System Design Specification of Automated House Planning and Visualization System for Real Estate Using Machine Learning

Name	Registration Number	Index Number
L. G. R. J. Lindapitiya	ICT/19/20/132	5066
R. M.V. P. B. Udagama	ICT/19/20/138	5071
B. M. C. B. K. Bandaranayake	ICT/19/20/016	4957
A. G. N. D. Kaluwelgoda	ICT/19/20/059	4997
T. M. M. M. B. Abeysinghe	ICT/19/20/002	4944

2024/03/18

Abstract

Traditional home design systems often struggle to accommodate the complexities posed by irregularly shaped land parcels, leaving individuals with limited options to realize their dream homes. In response, this project introduces an innovative solution poised to transform the design experience for such scenarios.

Our approach centers on automating the generation of floor plans, whether flat or angled, and providing immersive 3D visualizations to empower users in crafting their ideal living spaces. Through the integration of intuitive design tools and techniques, our system aims to democratize the process of personalized home design.

Key considerations such as usability testing, web application development, database management, and user interface optimization have been meticulously addressed to ensure a seamless user experience. Compliance, security, and thorough documentation further underpin the reliability and trustworthiness of our solution.

Delving into the realm of visual elements, we explore aspects of layout, typography, color palette, and icons to foster an aesthetically pleasing and engaging interface. Responsiveness, accessibility, and interactive elements are prioritized to accommodate diverse user needs and preferences.

Moreover, our project encompasses the implementation of state machines, specialized algorithms, and efficient processes to drive effective execution and performance. Feedback mechanisms, language clarity, brevity, and user assistance mechanisms enrich the user journey, enhancing comprehension and engagement throughout.

Acknowledgement

We wish to express our deepest gratitude to our supervisor, who has been the guiding force behind the development of this Software Design Specification (SDS) document for the automated house planning and visualization system. His expertise, mentorship, and unwavering support have been invaluable throughout every stage of this university research project.

We also acknowledge the significant contributions of our supervisor, who not only provided direction and corrected our mistakes but also shared valuable insights and knowledge as one of our esteemed lecturers. His dedication to our learning and project success has been truly commendable.

Furthermore, we extend our appreciation to all team members and lecturers who have played a role in shaping this document and project. It is through their collective effort and collaboration that we have been able to reach this milestone in our research journey.

Contents

List of Tables and Figures

1. INTRODUCTION

1.1. Problem to be addressed

The issue with current manual home design systems is their limitation to rectangular plots, leaving irregular and angled lands with inadequate solutions. Moreover, these systems lack fully automated 3D house views, focusing mainly on rectangular and square land shapes and neglecting sloped or non-standard plots. This limitation proves challenging for individuals with uniquely shaped land parcels, hindering them from creating house plans that optimize both space and aesthetics.

In the contemporary landscape, numerous house planning systems exist. However, upon closer examination, we discovered that most of them are manual or only partially automated. These systems prove to be time and cost-consuming, restricted to rectangular or square survey plans, and lacking 3D modeling capabilities. This limitation could pose a problem for some individuals with different needs. Traditional house planning methods within the architectural realm also share similar drawbacks, being both time and cost-intensive.

1.2. Objectives of the Project

Objective	Description
Generate Floor Plans	Create a plan for houses based on what users want, like how many rooms and their sizes.
Create 3D House Views	Make detailed pictures of houses so users can see what they look like before building them.
Build Easy-to-Use Website	Make a website where people can design and see their dream houses easily.
Completion Time	Finish making the plans and website within specific time frames.

Table 1: Objectives of the project

- Specific:
    - Develop an algorithm to generate floor plans based on user inputs like room configurations, dimensions, and preferences.
    - Implement a feature to create detailed 3D views of houses automatically.
    - Create an intuitive and accessible web application for designing and visualizing dream homes.
- Measurable:
    - The algorithm should accurately generate floor plans based on user-provided inputs.
    - The 3D views should provide detailed visualizations of the house designs.
    - The web application should allow users to design and visualize their dream homes effectively.
- Achievable:

    - The algorithm will utilize conditional generative models and convolutional neural networks to ensure accurate floor plan generation.
    - Implementing 3D visualization features will require integrating appropriate software libraries and tools for rendering realistic views.
    - Developing a web application using modern web development frameworks and technologies can provide the required functionality and accessibility.
- Relevant:
    - Providing users with precise floor plans and detailed 3D views aligns with the goal of helping them make informed decisions about their architectural choices.
    - Creating an intuitive and accessible web application addresses the need for a user-friendly platform to design and visualize dream homes effectively.
- Time-bound:
    - Aim to complete the algorithm development and integration within three months.
    - Implement the 3D visualization feature within two months after completing the algorithm.
    - Launch the web application within six months, incorporating both the algorithm and 3D visualization features for users to access.

1.3. Project Deliverables

- A web application that automatically generates house floor plans for regular and irregular land shapes based on user inputs.
- An algorithm that can create flat or angled floor plans for homes other than square and rectangular shapes.
- A feature that produces 3D views of the house designs to allow users to visualize their designs.
- A database system capable of storing a minimum of 100 user-generated 2D floor plans and 3D views.
- Compatibility with major web browsers and mobile devices as specified in the non-functional requirements.
- A user interface optimized for a display resolution of 1920x1080 pixels with a System Usability Scale (SUS) score of at least 10 in user testing.

- Compliance with IEEE guidelines, Rajarata University's Faculty of Applied Sciences guidelines, and ISO 27001 standards for information security management.
- Implementation of strong hashing methods to secure user passwords.
- Compliance with GDPR regulations for user privacy and data rights.
- Ensuring the web application does not include any features or content that pose a safety risk to users

1.4. System design approach

- The system design approach for this project combines the SCRUM model with a 3-tier architecture to facilitate efficient development and deployment. Using the SCRUM methodology, the project is divided into sprints, each focusing on specific functionalities like user authentication and generating 2D/3D views. The 3-tier architecture segregates presentation, application logic, and data layers, ensuring modularity and scalability. This design approach fosters collaborative development, with agile practices emphasizing adaptability and continuous improvement. By leveraging SCRUM and 3-tier architecture, the project aims to deliver a robust and responsive system tailored to user requirements.

1.4.1. Process models

- **Agile Software Model**

Functionality	Agile Approach	Activity Diagrams
Create House Floor Plans	Develop the plan step by step. Make it better each time. Listen to users.	Keep changing the plan based on user ideas.
3D House Views	Show the house in 3D. Add more details slowly. Ask people what they think.	Improve the 3D view based on feedback.
Manage User Accounts	Make it easy for people to make and delete accounts. Add more features over time.	Keep improving how users make and delete accounts.
Upload Survey Plan Image	Let people add their pictures early. Make it better as we go. Listen to feedback.	Change how pictures are added based on what people say.
Calculate Land Area	Figure out how big the land is. Make it more accurate later. Ask people if it's helpful.	Keep improving how we find out how big the land is.
Download Floor Plans and 3D Views	Let people download plans and views. Make it better later. Listen to what people want.	Change how plans and views are downloaded based on feedback.
User Authentication	Make it easy for people to log in. Add security features over time. Listen to feedback.	Keep improving how people log in and how safe it is.
Admin Privileges	Give admins the power to do things. Add more features over time. Listen to what admins need.	Change what admins can do based on what they say.
Image Validation	Check if pictures are good. Make the checks better later. Ask people if it works.	Keep improving how we check pictures based on feedback.
Generate and Display 2D Floor Plan	Make a simple floor plan first. Make it better each time. Ask people if they like it.	Change the floor plan based on what people say.

Generate and Display 3D View	Show a basic 3D view first. Make it more detailed later. Listen to feedback.	Improve the 3D view based on what people want.
Save Floor Plan and 3D View	Let people save their plans and views. Make it better over time. Ask people if it's helpful.	Change how we save plans and views based on feedback.

Table 2: Agile Approach and Activity Table

- Agile Principles:
  - Individuals and interactions over processes and tools: Emphasizes the importance of communication and collaboration within the development team and with stakeholders.
  - Working software over comprehensive documentation: Prioritizes delivering functional software incrementally, focusing on tangible outcomes rather than exhaustive documentation.
  - Customer collaboration over contract negotiation: Encourages continuous engagement with customers to gather feedback and adapt to changing requirements.
  - Responding to change over following a plan: Advocates for flexibility and responsiveness to changing needs, allowing for adjustments throughout the development process.

Functional Requirements and Activity Diagrams Aligned with Agile:

Iteration /Sprint	Function	Deliverable	Task	Timeline
1	User Authentication	Implement user sign-up and sign-in functionality.	Design user interface for registration and login forms. Develop backend logic to handle user authentication using secure	Complete within Sprint 1.

			password hashing. Write unit tests to ensure proper validation of user credentials.	
	Upload Survey Plan Image	Enable users to upload survey plan images.	Implement image upload functionality with validation for supported formats (e.g., JPEG, PNG). Store uploaded images securely in the database.	Complete within Sprint 1.
	Generate and Display 2D Floor Plan	Implement basic floor plan generation and rendering.	Develop algorithms to generate 2D floor plans based on user-provided room dimensions. Integrate rendering components to display generated floor plans.	Complete within Sprint 1.
2	Admin Privileges	Implement basic admin functionalities (e.g., user account management).	Design admin dashboard for managing user accounts. Implement CRUD operations for user	Complete within Sprint 2.

			management (view, create, update, delete).	
	Calculate Land Area	Develop algorithms to calculate land area based on user-provided data.	Define data structures and methods to calculate land area from room dimensions. Write tests to validate accuracy and correctness of land area calculations.	Complete within Sprint 2.
3	Generate and Display 3D View of House	Enhance 3D visualization features for house views.	Integrate 3D rendering libraries to improve visual fidelity and interactivity. Optimize performance for rendering complex 3D models.	Complete within Sprint 3.
	Download Floor Plans and 3D Views	Implement functionality to download saved floor plans and 3D views.	Design download options for users to access their saved designs. Implement backend logic to serve downloadable files securely.	Complete within Sprint 3.

	Image Validation	Continuously improve image validation capabilities	Enhance image processing algorithms to verify image clarity and correctness. Incorporate user feedback to refine image validation criteria.	Ongoing refinement throughout all sprints based on user feedback.

Table 3: Functional Requirements and Activity Diagrams Aligned with Agile

3-Tier Architecture

- The Automated House Planning and Visualization System will be designed using a three-tier architecture to ensure modularity, scalability, and maintainability of the system. Each tier will have specific responsibilities and interactions, contributing to the overall functionality of the system.

- Presentation Layer:

  The presentation layer, also known as the user interface layer, will be responsible for interacting with users and presenting information to them. In this system, the presentation layer will include the web interface where users can sign up, log in, upload images, provide inputs, view 2D floor plans, and visualize 3D views of their house designs. This layer will focus on providing a user-friendly interface for seamless interaction with the system.

- Application Layer:

The application layer, also known as the business logic layer, will contain the core functionality of the system. It will handle the processing of user inputs, generation of 2D floor plans and 3D views, validation of data, and communication between the presentation layer and the data layer. This layer will implement the algorithms for automatically generating floor plans and 3D views based on user-provided inputs.

- Data Layer:

  The data layer, also known as the database layer, will be responsible for storing and managing the system's data. It will store user accounts, uploaded images, generated floor plans, 3D views, and other relevant information. The data layer will ensure data integrity, security, and efficient retrieval of information for the application layer to process and present to users

Layer	Functions	Methodologies	Tools	Frameworks/Libraries
Presentation Layer	User Authentication	User-Centered Design (UCD) or Design Thinking	Visual Studio Code, Adobe Photoshop	appear.js, bootstrap.min.js, jquery.js, popper.min.js
	Generate and Display 2D Floor Plan			
	Generate and Display 3D View of House			
	Download Floor Plans and 3D Views			
Application Logic Layer	Generate House Floor Plans	Agile Software Development	Visual Studio	Google Colab, Keras

(Business Logic Layer)	3D House Views		Code, Anaconda	
	User Account Management			
	Upload Survey Plan Image			
	Calculate Land Area			
Data Access Layer	Save Floor Plan and 3D View	Relational Database Design	Cloud SQL, FireStore	Google Cloud
	View and Delete Account (Admin)			
	Image Validation			

Table 4: Methodologies and tools

- **SCRUM Model**:

Implementation	Tasks	Sprint
1st Implementation	- Develop models -Identify and implement algorithms -Setup database and schema - Design user interface - Integrate TensorFlow and Keras -Implement basic functionalities -Conduct initial testing	2 months
2nd Implementation	- Review and address errors from initial testing - Refine user interface - Enhance system functionalities - Optimize algorithms - Improve database performance - Conduct comprehensive testing	1 month
3rd Implementation	- Validate system against requirements - Address any remaining issues - Finalize documentation - Prepare for deployment - Conduct user acceptance testing	1 month

Table 5: SCRUM View

- **1st Implementation: Model Completion**
  - ✓ Backend Development:
    - Define database schema and structure.
    - Implement data management functionalities (organizing, storing, processing).
    - Develop algorithms for data processing tasks.
  - ✓ Frontend Development:
    - Design UI/UX wireframes and layouts.
    - Develop frontend components for user interaction.
  - ✓ Security and Deployment:
    - Research and plan security measures.
    - Set up initial deployment environments.

- **2nd Implementation: Error Correction**
  - ✓ Backend Development:
    - Debug and refine data management functionalities.
    - Address any issues with database schema.
    - Refactor and optimize algorithms.
  - ✓ Frontend Development:
    - Debug and refine UI components.
    - Ensure seamless integration with backend functionalities.
  - ✓ Security and Deployment:
    - Implement security measures and address any vulnerabilities.
    - Fine-tune deployment environments for stability and scalability.

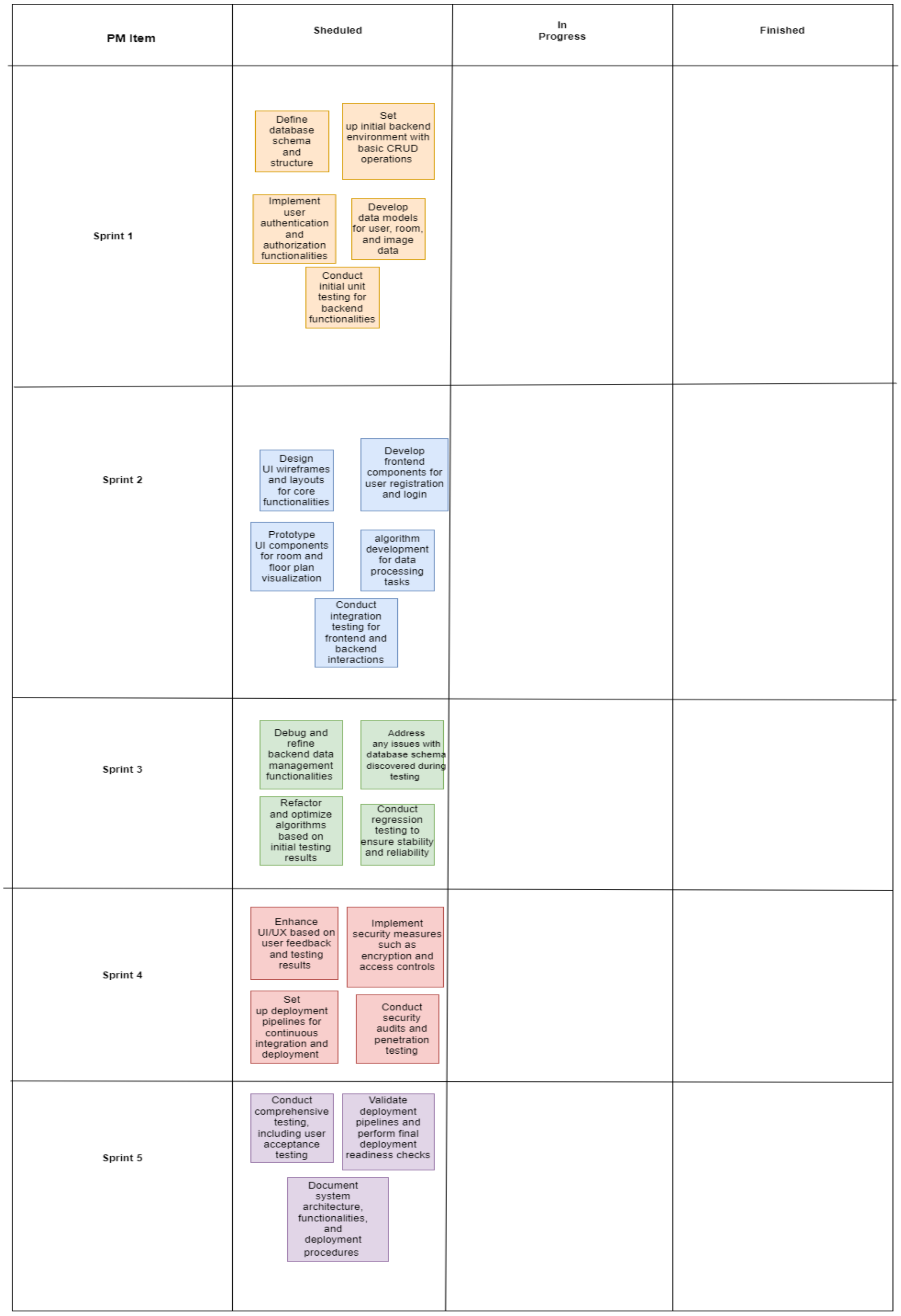
- **3rd Implementation: Testing and Validation**
  - ✓ Backend Development:
    - Conduct unit tests for data management and algorithms.
    - Implement error handling and logging mechanisms.
  - ✓ Frontend Development:
    - Perform user acceptance testing (UAT) for UI/UX.
    - Ensure compatibility across different devices and browsers.
  - ✓ Security and Deployment:
    - Conduct security audits and penetration testing.
    - Validate deployment pipelines for reliability and automation.

