PDF Active-Reading Assistant
Initial Project Plan

Ryan Helms (rh), William Qiu (wq), Nikhar Ramlakhan (nr), Abie Safdie (as), Caleb Sutherland (cs)

4-12-2024 - v1.02

Table of Contents

1. Project Plan Revision History

Date	Author	Description
4-10-2024	nr	Created the initial document.
4-11-2024	rh	Added project Gantt chart.
4-12-2024	team	Finalized the initial document.

2. Management Plan

2.1. Team organization

The team consists of Ryan Helms, William Qiu, Abie Safdie, Caleb Sutherland, and Nikhar Ramlakhan, all computer science students in CS 422. Each member contributes to both technical and organizational aspects of the project. Our team is structured to ensure collaboration and expertise in all aspects of the project. Nikhar Ramlakhan will also oversee project management and ensures adherence to documentation.

2.2. Work division amongst members

Tasks are allocated based on individual strengths and interests, ensuring a balanced workload and diverse skill set utilization. Nikhar and William focus on system setup and backend development, while Abie, Caleb, and Ryan handle user interface design and front-end development. Collaboration and cross-functional involvement are encouraged to foster a holistic understanding of the project.

2.3. Decision making protocols

Decisions are made collaboratively, with input from all team members. Major decisions require a super-majority vote (3 out of 5) to ensure consensus and representation of all perspectives. Major decisions will be documented and signed / initialed by all team members to ensure that a record is kept in case any future disputes or disagreements.

2.4. Team meetings and communication

Communication is facilitated through a Discord channel and a WhatsApp group channel for any group discussions outside of meetings. Members are expected to communicate respectfully and within designated hours (8:00am to 10:00pm). In-person meetings are held at the Allan Price Science Commons Library on the following dates and times:

- Monday, 8 April | 6:00pm to 8:00pm
- Friday, 12 April | 4:00pm to 5:00pm
- Monday, 15 April | 6:00pm to 8:00pm
- Friday, 19 April | 4:00pm to 5:00pm
- Monday, 22 April | 6:00pm to 8:00pm
- Friday, 26 April | 4:00pm to 6:00pm
- Additional meetings can be organized at least 24 hours in advance.

3. Work breakdown schedule

3.1. Milestones

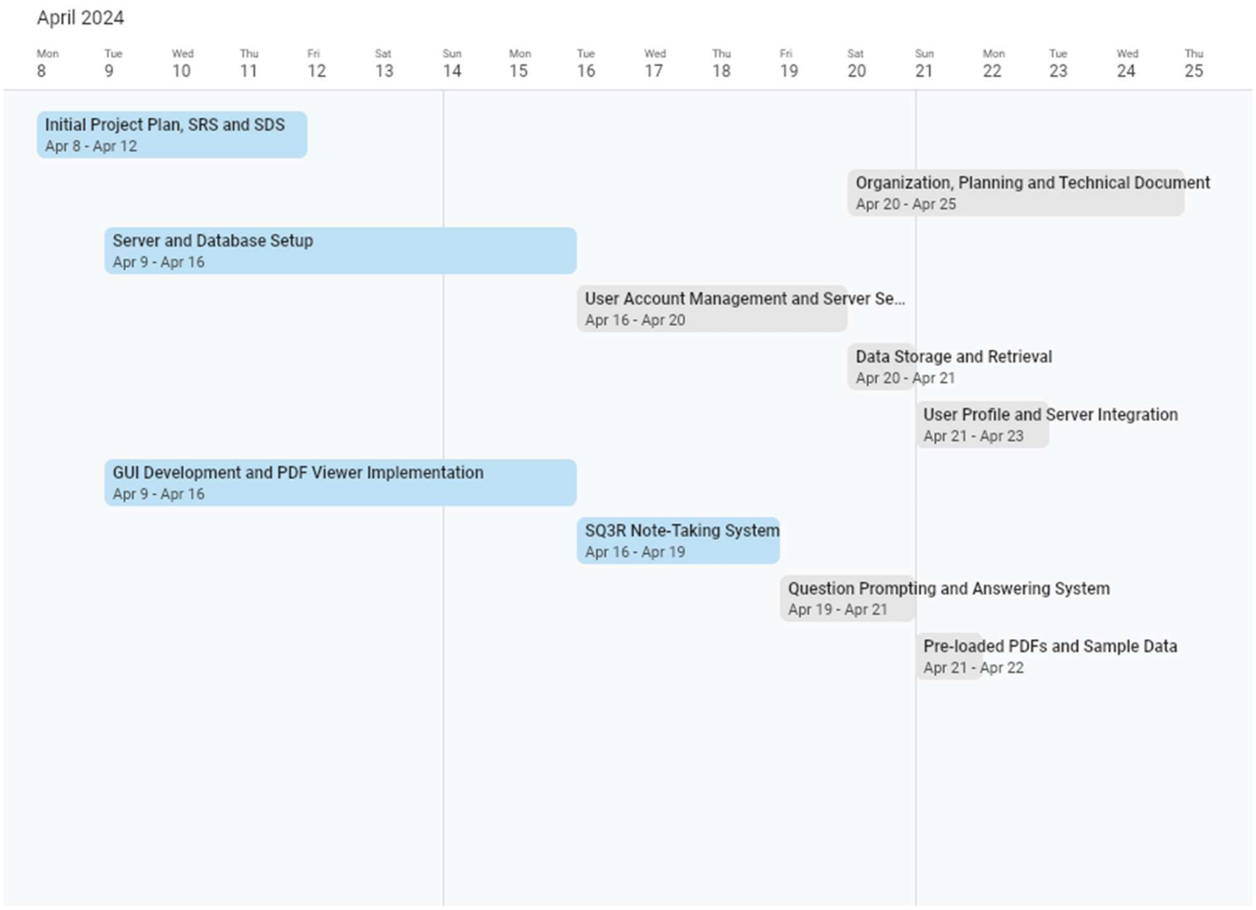
#	Projected Milestone	Projected Date
1.	Initial Project Plan, SRS and SDS	04/12/2024
2.	Server and Database Setup	04/16/2024
3.	GUI Development and PDF Viewer Implementation	04/16/2024
4.	SQ3R Note-Taking System	04/19/2024
5.	Login Page	04/20/2024
6.	User Account Management and Server Setup Prompt	04/21/2024
7.	Data Storage and Retrieval	04/21/2024
8.	Question Prompting and Answering System	04/22/2024
9.	Pre-loaded PDFs and Sample Data	04/23/2024
10.	User Profile and Server Integration	04/25/2024
11.	Organization, Planning and Technical Document	04/26/2024

