SOFTWARE DESGIN DOCUMENTATION

for

Learnix

Prepared by Vincent Le, Bayasgalan Battogtokh, Devin Irby, Miguel Lima, Wyatt Stohr, Xander Thompson, Boscoe Lindholm

Project 1: AcademicGPT

March 12, 2024

Contents

List of Figures

1 Introduction

1.1 Purpose of the Document

The Learnix Software Design Documentation is intended to serve as the blueprint for implementation of the requirements set out in the Software Requirements Specification. This document intents to:

- Communicate rationale for design decisions to key stakeholders and the development team
- Establish guidance on implementation of architecture, modules, components, and their interactions
- Ensure proper project planning
- Allow for proper traceability from the requirement analysis to the implementation of the software

In short, the Learnix Software Design Document seeks to document and communicate how we plan to create the software.

1.2 Scope

1.2.1 Functional Scope

Learnix will have the following functional scope:

- Interactive UX: Enabling users to interact with the application through queries and presenting results clearly and concisely
- User profiles: Enabling users to set up profiles that keep track of their search history, login information, and number of queries. This feature lets users track their research progress and personalizes the user experience
- Sign in/Log out feature: Users will gain access to their profiles and save their information by using the sign in feature. By logging out, users can safely exit the app and protect their privacy.
- Filter function: Will recognize and remove non-academic results will ensure that the results returned are relevant to research.
- Search Query Results: Display relevant information, and present search results in an approachable format.
- Copy/paste feature: Users will be able to save the copied text to their clipboard and paste it somewhere else.

1.2.2 Non-Functional Scope

Learnix will have the following nonfunctional scope

- Performance:Because Learnix is intended to be proof-of-concept, the system shall respond to user queries within a reasonable period, not exceeding 20 seconds under normal load conditions. It should also implement caching strategies to improve response times.
- Scalability: The system shall be designed with scalability in mind, capable of supporting an increasing number of users and data volume without a significant drop in performance. It should be able to scale horizontally and vertically as required
- Usability: The UI should be intuitive for use in academic research at a university level
- Maintainability: The system shall be maintainable, with clear documentation and modular code to facilitate updates and bug fixes. It should follow best practices for code quality and testing

1.3 Supported Platforms

As a proof-of-concept, Learnix will only be available on the Android platform.

1.3.1 User Profiles

The intended userbase for Learnix will be college students. The intended age range is 18 years or older. Reading level for Learnix should be at the college level. Learnix's training database consists of academic papers written by and for scholars. Therefore, the Llama 2 responses should be a summary of its findings an undergraduate student could understand. The user should be able to operate a phone and web browser with some accessibility assistance where it may be necessary. Learnix assumes the user can use a touch screen and see it. If this is a problem for the user, then the software should still be usable with external assistance, such as OS level accessibility features.

1.3.2 Limitations

- Regulatory policies: Llama 2 License rights and agreements.
- Hardware Limitations (e.g., signal timing requirements): 64GB of memory to run a quantized 70B Llama 2 Model is the minimum requirement. For faster generation 48GB VRAM is needed to fit entire model.
- Data availability and Quality: Availability of high-quality training data is crucial for the AI model's performance.
- Security Constraints: User information should be stored encrypted. Sensitive information during
- the process should be left out of the GitHub repo.
- Language Support: Software requires user to insert input in English measures within reason
- should be applied to ensure input is in English.
- Cost Constraints: The developer budget is low. Using technologies that are low cost and open
- source.

1.3.3 Intended Usage

Learnix will be a proof-of-concept android application that will allow users to ask an AI chatbot to search for academic research on any subject they want. Learnix is intended to make research tasks easier by saving time and effort though quickly finding and retrieving relevant and credible scholarly articles. Through Learnix, researchers will improve their knowledge in their research topics. Learnix is the solution to increase research productivity and efficiency all while improving research quality by providing access to credible scholarly sources.

1.3.4 Future Enhancements

Future enhancements can include:

- Deploying to live audiences
- Expanding to Apple platforms
- More granular filterin

1.3.5 Version Control

Version control will be conducted through a project page hosted on GitHub.

2 High-Level Design

The front-end will be made with React, written in JavaScript/TypeScript. The backend will use the Llama-2 AI model and MySQL for the database.