Sprint Overview:

- **Sprint 1: Backend Setup and Database Schema (2 weeks)**
    - Tasks:
        - ✓ Define database schema and structure (2 days).
        - ✓ Set up initial backend environment with basic CRUD operations (3 days).
        - ✓ Implement user authentication and authorization functionalities (3 days).
        - ✓ Develop data models for user, room, and image data (3 days).
        - ✓ Conduct initial unit testing for backend functionalities (1 day).
    - Goal:
        - ✓ Complete the foundational backend setup and database schema.
- **Sprint 2: Frontend Prototyping and Algorithm Development (3 weeks)**
    - Tasks:
        - ✓ Design UI wireframes and layouts for core functionalities (3 days).
        - ✓ Develop frontend components for user registration and login (4 days).
        - ✓ Prototype UI components for room and floor plan visualization (5 days).
        - ✓ Begin algorithm development for data processing tasks (5 days).
        - ✓ Conduct integration testing for frontend and backend interactions (2 days).
    - Goal:
        - ✓ Prototype frontend interfaces and initiate algorithm development.
- **Sprint 3: Error Correction and Algorithm Refinement (2 weeks)**
    - Tasks:
        - ✓ Debug and refine backend data management functionalities (5 days).
        - ✓ Address any issues with database schema discovered during testing (3 days).
        - ✓ Refactor and optimize algorithms based on initial testing results (5 days).
        - ✓ Conduct regression testing to ensure stability and reliability (2 days).
    - Goal:
        - ✓ Resolve errors and refine algorithms for improved performance.
- **Sprint 4: UI/UX Enhancement and Security Measures (3 weeks)**
    - Tasks:
        - ✓ Enhance UI/UX based on user feedback and testing results (7 days).
        - ✓ Implement security measures such as encryption and access controls (7 days).

        - ✓ Set up deployment pipelines for continuous integration and deployment (7 days).
        - ✓ Conduct security audits and penetration testing (5 days).
    - Goal:
        - ✓ Improve user experience, implement security measures, and prepare for deployment.

- **Sprint 5: Testing, Validation, and Documentation (2 weeks)**
    - Tasks:
        - ✓ Conduct comprehensive testing, including user acceptance testing (UAT) (7 days).
        - ✓ Validate deployment pipelines and perform final deployment readiness checks (5 days).
        - ✓ Document system architecture, functionalities, and deployment procedures (4 days).
    - Goal:
        - ✓ Ensure system stability, validate deployment readiness, and prepare documentation for release.



PM Item	Sheduled	In Progress	Finished
Sprint 1	Define database schema and structure; Set up initial backend environment with basic CRUD operations; Implement user authentication and authorization functionalities; Develop data models for user, room, and image data; Conduct initial unit testing for backend functionalities		
Sprint 2	Design UI wireframes and layouts for core functionalities; Develop frontend components for user registration and login; Prototype UI components for room and floor plan visualization; algorithm development for data processing tasks; Conduct integration testing for frontend and backend interactions		
Sprint 3	Debug and refine backend data management functionalities; Address any issues with database schema discovered during testing; Refactor and optimize algorithms based on initial testing results; Conduct regression testing to ensure stability and reliability		
Sprint 4	Enhance UI/UX based on user feedback and testing results; Implement security measures such as encryption and access controls; Set up deployment pipelines for continuous integration and deployment; Conduct security audits and penetration testing		
Sprint 5	Conduct comprehensive testing, including user acceptance testing; Validate deployment pipelines and perform final deployment readiness checks; Document system architecture, functionalities, and deployment procedures		

Figure 1: SCRUM Plan

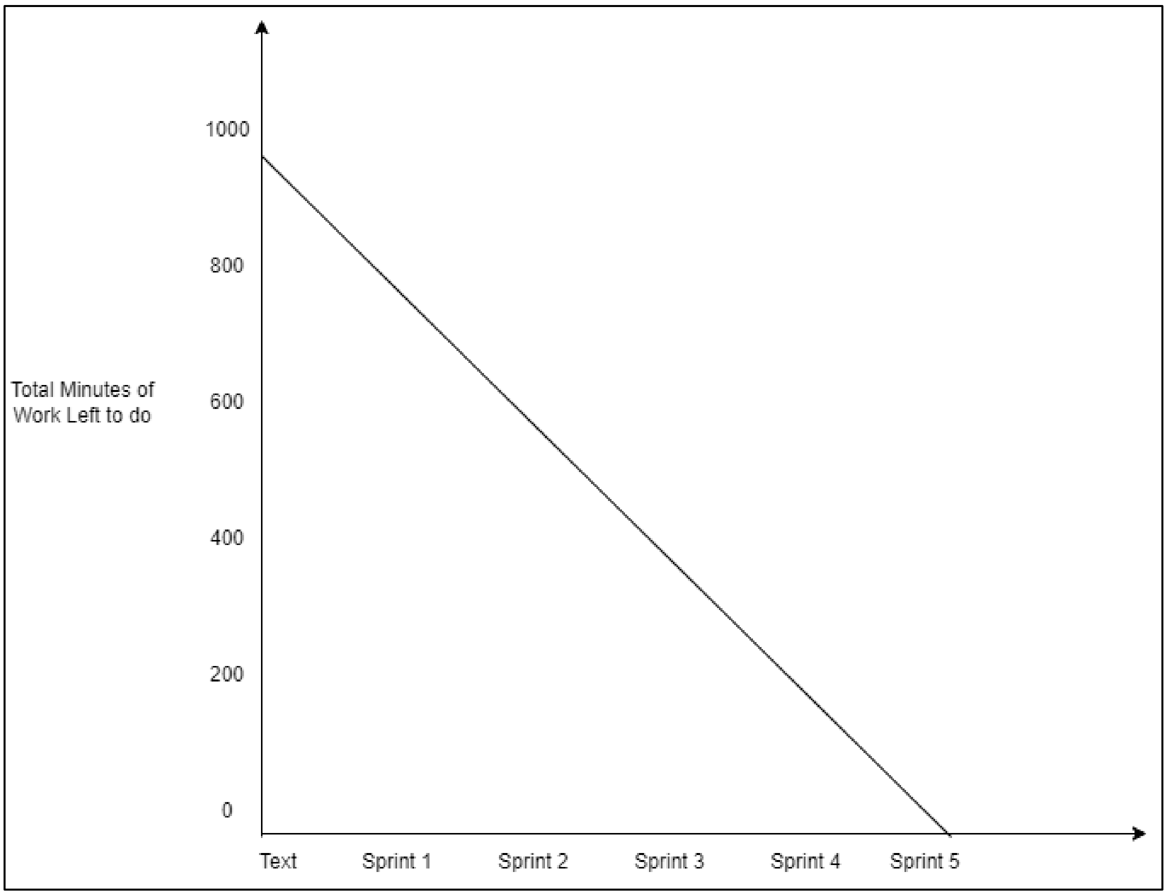


Figure 2: Burndown Chart

1.5. Standards to be followed

Layer	Requirement	Details
Presentation	Usability	- Optimize UI for 1920x1080 display.
		- Achieve SUS score ≥ 10.
Application	Performance	- Store 100+ user-generated plans/views.
		- Limit storage to 1 TB.
	Scalability	- Handle larger file sizes.
	Interoperability	- Compatible with major browsers/devices.
Data	Documentation	- Follow IEEE and university guidelines.
	Standard	- Comply with ISO 27001.
	Security	- Use strong password hashing.
External	Privacy	- GDPR compliance.
		- Notify breaches within 72 hours.
	Safety	- Ensure web application safety for users.

Table 6: System Layer Requirements

Overview of System Layer Requirements:

- Presentation Layer (User Interface):
  - Usability:
    - The user interface should be optimized for a display resolution of 1920x1080 pixels.
    - Ensure a System Usability Scale (SUS) score of at least 10 in user testing.
- Application Layer (Business Logic Layer):
  - Performance:
    - Store a minimum of 100 user-generated 2D floor plans and 3D views in the database.
    - Limit database storage allocation for user-generated content to 1 TB.
  - Scalability:
    - Handle larger file sizes, especially for GLTF/GLB files with standard-resolution textures (1024x1024 to 2048x2048 pixels).
  - Interoperability:
    - Ensure compatibility with major web browsers (Chrome version 122 or higher, Microsoft Edge version 121 or higher, Opera version 107

or higher) and mobile devices (iOS version 14 or higher, Android version 10 or higher).

- Data Layer (Data Access Layer):
  - Documentation:
    - Ensure compliance with IEEE guidelines and Rajarata University's Faculty of Applied Sciences, Department of Computing Learning Management System Software Requirement Specification Submission guidelines.
  - Standard:
    - Comply with ISO 27001 standards for information security management, ensuring the protection of user data and privacy.
  - Security:
    - Use a strong hashing method to keep passwords secure.
  - External Requirements:
    - Privacy Requirements:
      - Comply with GDPR regulations to protect user privacy and data rights according to EU standards.
      - Store and process user data in compliance with relevant data protection laws, with a data breach notification time frame of 72 hours.
    - Safety Requirement:
      - Ensure that the web application does not include any features or content that pose a safety risk to users, providing a safe and secure user experience.

1.6. Organization of the SDS

- The Software Design Specification (SDS) document for the automated house planning and visualization system is organized into different sections to explain how the system will be designed and what standards and processes will be followed.

- Here is a simple breakdown of the document:

1.6.1. Introduction:

- This part explains why the document exists and what it covers.

1.6.2. System Design Approach:

- It talks about how the project will be divided into smaller parts for easier development and how different parts of the system will work together.

1.6.3. Process Models:

- It describes the methods used to develop the system and shows diagrams for different functions like creating house plans and managing user accounts.

1.6.4. Standards to be Followed:

- This section lists the rules and requirements that the system must meet, such as being easy to use and secure.

1.6.5. Sprint Tasks Overview:

- It explains the specific tasks that will be done in each development phase, like creating user logins and generating 3D views.

1.6.6. Validation Methods and Measurements:

- This part talks about how the system will be tested to make sure it works well and is easy to maintain.

1.6.7. External Requirements:

- It covers additional needs like privacy rules and safety measures to protect users.

1.6.8. Writing Code Safely:

- This section gives advice on writing code that is secure and how to handle emergencies like data loss.

1.6.9. Research Design:

- It details the specific design choices for the system, like how the user interface will look and what technologies will be used.

1.6.10. Conclusion:

- It wraps up the document by summarizing the main points and emphasizing the importance of meeting user needs and project goals.

This organization helps ensure that everyone involved in the project understands how the system will be built and what standards it needs to meet for a successful outcome.

2. ARCHITECTURAL DESIGN

- This chapter includes the system architecture, objects and communication, state machines and special algorithms, techniques, libraries, and implementation environments in detail.

2.1. System Architecture

- The system architecture outlines how all the parts of the software system fit together and collaborate to accomplish its goals. It covers the design and arrangement of components, modules, and their interactions, giving a broad view of the system's structure and functionality.