3.2. Project Schedule

Following the above milestones, the project schedule can be derived as follows:

Milestone	Projected Date	Assigned Member(s)
Initial Project Plan, SRS and SDS	04/12/2024	team
Server and Database Setup	04/16/2024	wq, nr
GUI Development and PDF Viewer Implementation	04/16/2024	rh, as, cs
SQ3R Note-Taking System	04/19/2024	rh,as
Login Page	04/20/2024	as, cs
User Account Management and Server Setup Prompt	04/21/2024	wq, nr
Data Storage and Retrieval	04/21/2024	wq, nr
Question Prompting and Answering System	04/22/2024	rh, cs
Pre-loaded PDFs and Sample Data	04/23/2024	rh, as, cs
User Profile and Server Integration	04/25/2024	wq, nr
Organization, Planning and Technical Document	04/26/2024	team

3.3. Project Gantt Chart



4. Monitoring and reporting

4.1. Individual progress monitoring

Individual progress will be primarily monitored through the GitHub repository and the developer log / document log markdown file. Each team member is responsible for regularly updating the developer log / document log with their contributions, changes made, and revisions. This log will serve as a central hub for tracking individual progress and documenting project developments.

Regular team meetings will be held, providing an opportunity for members to report their progress, discuss any challenges or roadblocks, and coordinate on tasks. At the beginning of each meeting, members will provide updates on their individual contributions, which will be documented in the developer log / document log.

4.2. Project progress monitoring

Project progress will be monitored through various channels to ensure alignment with the project plan and objectives:

- Documentation Updates: Documentation, including the project plan, SRS, and SDS, will be continuously updated to reflect project progress and any changes or refinements made. During team meetings, documentation will be reviewed and revised as necessary to ensure accuracy and alignment with project developments.
- Communication Channels: WhatsApp and Discord will serve as additional avenues for minor updates, progress reviews, and ad-hoc discussions among team members. While major updates and progress reports will be documented in the developer log / document log, these communication channels will facilitate real-time collaboration and information sharing.
- Gantt Chart: A Gantt chart will be developed as part of project monitoring, providing a visual representation of project timelines, milestones, and dependencies. The Gantt chart will be updated regularly to reflect progress and any adjustments to the project schedule.