2.1 Data Structures and Types

The software will utilize specific data structures and data types for expression evaluation and storage:

- User Information
- E-mail (Primary)
- Password
- Query IDs (Foreign 1:N)
- Search Query
- Query ID (Primary)
- User ID (Foreign)
- User prompt
- Response

2.2 User Interface (UI/UX)

The user interface will be simple in design with minimal displays made using React software. A main page, search bar, and a side bar menu that can open anywhere.

- Main Page: The current page. It takes up most of the screen and displays whatever the page has, whether that be the home page or the search results.
- Search Bar: Placed at the top of the screen. The user can type in whatever they want and once confirmed, the Llama2 AI will generate a response. The main page will show the search query and response below the search bar.
- Side Bar: A side menu that pops up when a user swipes from the left edge to the right or with a button. It shows the last five searches a user made, including other options like settings and logging out.

React will be used since it's a well-liked JavaScript user interface library that is frequently used in front-end development. Additionally, React is allows for:

- Cross development: React Native saves time and money by enabling to write code once and have it run in both Android and IOS
- Faster development cycles: Code can instantly rebuild the entire app thanks to React Native's hot reloading feature.
- Reusable components: Code can be written once and employed in multiple instances. This feature not only promotes efficiency but also converses time and energy.

2.3 Usability and Functionality

Users of the application can login and logout of the app. While using the app, the users can type a question into the search bar and when the search bar button is pressed the question will be sent as a question to the backend Llama-2 AI model. The Llama-2 model will return a result that will be shown for the user to read. The most recent five searches will be found on the side bar. The side bar will also hold the logout button and settings button.

2.4 Non-Functional Requirements

In this section, we outline the non-functional requirements that the software must meet. Non-functional requirements describe the attributes and characteristics of the software beyond its specific functionalities.

2.4.1 Performance

- Response Time: The system should respond to user inputs within an acceptable time frame.
- Scalability: The software should be able to handle an increasing number of users or computations efficiently.
- Resource Utilization: The software should use system resources, such as memory and CPU, efficiently.
- User Interface: The user interface should be intuitive and easy to use.
- Accessibility: The software should be accessible to users with disabilities.
- Compatibility: It should be compatible with different devices and web browsers.

2.4.2 Security

- Data Security: User data should be stored securely and protected from unauthorized access.
- Authentication and Authorization: The software should provide appropriate authentication
- and authorization mechanisms.
- Data Privacy: Ensure that user data is not shared or used without explicit consent.

2.4.3 Reliability

- Availability: The software should be available with minimal downtime.
- Fault Tolerance: It should continue to operate even in the presence of software or hardware failures.
- Data Backup and Recovery: Implement regular data backups and provide mechanisms for data recovery

2.4.4 Performance Testing

- Load Testing: Conduct load testing to evaluate performance under heavy user loads.
- Stress Testing: Test the software's limits to determine breaking points.

3 System Architecture

In this section, we provide an overview of the software's architecture, describing how different modules interact and the overall structure of the system.

3.1 Architecture Overview

The software is designed using a modular and layered architecture to ensure flexibility, scalability, and maintainability. The system is divided into several key modules that work together to provide the desired functionality. The core components of the architecture include:

3.1.1 User Interface (UI)

The user interface module is responsible for interacting with users, collecting input, and displaying results. It provides an intuitive and user-friendly interface for users to interact with the generative AI.

3.1.2 Parsing

The parsing module handles the conversion of the user's input into a format suitable for evaluation. It includes a lexer and parser to break down complex input into simpler components.

3.1.3 Evaluation Engine

The evaluation engine module is at the heart of the system, responsible for evaluation and solutions on parsed inputs. It utilizes the LLama 2 AI model for evaluation and solution.

3.1.4 Data Structures

Data structures, including stacks for managing input and output, are utilized by the Llama 2 powered AI engine to process queries effectively. These data structures ensure the proper handling of the application with its levels of precedence and associativity.

3.1.5 Output Rendering

The output rendering module formats and displays the results to the user. It is responsible for presenting the results in a clear and understandable manner.