2.1.1. High-Level Architecture

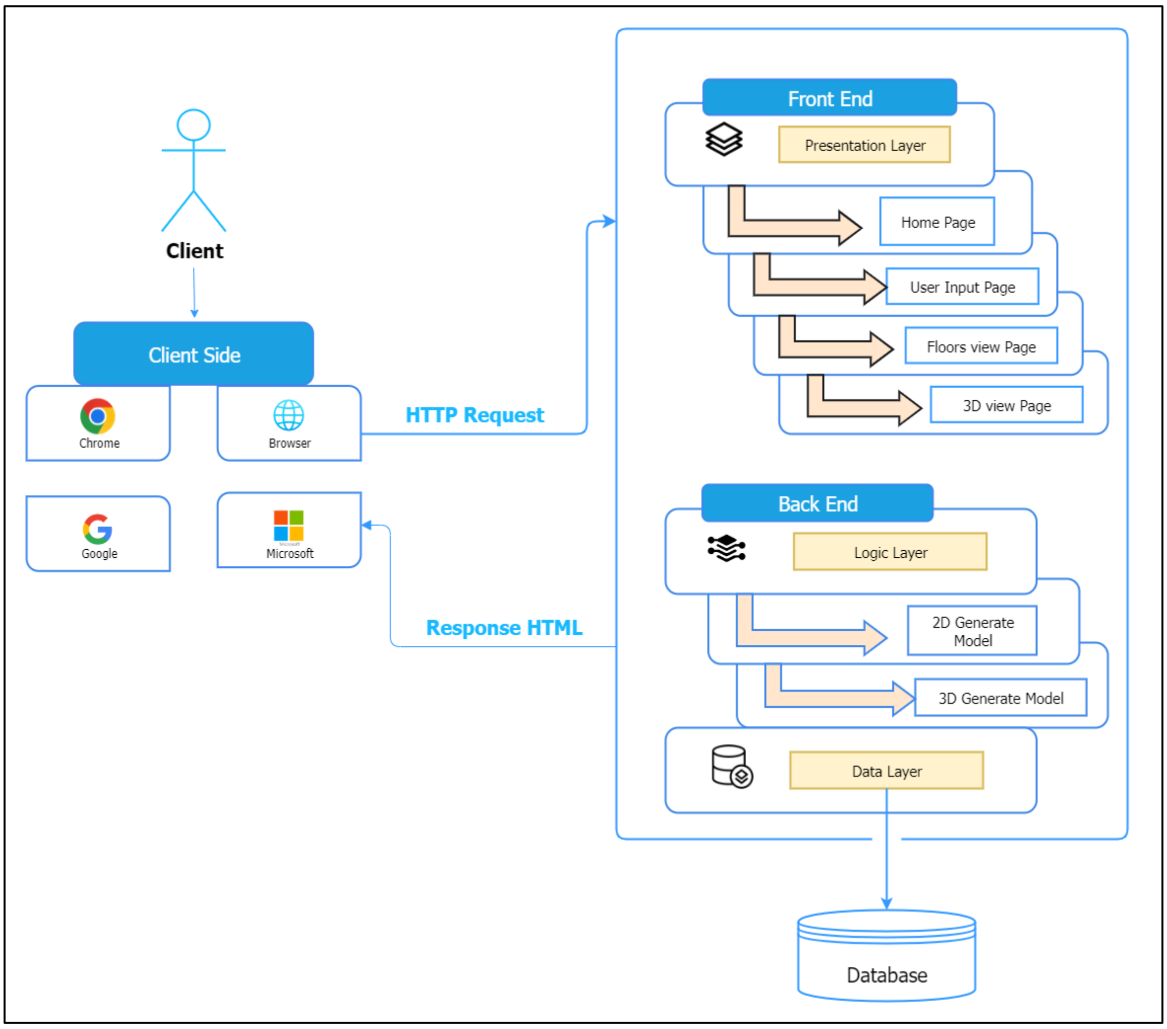


Figure 3: High-Level Architecture diagram of system

2.1.2. Component Diagram

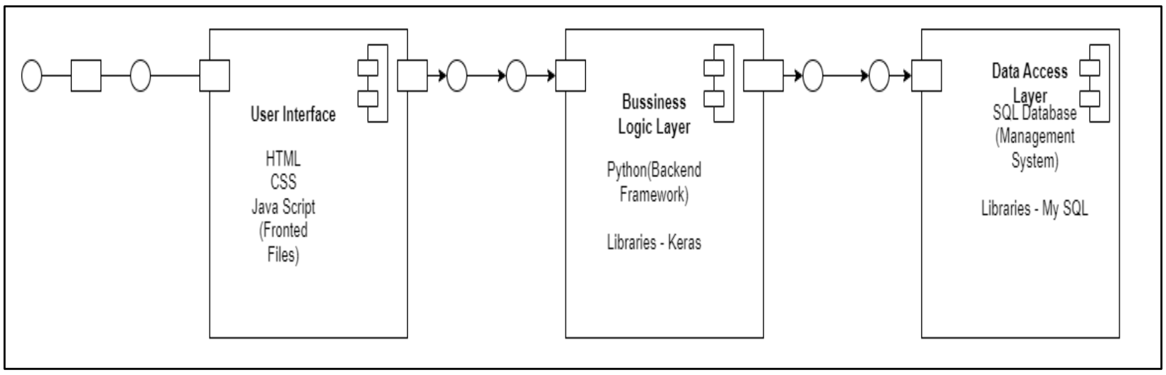


Figure 4: Component Diagram

2.2. Objects and communication

2.2.1. Class diagram

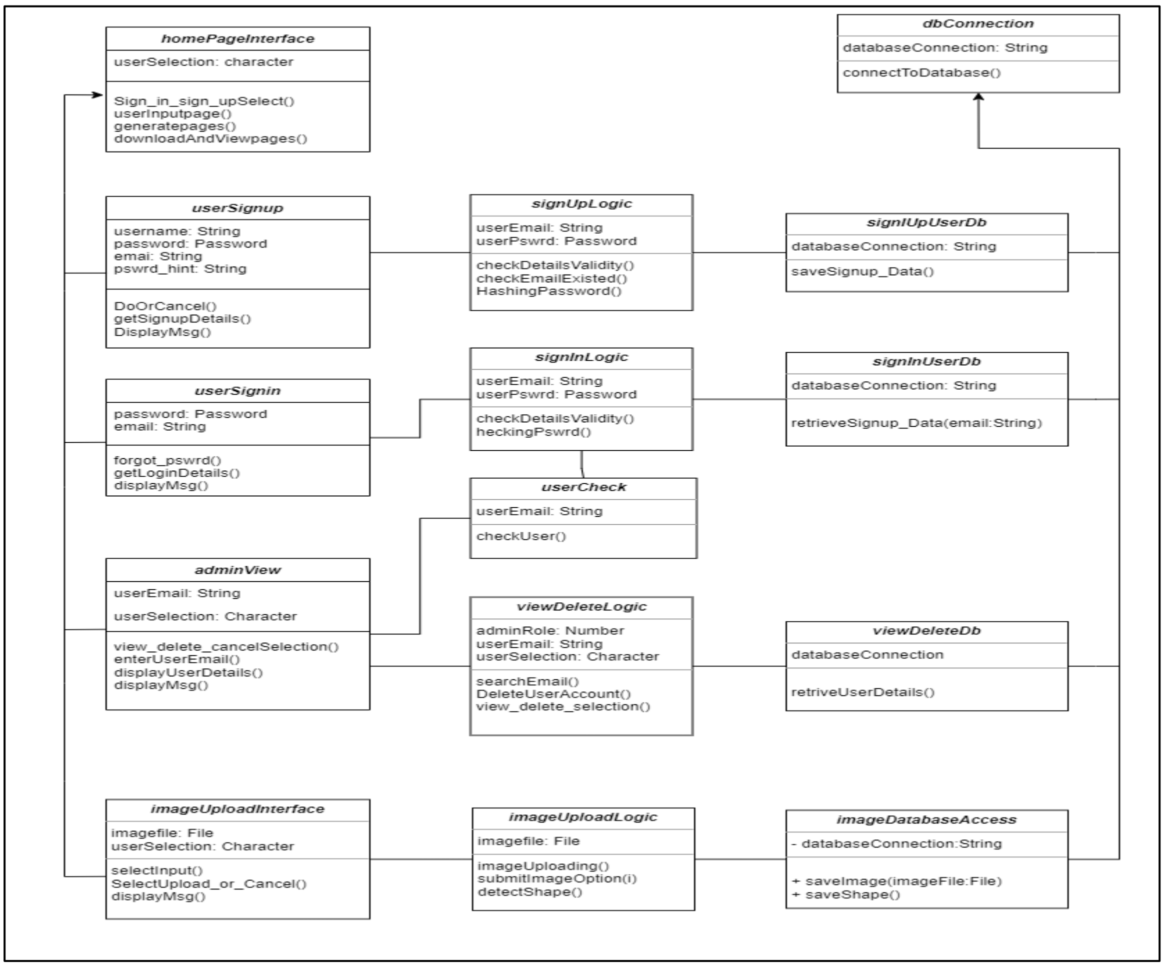


Figure 5: Class Diagram 1

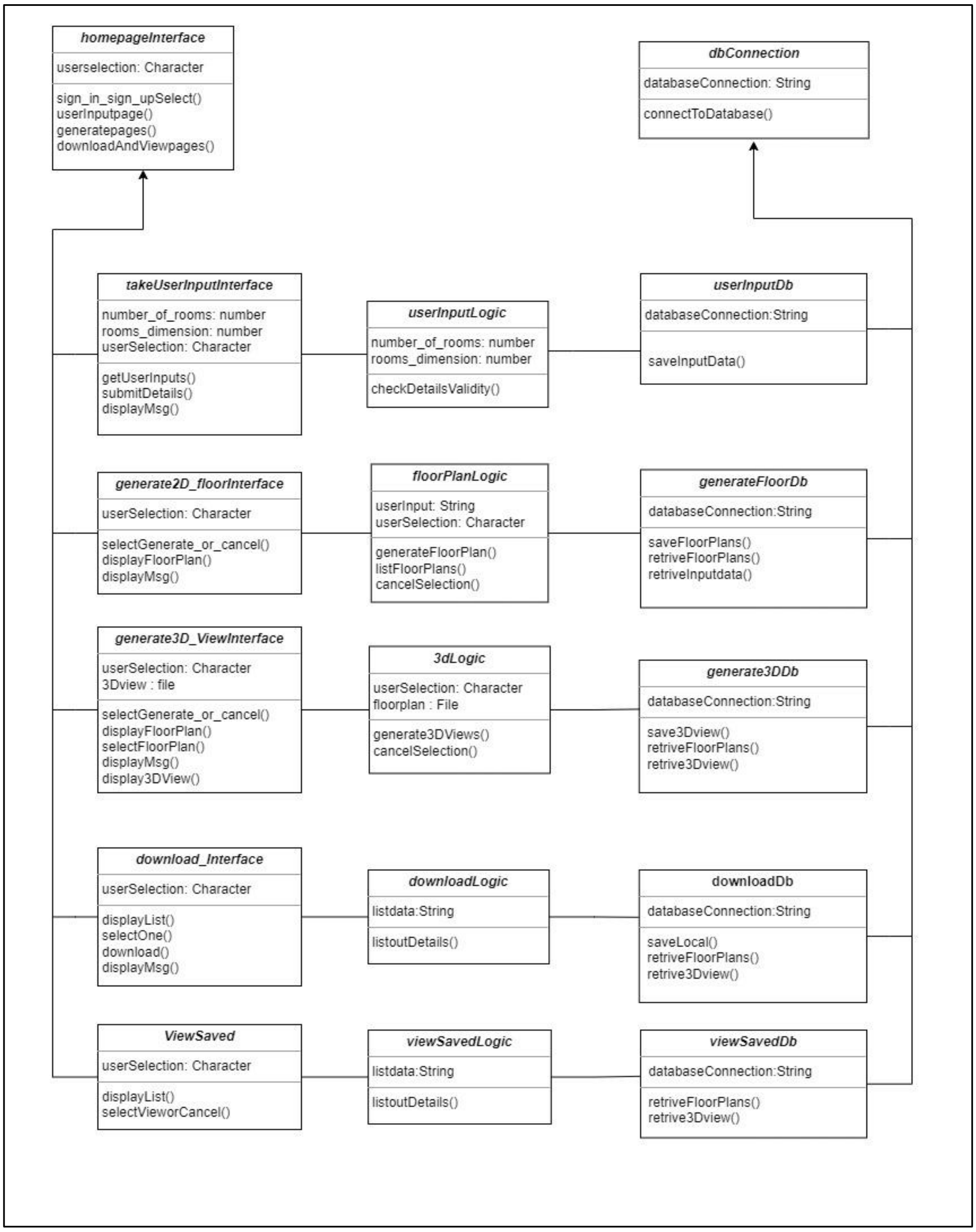


Figure 6: Class Diagram 2

2.2.2. Sequence Diagrams

- **SignUp**

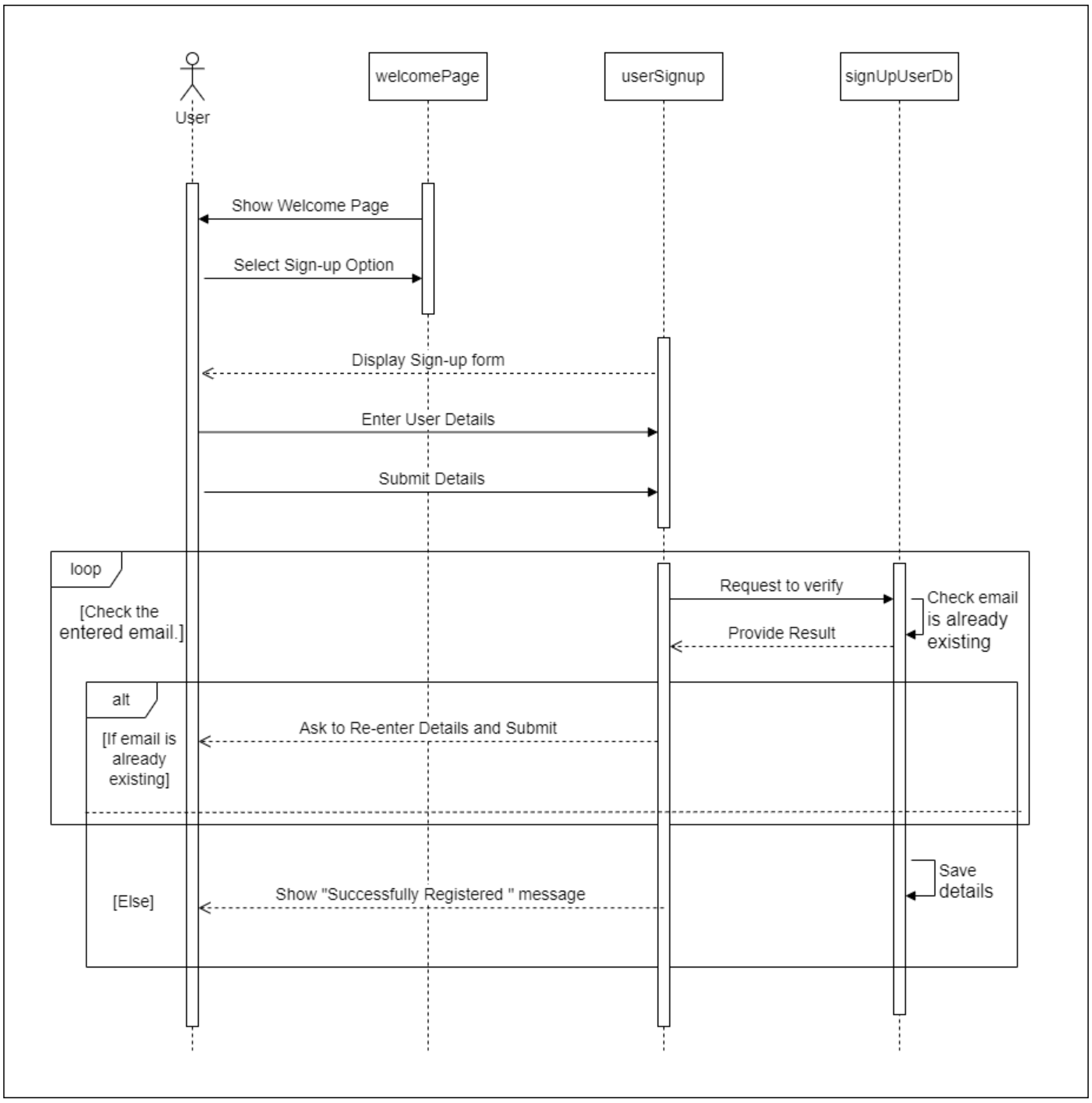


Figure 7: SignUp Sequence Diagram

- **SignIn**

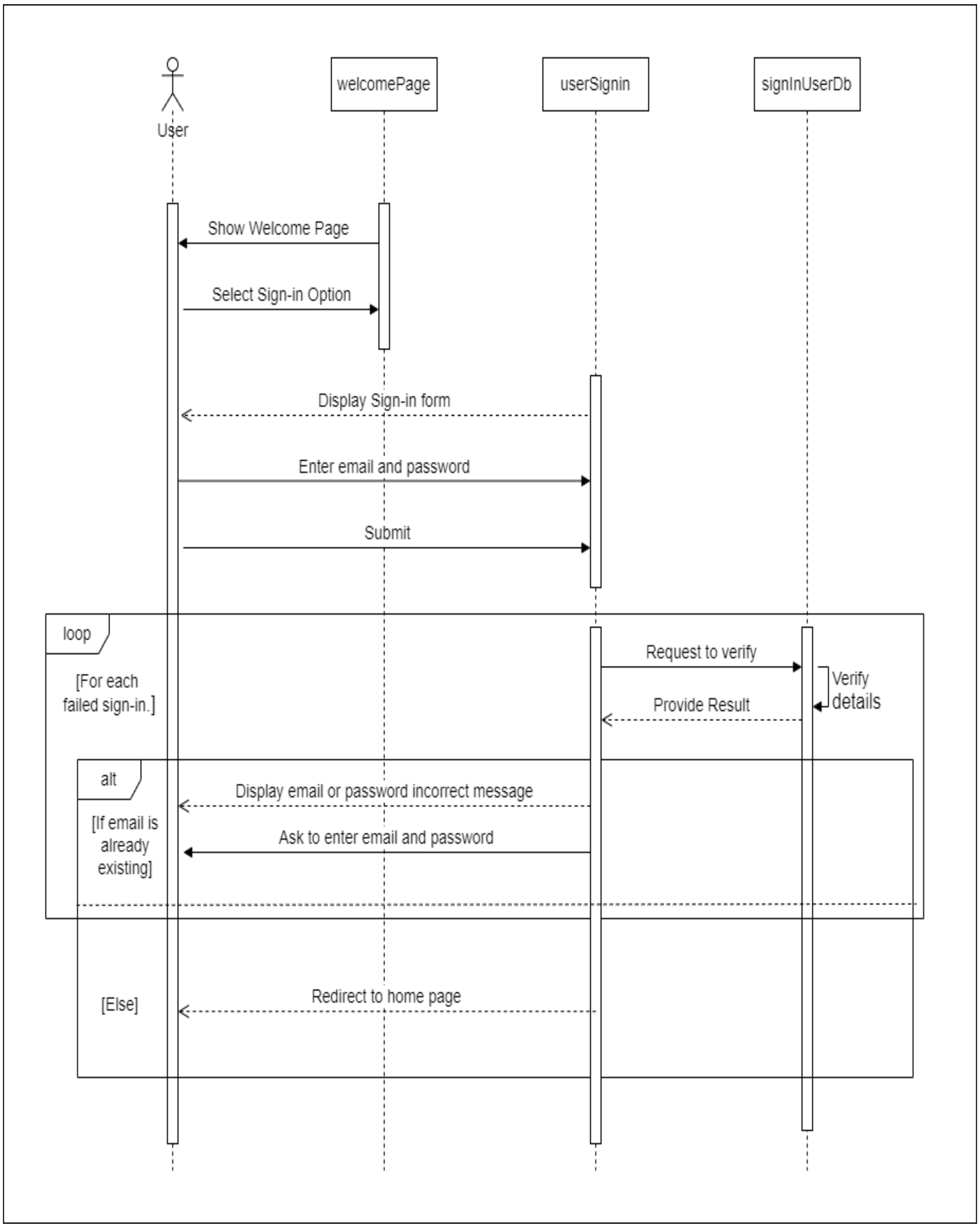


Figure 8: SignIn Sequence Diagram

- **View and Delete Account**

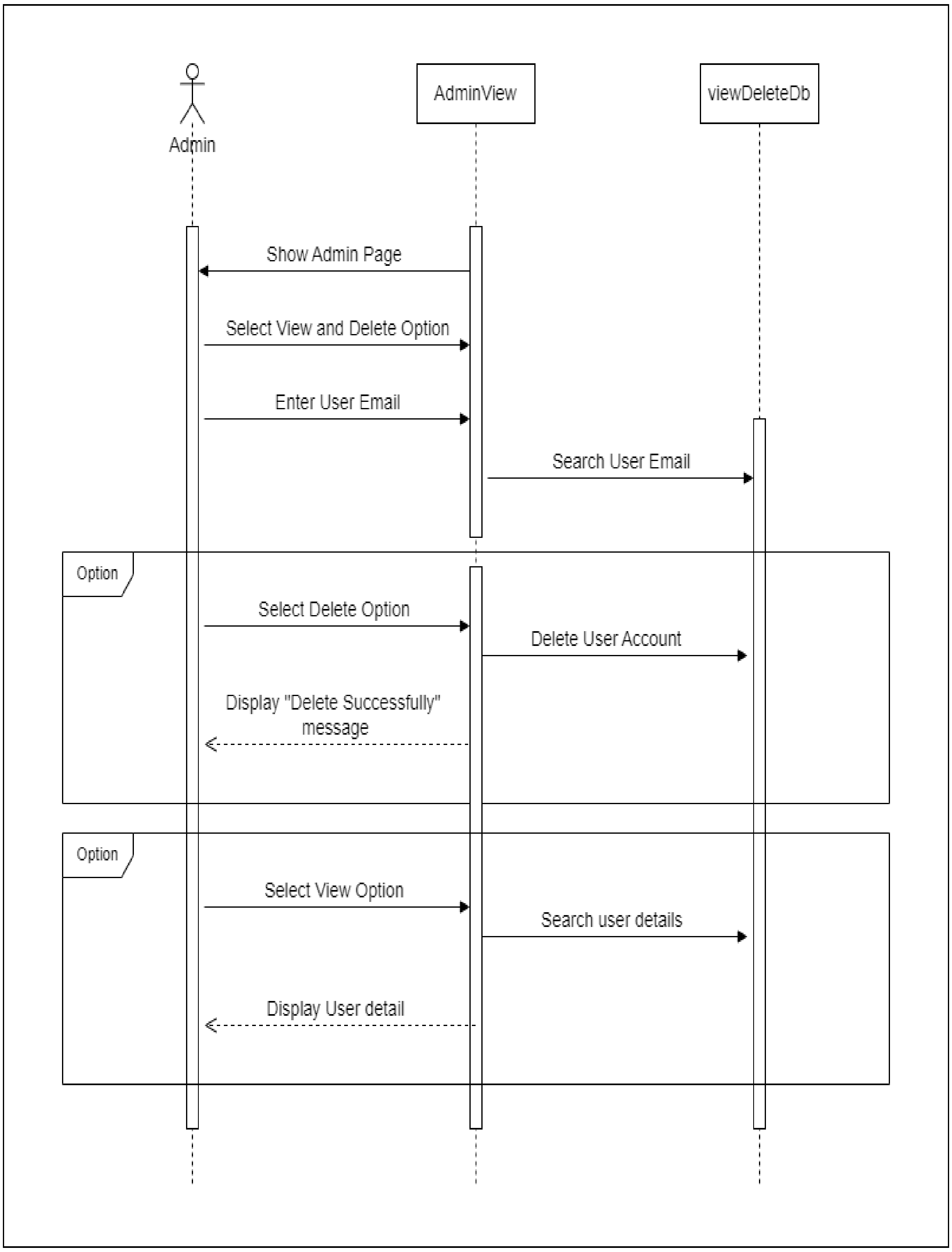


Figure 9: View and Delete Sequence Diagram

- **Upload Image & Take User Input**

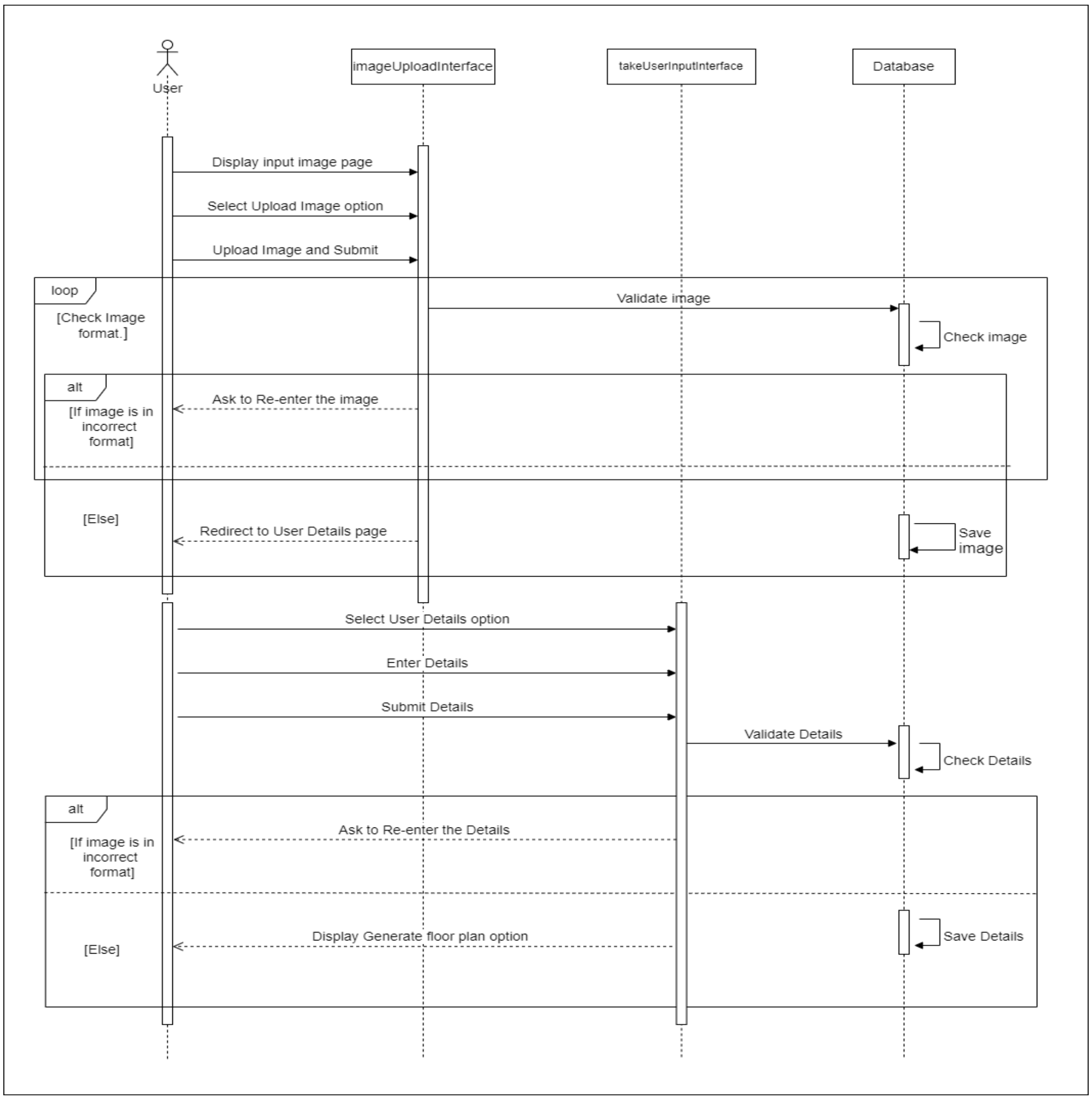


Figure 10: Upload image Sequence Diagram

- **Generate Floor Plan**

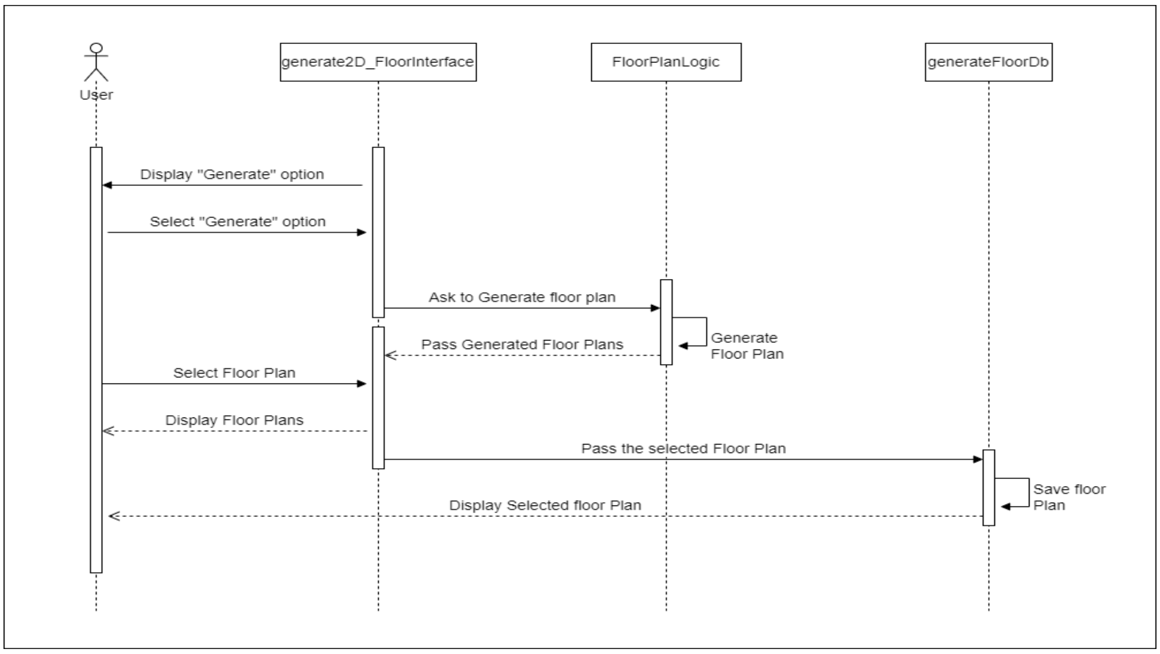

User
generate2D_FloorInterface
FloorPlanLogic
generateFloorDb
Display "Generate" option
Select "Generate" option
Ask to Generate floor plan
Generate Floor Plan
Pass Generated Floor Plans
Select Floor Plan
Display Floor Plans
Pass the selected Floor Plan
Save floor Plan
Display Selected floor Plan

Figure 11: Generate floor plan Sequence Diagram

- **Generate 3D view**

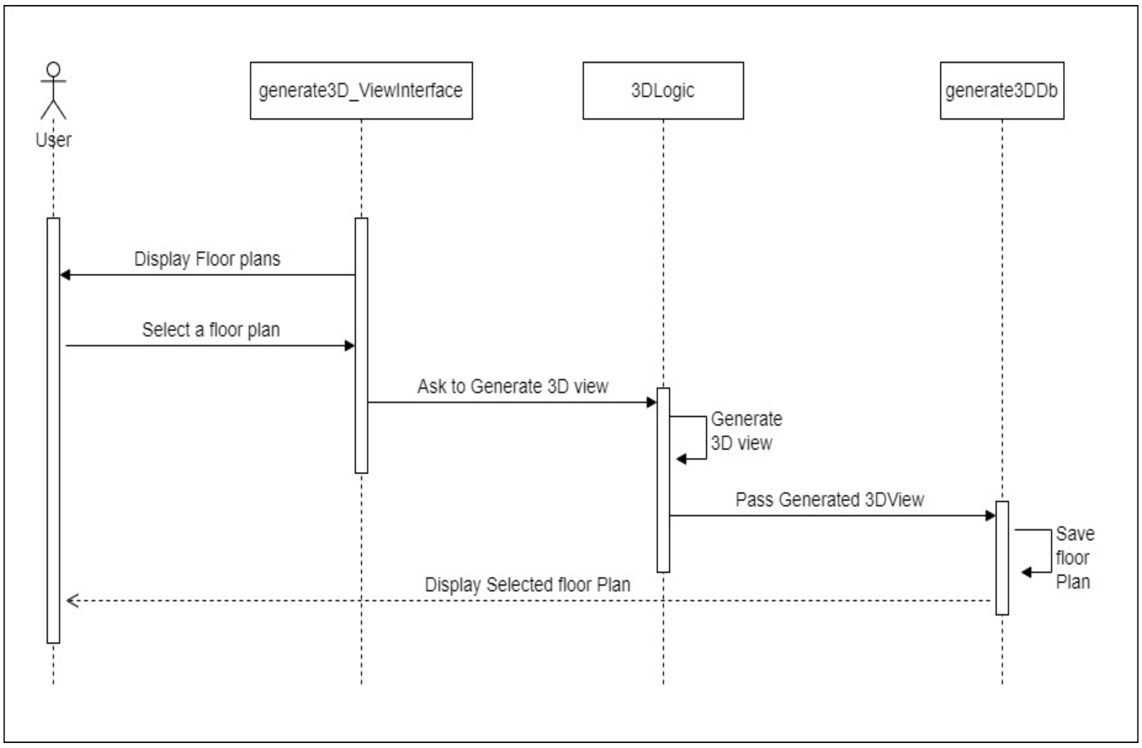


Figure 12: Generate 3D view Sequence Diagram

- **Download Floor plans and 3D views**

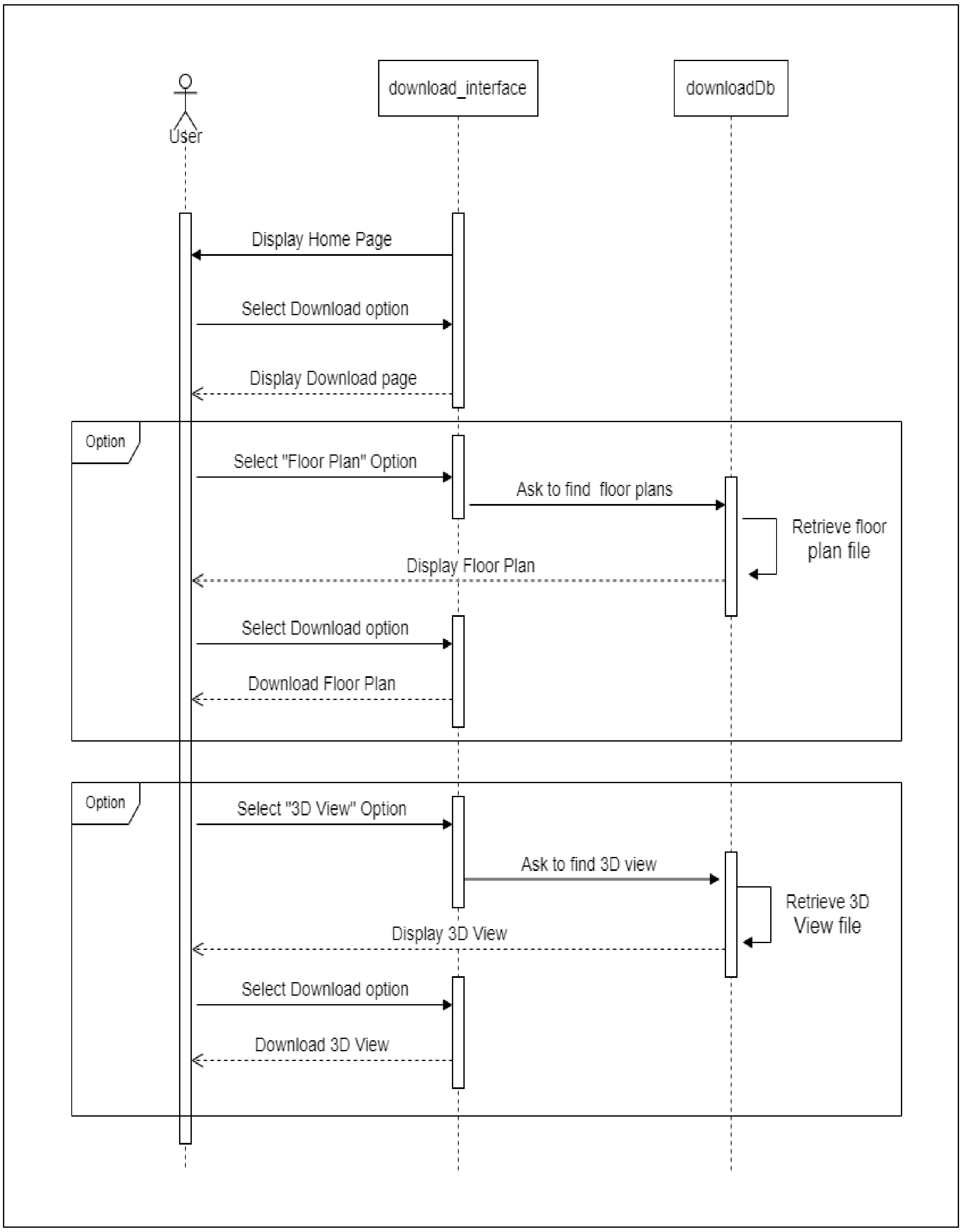


Figure 13: Download floor plan Sequence Diagram

- **View and Saved Plans and 3D views**

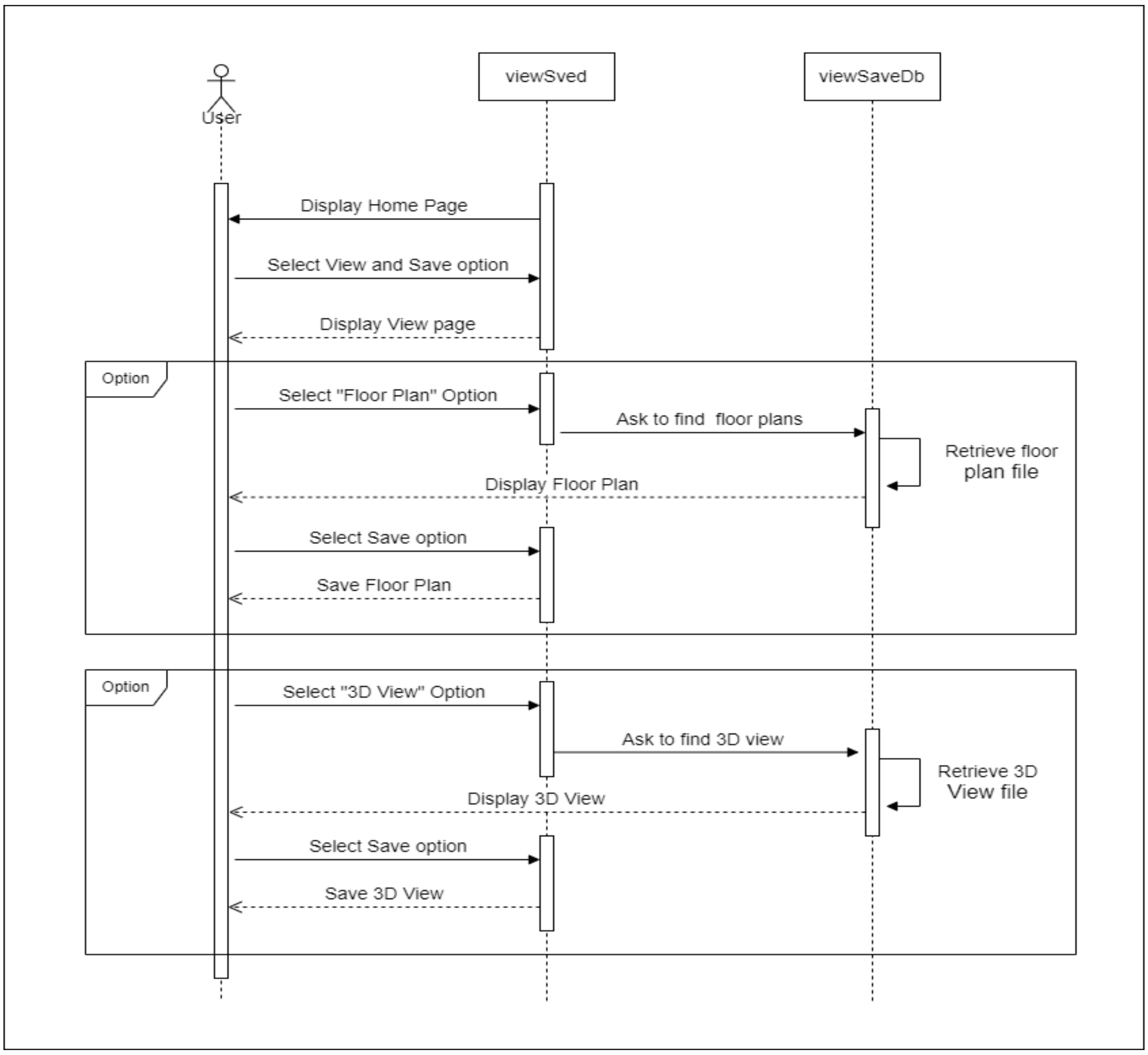


Figure 14: View and Save Sequence Diagram

2.3. Processes and special algorithms

Process/Algorithm	Description	Algorithm	Purpose	Example
User Registration Process	Users provide personal details to create an account.	Verify user information and assign a unique ID for each registered user.	Enable users to sign up and access the system securely.	When a new user signs up, their details are checked to ensure they meet the system's requirements.
User Authentication	Users enter credentials (username, password) to log in and access their account.	Check entered credentials against stored data to confirm the user's identity.	Ensure only authorized users can access the system.	When a user logs in, the system verifies their credentials before granting access.
Room Dimensions Calculation	Calculate room sizes based on user-provided measurements (e.g., length, width).	Use given measurements to compute accurate room dimensions.	Provide precise room details for creating floor plans.	After receiving room measurements, the system calculates the room dimensions automatically.
Floor Plan Generation Algorithm	Create 2D floor plans using room dimensions and layout data.	Convert room measurements into graphical floor plans.	Generate visual representations of building layouts.	The system generates a detailed floor plan for visualization based on room measurements.
3D House View Rendering	Display 3D views of houses using generated 2D floor plans.	Transform 2D floor plans into realistic 3D house models.	Allow users to visualize architectural designs in a three-	Users can view their floor plans as fully rendered 3D models to

			dimensional space.	understand the building's layout better.