5. Build plan

5.1. System build plan

1. Initial Setup
    - Establish development environment
    - Set up GitHub repository
    - Create project files and directories
2. Backend Development
    - Implement the server for storing program data
    - Develop system for storing user data and profiles
    - Set up database for storing user information
    - Create system to prompt user inputs for server setup
3. Frontend Development
    - Design and develop GUI for accessing PDF files and parsing text
    - Implement hierarchical SQ3R structure for capturing user notes
    - Develop system for hiding and unhiding notes
    - Create system for cycling through questions in user notes
4. Login Page
    - Develop system to validate user access.
5. User Management and Authentication
    - Create system for managing user accounts and profiles
    - Develop functionality for pre-defining user accounts
6. Integration and Testing
    - Integrate frontend and backend components
    - Conduct testing to ensure functionality and reliability
    - Address any bugs or errors identified during testing
7. Deployment and Release
    - Prepare system for deployment to production environment
    - Develop deployment scripts and procedures
    - Release system to end-users
8. Sample Data Preparation
    - Generate pre-loaded complete notes
    - Populate database with sample data for testing and demonstration purposes

5.2. Explanation of system build plan

1. Modularity and Scalability: Breaking the system into distinct components allows for modularity, making it easier to manage and scale the project. Each component can be developed, tested, and deployed independently, reducing complexity and facilitating future updates or expansions.

2. Clear Functionalities: Each step in the build plan corresponds to a specific functionality or feature of the PDF Active-Reading Assistant. This approach ensures clarity and focus, enabling the team to prioritize tasks effectively and deliver incremental value to users.

3. Risk Management: By identifying and addressing potential risks early in the development process, the build plan aims to mitigate project risks and uncertainties.

4. User-Centric Design: The build plan prioritizes functionalities that directly impact the user experience, such as GUI development, note-taking features, and PDF parsing capabilities. This user-centric approach ensures that the system meets the needs and expectations of its intended users, enhancing usability and satisfaction.

5. Progressive Enhancement: The build plan follows a progressive enhancement approach, starting with essential functionalities and gradually adding more advanced features. This incremental development strategy allows for early feedback from developers and users, enabling iterative improvements based on real-world usage and feedback.

5.3. Risks and risk reduction strategies

1. Research and Prototyping: Before proceeding with development, conduct thorough research and prototyping to explore potential solutions, identify technical challenges, and validate assumptions. This helps mitigate risks associated with unknowns and uncertainties.

2. Regular Testing and Quality Assurance: Implement robust testing processes, including unit tests, integration tests, and end-to-end tests, to identify and address bugs, errors, and inconsistencies. Prioritize quality assurance throughout the development lifecycle to ensure the reliability and stability of the system.

3. Continuous Communication and Collaboration: Foster open communication and collaboration within the team to address challenges, share insights, and align on objectives. Regular team meetings, progress updates, and discussions help identify and resolve issues promptly, minimizing the impact on project timelines and deliverables.

6. Acknowledgements

The content of this document is inspired by the Project 1 Evaluation Criteria provided by Prof. Anthony Hornof.

This document template is built and derived from SRS/SDS template provided by Prof. Anthony Hornof. Additionally, it builds on a document developed by Stuart Faulk in 2017, and on the publications cited within the document, such as IEEE Std 1016-2009.

PDF Active-Reading Assistant
Initial Software Requirements Specification

Ryan Helms (rh), William Qiu (wq), Nikhar Ramlakhan (nr), Abie Safdie (as), Caleb Sutherland (cs)

4-12-2024 - v1.00

For this document, we will be using the provided PSD Active-Reading Assistance Software Requirements Specification issued by Prof. Anthony Hornof.

PDF Active-Reading Assistant
Initial Software Design Specification

Ryan Helms (rh), William Qiu (wq), Nikhar Ramlakhan (nr), Abie Safdie (as), Caleb Sutherland (cs)

4-12-2024 - v1.02

Table of Contents

1. SDS Revision History

Date	Author	Description
4-10-2024	nr	Created the initial document.
4-11-2024	as, cs	Provided models, descriptions, and general edits for Software Modules Section
4-12-2024	team	Added static and dynamic module diagrams.

2. System Overview

The system, named the PDF Active-Reading Assistant, aims to provide users with tools to enhance their active reading experience with PDF documents. It will consist of components for accessing, note-taking, and organizing PDF files, as well as a server component for storing user data.

3. Software Architecture

The software architecture of the PDF Active-Reading Assistant is designed to facilitate efficient interaction between its various components, ensuring seamless functionality and ease of maintenance. The architecture is composed of several key components, each serving a specific role in the system.

