ALEP: Advanced Loan Eligibility Predictor

# System Design Document

Version 2.0

2024-05-09

Table of Contents

[1. Introduction 1](#_Toc160112726)

[**1.1** Purpose of the SDD 1](#_Toc160112727)

[2. General Overview and Design Guidelines/Approach 2](#_Toc160112728)

[**2.1** General Overview 2](#_Toc160112729)

[**2.2** Assumptions/Constraints/Risks 2](#_Toc160112730)

[2.2.1 Assumptions 2](#_Toc160112731)

[2.2.2 Constraints 3](#_Toc160112732)

[2.2.3 Risks 3](#_Toc160112733)

[**2.3** Alignment with an Enterprise Architecture framework 3](#_Toc160112734)

[3. Design Considerations 5](#_Toc160112735)

[**3.1** Goals and Guidelines 5](#_Toc160112736)

[**3.2** Development Methods & Contingencies 5](#_Toc160112737)

[**3.3** Architectural Strategies 5](#_Toc160112738)

[**3.4** Performance Engineering 6](#_Toc160112739)

[4. System Architecture and Architecture Design 8](#_Toc160112740)

[**4.1** Logical View 8](#_Toc160112741)

[**4.2** Hardware Architecture 8](#_Toc160112742)

[4.2.1 Security Hardware Architecture 8](#_Toc160112743)

[4.2.2 Performance Hardware Architecture 8](#_Toc160112744)

[**4.3** Software Architecture 8](#_Toc160112745)

[4.3.1 Security Software Architecture 9](#_Toc160112746)

[4.3.2 Performance Software Architecture 9](#_Toc160112747)

[**4.4** Information Architecture 9](#_Toc160112748)

[4.4.1 Records Management 9](#_Toc160112749)

[**4.5** Internal Communications Architecture 10](#_Toc160112750)

[**4.6** Security Architecture 10](#_Toc160112751)

[**4.7** Performance 10](#_Toc160112752)

[**4.8** System Architecture Diagram 11](#_Toc160112753)

[5. System Design 12](#_Toc160112754)

[**5.1** Business Requirements 12](#_Toc160112755)

[**5.2** Database Design 12](#_Toc160112756)

[5.2.1 Data Objects and Resultant Data Structures 12](#_Toc160112757)

[5.2.2 File and Database Structures 12](#_Toc160112758)

[**5.3** Data Conversion 14](#_Toc160112759)

[**5.4** User Machine-Readable Interface 14](#_Toc160112760)

[5.4.1 Inputs 14](#_Toc160112761)

[5.4.2 Outputs 15](#_Toc160112762)

[**5.5** User Interface Design 16](#_Toc160112763)

[5.5.1 Section 508 Compliance 16](#_Toc160112764)

[6. Operational Scenarios 17](#_Toc160112765)

[7. Detailed Design 19](#_Toc160112766)

[**7.1** Hardware Detailed Design 19](#_Toc160112767)

[**7.2** Software Detailed Design 19](#_Toc160112768)

[**7.3** Security Detailed Design 20](#_Toc160112769)

[**7.4** Performance Detailed Design 20](#_Toc160112770)

[**7.5** Internal Communications Detailed Design 20](#_Toc160112771)

[8. System Integrity Controls 21](#_Toc160112772)

[9. External Interfaces 22](#_Toc160112773)

[**9.1** Interface Architecture 22](#_Toc160112774)

[**9.2** Interface Detailed Design 22](#_Toc160112775)

[Appendix A: Record of Changes 23](#_Toc160112776)

[Appendix B: Acronyms 24](#_Toc160112777)

[Appendix C: Glossary 25](#_Toc160112778)

[Appendix D: Referenced Documents 26](#_Toc160112779)

[Appendix E: Approvals 27](#_Toc160112780)

[Appendix F: Additional Appendices 28](#_Toc160112781)

[Appendix G: Database Design Document – Sample Template 30](#_Toc160112782)

[5.2. Functional Design Decisions 32](#_Toc160112783)

[**9.3** 6.1. Data Software Objects and Resultant Data Structures 32](#_Toc160112784)

[**9.4** 6.2. Database Management System Files 33](#_Toc160112785)

[**9.5** 7.1. Roles and Responsibilities 33](#_Toc160112786)

[**9.6** 7.2. System Information 34](#_Toc160112787)

[**9.7** 7.2.1. Database Management System Configuration 34](#_Toc160112788)