Image Processing Algorithm	Validate and process uploaded images to ensure quality and compatibility.	Analyze images for clarity and correctness before using them in the system.	Ensure user-provided images meet the system's standards.	When users upload images for floor plans, the system checks them to verify they are clear and suitable.
Data Storage and Retrieval	Store user data securely and retrieve it when needed.	Use database queries to manage and access user-generated content.	Ensure reliable data management within the system.	User information and generated floor plans are stored securely in the system's database.
Auto-Save Functionality	Automatically save user progress and work to prevent data loss.	Implement timed or event-based saving mechanisms to preserve user input.	Provide a seamless user experience with data continuity.	As users input data or create designs, the system saves their work automatically to avoid losing progress.
Security Encryption	Use encryption techniques (e.g., HTTPS) to protect data during transmission.	Encrypt data before sending it over the internet to ensure privacy and security.	Safeguard sensitive information from unauthorized access.	When users log in or submit data, the system encrypts the information to prevent interception by unauthorized parties.
Feedback Collection Process	Gather user feedback and suggestions	Provide interactive forms or	Gather insights to enhance system	Users can submit feedback through the

	for system improvement.	feedback mechanisms to capture user input.	functionality and user satisfaction.	system's interface to suggest improvements or report issues.

Table 7: Processes and special algorithms

Overview of processes and special algorithms:

- User Registration Process:
    - Description: Users enter their details like name, email, and password to create an account.
    - Algorithm: Verify user information and assign a unique ID for each registered user.
    - Purpose: Enable users to sign up and access the system securely. Example: When a new user signs up, their details are checked to ensure they meet the system's requirements.

- User Authentication:
    - Description: Users enter their username and password to log in and access their account.
    - Algorithm: Check the entered credentials against stored data to confirm the user's identity.
    - Purpose: Ensure only authorized users can access the system. Example: When a user logs in, the system verifies their credentials before granting access.

- Room Dimensions Calculation:
    - Description: Calculate room sizes based on user-provided measurements (e.g., length, width).
    - Algorithm: Use the given measurements to compute accurate room dimensions.
    - Purpose: Provide precise room details for creating floor plans. Example: When a user enters room measurements, the system calculates the room dimensions automatically.

- Floor Plan Generation Algorithm:
    - Description: Create 2D floor plans using room dimensions and layout data.

    - Algorithm: Convert room measurements into graphical floor plans.
    - Purpose: Generate visual representations of building layouts. Example: After receiving room measurements, the system generates a detailed floor plan for visualization.

- 3D House View Rendering:
    - Description: Display 3D views of houses using generated 2D floor plans.
    - Algorithm: Transform 2D floor plans into realistic 3D house models.
    - Purpose: Allow users to visualize architectural designs in a three-dimensional space. Example: Users can view their floor plans as fully rendered 3D models to understand the building's layout better.