1. Set of Components:
   - PDF Viewer: Responsible for accessing PDF files and rendering their content for display to the user.
   - Annotation System: Facilitates user annotation of PDF documents, providing tools for highlighting, commenting, and organizing annotations.
   - Login Page: Responsible for validating users.
   - Quizzing: Responsible for testing knowledge acquired by applying the SQ3R method.
   - Server: Manages the storage and retrieval of user data, including annotations and user profiles.

2. Functionality Provided by Each Component:
   - PDF Viewer: Renders PDF content on the user interface, allows navigation through the document, and interfaces with the Annotation System to display user annotations.
   - Annotation System: Provides tools for creating, editing, and organizing user annotations, stores annotations in the server, and interfaces with the PDF Viewer for display.
   - Login Page: Validates user access.
   - Quizzing: Displaying and hiding note taking components.
   - Server: Stores user data, manages user authentication, and provides APIs for communication between client-side components.

3. Interaction Between Modules:
   - The PDF Viewer interacts with the Annotation System to display user annotations alongside the PDF content, enabling users to view and manage their annotations in real-time.
   - The Annotation System communicates with the Server to store and retrieve user annotations, ensuring data persistence across sessions and devices.
   - Login Page links each user to the database.
   - Quizzing: Interacts with the annotation system.

4. Rationale for the Architectural Design:

   The architectural design was chosen to promote modularity, scalability, and maintainability of the system. By decomposing the system into separate components, each responsible for a specific aspect of functionality, we ensure that changes or updates to one component do not affect the others. Additionally, the use of a server component allows for centralized data storage and management, facilitating collaboration and synchronization among multiple users. Overall, this architecture provides a solid foundation for the development of a robust and feature-rich PDF Active-Reading Assistant.

4. Software Modules

4.1. PDF Viewer

Role and Primary Function

The PDF Viewer module is responsible for rendering PDF files within the application's user interface. Its primary function is to display PDF content to users, allowing them to navigate through the document, zoom in/out, and interact with annotations.

Interface to Other Modules

- Interfaces with the Notetaking System module to display user notes alongside the PDF content.
- Communicates with the application to provide tools for viewing the PDF. Such as loading, zooming, and scrolling.

Static Model

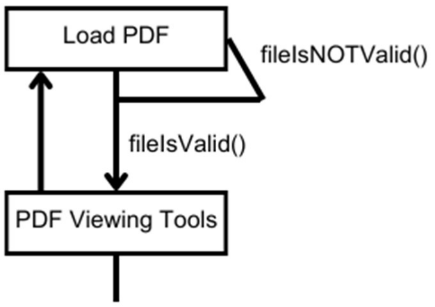


Figure 1

The static model (Figure 1) of the PDF Viewer module depicts its key components and their relationships. It includes components for loading a PDF, zooming, scrolling, and showing/hiding the notetaking area.

Dynamic Model

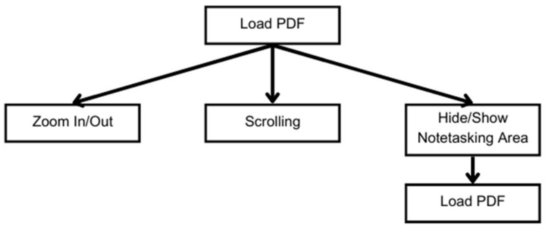


Figure 2

The dynamic model (Figure 2) illustrates the flow of control and data within the PDF Viewer module. For instance, if the user attempts to load a file that is invalid, it will be prompted to load another PDF. Otherwise, it will be granted access to the PDF viewing tools.

Design rationale

The PDF Viewer module was designed to provide users with a seamless and intuitive experience for viewing PDF documents within the application. Its separation from other modules allows for independent development and maintenance, ensuring flexibility and scalability. By incorporating both static and dynamic models, the design rationale emphasizes clarity and completeness in describing the module's functionality and interactions.

Alternative designs

Several alternative designs could be considered for the PDF Viewer module, including different rendering engines, user interface layouts, and interaction paradigms. Ultimately, the chosen design will be selected for its balance of performance, usability, and compatibility with other system components. Alternative approaches will be documented and evaluated to ensure that the final design met the project's requirements and constraints.

4.2. Annotation System

Role and Primary Function

The Annotation System module is responsible for managing user-generated annotations within the application. Its primary function is to allow users to create, edit, and delete annotations alongside PDF documents, providing a collaborative environment for document review and discussion.

Interface to Other Modules

- Interfaces with the PDF Viewer module to display annotations alongside the PDF content.
- Communicates with the Server module to store and retrieve annotation data from the database.