3.1.6 Module Interaction

The software modules interact in the following sequence:

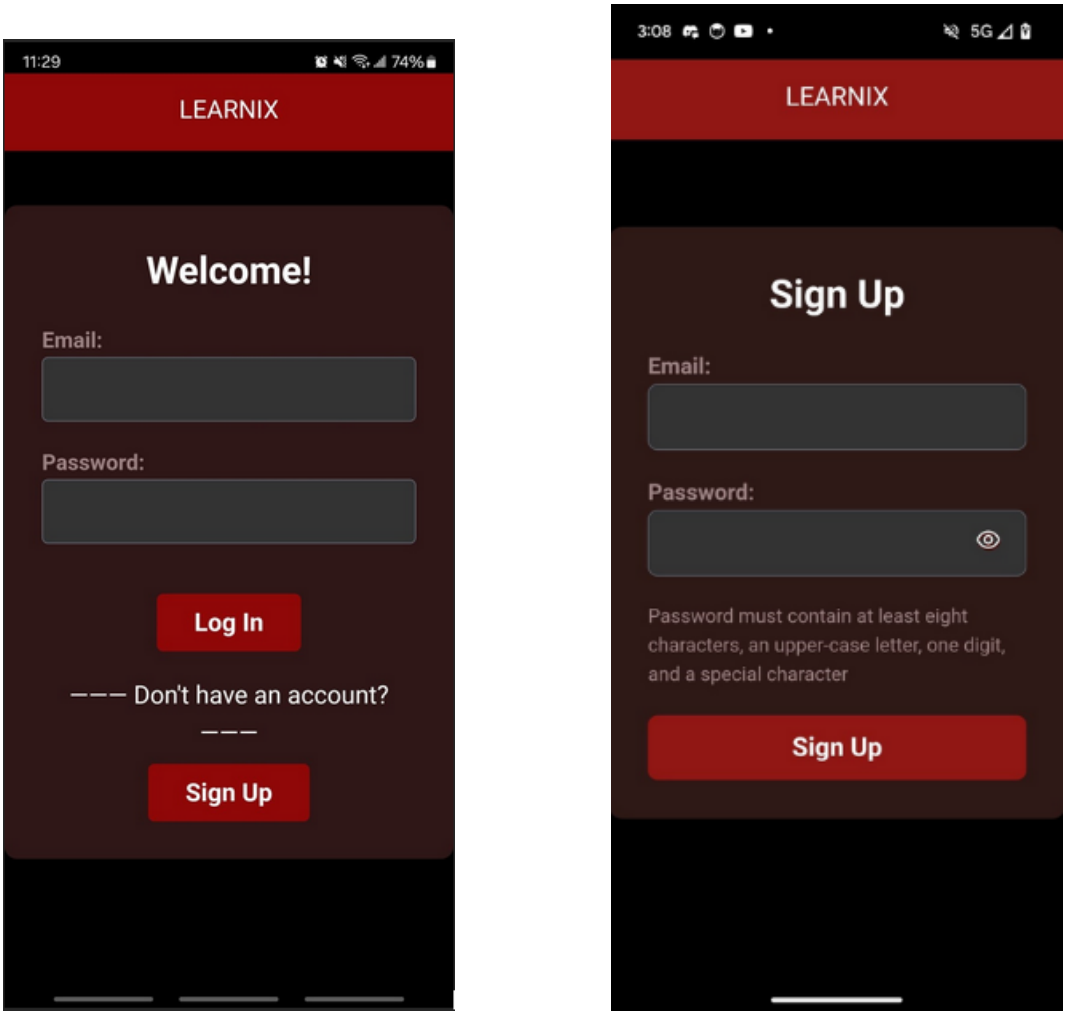
- The User Interface (UI/UX) module collects the input from the user and forwards it to the AI module.
- The Llama 2 module tokenizes and parses the input, converting it into a format suitable for evaluation.
- The parsed input is passed to the AI Engine module, which uses the Llama 2 model to perform responses.
- The Function Library is accessed by the AI Engine to handle built-in functions. Data structures are used to manage inputs and responses during evaluation. The result is then passed back to the User Interface module for display.

This layered architecture ensures a separation of concerns and allows for flexibility in extending the system's functionality in the future.

3.2 GUI/GUX Design

3.2.1 Welcome Page

The welcome page has the authentication system with a function responsible for handling user login by sending credentials to a backend API. The welcome page component also manages the user interface for the login page, interacting with the login function and updating the UI based on the authentication response.