- Image Processing Algorithm:
    - Description: Validate and process uploaded images to ensure quality and compatibility.
    - Algorithm: Analyze images for clarity and correctness before using them in the system.
    - Purpose: Ensure user-provided images meet the system's standards. Example: When users upload images for floor plans, the system checks them to verify they are clear and suitable.

- Data Storage and Retrieval:
    - Description: Store user data securely and retrieve it when needed.
    - Algorithm: Use database queries to manage and access user-generated content.
    - Purpose: Ensure reliable data management within the system. Example: User information and generated floor plans are stored securely in the system's database.

- Auto-Save Functionality:
    - Description: Automatically save user progress and work to prevent data loss.
    - Algorithm: Implement timed or event-based saving mechanisms to preserve user input.
    - Purpose: Provide a seamless user experience with data continuity. Example: As users input data or create designs, the system saves their work automatically to avoid losing progress.

- Security Encryption:
    - Description: Use encryption techniques (e.g., HTTPS) to protect data during transmission.
    - Algorithm: Encrypt data before sending it over the internet to ensure privacy and security.
    - Purpose: Safeguard sensitive information from unauthorized access. Example: When users log in or submit data, the system encrypts the information to prevent interception by unauthorized parties.

- Feedback Collection Process:
    - Description: Gather user feedback and suggestions for system improvement.
    - Algorithm: Provide interactive forms or feedback mechanisms to capture user input.
    - Purpose: Gather insights to enhance system functionality and user satisfaction.

      Example: Users can submit feedback through the system's interface to suggest improvements or report issues.

2.4. State machines

2.4.1. Sign-In

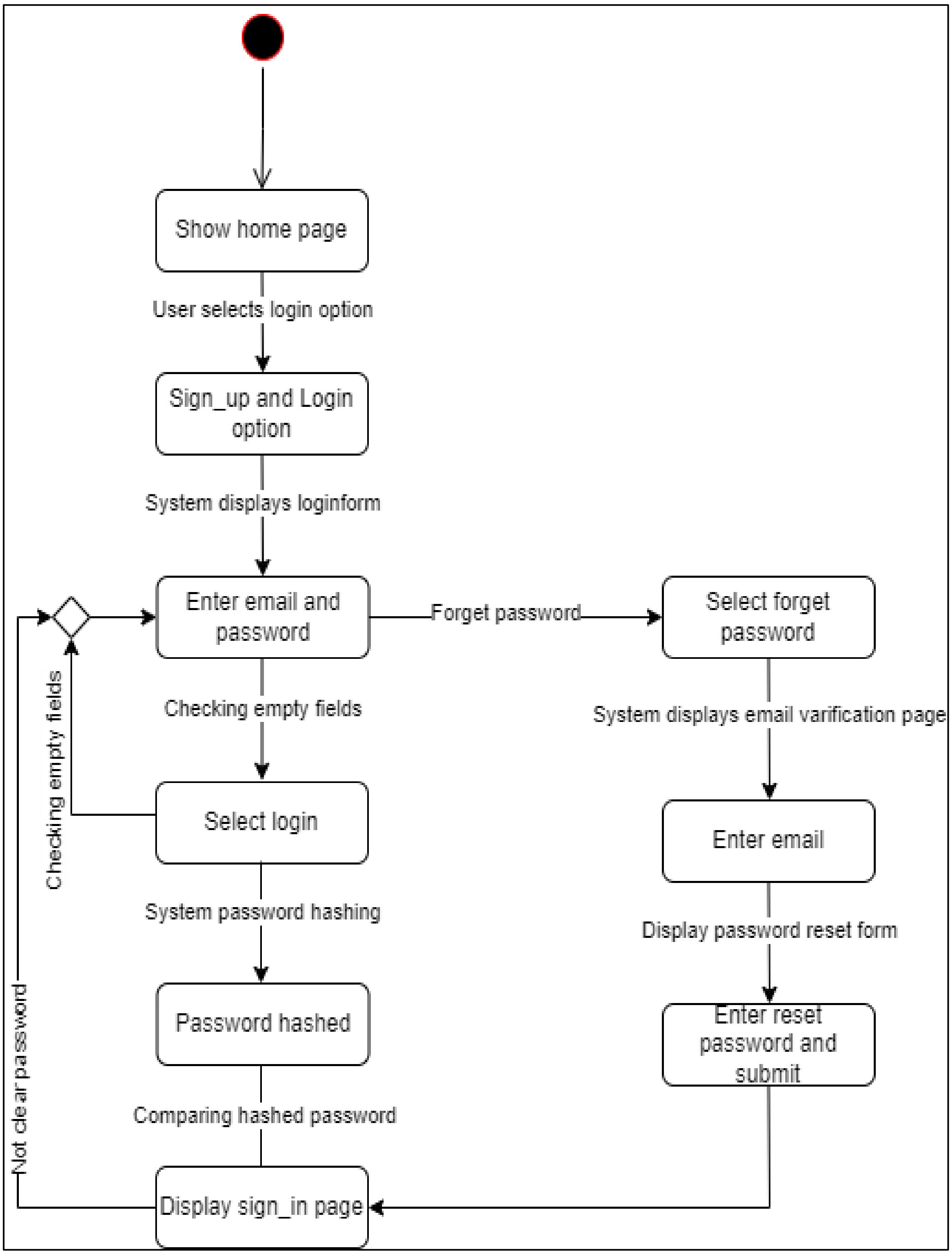


Figure 15: Sign-In State diagram

2.4.2. SignUp

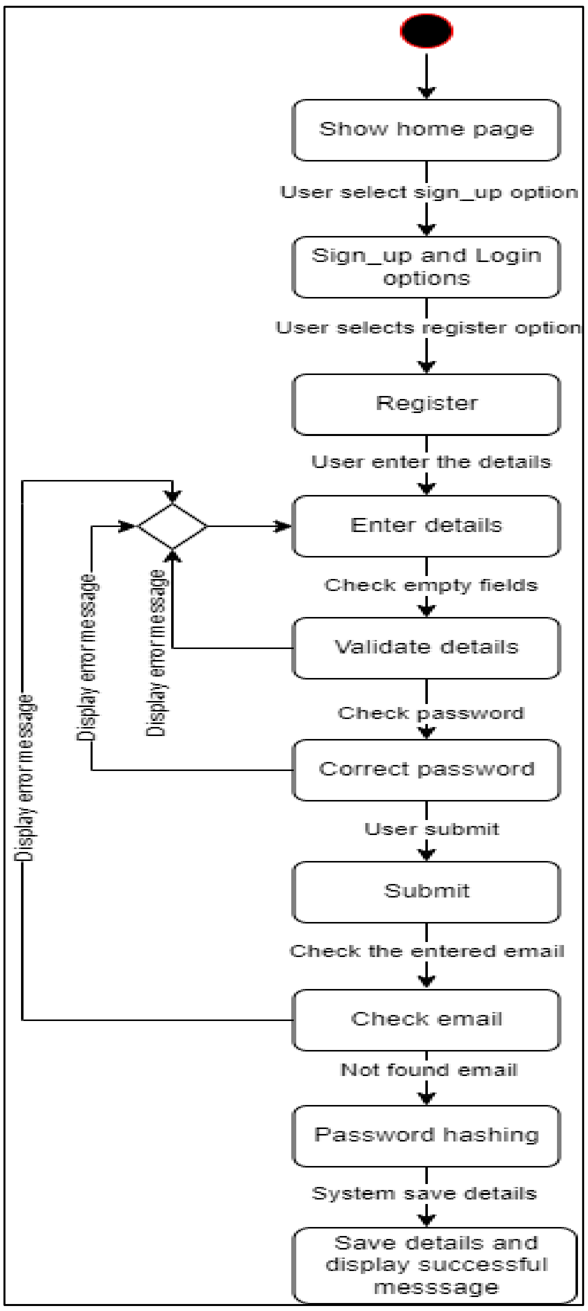


Figure 16: Sign_up state diagram

2.4.3. Take User Inout

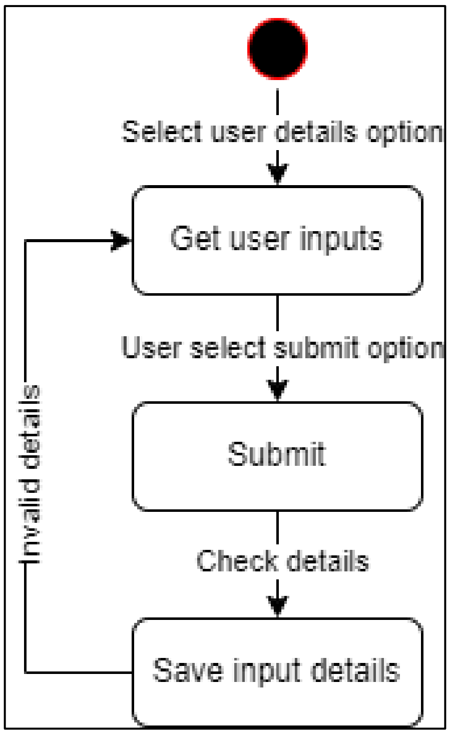


Figure 17: Take User Input

2.4.4. Upload Image

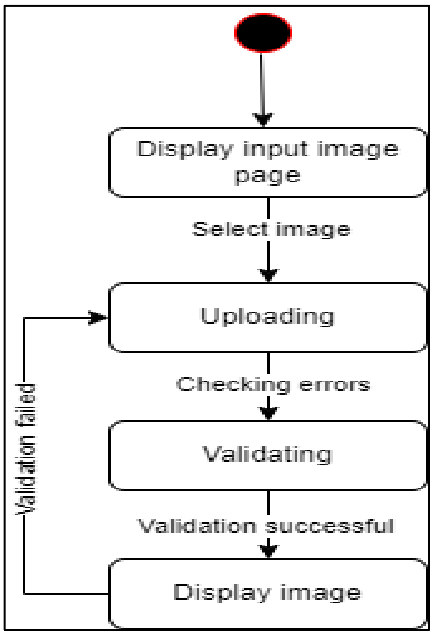


Figure 18: Upload Image

2.4.5. Shape Detect

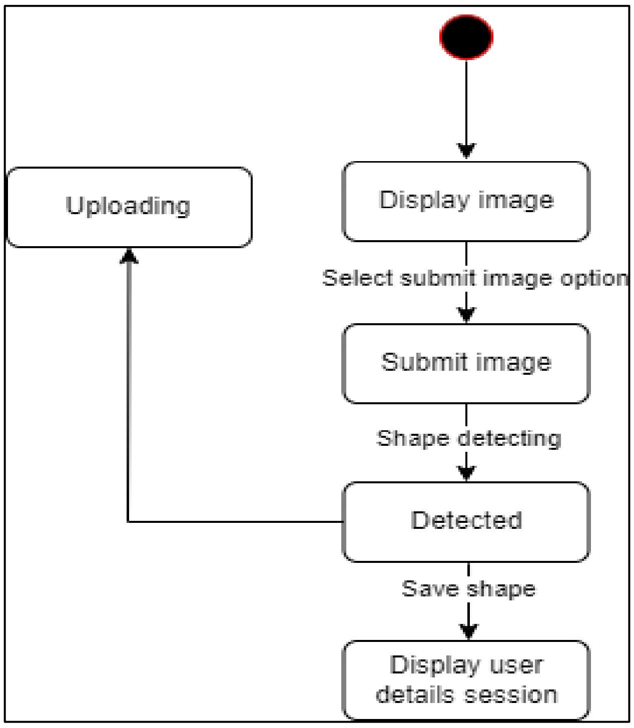


Figure 19: Shape Detect State diagram

2.4.6. View and Delete Account

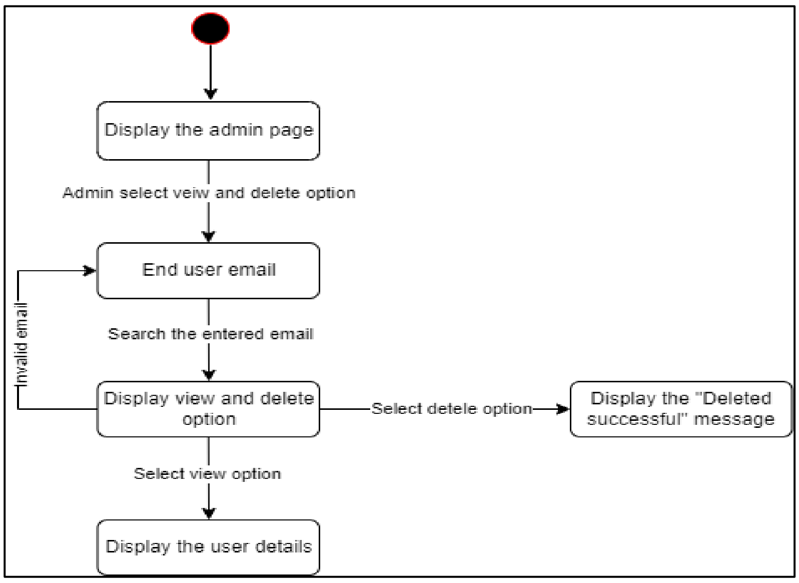


Figure 18: View and Delete State Diagram

2.4.7. Generate 2D floor plans

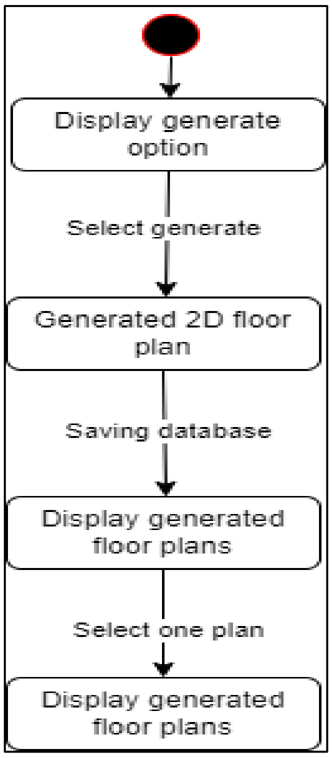


Figure 19: Generate 2D floor plan State diagram

2.4.8. Generate 3D view

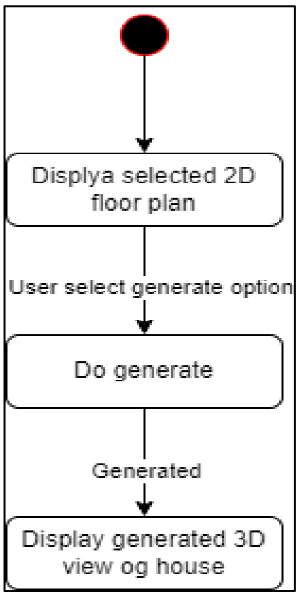
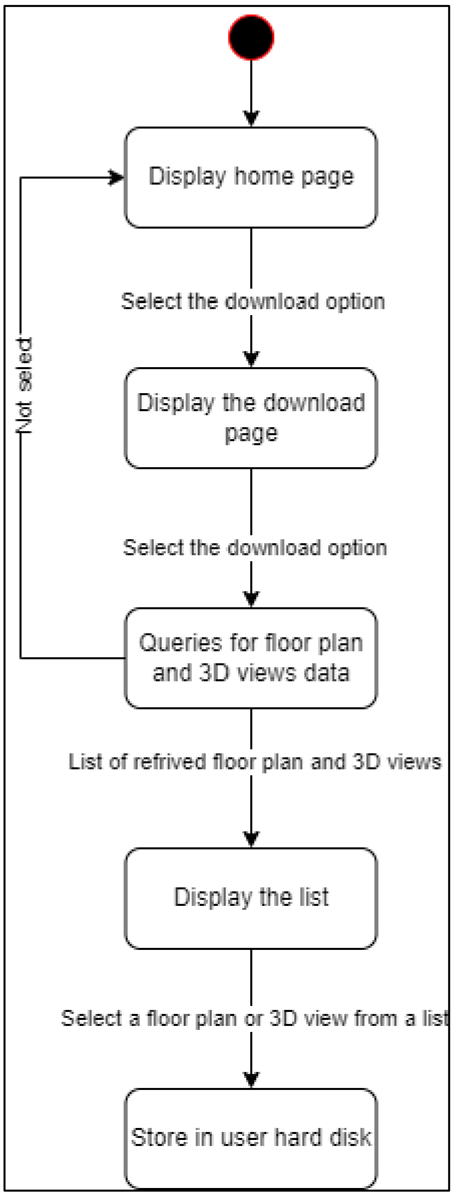

Displya selected 2D floor plan
User select generate option
Do generate
Generated
Display generated 3D view og house

Figure 20: Generate 3D view

2.4.9. Download floor plans and 3D views



Display home page
Select the download option
Display the download page
Select the download option
Queries for floor plan and 3D views data
Not seled
List of refrived floor plan and 3D views
Display the list
Select a floor plan or 3D view from a list
Store in user hard disk

Figure 21: Download State Diagram

2.5. Tools, techniques, libraries, 3rd party tools and implementation environment

Category	Tools/Techniques/Libraries
Frontend Development	Visual Studio Code, Adobe Photoshop
Backend Development	Google Colab
Database	Google Cloud
Data Visualization	Three.js (for 3D visualization in the browser)
User Authentication	Passport.js (for authentication)
Image Processing	OpenCV (for image processing tasks)
Machine Learning	Keras (for developing AI models), Google Colab
Version Control	Git (for version control)
Integrated Development	Visual Studio Code, Anaconda

Table 8: Tools and Techniques

- Presentation Tier (Frontend):
    - Frontend Development: Involves creating the user interface (UI) of the web application using languages and technologies such as HTML, CSS, and JavaScript. HTML is used for structuring web pages, CSS for styling and layout, and JavaScript for adding interactivity.
- Application Tier (Backend):
    - Programming Language (Python): Used for backend development and creating AI models. Python is known for its simplicity and versatility, making it a popular choice for web development and machine learning tasks.
    - Backend Development (Node.js, Express.js): Node.js is a JavaScript runtime used for building scalable network applications. Express.js is a web application framework for Node.js, providing features for building web servers and APIs.
    - Web Framework (Flask): Flask is a lightweight Python web framework used for developing web applications. It offers simplicity and flexibility, making it suitable for building various types of web applications.
- Data Tier (Database):
    - Database: SQL databases such as MySQL and PostgreSQL are used for storing and managing data. These databases offer structured query language (SQL) for querying and manipulating data, providing reliability and scalability for web applications.

3. UI DESIGN

- In this chapter, we intend to discuss the UI design of the automated house plan generating and visualizing system based on machine learning. Under this chapter we elaborate on the PACT(People, Activities, Contexts, and Technologies) analysis, UI Design considerations and approaches, UI Design considerations and approaches, Input Design aspects, Output Design aspects, Dialogue design aspects, and Hosting/ installation environment in detail.