[7.2.2. Database Support Software 35](#_Toc160112789)

[7.2.3. Security and Privacy 35](#_Toc160112790)

[7.3. Performance Monitoring and Database Efficiency 35](#_Toc160112791)

[7.3.1. Operational Implications 35](#_Toc160112792)

[7.3.2. Data Transfer Requirements 36](#_Toc160112793)

[7.3.3. Data Formats 36](#_Toc160112794)

[**9.8** 7.4. Backup and Recovery 36](#_Toc160112795)

[Appendix H: Interface Control Document 37](#_Toc160112796)

[**1.1** Interface Overview 37](#_Toc160112797)

[1.1.1 Interface Controls 37](#_Toc160112798)

[**1.2** Functional Allocation 38](#_Toc160112799)

[**1.3** Data Transfer 38](#_Toc160112800)

[**1.4** Transactions 38](#_Toc160112801)

[**1.5** Security and Integrity 39](#_Toc160112802)

[**2.** Detailed Interface Requirements 40](#_Toc160112803)

[**2.1** Requirements for Interface with an External System 40](#_Toc160112804)

[2.1.1 Assumptions 40](#_Toc160112805)

[2.1.2 Technical Interface Requirements 40](#_Toc160112806)

[2.1.3 General Processing Steps 40](#_Toc160112807)

[2.1.4 Interface Processing Time Requirements 40](#_Toc160112808)

[2.1.5 Message Format (or Record Layout) and Required Protocols 40](#_Toc160112809)

[2.1.6 Communication Methods 41](#_Toc160112810)

[2.1.7 Security Requirements 41](#_Toc160112811)

[**3.** Quality Assurance 41](#_Toc160112812)

List of Figures

[Figure 1 – Software Architecture Diagram 8](#_Toc166203899)

[Figure 2 – System Architecture Diagram 11](#_Toc166203900)

[Figure 3 – Database Relational Schema 13](#_Toc166203901)

[Figure 4 – UI Interface Diagram 16](#_Toc166203902)

List of Tables

[Table 1 - Record of Changes 23](#_Toc160119796)

[Table 2 - Acronyms 24](#_Toc160119797)

[Table 3 - Glossary 25](#_Toc160119798)

[Table 4 - Referenced Documents 26](#_Toc160119799)

[Table 5 - Approvals 27](#_Toc160119800)

[Table 6 - OSI Application Layer 37](#_Toc160119801)

[Table 7 - OSI Presentation Layer 37](#_Toc160119802)

## Introduction

This System Design Document (SDD, applies to the development of the Automated Loan Eligibility Prediction (ALEP) software. This SDD describes the technical specifications and architectural blueprint necessary for the implementation of the ALEP system. The SDD is expected to change throughout the development of ALEP, reflecting any changes or refinements in the technical design specifications as the project progresses. As ALEP will be handling financial information, sufficient security and privacy measures will need to be considered. Measures will be implemented to safeguard the sensitive applicant information and ensure compliance with regulatory standards.

### Purpose of the SDD

The purpose of the SDD is to document the necessary information required to define the architecture and system design. It is to serve as a guide for the development team, providing guidance on the technical infrastructure and design considerations essential for the development of the ALEP system, so that they may understand the goals, functionality, and constraints.

## General Overview and Design Guidelines/Approach

This section describes the principles and strategies to be used as guidelines when designing and implementing the system.

### General Overview

This software is intended for financial institutions, including banks and credit unions, to automate the process of evaluating loan eligibility for applicants. End-users include loan officers, financial analysts, and other staff involved in the loan approval process. The ALEP software aims to streamline the loan approval process by analyzing applicant data that was submitted by the application form and determining the applicant’s eligibility for the loan. The scope of the ALEP software includes analyzing applicant data from the loan application form to assess the applicant’s eligibility for a loan. The main goals are to automate the loan approval process, reduce manual errors, and enhance the overall efficiency of the lending process.

### Assumptions/Constraints/Risks

#### Assumptions

ALEP is designed to operate primarily on Windows/macOS environments, ensuring flexibility across different server setups. The frontend is built using React.js and is optimized for modern web browsers including Google Chrome, Firefox, Safari, and Microsoft Edge, facilitating access from a variety of devices. Users are assumed to have basic computer literacy to navigate the web interface efficiently.

The system utilizes SQLite for database management, chosen for its lightweight nature and suitability for smaller, less complex applications. This setup assumes the availability of continuous internet connectivity to access all functionalities, including loan application processes and status updates.