Static Model

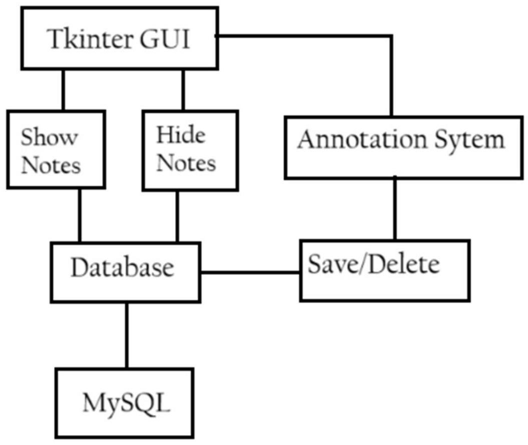


Figure 3

The static model (Figure 3) of the Annotation System module outlines its components for managing annotations, including storage, rendering, and user interaction functionalities.

Dynamic Model

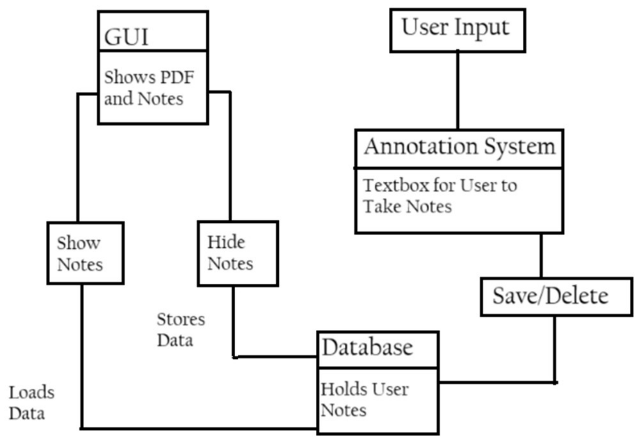


Figure 4

The dynamic model (Figure 4) illustrates the flow of annotation creation, editing, and deletion within the Annotation System module. It demonstrates how user actions trigger updates to annotation data and presentation.

Design rationale

The design of the Annotation System module prioritizes flexibility and scalability to accommodate various types of annotations and user interactions. By separating annotation management from PDF rendering, the module allows for independent development and extensibility. The dynamic model reflects the module's responsiveness to user actions and its integration with other system components.

Alternative designs

Alternative designs for the Annotation System module will be explored, including different data structures for storing annotations, user interface layouts for annotation editing, and synchronization mechanisms for collaborative annotation. The chosen design will be selected based on its suitability for supporting real-time collaboration, efficient data storage, and seamless integration with the PDF Viewer and Server modules.

4.3. Login Page

Role and Primary Function

The Login Page module is responsible for verifying which user is requesting to access the application. Its primary function is to grant access and provide the database component of the software with the information needed to load the appropriate data stored on the server.

Interface to Other Modules

- Interfaces with the application to provide user identification.
- Communicates with the Server module to store data with respect to each user.

Static Model

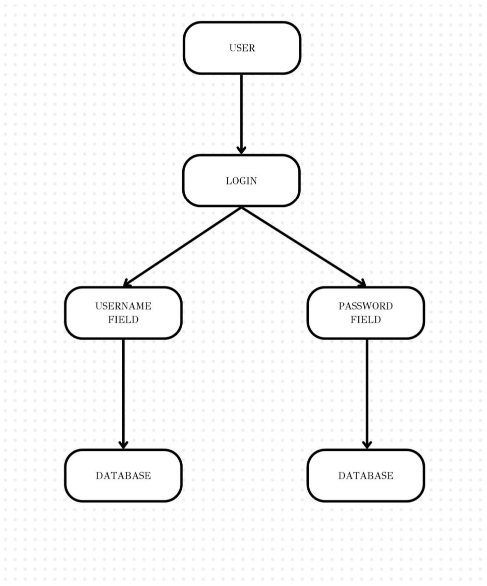


Figure 5

The static model (Figure 5) of the Login Page module outlines what components the login page will interact with. These include the username and password input fields and the database associated with the application. The login page will check the database to ensure the username and password the user submitted matches the information stored.