3.1. PACT (People, Activities, Contexts, Technologies) analysis

Category		Description
People	Users	- Target Audience: Homeowners, interior designers, architects
		- Demographics: Varied demographics including age, gender, occupation
		- Technical Proficiency: Varies from beginners to advanced users
	Administrators	- Responsibilities: Manage user accounts, monitor uploaded images
		- Technical Proficiency: Intermediate to advanced technical skills
Activities	User Activities	- Registering an Account
		- Logging In
		- Uploading Images
		- Designing Floor Plans
		- Viewing 3D House Views
		- Managing Account Settings
	Administrator	- Monitoring User Accounts
		- Reviewing Uploaded Images

		- Managing Database Resources
Contexts	Environment	- Access from various devices (desktops, laptops, tablets, smartphones)
	Time Constraints	- Varying internet connectivity
		- Limited time to complete tasks
	Privacy and Security	- Expectation of privacy and security of personal information and uploaded images
	User Preferences	- Preferences for design styles, user interface elements, and customization options
Technologies	Frontend Development	- HTML, CSS, JavaScript
		- Frameworks like React.js or Vue.js
	Backend Development	- Node.js with Express.js
		- Authentication libraries like Passport.js
	Database	- SQL database (e.g., MySQL, PostgreSQL)
	Libraries/Frameworks	- Three.js for 3D visualization
		- OpenCV for image processing tasks
		- Bootstrap or Materialize CSS for frontend design elements
Deployment	- Cloud platforms (Heroku, AWS, Microsoft Azure) for hosting and scaling the website	Deployment

Table 9: PACT

Screen	People				Activity						Conte xt		Technology				
	Demographics	Skill Level	Language (English)	Interaction Level	Filling Details	Navigation	Readability (Easy)	Camera Using	Model Mapping	Message Displaying	Physical Environment	Social Integration	Input Method	Output Method	Response	Colors	Typography
Register	-	Y	Y	Y	Y	Y	Y	-	-	Y	Y	-	Y	-	Y	Y	Y
Login	-	Y	Y	Y	Y	Y	Y	-	-	Y	Y	-	Y	-	Y	Y	Y
Home Screen	-	-	Y	Y	-	Y	Y	-	-	-	Y	-	-	-	Y	Y	Y
Input Screen	-	-	Y	Y	Y	Y	Y	-	-	Y	Y	-	Y	-	Y	Y	Y
User Dashboard Screen	-	-	Y	Y		Y	Y	-	-	Y	Y	-	-	Y	Y	Y	Y
2D Floor Plan Display Screen	-	-	Y	Y	Y	Y	Y	-	-	Y	Y	-	Y	-	Y	Y	Y
3D House View Display Screen	-	-	Y	Y	-	Y	Y	-	Y	-	Y	-	-	Y	Y	Y	Y
Admin Dashboard Screen	Y	Y	Y	Y	Y	Y	Y	-	-	Y	Y	-	Y	Y	Y	Y	Y
User Account Management Screen	Y	Y	Y	Y	Y	Y	Y	-	-	Y	Y	-	Y	Y	Y	Y	Y
Download Floor Plan Screen	-	-	Y	Y	Y	Y	Y	-	-	Y	Y	-	Y	Y	Y	Y	Y

Table 10: System Interaction

3.2. UI Design consideration and approaches

- Responsive Web Design: Ensure that the web application layout adjusts dynamically to various screen sizes and devices, providing a consistent user experience across desktops, tablets, and smartphones.
- Cross-Browser Compatibility: Design and test the web application to work seamlessly across different web browsers (e.g., Chrome, Firefox, Safari, Edge) to ensure consistent functionality and appearance for all users.
- Intuitive Navigation: Implement clear and intuitive navigation menus, breadcrumbs, and links to help users easily navigate through different sections and features of the web application.
- Consistent Design Language: Maintain a consistent design language, including color schemes, typography, and UI elements, throughout the web application to establish familiarity and improve usability.
- Accessibility: Ensure that the web application is accessible to users with disabilities by adhering to accessibility standards (e.g., WCAG) and incorporating features such as keyboard navigation, alt text for images, and semantic HTML markup.
- Performance Optimization: Optimize the performance of the web application by minimizing page load times, reducing HTTP requests, and optimizing assets (e.g., images, scripts, stylesheets) to enhance user experience and engagement.
- User Feedback Mechanisms: Implement interactive elements such as forms, surveys, and feedback buttons to collect user feedback and suggestions, allowing for continuous improvement of the web application based on user input.
- Security Considerations: Incorporate security measures such as HTTPS, data encryption, and user authentication to protect user data and prevent unauthorized access to sensitive information.
- Mobile-First Approach: Adopt a mobile-first design approach, focusing on designing for mobile devices initially and then scaling up to larger screens, to ensure a user-friendly experience for mobile users, who represent a significant portion of web traffic.
- Progressive Enhancement: Design the web application to progressively enhance the user experience by providing basic functionality to all users while leveraging advanced features and technologies for users with modern browsers and devices.

- Content Strategy: Develop a content strategy to organize and present information effectively, ensuring that the most relevant content is easily accessible to users and presented in a clear and concise manner.
- Usability Testing: Conduct usability testing sessions with real users to identify usability issues, gather feedback, and validate design decisions, enabling iterative improvements to the web application's UI and UX.

UI Design Consideration	**Description**	**Examples**
Responsive Web Design	Ensure the layout adjusts dynamically to various screen sizes and devices.	A website that displays properly on both desktop and mobile devices.
Cross-Browser Compatibility	Design and test the application to work seamlessly across different web browsers.	Compatibility with Chrome, Firefox, Safari, and Edge browsers.
Intuitive Navigation	Implement clear and intuitive navigation menus, breadcrumbs, and links.	Menu bars, search bars, and clickable buttons for easy navigation.
Consistent Design Language	Maintain a consistent design language, including color schemes, typography, and UI elements.	Using the same font style, color palette, and button styles throughout the site.
Accessibility	Ensure the application is accessible to users with disabilities.	Providing alt text for images and keyboard navigation options.
Performance Optimization	Optimize performance by minimizing page load times and reducing HTTP requests.	Compressing images and using caching mechanisms to speed up loading times.
User Feedback Mechanisms	Implement interactive elements such as forms and feedback buttons.	Feedback forms, surveys, and rating systems for user engagement.
Security Considerations	Incorporate security measures such as HTTPS and data encryption.	SSL certificates for secure connections and password hashing for user authentication.

Mobile-First Approach	Adopt a mobile-first design approach.	Designing for smaller screens first and then scaling up for larger screens.
Progressive Enhancement	Design the application to progressively enhance the user experience.	Providing basic functionality for all users and additional features for modern browsers.
Content Strategy	Develop a content strategy to organize and present information effectively.	Categorizing content into sections, using headers and subheadings for clarity.
Usability Testing	Conduct usability testing sessions with real users.	Observing user interactions and gathering feedback on navigation and layout.

Table 11: Summary of UI Consideration

3.3. Design tools, techniques, templates

3.3.1. Design Tools:

- Canva: An easy-to-use graphic design platform that allows you to create various designs, including social media graphics, presentations, posters, and more, using drag-and-drop features and customizable templates.
- Adobe Spark: A web-based design tool that enables you to create graphics, web pages, and videos quickly and easily, with no design experience required. It offers a variety of templates and intuitive editing features.
- Piktochart: A tool for creating infographics, presentations, posters, and reports using customizable templates and an intuitive interface. It's suitable for non-designers and offers drag-and-drop functionality.

3.3.2. Design Techniques:

- Sketching: Start by sketching out your ideas on paper or using digital sketching tools like Sketchboard or Balsamiq. Sketching helps visualize layouts and interactions before diving into detailed designs.
- Mood Boards: Gather images, colors, typography samples, and other visual inspirations on a digital or physical mood board to establish the overall look and feel of your project.
- Paper Prototyping: Create paper prototypes of your web pages by sketching out different elements and arranging them on paper. This

technique allows for quick iteration and feedback before moving to digital prototypes.

- Storyboarding: Create a series of sketches or screenshots to outline the user journey and interaction flow within your web application. Storyboards help visualize how users will navigate through your site.

3.3.3. Design Templates:

- Home Page

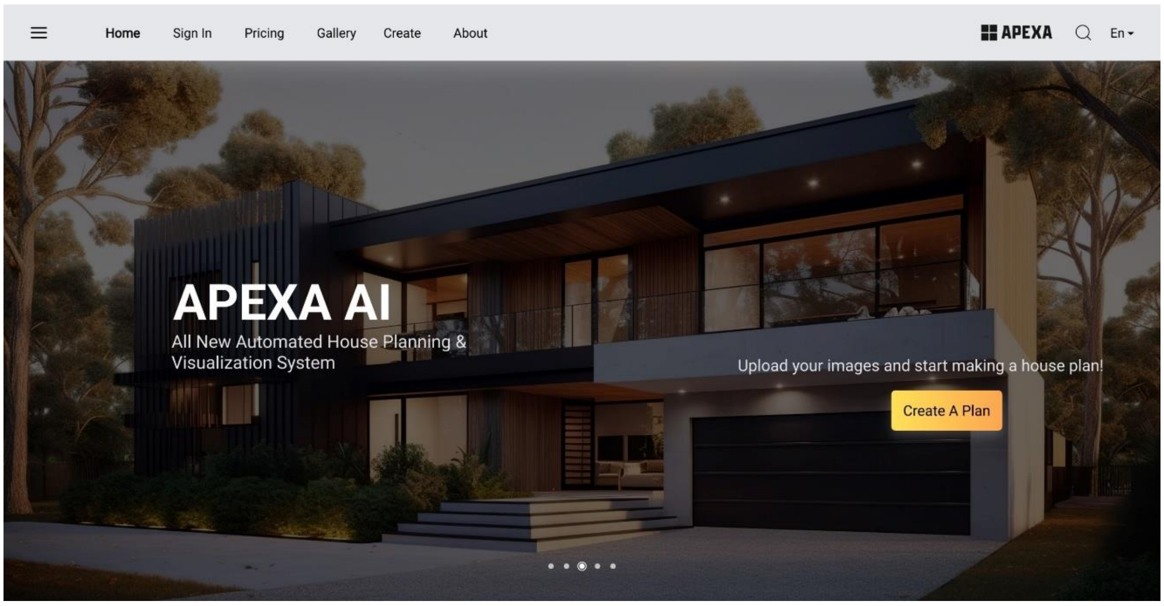

Home
Sign In
Pricing
Gallery
Create
About
APEXA
En
APEXA AI
All New Automated House Planning & Visualization System
Upload your images and start making a house plan!
Create A Plan