Dependencies include several Python libraries such as Flask for the backend, React for the frontend, and Joblib for model management, which must be maintained with current versions to ensure ongoing stability and security. Looking ahead, potential scalability needs may require transitioning to a more robust database system like PostgreSQL or MySQL if user demand increases significantly. Additionally, future enhancements might incorporate more sophisticated machine learning models, integration with external financial APIs for real-time data verification, and advanced user profile management features.

#### Constraints

For the ALEP system, the design constraints primarily revolve around the software environment and end-user considerations. The system is developed using React.js for the frontend and Flask for the backend, which necessitates environments capable of running Node.js and Python. This software setup is essential for maintaining the system's operational integrity and compatibility.

The ALEP system operates within a web-based environment, accessible via standard web browsers, which assumes users have access to an internet connection and devices that support modern web standards. The use of SQLite for the database ensures lightweight data management suitable for the scale of operations envisioned in ALEP, avoiding the need for more complex database systems and minimizing hardware demands.

Security is a critical constraint, with the system designed to adhere to standard web security protocols, including HTTPS. Data handling within ALEP complies with general data protection regulations to safeguard user information, necessitating robust security measures in both application development and deployment phases.

Performance constraints are addressed through efficient coding practices and lightweight technology choices, ensuring the system remains responsive and performs adequately under typical usage scenarios. This focus on optimized performance and security reflects the primary constraints influencing ALEP's system design.

#### Risks

In the design of the ALEP system, there are risks that could potentially impact the successful development and deployment of the system. It is necessary to identify these risks and implement appropriate mitigation strategies to address them.

* Data Privacy and Security Risks:
  + Risk: The ALEP system deals with sensitive financial data, including income, credit history, and employment status, which poses a risk of data breaches or unauthorized access.
  + Mitigation: Implement encryption techniques, access controls, and data anonymization to protect sensitive information. Conduct regular security audits and compliance assessments to ensure adherence to data privacy regulations.
* Model Accuracy Risks:
  + Risk: The accuracy of loan eligibility predictions generated by the ML model may be affected by biases in the training data or changes to lending criteria over time.
  + Mitigation: Continuously monitor and evaluate the performance of the ML model using historical data and feedback from end-users. Implement bias detection and mitigation techniques, such as dataset preprocessing, threshold tuning and fairness constraints, to minimize inaccuracies in predictions.

## Design Considerations

### Goals and Guidelines

Developers are expected to adhere to the PEP8 and Airbnb style guides for Python and JavaScript coding, respectively, to ensure consistency across the codebase. Version control is managed through Git, employing a branching model that clearly delineates development, features, and production branches for effective code management and integration. Additionally, secure coding practices are emphasized, especially in handling user data and authentication, with all API endpoints secured using JWT (JSON Web Tokens) to prevent CSRF (Cross-Site Request Forgery) attacks.

A modular design approach is adopted, with the backend and frontend developed as separate modules linked via RESTful APIs, facilitating easier updates and maintenance. Systematic error handling is also integral, aimed at managing and logging errors gracefully to maintain service availability and reliability, while providing user-friendly error messages to guide users towards easy resolution.

The server’s RESTful APIs adhere to industry standards, with endpoints that reflect the resources they manage (e.g., /api/users, /api/loans) and utilize appropriate HTTP verbs (GET, POST, PUT, DELETE) to indicate actions. The user interface prioritizes intuitiveness and responsiveness, adopting a mobile-first design philosophy. All user interactions, particularly form submissions and inputs, are designed to provide immediate feedback to correct errors or invalid entries, thereby enhancing the overall user experience.

### Development Methods & Contingencies

* **Iterative Development Approach:** ALEP is built using an Agile framework that emphasizes iterative development and continuous feedback. This approach allows for frequent refinements and adjustments in response to user feedback and evolving business requirements.
* **Architectural Adaptability:** ALEP's architecture is designed to be flexible, facilitating easy integration of new features and adaptability to changing technological landscapes.