Dynamic Model

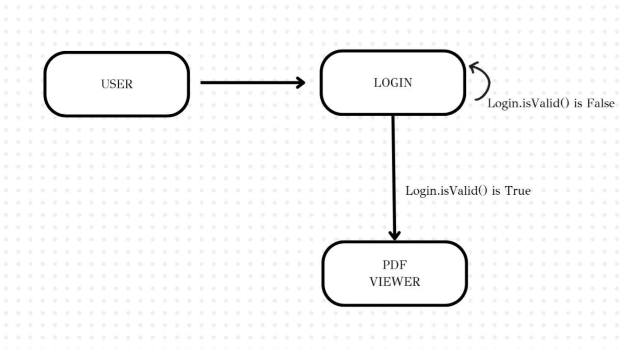


Figure 6

The dynamic model (Figure 6) illustrates the login page with respect to valid information. If the user inputs invalid information, they will have to re-input their data. If the data they enter is valid, they will gain access to the PDF viewer.

Design rationale

The design of the Login Page module focuses on ease-of-use and security. By having a simple username and password field, the user will be able to quickly input their credentials to access the site. Security is ensured by deploying access control. Meaning, if the user inputs valid credentials they will be given access to *only* their notes and other information that they stored on the server.

Alternative designs

Alternative designs for the Login module will be considered on different effectiveness and ease of use for the user. The intent is for the user to sign-in and access their respective data with ease.

4.4. Server

Role and Primary Function

The Server module is responsible for managing application data, including user accounts, annotations, and PDF document metadata. Its primary function is to provide a centralized repository for storing and retrieving data, ensuring data integrity, security, and scalability.

Interface to Other Modules

- Interfaces with the Annotation System module to store and retrieve annotation data.
- Communicates with the PDF Parser module to store parsed content metadata.
- Provides APIs for client modules to access and manipulate data stored in the database.

Static Model

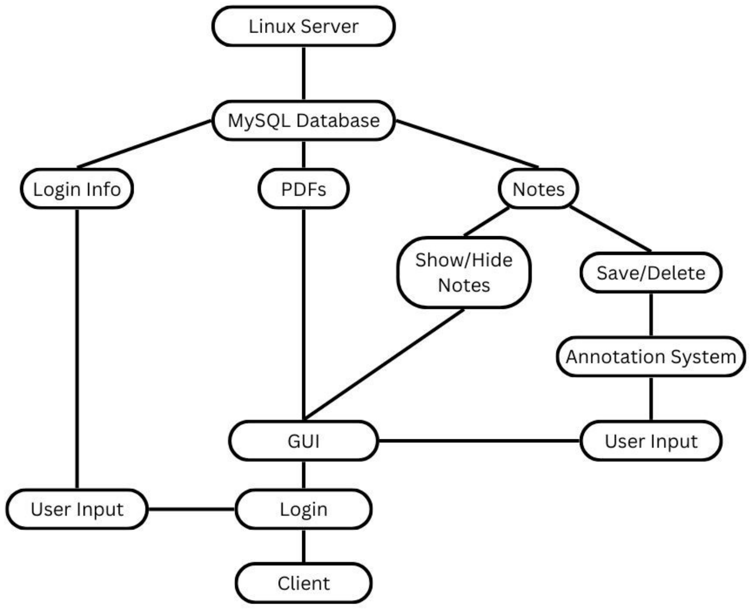


Figure 7

The static model (Figure 7) of the Server module outlines its components for data management, including database management, authentication, and data access functionalities.