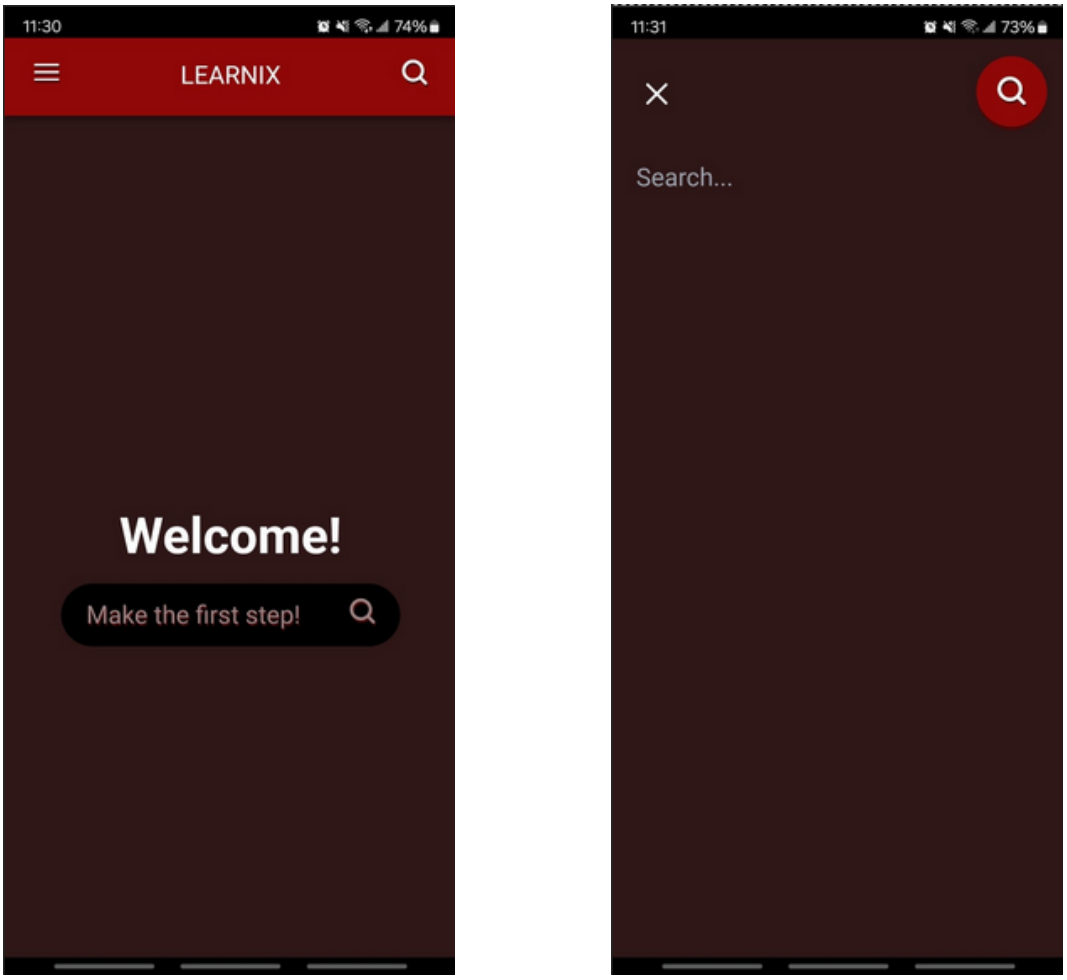
(a) Welcome Page

(b) Sign Up Page

Figure 1: The Welcome and Sign-Up Pages

3.2.2 Sign Up Page

The sign up page will allow users to make an account for Learnix by adding and email and password. This page will also notify users if they have created password of sufficient strength.



(a) Search Page (b) Input Page

Figure 2: The Search and Input Pages

3.2.3 Input Field

The main input field is where the user enters the questions. It is a text entry area that allows users to input using the keyboard.

3.2.4 Sidebar

The sidebar allows the user to view and pull up their piror searches, logout, and go back to the homepage.

Figure 3: The Sidebar

3.2.5 Output Page

The output page shows the results from the search input.