### Architectural Strategies

In the development of ALEP, several strategic architectural decisions were made to optimize both the system's structure and functionality. These decisions directly reflect our design goals of efficiency, scalability, and robust integration of machine learning technologies. Here are the refined strategies:

* **Programming Language and Framework:** We selected Python due to its strong reputation in both web development and machine learning communities. Python's extensive libraries and frameworks expedite development and facilitate complex computations. For the web application's backend, we chose Flask because of its simplicity and flexibility, which allow for quick deployment and easy integration with Python's data science stacks.
* **Machine Learning Model Integration:** A pivotal decision in ALEP’s architecture was the integration of a Random Forest classifier, chosen for its effectiveness in handling binary classification tasks like loan eligibility prediction. This model was trained on a dataset featuring a range of attributes from credit history to income levels, which are critical in predicting loan approval outcomes. The model training involved using Scikit-learn, a machine learning library in Python that provides tools for model training, validation, and testing. This approach not only simplifies the model deployment process but also enhances the system’s ability to evolve as the dataset grows.
* **Model Training and Validation:** The Random Forest model was trained with a 80-20 train-test split, ensuring that the model was evaluated on unseen data to gauge its real-world applicability accurately. Additionally, the use of techniques like cross-validation helped in mitigating overfitting and optimizing the model’s parameters.
* **Web Application Framework:** The choice of Flask was dictated by the need for a lightweight, flexible solution that could seamlessly integrate with Python's powerful machine learning libraries. Flask's ability to serve as both a web server and a backend application framework allowed us to streamline development, focusing on features and integration rather than underlying complexities.
* **Frontend Technology:** For the frontend, React.js was employed to leverage its component-based architecture, which aligns well with the dynamic nature of loan application processes. React's state management capabilities make it ideal for handling real-time data updates and user interactions in a responsive and efficient manner.
* **Data Management:** SQLite was chosen for data storage due to its lightweight nature and compatibility with Flask. This choice supports our goal of maintaining a simple, deployable application without the overhead of more complex database systems, which are unnecessary given the current scale of data.
* **User Interface Design:** Emphasis was placed on creating a responsive and accessible user interface using HTML, CSS, and React.js. The design principles focused on simplicity and user engagement, ensuring that the application is intuitive for users with varying levels of technical proficiency.

### Performance Engineering

The ALEP system's design incorporates specific performance requirements to ensure scalability and responsiveness, crucial for efficiently processing loan applications. Here are the strategies and methods implemented directly within ALEP:

* **Scalability:** The architecture of ALEP is designed to efficiently handle an increasing workload without degradation in performance. This is primarily achieved through its backend service setup using Flask, which can be scaled up with additional server instances as needed. The use of a lightweight framework helps in maintaining performance even as user demand grows.
* **Response Time:** To deliver swift responses to loan eligibility queries, ALEP minimizes processing delays through optimized backend logic. The Random Forest algorithm, utilized for decision making, is tuned for quick execution, which significantly enhances the response time for end users.
* **Resource Optimization:** ALEP is optimized for resource use to ensure that the system utilizes server capabilities efficiently. This includes careful management of memory and efficient database queries, which help in maintaining high throughput levels and low latency in processing loan applications.
* **Load Testing:** Prior to deployment, ALEP was subjected to load testing to ensure it could handle expected traffic volumes and simultaneous operations without performance drops. This testing was crucial in validating the system's capacity to manage real-world operational stresses.
* **Performance Monitoring:** ALEP includes basic monitoring setups that track system performance metrics such as response times and system health. This monitoring is essential for ongoing performance management and ensures that the system continues to meet performance standards post-deployment.

These strategies are integral to maintaining the high performance of the ALEP system, ensuring that it remains responsive and efficient under varied operational conditions. By focusing on these key areas, ALEP is able to provide reliable and fast processing of loan applications, meeting the needs of its users effectively.

## System Architecture and Architecture Design

### Logical View

The ALEP system is structured around a client-server architecture model, emphasizing simplicity and functionality. Below is an overview of the logical structure of ALEP:

* **Frontend (Client-Side):** The user interface of ALEP is developed using React.js, providing a responsive and interactive experience. It communicates with the backend via HTTP requests, handling user interactions such as signing in, submitting loan applications, and viewing the status of previous applications.
* **Backend (Server-Side):** The server side of ALEP is implemented using the Flask web framework. It manages the core functionalities including user authentication, request handling, data processing, and interfacing with the SQLite database for persistent data storage.
* **Database:** ALEP utilizes an SQLite database to store user data, loan application details, and decision results. This relational database setup facilitates efficient data retrieval and storage, supporting the application's need for quick data processing and decision-making.
* **Machine Learning Model:** The decision-making core of ALEP includes a pre-trained Random Forest model, responsible for predicting loan eligibility. This model operates within the Flask application, processing input data received from the frontend and storing results back to the database.