Dynamic Model

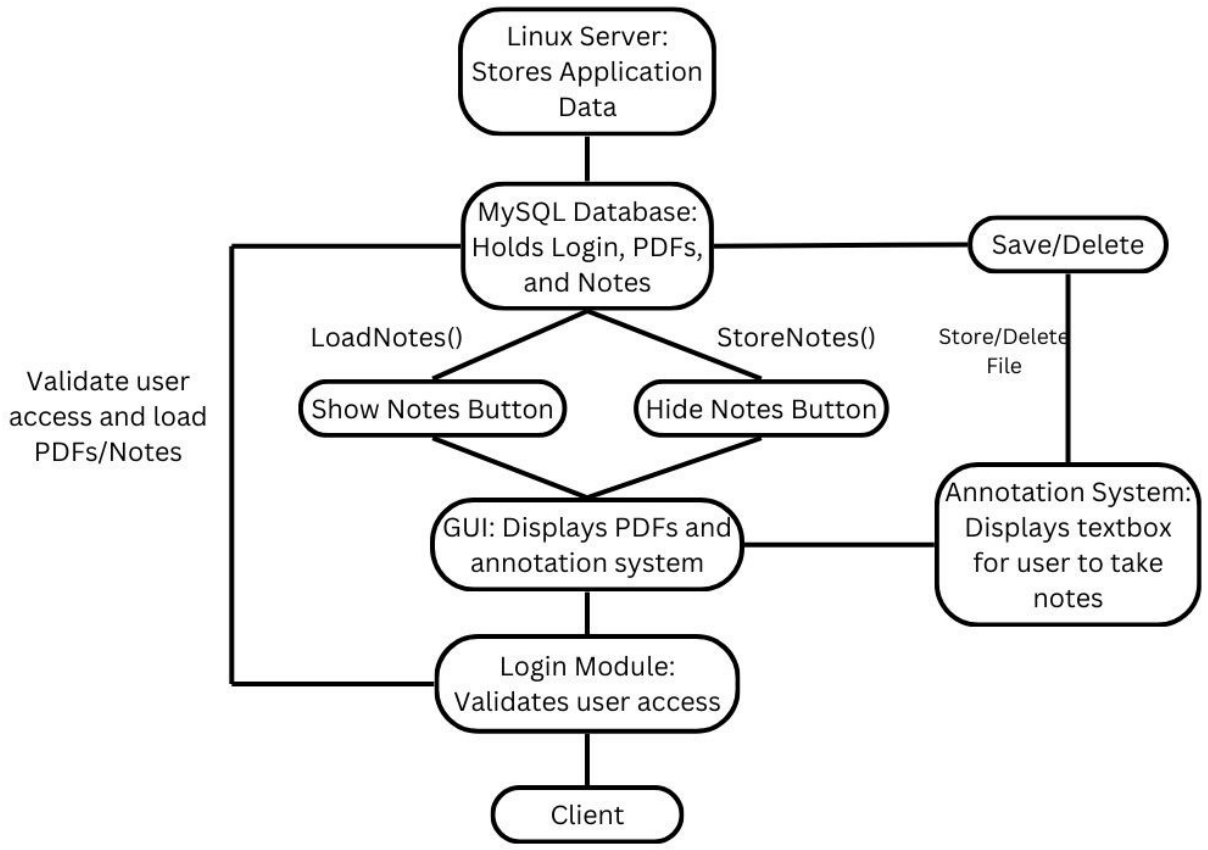

Linux Server: Stores Application Data
MySQL Database: Holds Login, PDFs, and Notes
Save/Delete
LoadNotes()
StoreNotes()
Store/Delete File
Validate user access and load PDFs/Notes
Show Notes Button
Hide Notes Button
GUI: Displays PDFs and annotation system
Annotation System: Displays textbox for user to take notes
Login Module: Validates user access
Client

Figure 8

The dynamic model (Figure 8) illustrates the flow of data within the Server module, demonstrating how client requests are processed, authenticated, and served, and how data is retrieved from and stored in the database.

Design rationale

The design of the Server module emphasizes reliability, security, and scalability in managing application data. By employing a client-server architecture, the module enables concurrent access from multiple clients while enforcing access controls and data consistency. The dynamic model reflects the module's handling of client requests and its integration with other system components for seamless data exchange.

Alternative designs

Alternative designs for the Server module explored different database technologies, authentication mechanisms, and data storage models. The chosen design was selected based on its suitability for supporting concurrent access, data encryption, and backup/restore functionalities, while allowing for future enhancements and optimizations.

4.5. Self-Quizzing

Role and Primary Function

The Self-quizzing module is responsible for allowing the user to quiz themselves with the knowledge they acquired from reading the PDF document in a SQ3R manner.

Interface to Other Modules

- Interfaces with the Annotation System module to show/hide notes that the user took.
- Interfaces with the Database to load notes that were inputted by the user.

Static Model

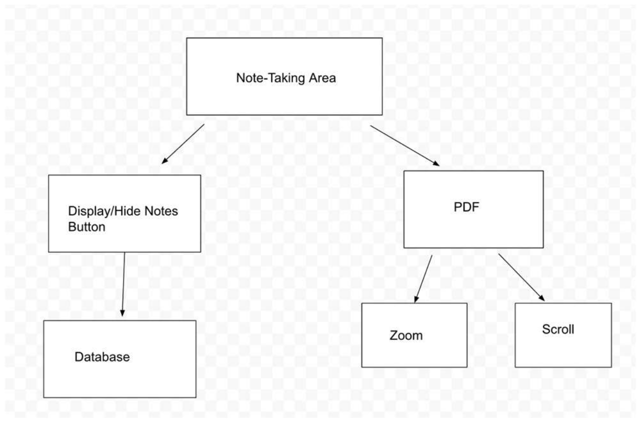


Figure 9

The static model of the Self-Quizzing module (Figure 9) showcases its interaction to different components of the application. These include the Notetaking area, the hide/show button, the database, and the PDF. As a result, the self-quizzing module will have the ability to show/hide notes, load/store notes to the database, and change the size and view of the PDF.

