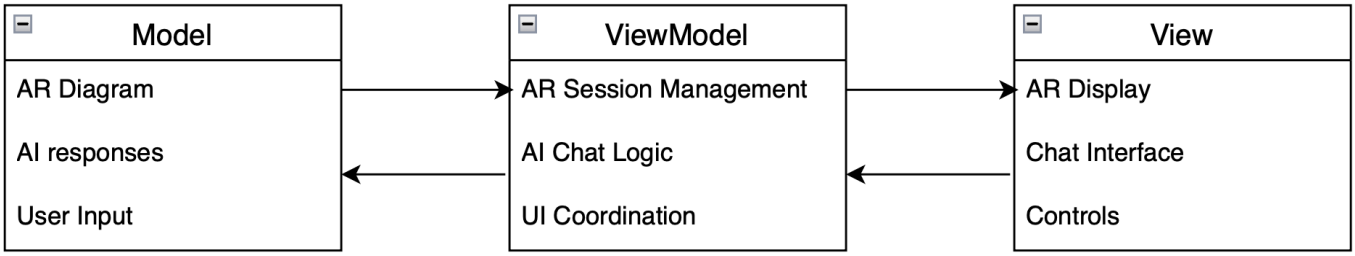


Figure 2: MVVM Architecture Pattern

2.2 Frontend Design

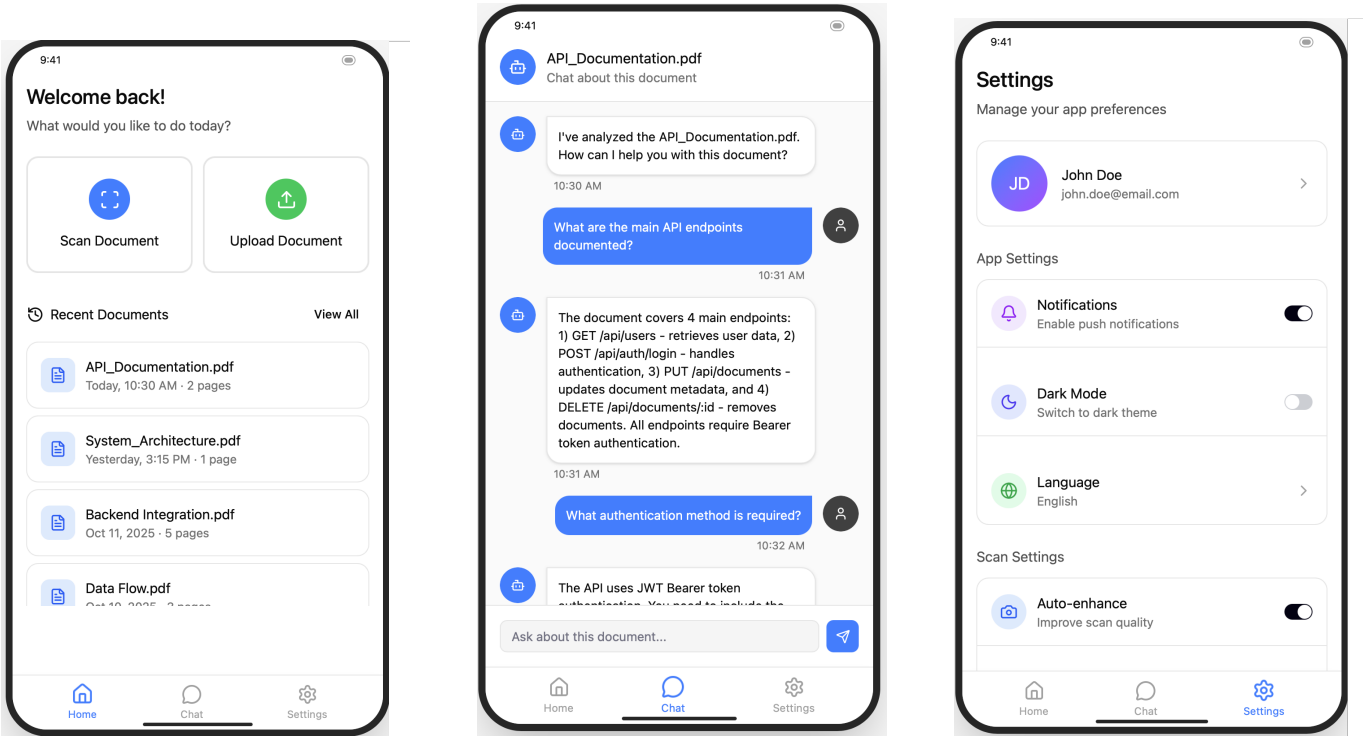
2.2.1 User Interface and User Experience

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, ensuring a seamless user experience. When the user open 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 UI design for the application:

Upon selecting the scan option, the user will be directed to the camera interface, where they can capture pictures of the technical documentation. Alternatively, the user can choose to upload the document from their device's storage and access the same features. After scanning or uploading a document, the app will process the text, images and diagrams, and give the user the option to view the AR overlays or interact with the AI assistant. The AR overlays will provide interactive elements that explain components, relationships and interactions within the diagrams based on the user's input and the information extracted from the document. The image below illustrates a flow chart of the app's user interface and navigation:

2.2.2 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



(a) Home Screen (b) Chatbot Tab (c) Settings Tab

Figure 3: Proposed UI Design for the Application

example, if the app fails to recognize 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 minimize their occurrence and impact on the user experience. Such strategies include input validation, network status checks and AR tracking optimizations. This approach will ensure that users can effectively utilize the app's features without being hindered by technical issues.

2.2.3 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. The data flows as follows:

1. The user scans or uploads a document using the app's interface.
2. The frontend validates the input and sends the document data to the backend for processing.
3. The validated data is sent to the backend services for text recognition, diagram analysis and AI processing.
4. The backend processes the data and generates AR overlays and AI responses based on the document content.
5. The generated AR overlays and AI responses are sent back to the frontend for display to the user.
6. The user interacts with the AR overlays and AI assistant, and any further queries or actions are sent back to the backend for processing.

2.3 Backend Design

2.3.1 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.

2.3.2 Workflow

The backend workflow consists of several key steps:

1. **Request Reception:** The backend receives the scanned images or uploaded from the frontend via RESTful API calls.

2. **Image and Data Processing:**
    - If an image is recieved, it is processed using IBM Granite Vision to extract text and identify diagrams, returning structured data.
    - If text data is recieved, it is analyzed to identify key components, relationships and interactions.
3. **AI Reasoning and Response Generation:** The backend composes of 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 send a single JSON payload back to the mobile app. The fronted then displays the AR overlays and AI responses to the user.

2.3.3 Data Management and Caching

To improve efficiency, the backend implements:

- **Caching:** Previously processed diagrams are stored to reduce redundant processing using Redis or local storage.
- **Error Handling:** AI or network failures return structured JSON error messages to the frontend for user notification.
- **User session logs:** Stored in a database for analytics and improvement (PostgreSQL or Firebase).

2.4 Technology Stack

The project will utilize the following technologies:

- **Frontend:** React Native for cross-platform mobile development, ARCore/ARKit/ViroReact for augmented reality functionalities.
- **Backend:** Node.js with Express for server-side logic, IBM Granite 4.0 for AI capabilities, IBM Granite Vision for image recognition, and MongoDB for data storage.
- **Development Tools:** Visual Studio Code for code editing, Git for version control, REST APIs for communication between frontend and backend.