The system's architecture is designed to ensure that each component functions both independently and in concert with others, optimizing maintainability and scalability. All system components are tightly integrated, providing a seamless flow of data across the user interface, server logic, and database systems. This logical view is designed to be an integral part of the overall system documentation and is stored within the project’s repository under the documentation section for easy access and reference.

### Hardware Architecture

Not Applicable.

#### Security Hardware Architecture

Not Applicable.

#### Performance Hardware Architecture

Not Applicable.

### Software Architecture

The software architecture for the ALEP system comprises various components distributed across different layers to support its functionality. Below is an overview of the software components needed to support the system:

A diagram of a server

Description automatically generated

Figure 1 – Software Architecture Diagram

#### Security Software Architecture

**Authentication:**

ALEP uses Flask to handle user authentication, which verifies user credentials before granting system access. User passwords are securely hashed using bcrypt, ensuring that password data stored in the database is protected against unauthorized access.

**Authorization:**

Post-authentication, ALEP ensures that only authenticated users can access specific functionalities such as submitting and querying loan applications. The system access is managed based on user session status.

#### Performance Software Architecture

ALEP is designed primarily for demonstration or small-scale deployment, using SQLite and Flask, which are suitable for moderate loads. Currently, there are no specific mechanisms implemented for high availability, redundancy, or handling high traffic volumes, which are typically necessary for larger-scale deployments. The current setup does not include load balancing or database clustering.

### Information Architecture

ALEP system stores various types of data critical for processing loan applications and managing user accounts. The data includes:

* **User Information**: Includes personally identifiable information (PII) such as first and last names, usernames, and email addresses. Passwords are securely hashed and stored.
* **Loan Application Data**: Consists of non-PII data such as loan amount, applicant income, coapplicant income, and other loan-related details necessary for determining loan eligibility.
* **Loan Decision Data**: Stores outcomes of loan applications including decision status and reasons for rejection, if applicable.

#### Records Management

The data handled by ALEP is subject to the General Data Protection Regulation (GDPR), ensuring that personal data is processed under strict conditions and for a legitimate purpose.

##### Data

Data supplied to ALEP includes:

* **User Input**: Users manually input their personal and loan-related information through the web interface.
* **Automated Input**: Loan decisions are processed automatically based on the model's predictions and stored electronically.

##### Manual/Electronic Inputs

Once entered into ALEP, manual and electronic inputs are:

* **Validated**: Checks are performed to ensure data integrity and correctness.
* **Processed**: Data is used for loan eligibility processing and user management.
* **Stored**: All validated data is stored in the SQLite database.

##### Master Files

The SQLite database maintains:

* **Users Table**: Stores user credentials and personal information.
* **Loan Applications Table**: Records details of each loan application.
* **Loan Decisions Table**: Logs the outcome of loan applications along with decision reasons.

### Internal Communications Architecture

The internal communications architecture of ALEP defines the network topology and communication protocols used for inter-component communication. The system relies on standard network protocols such as TCP/IP for data transmission between application servers, databases, and external interfaces. Secure communication channels are established using encryption and digital certificates to protect data in transit.

### Security Architecture

The security architecture of ALEP encompasses a comprehensive set of security controls and measures to protect the confidentiality, integrity, and availability of system resources and data. Access controls, encryption, intrusion detection, and logging mechanisms are implemented to mitigate security risks and vulnerabilities. Regular security assessments and audits are conducted to ensure compliance with industry standards and regulatory requirements.

### Performance

The performance of ALEP is optimized through efficient resource utilization, scalability, and responsiveness. Performance monitoring tools and metrics are employed to track system performance and identify areas for optimization. Load testing and capacity planning exercises are conducted to assess system scalability and determine hardware and software requirements for handling expected workloads.

### System Architecture Diagram

A detailed system architecture diagram illustrating the interconnected components of ALEP is provided. The diagram, referenced in Appendix D, showcases the interaction between software modules and communication components, emphasizing the flow of data and control within the system.

A diagram of a system

Description automatically generated

Figure 2 – System Architecture Diagram

## System Design

### Business Requirements

Business requirements for the ALEP system focus primarily on efficiently processing and managing loan applications with a high degree of accuracy and security. The main business goals include:

* Providing quick loan eligibility decisions.
* Ensuring data privacy and security.
* Maintaining scalability to handle varying loads of user requests.