Dynamic Model

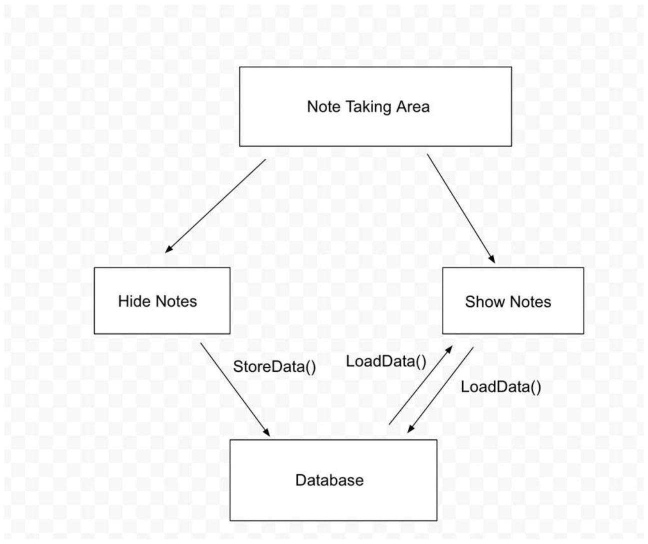


Figure 10

The dynamic model (Figure 10) illustrates the flow of control for the self-quizzing module. The user will be able to hide/show notes and load/store data from the database.

Design rationale

The design of the self-quizzing module will be focused on the showing/hiding of the notetaking area. At the forefront of design will be ease-of-use and reliability. The showing/hiding of notes will be enacted via a single button press. The reliability of notes loading/storing from the database will be necessary to establish.

Alternative designs

Alternative designs for the self-quizzing module will be considered as development progresses. If there is an optimization that improves ease-of-use and reliability it will be enacted.

5. Dynamic Models of Operational Scenarios

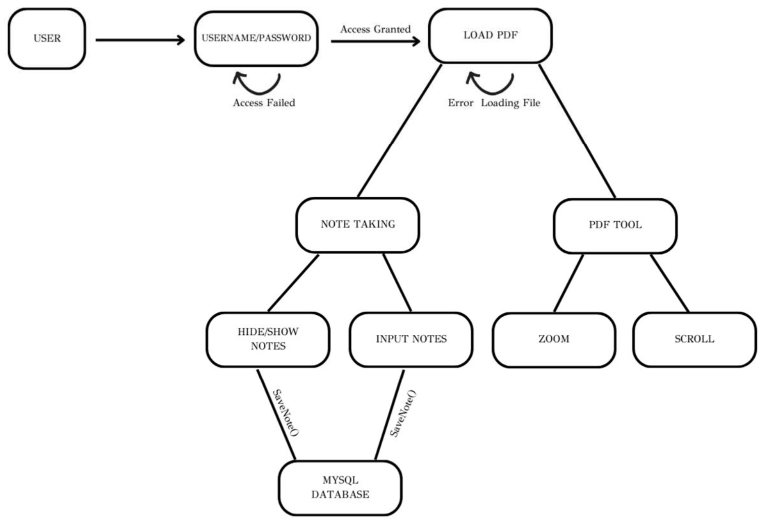


Figure 11

Figure 11 illustrates the primary use case scenario for PDF Active-Reading-Assistant. The diagram highlights that the initial step involves user authentication, followed by the ability to upload a PDF for note-taking or utilizing PDF tools. Regarding note-taking functionality, users can seamlessly display, load, and store notes within the database. As for PDF tools, users have the capability to zoom in and scroll through the document effortlessly.

6. References

Faulk, Stuart. (2011-2017). CIS 422 Document Template. Downloaded from https://uocis.assembla.com/spaces/cis-f17-template/wiki in 2018. It appears as if some of the material in this document was written by Michal Young.

IEEE Std 1016-2009. (2009). IEEE Standard for Information Technology—Systems Design—Software Design Descriptions. https://ieeexplore.ieee.org/document/5167255

J, Anthony Hornof. "Software Design Specification [Template]." 6 May 2019.

Sommerville, I. (2000) Software Engineering. Addison-Wesley, Harlow.

7. Acknowledgements

This document is built with reference to the SDS template provided by Prof. Anthony Hornof. Additionally, it builds on a document developed by Stuart Faulk in 2017, and on the publications cited within the document, such as IEEE Std 1016-2009.