Figure 22: Home Interface

- Sign-in/Sign-up Page

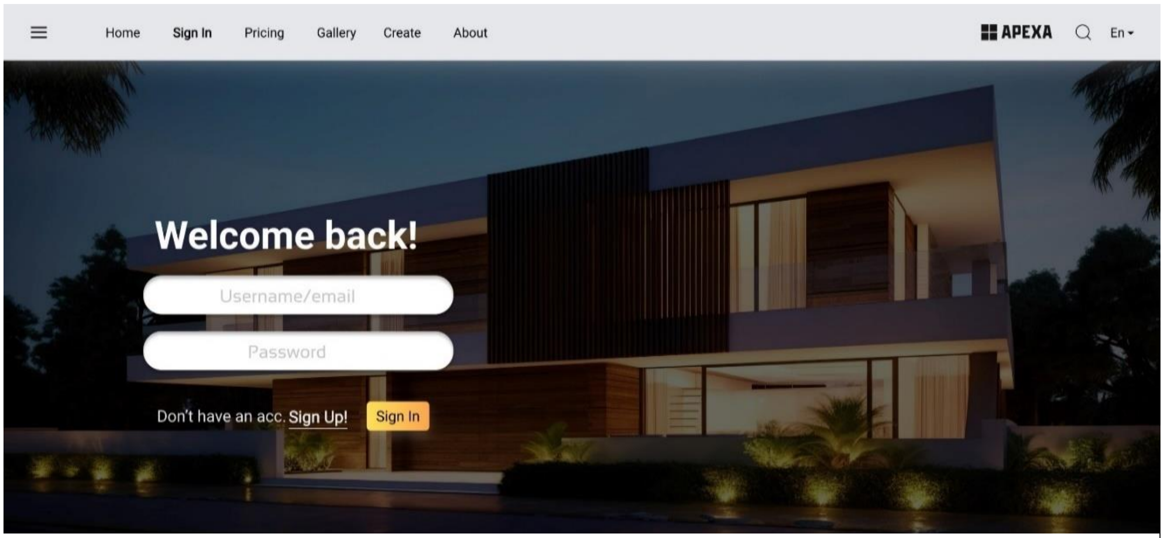


Figure 23: Sign-in/Sign-up Page

- Upload Image page

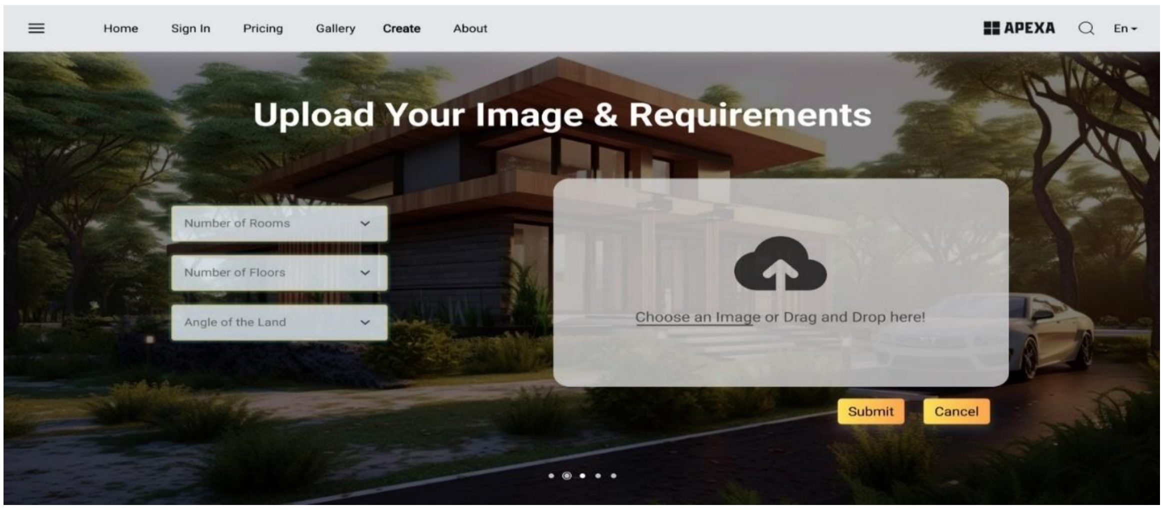


Figure 24: Upload Image Interface

- Admin Page

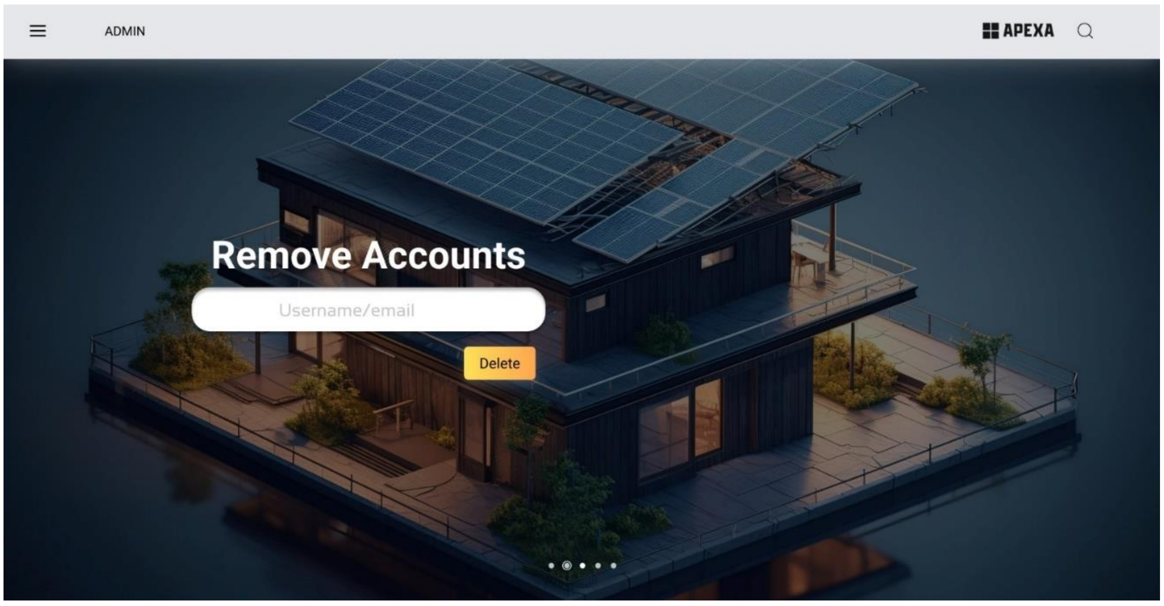


Figure 25: Admin page Interface

- Generate Floor Plan Page

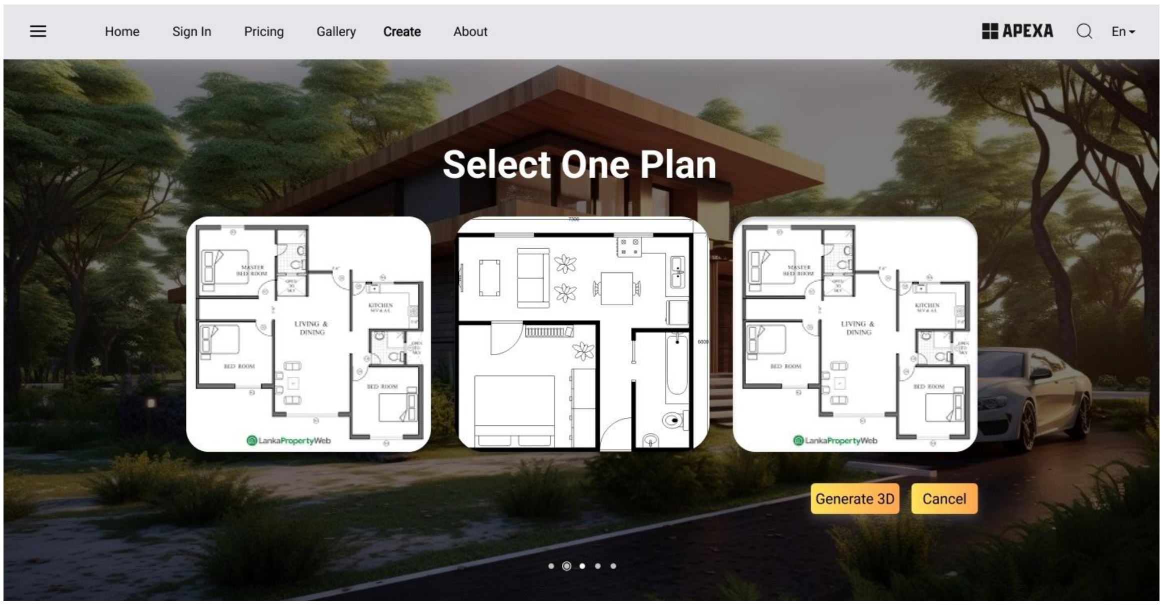


Figure 26: Generate Floor Plan Page Interface

- 3D view Page

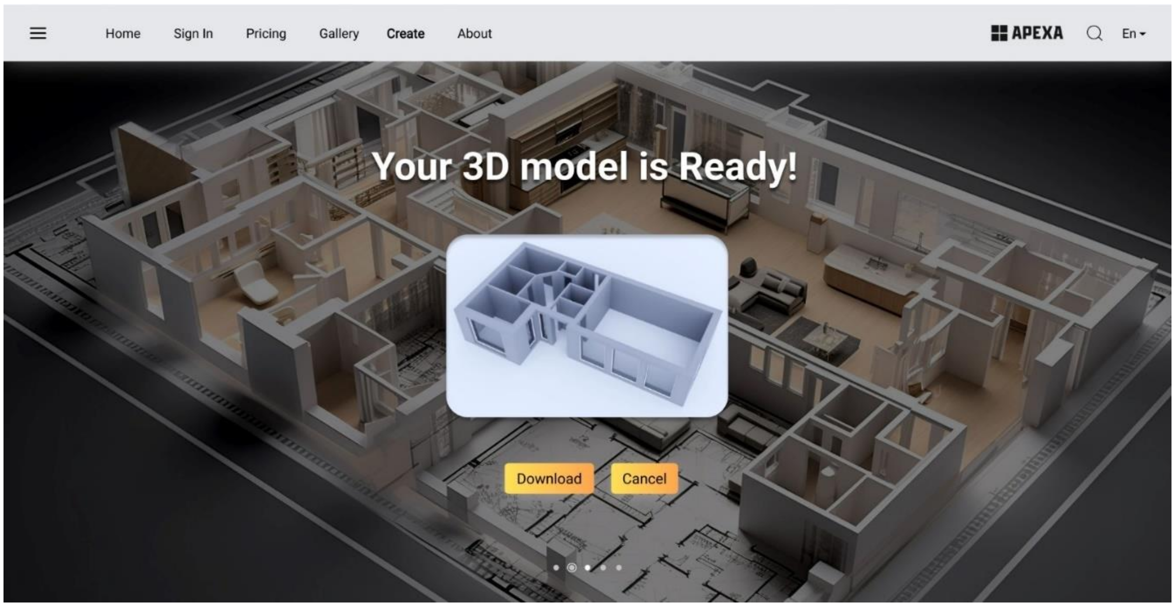


Figure 27: 3D View Page

3.4. Input Design aspects

Aspect	Description
Input Fields	Boxes where users type information.
Labels	Clear instructions next to input fields.
Validation	Checking input for correctness and providing feedback on errors.
Error Handling	Displaying helpful messages when users make mistakes.
Feedback	Providing immediate feedback on user input.
Accessibility	Ensuring all users, including those with disabilities, can use input fields easily.
Input Masks	Formatting input as users type (e.g., adding dashes to phone numbers).
Auto-Suggestions	Offering helpful suggestions as users type.
Mobile Optimization	Optimizing input fields for easy use on mobile devices.
Consistency	Maintaining uniformity in design and functionality across the project.

Table 12: Table of Input Design Aspects

- Input Fields: Designing intuitive and visually appealing input fields for users to enter text, numbers, dates, and other data.
- Labels: Providing clear and descriptive labels for input fields to guide users on what information to enter.
- Validation: Implementing validation mechanisms to ensure that users enter valid data according to specified formats and requirements.

- Error Handling: Designing error messages and notifications to effectively communicate validation errors or incorrect inputs to users and guide them on how to correct the issue.
- Feedback: Providing immediate feedback to users upon input, such as success messages for valid inputs or real-time validation feedback as users type.

- Accessibility: Ensuring that input elements are accessible to users with disabilities by implementing features like keyboard navigation, focus indicators, and screen reader compatibility.

- Input Masks: Using input masks to format and validate input data, such as phone numbers, credit card numbers, or dates, as users type.

- Auto-Suggestions: Implementing auto-suggestion or autocomplete features to assist users in completing input fields more efficiently, especially for lengthy or complex data.

- Mobile Optimization: Designing input elements to be mobile-friendly, with larger touch targets and optimized layouts for smartphones and tablets.

- Consistency: Maintaining consistency in the design and behavior of input elements across the application to provide a cohesive user experience.

3.5. Output Design aspects

Aspect	Description
Visual Elements	Pictures, words, and graphics used to show information.
Layout	How everything is arranged on the screen.
Typography	The style and size of the text used in the design.
Color Palette	The colors chosen to make the design look good and easy to understand.
Icons and Symbols	Small pictures or signs that represent different things.
Consistency	Making sure everything looks the same across the whole design.
Responsiveness	Making sure the design works well on different devices, like phones and computers.

Accessibility	Making sure everyone, including people with disabilities, can use the design easily.
Interactive Elements	Buttons or links that people can click on or interact with.
Feedback Mechanisms	Giving people clear signals when they do something or make a mistake.

Table 13: Table of Output Design Aspects

3.6. Dialogue design aspects

Aspect	Description
Language and Tone	The words and how they sound, matching who we're talking to and how we want to sound.
Clarity	Making sure the words are easy to get, avoiding hard words or too much talk.
Brevity	Keeping it short and sweet, saying just what's needed without extra stuff.
Staying Relevant	Talking about things that make sense right now, giving info that matters.
Helping Users	Giving clear instructions to users, showing them how to do things.
Fixing Mistakes	Helping users when things go wrong, giving them a way to fix stuff.
Saying "Yes" and "No"	Saying when something's okay and when it's not, letting users know what's happening.
Keeping Things Alike	Making sure all the talking feels the same, so users know what to expect.
Open to Everyone	Making sure everyone can understand and use what we say, no matter who they are.

Table 14: Dialogue Design Aspects

- Language and Tone: Choose words and a style that matches our audience and the image we want to convey. For example, using friendly language for a casual audience
.
- Clarity: Ensure that our messages are easy to understand by avoiding complex language or long-winded explanations.
- Brevity: Keep messages short and concise, getting to the point without unnecessary information.
- Staying Relevant: Provide information that is timely and useful to users, avoiding irrelevant or outdated content.
- Helping Users: Offer clear instructions and guidance to users to help them complete tasks or resolve issues.
- Fixing Mistakes: Provide helpful error messages or prompts to assist users when they make errors or encounter problems.
- Saying "Yes" and "No": Clearly communicate acceptance or rejection of user actions, ensuring users understand the outcome of their actions.
- Keeping Things Alike: Maintain consistency in language and style across different parts of the application to provide a cohesive user experience.
- Open to Everyone: Ensure that our messages are accessible and understandable to all users, regardless of their background or abilities.

3.7. Hosting/ installation environment

Step	Description
1. Sign Up	Create an account on Google Cloud Platform (GCP) and provide billing information if required.
2. Set Up Project	Create a new project in the Google Cloud Console to contain all resources related to your application.
3. Choose Hosting	Select the appropriate GCP hosting service for your project, such as Google App Engine, Compute Engine, or Kubernetes Engine (GKE).
4. Deploy Application	Package your application code and deploy it to the chosen hosting service using the gcloud command-line tool or Google Cloud Console.
5. Configure Domain	Set up your custom domain and configure DNS records to point to your GCP-hosted application.
6. Database & Storage	Utilize GCP services like Cloud SQL for databases and Cloud Storage for storing files and media assets as per your project requirements.
7. Implement Security	Implement security measures provided by GCP, such as IAM, encryption, and firewall rules, to protect your application and data.
8. Monitor & Scale	Set up monitoring and logging using Stackdriver to track performance and configure auto-scaling to handle traffic spikes.
9. CI/CD	Implement Continuous Integration and Deployment (CI/CD) pipelines to automate the build, test, and deployment processes for your application updates.
10. Cost Management	Monitor resource usage and optimize costs by choosing appropriate instance types, scaling strategies, and resource configurations.

Table 15: Hosting and Installation

- Sign Up for Google Cloud Platform: Create an account on GCP if you haven't already. You may need to provide billing information, but Google often offers free credits for new users.
- Set Up a Project: Create a new project in the Google Cloud Console. This project will serve as the container for all the resources related to your application.
- Choose Hosting Service: Decide which GCP hosting service best suits your project needs. Options include:
  - Google App Engine: Ideal for hosting scalable web applications with auto-scaling and managed infrastructure.
  - Google Compute Engine: Provides virtual machines (VMs) for more control over the hosting environment.
  - Google Kubernetes Engine (GKE): Offers containerized applications managed by Kubernetes.
- Deploy Your Application: Depending on the hosting service chosen:
  - For App Engine: Package your application code into a deployment package (e.g., a Docker container) and deploy it using the gcloud command-line tool or Google Cloud Console.
  - For Compute Engine: Create a VM instance, configure it with the necessary software stack, and deploy your application code.
  - For GKE: Deploy your containerized application to a Kubernetes cluster using kubectl or Google Cloud Console.
- Configure Domain and DNS: If you have a custom domain, configure it to point to your GCP-hosted application. Set up DNS records to direct traffic to your GCP resources.
- Set Up Database and Storage: Use GCP services like Cloud SQL for managing databases, Cloud Storage for storing files and media assets, and other relevant services based on your project requirements.
- Implement Security Measures: Utilize GCP's security features, such as IAM, encryption, and firewall rules, to protect your application and data from security threats.
- Monitor and Scale: Set up monitoring and logging using Stackdriver to track performance, availability, and errors. Configure auto-scaling to handle traffic spikes and ensure reliability.
- Continuous Integration and Deployment (CI/CD): Implement CI/CD pipelines to automate the build, test, and deployment processes for your application updates.
- Cost Management: Monitor your resource usage and optimize costs by choosing the appropriate instance types, scaling strategies, and resource configurations.

4. DATA MANAGEMENT

Data management in this system involves organizing, storing, and processing information efficiently. Here's a simple explanation:

- Organizing Data: Structuring information into categories or formats that make it easy to access and use. For example, storing user details like usernames, passwords, and email addresses in a database table.

- Storing Data: Saving information securely in a database or storage system. This ensures data is preserved and can be retrieved when needed. For instance, saving uploaded images and generated floor plans in cloud storage.

- Processing Data: Performing operations on data to generate insights or outcomes. This could involve calculating room dimensions to create a floor plan or generating 3D views based on 2D plans.

- Ensuring Data Integrity: Maintaining the accuracy and consistency of data throughout its lifecycle. Validating user inputs and implementing constraints to prevent errors or inconsistencies in the stored data.

- Securing Data: Implementing measures to protect data from unauthorized access or loss. This includes using encryption, access controls, and regular backups to safeguard information.

In summary, data management in this system involves handling data effectively, from its creation and storage to processing and protection, ensuring that information is organized, accessible, and secure for users and system operations.

4.1. Data requirements

Data Category	Description
User Data	- User details (name, email, password)
	- User account information (creation date, last login)
	- Admin privileges (if applicable)
Room and Floor Plan Data	- Room dimensions (length, width, height)
	- Floor plan configurations (room layout, wall positions)
	- Generated 2D floor plans (image files)
Image Data	- Uploaded survey plan images (floor plans, land layouts)
	- Processed images (identified shapes, floor plan representations)
Authentication Data	- User credentials (username, password hashes)
	- Access tokens (for session management)
System Logs and Audit Trails	- User activity logs (login history, account modifications)
	- Admin actions (user deletions, data modifications)
Configuration Data	- Application settings (default preferences, system configurations)
	- Database schema and structure definitions
Feedback and Interaction Data	- User feedback (comments, ratings)
	- Interaction history (user actions, system responses)
Metadata and Relationships	- Data relationships (user-to-account, image-to-plan associations)
	- Metadata (timestamps, unique identifiers)
	- Access control lists (ACLs) and permissions

Security and Access Control Data	- Encryption keys and security configurations
Error and Exception Data	- Error logs (application errors, debugging information)
	- Exception handling data (error codes, stack traces)

Table 16: Data Requirements

4.2. Design tools, techniques

- Database Management System (DBMS): Use MySQL or PostgreSQL to manage structured data efficiently.
- Entity-Relationship (ER) Diagrams: Create ER diagrams to model relationships between database entities.
- Normalization: Apply normalization techniques to eliminate redundancy and improve data integrity.
- Indexing: Use indexing to optimize database queries and improve data retrieval performance.
- Stored Procedures: Implement stored procedures to encapsulate business logic within the database.
- Backup and Recovery: Set up regular backups and implement recovery procedures to ensure data availability and integrity.

4.3. Conceptual database design

4.3.1. Identify Entities and Attributes

- User Entity:
    - Represents individuals using the system and is identified uniquely by their email address. Attributes include User ID, email (primary key), role, username, and password.
- User_Input Entity:
    - Captures user-provided data with attributes such as Input_ID (primary key), input_timestamp, and room information including number_of_rooms, room_dimensions, and floor_angle.

- User_Output Entity:
    - Stores results delivered to users, featuring attributes like Output_ID (primary key), output_timestamp, and a 3D_view.

Entity	Attributes
User	Email (Primary Key), Role, Username, Password
User_Input	Input_ID (PrimaryKey), Input_timestamp, Number_of_rooms, Room_Dimension, Floor_angle
User_Output	Output_ID (Primary Key), Output_timestamp, 3D_view, 2D_floor_plan

Table 17: Entity and Attributes

4.3.2. Establish Relationships

- User Entity:
    - This entity has a one-to-many relationship with User_Input and User_Output entities.

- User_Input Entity:
    - Linked to the User entity via a one-to-many relationship.

- User_Output Entity:
    - Connected to the User entity in a one-to-many relationship to track user interactions and outputs.

Entity	Relationships
User	One-to-Many with User_Input, User_Output
User_Input	Many-to-One with User
User_Output	Many-to-One with User

Table 18: Entity with Relationships

4.3.3. Create Entity-Relationship Diagram (ERD):

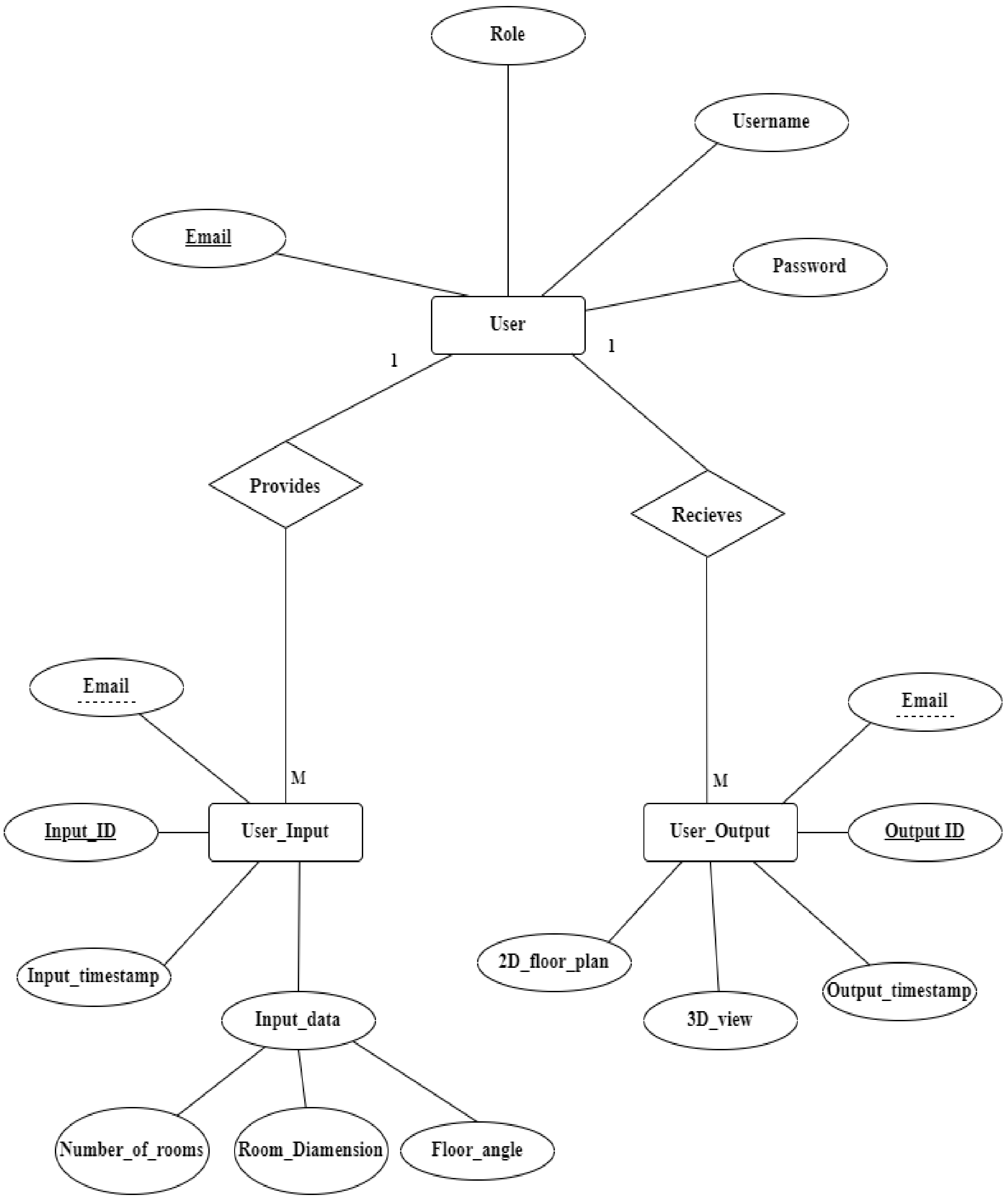


Figure 20: EER Diagram

4.4. Logical database design

- This logical database design establishes the structure, relationships, and properties of the tables and columns based on the conceptual design.