### Database Design

ALEP utilizes a SQLite database due to its lightweight nature and ease of integration with Flask applications. Below is an overview of the database design:

**Users Table**

* + **id** (Integer): Primary key.
  + **firstname** (String): User's first name.
  + **lastname** (String): User's last name.
  + **username** (String): Unique identifier for the user.
  + **email** (String): User's email address.
  + **password\_hash** (String): Hashed password for security.

**Loan Applications Table**

* + **id** (Integer): Primary key.
  + **user\_id** (Integer): Foreign key linked to Users table.
  + **gender**, **married**, **dependents**, **education**, **self\_employed** (String): Various demographic and personal data.
  + **applicant\_income**, **coapplicant\_income**, **loan\_amount** (Float): Financial data.
  + **loan\_term** (Integer): Duration of the loan.
  + **credit\_history** (Integer): Credit history status.
  + **property\_area** (String): Urban, Rural, or Semiurban.
  + **purpose** (String): Purpose of the loan.

**Loan Decisions Table**

* + **id** (Integer): Primary key.
  + **application\_id** (Integer): Foreign key linked to Loan Applications table.
  + **answer** (String): Decision on the loan application.
  + **reason** (String): Reason for the loan decision.
  + **decision\_date** (DateTime): Date and time of the decision.

**CRUD Capabilities**

Each table supports full CRUD (Create, Read, Update, Delete) capabilities through Flask and SQLAlchemy, with specific validation rules ensuring data integrity and alignment with business rules.

#### Data Objects and Resultant Data Structures

Data objects are managed through SQLAlchemy ORM models, which encapsulate the database structure in Python classes. These models facilitate data handling between the SQLite database and the Flask application, ensuring structured data flow and integrity.

#### File and Database Structures

The data is stored in a SQLite file located within the project directory. This setup supports the system architecture by providing a self-contained database environment that is easy to maintain and deploy. The database schema, managed through SQLAlchemy, reflects the logical data model with relationships between tables like Users, Loan Applications, and Loan Decisions defined explicitly.

The simplicity of SQLite meets ALEP's requirements without the overhead of more complex database management systems, aligning with the system's need for a lightweight, efficient data management solution.

##### Database Management System Files

A diagram of a loan application

Description automatically generated

Figure 3 – Database Relational Schema

The detailed design of the DBMS files, including the physical description of schemas, records, and storage configurations, can be found in the Database Design Document (refer to Appendix G: Database Design Document - Detailed Database Design).

##### Non-Database Management System Files

Not Applicable.

### Data Conversion

Data conversion processes within the ALEP system are limited to user input transformations necessary for the operation of the machine learning model. All conversions, such as the squaring of income values and normalization performed during preprocessing, are defined within the system's data preprocessing notebook.

### User Machine-Readable Interface

**User Classes/Roles:**

1. **Operational Users:** These users are primarily responsible for interacting with the system on a day-to-day basis, submitting loan applications, and accessing decisions.
2. **System Operators:** System operators oversee the overall functioning of the ALEP system, monitor system health, and manage user accounts.
3. **Administrators:** Administrators have elevated privileges to configure system settings, manage security permissions, and generate reports. We expect 2 administrators who may access the system concurrently during administrative tasks.

#### Inputs

The primary method for user input is through web-based forms on the React.js frontend, where users submit loan application details. Each input form includes fields like income, loan amount, education level, and employment status, directly mapped to the database fields in the SQLite database. Data validations are enforced both client-side in the React application and server-side in the Flask application to ensure data integrity and security.

**Data Entry Controls**:

* Mandatory fields are marked and validated for completeness.
* Input values for fields like income and loan amount are checked for numerical constraints.
* Dropdowns for fields like education and employment status prevent invalid entries.

**Security and Accessibility**:

* Access to the loan application interface requires user authentication.
* Session management is handled securely to prevent unauthorized data access.
* The interface complies with basic accessibility standards to ensure it is usable by a broad audience.

#### Outputs

Outputs from the ALEP system include:

* **Loan Decision Reports**: Displayed immediately upon processing a user's loan application. These reports show the decision ("Approved" or "Denied") and, in cases of denial, reasons for the decision.
* **User Dashboard**: Shows past loan applications and their statuses. Users can query past applications and see detailed views of each application.

**Report Details**:

* Each report is dynamically generated based on the user's loan application.
* Outputs are secured to ensure only authenticated users can access their information.