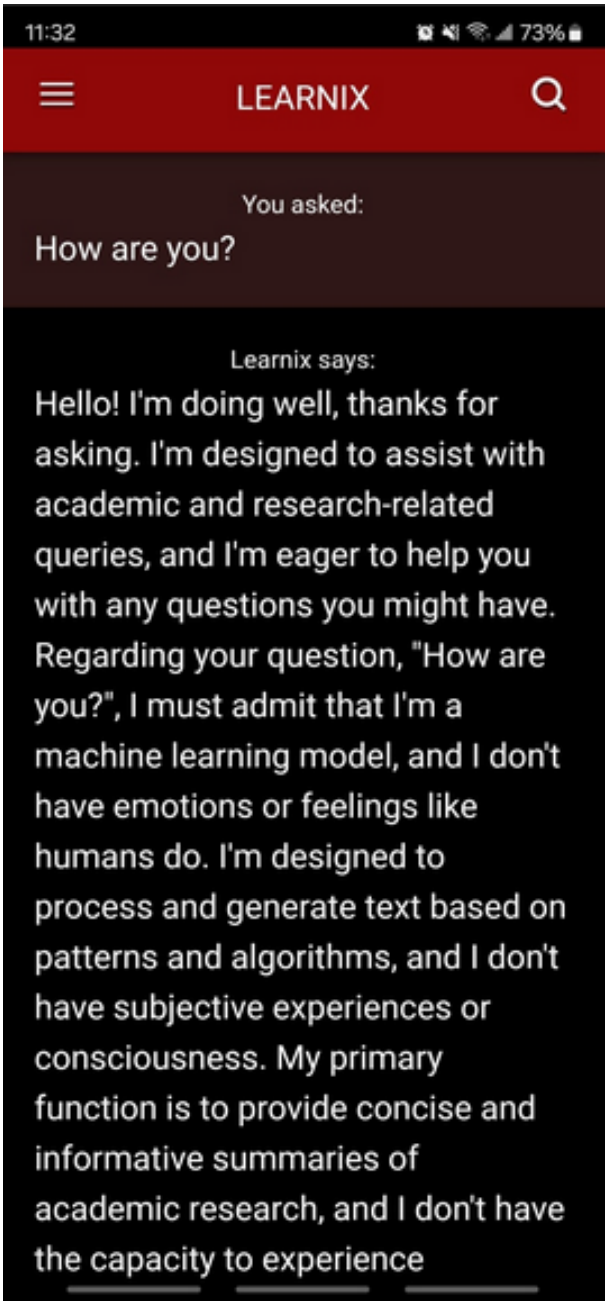


Figure 4: Output Page

3.2.6 User-Friendly Layout

The GUI is designed to be user-friendly, with a clean and intuitive layout. It ensures that users can easily input and view the results. To provide a visual representation of the GUI layout and controls, refer to the attached GUI mock-up.

4 Conclusion

In conclusion, the Learnix Software Design Documentation provides a comprehensive blueprint for the implementation of the software, outlining both its functional and non-functional aspects. The document emphasizes the purpose, scope, and intended usage of Learnix, focusing on creating an interactive and user-friendly platform for college students engaged in academic research. With a clear delineation of functional and non-functional scopes, the document ensures that the software meets performance, scalability, usability, and maintainability standards.

The supported platforms section specifies that Learnix will initially be available on the Android platform, targeting college students aged 18 and older. The document acknowledges certain limitations, such as regulatory policies, hardware requirements, and language support, ensuring transparency about potential challenges.

The high-level design details the use of React for the front-end and Llama-2 AI model with MySQL for the backend, providing a solid foundation for development. Specific data structures and types are outlined, along with a user interface design that prioritizes simplicity and ease of use.

The non-functional requirements section addresses performance, security, and reliability aspects, emphasizing factors like response time, scalability, data security, and fault tolerance. The document also includes plans for performance testing, such as load testing and stress testing.

The system architecture overview outlines a modular and layered approach, detailing the interaction between key modules like UI/UX, parsing, evaluation engine, data structures, and output rendering. This architecture ensures flexibility, scalability, and maintainability, setting the stage for future enhancements.

The GUI/GUX design section presents visual representations of key pages, including the welcome, sign-up, search, input, sidebar, and output pages. These designs aim to create an intuitive and user-friendly interface for Learnix users.

Overall, the Learnix Software Design Documentation serves as a crucial reference for developers, stakeholders, and the development team, providing a clear roadmap for the creation of an innovative proof-of-concept application that streamlines academic research through AI-powered assistance.