4.4.1. Normalize Data:

  - Let's break down the normalization process for each entity:
  - User Entity:
    - No normalization is required for this entity as it already seems to be in 1NF, 2NF, and 3NF.
    - All attributes are atomic.
    - There are no partial or transitive dependencies.
  - User_Input Entity:
    - 1NF: The entity appears to be in 1NF as there are no repeating groups or arrays.
    - 2NF: Since Input_ID is a composite primary key and there are no attributes that are partially dependent on only a part of the primary key, it's already in 2NF.
    - 3NF: The entity appears to be in 3NF as there are no transitive dependencies.
  - User_Output Entity:
    - 1NF: The entity appears to be in 1NF as there are no repeating groups or arrays.
    - 2NF: Since Output_ID is the primary key and there are no attributes that are partially dependent on only a part of the primary key, it's already in 2NF.
    - 3NF: The entity appears to be in 3NF as there are no transitive dependencies.
- Therefore, all entities seem to be already normalized up to 3NF, and no further normalization is required based on the provided information.

4.4.2. Define table and Columns

  - User Table:
    - Email (Primary Key)
    - Role
    - Password
    - Username
  - User_Input Table:
    - Input_ID (Primary Key)
    - Input_data
    - Input_timestamp
    - Email (Foreign Key referencing User.Email)

- User_Output Table:
  - Output_ID (Primary Key)
  - 3D_view
  - Output_timestamp
  - Email (Foreign Key referencing User.Email)
  - 2D_floor_plan

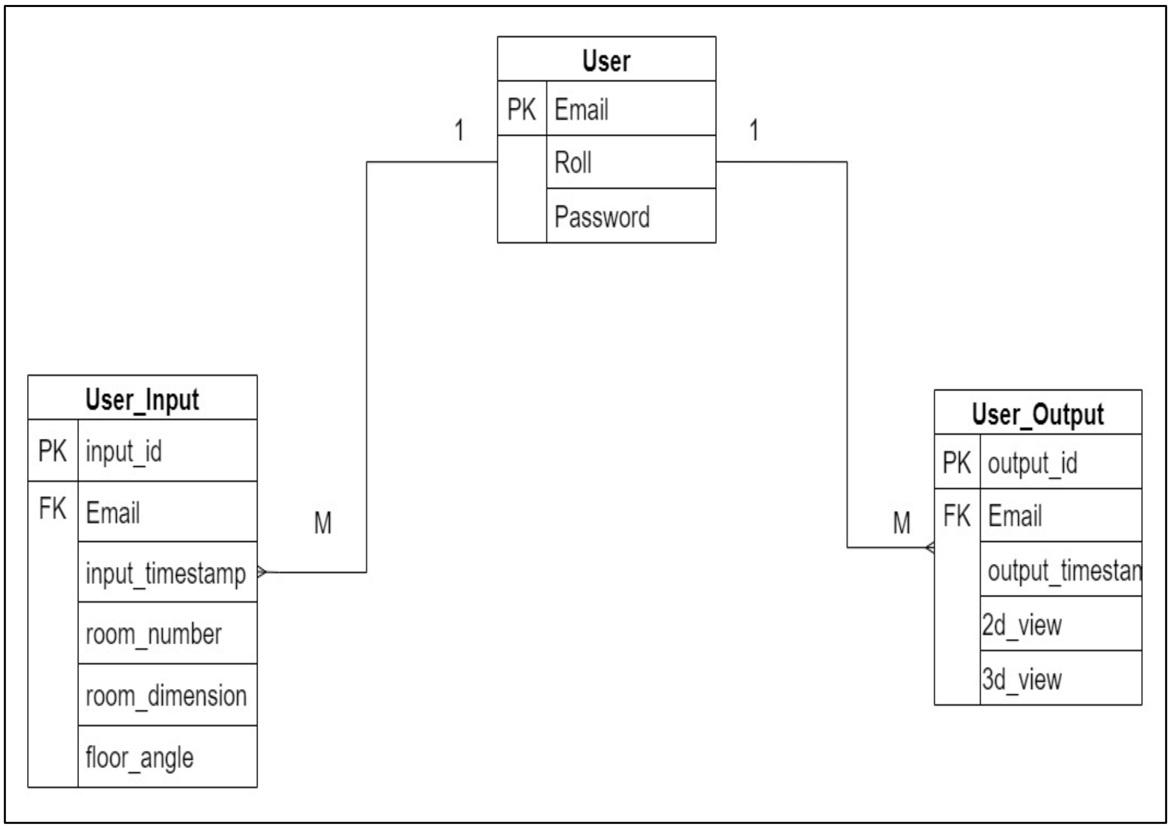


Figure 29: Logical Database Design

4.4.3. Develop Data Dictionary:

Table Name	Column Name	Description	Data type
User	Email	Primary key, unique identifier for each user	VARCHAR
	Role	Role of the user	VARCHAR
	Password	Password of the user.	VARCHAR
	Username	Username of the user	VARCHAR
User_Input	Input_ID	Primary key, unique identifier for each input	INTEGER
	Input_data	Data input by the user	VARCHAR.
	Input_timestamp	Timestamp of input	DATETIME
	Email	Foreign key referencing User table Email	VARCHAR
User_Output	Output_ID	Primary key, unique identifier for each output	INTEGER
	3D_view	Output of 3D view.	FILE
	Output_timestamp	Timestamp of output	DATETIME

	Email	Foreign key referencing User.Email	VARCHAR
	2D_view	Output of 2D view	FILE

Table 129: Data Dictionary

4.5. Schema refinement

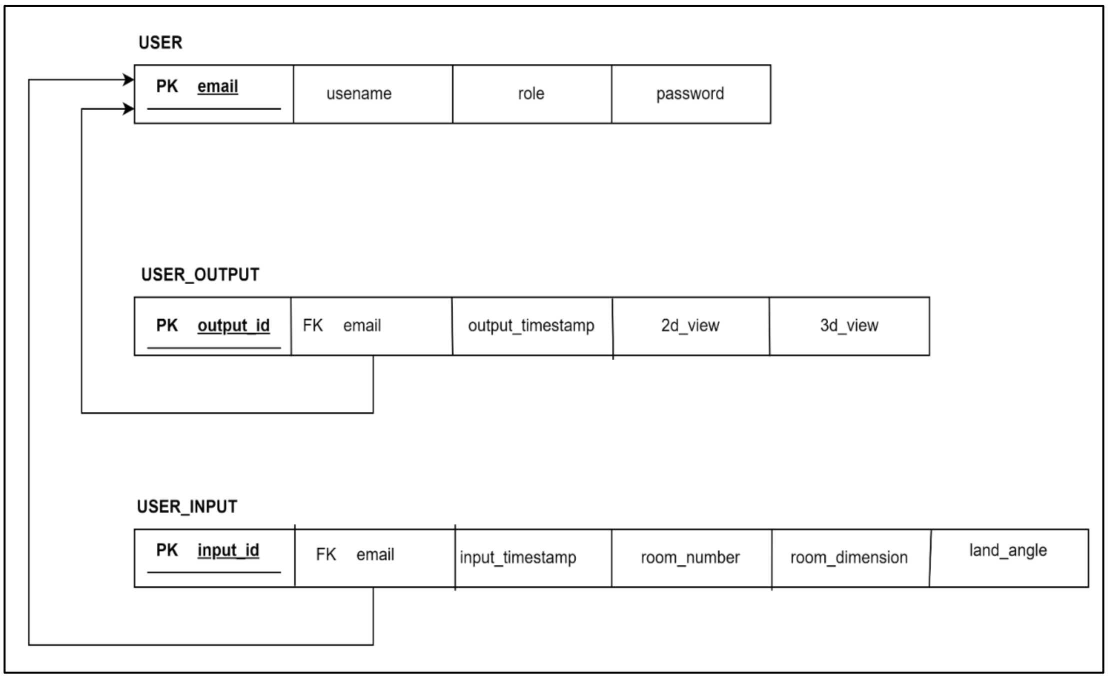


Figure 30: Schema Refinement

4.6. Physical database design

```
CREATE TABLE User (
    UserID INT PRIMARY KEY,
    Email VARCHAR(255) UNIQUE,
    Role VARCHAR(50),
    Username VARCHAR(100),
    Password VARCHAR(255)
);
CREATE TABLE UserInput (
    InputID INT PRIMARY KEY,
    UserID INT,
    InputTimestamp TIMESTAMP,
    Number_of_rooms INT,
    Room_dimension VARCHAR(255),
    Floor_angle DECIMAL(5,2),
    FOREIGN KEY (UserID) REFERENCES User(UserID)
);

CREATE TABLE UserOutput (
    OutputID INT PRIMARY KEY,
    UserID INT,
    OutputTimestamp TIMESTAMP,
    3D_view BLOB,
    FOREIGN KEY (UserID) REFERENCES User(UserID)
);
```

Figure 31: Physical Database Design

4.7. Security Design

Security Design Considerations	Description
User Verification	Verify user identities during login using strong passwords and multi-factor authentication (MFA).
Keeping Information Safe	Encrypt sensitive data at rest and in transit using encryption protocols like HTTPS.
Following Rules	Adhere to data protection regulations (e.g., GDPR, HIPAA) and obtain user consent for data usage.
Checking for Problems	Conduct regular security audits and vulnerability assessments to identify and mitigate risks.
Writing Code Safely	Implement secure coding practices to prevent common security vulnerabilities (e.g., SQL injection, XSS).
Being Prepared for Emergencies	Establish data backup and disaster recovery plans to minimize impact in case of incidents.
Educating Everyone	Provide security awareness training to users and developers to promote best practices.
Protecting the Network	Utilize firewalls, intrusion detection systems (IDS), and monitoring tools to safeguard against unauthorized access.

Table 21: Security Design

5. RESEARCH DESIGN

- The Software Design Specification (SDS) document for the system outlines the detailed design specifications and architecture for a web-based application focused on creating and visualizing 2D floor plans and 3D house views. The scope of the SDS covers user interface design considerations, system architecture utilizing a three-tier model, and data management strategies. Key design considerations include a responsive and intuitive user interface, separation of concerns using three-tier architecture (Presentation Layer, Business Logic Layer, Data Access Layer), and use of scalable cloud-based technologies for hosting and data storage. The architectural design specifies technologies for each layer, such as HTML/CSS/JavaScript and React.js for the presentation layer, Node.js and Express.js for the business logic layer, and Sequelize ORM for data access. The SDS also includes component interactions, user interface mockups, and details on development tools and technologies like version control, IDEs, and CI/CD pipelines. Overall, the SDS serves as a blueprint for implementing the functional requirements outlined in the Software Requirements Specification (SRS) document and ensures alignment with stakeholder expectations and project objectives throughout the software development lifecycle.

5.1. Study objectives-based literature review

Study Objectives	Description	Examples
Understanding User Experience (UX) Design	Learn about designing user-friendly websites that are easy to navigate and accessible on all devices.	Analyze popular websites for their UX design.
Exploring Database Management Techniques	Investigate methods to efficiently organize and manage data for the web application.	Study relational databases like MySQL.
Analyzing Security and Privacy Measures	Study methods to ensure user data is secure and private, utilizing encryption and secure login.	Research HTTPS encryption protocols.
Investigating User Input Processing	Research ways to process user input, such as room dimensions, and convert it into meaningful data.	Explore algorithms for image processing.

Reviewing Web Development Technologies	Explore modern tools and technologies used in web development to create interactive and robust sites.	Learn frameworks like React and Angular.
Exploring User Feedback Mechanisms	Examine methods for collecting user feedback to improve the website's usability and functionality.	Implement surveys and feedback forms.

Table 22:Study objectives-based literature review

5.2. Formalizing high-level implementation components

- **User Authentication Module**:
  - Responsible for user account management, including sign-up, sign-in, and account verification.
  - Implements user authentication and authorization mechanisms to ensure secure access to the system.
- **Data Input Module**:
  - Handles user inputs such as room dimensions, land shapes, and survey plan images.
  - Validates and processes user inputs to generate accurate floor plans and 3D views.
- **Floor Plan Generation Module**:
  - Generates 2D floor plans based on user inputs and specifications.
  - Utilizes machine learning algorithms or geometric modeling techniques to create accurate floor plans for different land shapes.
- **3D Visualization Module**:
  - Generates 3D views of the house based on the 2D floor plans and additional inputs.
  - Utilizes rendering techniques and visualization algorithms to create realistic 3D representations of the house designs.
- **Data Storage and Management Module**:
  - Manages the storage and retrieval of user-generated floor plans, 3D views, and other project data.

    - Ensures data integrity, security, and scalability of the system's storage infrastructure.
- **User Interface Module**:
    - Provides a user-friendly interface for users to interact with the system.
    - Displays generated floor plans, 3D views, and allows users to input parameters for house design.
- **Error Handling and Logging Module**:
    - Handles system errors, exceptions, and logs relevant information for debugging and monitoring.
    - Ensures robust error handling mechanisms to maintain system stability and reliability.
- **System Maintenance and Updates Module**:
    - Manages system maintenance tasks, updates, and version control.
    - Implements mechanisms for deploying new features, bug fixes, and system enhancements.

5.3. Data extractions, sample design, test data sets, training data sets

5.3.1. Data extractions

The system may extract user-provided data such as room dimensions, survey plan images, and land shape information to generate floor plans and 3D views.

The system may need to extract data from various sources, including user inputs, databases, external APIs, or image repositories.

5.3.2. Sample Design

Sample design may involve selecting a diverse set of house layouts, room configurations, and land shapes to train the machine learning models for generating floor plans and 3D views.

5.3.3. Test Data Sets and Training Data Sets

We use samples of the same data set for training the model and testing the model. The samples are separated in a ratio of 70:30 and 70% are used to train the model and 30% are used to test the model.

5.4. Non-functional aspects

5.4.1. Product requirements

Performance:

- The system should be able to store a minimum of 100 user-generated 2D floor plans and 3D views in the database.
- Database storage allocation for user-generated content should not exceed 1 TB .

Scalability:

- The system should be capable of handling larger file sizes, especially for GLTF/GLB files with standard-resolution textures (1024x1024 to 2048x2048 pixels).

Interoperability:

- The system must be compatible with major web browsers (Chrome version 122 or higher, Microsoft Edge version 121 or higher, Opera version 107 or higher) and mobile devices (iOS version 14 or higher, Android version 10 or higher).

Usability:

- The user interface should be optimized for a display resolution of 1920x1080 pixels.
- The System Usability Scale (SUS) score should be at least 10 in user testing.

5.4.2. Organizational Requirements

Documentation:

- Compliance with IEEE guidelines and specific submission guidelines for documenting the system.

Standard:

- The system should comply with ISO 27001 standards for information security management to ensure the protection of user data and privacy.

Security:

- Use of a strong hashing method to keep passwords secure.

5.4.3. External Requirements

Privacy Requirements:

- Compliance with GDPR regulations to protect user privacy and data rights in accordance with EU standards.
- User data should be stored and processed in compliance with relevant data protection laws, with a data breach notification time frame of 72 hours.

Safety Requirement:

- The web application should not include any features or content that pose a safety risk to users, ensuring a safe and secure user experience.

5.5. Proposed validation methods and measurements

5.5.1. Validation

Tests for units:

- Unit Test Coverage: To confirm that each component and its features work independently, make sure there is a minimum of 90% thorough test coverage.

- Defect Density: During unit testing, keep the defect density rate low—less than 5 defects per 1000 lines of code—to guarantee high code quality.

- Code Complexity: To improve readability and maintainability of the code, keep the average code complexity per unit test under 10.

Testing for integration

- Integration Test Coverage: Aim for an integration test coverage of 80% to confirm how functions and components interact with one another once machine learning has been integrated.

- Machine Learning Integration: In order to validate the functionality, make sure that the machine learning components are successfully integrated and have a 100% pass rate in integration testing.

- Compatibility testing: To obtain a 95% compatibility rating, confirm that machine learning models and algorithms work together during integration testing.

5.6. Measurements

- Positive user feedback percentages are used to evaluate the application's usability, realism, and friendliness.
- Assess the comprehensiveness of the system design specification by making certain that sufficient documentation is provided for both functional and non-functional needs.
- Megabytes are used to measure memory and the memory space related to one user.

6. APPROVAL

Signature of the team members:

Registration Number	Index Number	Name	Signature
ICT/19/20/132	5066	L. G. R. J. Lindapitiya	[illegible]
ICT/19/20/138	5071	R. M.V. P. B. Udagama	[illegible]
ICT/19/20/016	4957	B. M. C. B. K. Bandaranayake	[illegible]
ICT/19/20/059	4997	A. G. N. D. Kaluwelgoda	[illegible]
ICT/19/20/002	4944	T. M. M. M. B. Abeysinghe	[illegible]

Date: 05.05.2024

Approval of the supervisor(s)

I agree / ~~disagree~~ with the design aspects stipulated in this Software Design Specification

Name: KHA Hettige

Department/ Organization: Department of Computing

Signature: [illegible]

References:

[1] Lucidchart. (n.d.). UML Sequence Diagram. [Online]. Available: https://www.lucidchart.com/pages/uml-sequence-diagram#:~:text=A%20sequence%20diagram%20is%20a,to%20document%20an%20existing%20process. Accessed on: April 18, 2024.

[2] "Unified Modeling Language (UML) Sequence Diagrams," GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/. [Accessed: April 18, 2024].

[3] IBM. (n.d.). Three-Tier Architecture. Retrieved from https://www.ibm.com/topics/three-tier-architecture

[4] "State Chart vs. Activity Diagram: A Comparison of Modeling Tools in Software Development." Visual Paradigm Guides, Visual Paradigm, URL: https://guides.visual-paradigm.com/state-chart-vs-activity-diagram-a-comparison-of-modeling-tools-in-software-development/. Accessed: April 18, 2024.

[5] Unknown. "Title of Video," Online video, n/a. [Online]. Available: https://youtu.be/F00PKjta-NM?si=NWFZXrIetAmFEjwX. [Accessed: Date Accessed].

Index Number	Name	Task
5066	L. G. R. J. Lindapitiya	Introduction Problem to be addressed Objectives of the Project System design approach (Process models, Patterns ex. MVC, SCRUM) Component Processes and special algorithms Data Management Data requirements Conceptual database design Logical database design Research Design
5071	R. M.V. P. B. Udagama	Project Deliverables Standards to be followed Organization of the SDS Class Diagrams Tools, techniques, libraries, 3rd party tools and implementation environment Schema refinement Physical database design Security design
4957	B. M. C. B. K. Bandaranayake	Abstract Sequence diagram objective of the project Data extractions, sample design, test data sets, training data sets Non function aspect
4997	A. G. N. D. Kaluwelgoda	State machine diagram , proposed validation methods measurements Introduction of topics
4944	T. M. M. M. B. Abeysinghe	Template Design Input Design aspects