### User Interface Design

**A screenshot of a computer screen

Description automatically generated**

Figure 4 – UI Interface Diagram

#### Section 508 Compliance

* **Keyboard Navigation:** All functionalities within the user interface are accessible via keyboard shortcuts to accommodate users with mobility impairments.

## Operational Scenarios

* **SCENARIO 1: Initial Login**
  + Description: The loan officer accesses the ALEP system for the first time and needs to set up their account and credentials.
  + Steps:
    1. The loan officer receives initial access credentials (username and temporary password) from the system administrator.
    2. The loan officer opens the ALEP system login page on their computer.
    3. They enter the provided username and password into the login fields.
    4. The ALEP system prompts the officer to create a new password for their account to enhance security.
    5. The loan officer enters a new password and confirms it to proceed with the account setup.
    6. The ALEP system verifies the password complexity and accepts the new password.
    7. The ALEP system confirms successful account setup and redirects the loan officer to the login page.
    8. The loan officer will then log into the system with their credentials and be sent to the home page.
    9. The loan officer explores the system interface and available features to familiarize themselves with the ALEP system’s functionality.
* **SCENARIO 2: Loan Application Submission**
  + Description: The loan officer receives a loan application from a customer and needs to assess the customer’s eligibility for a loan using the ALEP system.
  + Steps:
    1. The loan officer logs into the ALEP system using their credentials.
    2. They access the loan application form within the system.
    3. They enter the customer’s financial information into the system’s application form.
    4. They submit the application form to the system.
    5. The ALEP system processes the customer’s financial data using the ML prediction model.
    6. The system generates a loan eligibility prediction based on the customer’s financial information.
    7. The loan officer reviews the eligibility prediction and decides whether to approve or deny the loan application.
    8. The loan officer communicates the decision to the customer.
* **SCENARIO 3: Reporting Loan Approval Statistics**
  + Description: The loan officer needs to generate reports on loan approval statistics and trends using the ALEP system.
  + Steps:
    1. The loan officer logs into the ALEP system with their credentials and is taken to the home page.
    2. The loan officer accesses the reporting dashboard within the ALEP system.
    3. The loan officer selects the desired parameters for generating loan approval statistics.
    4. The ALEP system retrieves relevant data from the database.
    5. The ALEP system processes the data to generate reports on loan approval statistics and trends.
    6. The loan officer reviews the generated reports and analyzes loan approval trends over time.
    7. The loan officer uses the insights from the reports to make informed decisions about lending practices.

## Detailed Design

### Hardware Detailed Design

Not Applicable.

### Software Detailed Design

The software architecture of ALEP includes several key components detailed below:

* **Backend**: Developed in Python using the Flask framework, handling API requests, user authentication, and interaction with the SQLite database.
* **Frontend**: Developed using React.js, providing a responsive user interface for client interactions.
* **Database**: Utilizes SQLite for storing user data, loan applications, and decision history. The structure is designed to ensure quick queries and data integrity.
* **Machine Learning Model**: Integrated as a component of the backend, executed to predict loan eligibility. The model is serialized using Joblib and loaded into the Flask application at runtime.

### Security Detailed Design

Security within the ALEP system focuses on:

* **Authentication**: Utilizes Flask-Login for managing user sessions and authenticating users before granting access to the application.
* **Authorization**: Ensures users can only access their data and perform actions within their permissions.
* **Encryption**: Uses HTTPS to encrypt data transmitted between the client and server. Passwords are hashed using bcrypt before storage.
* **Logging and Auditing**: Standard logging is implemented for both access and errors, which aids in auditing and debugging.

### Performance Detailed Design

Not Applicable.

### Internal Communications Detailed Design

Internal communications within ALEP are managed through HTTP requests between the frontend and backend. Flask routes handle these interactions efficiently, ensuring data integrity and system responsiveness. The system's architecture supports robust error handling and responsive feedback to user actions, facilitating a seamless user experience.

## System Integrity Controls

* **Internal Security**:
  + **Role-Based Access Control (RBAC)**: ALEP utilizes RBAC to ensure that data access is granted based on user roles. This mechanism restricts access to sensitive data such as loan application details and user personal information, ensuring that only authorized users have access based on their roles.
* **Audit Procedures**:
  + **Logging**: ALEP has a logging system that captures user activities, system errors, and administrative actions. This helps in creating an audit trail for monitoring and security purposes.
  + **Data Retention**: The system maintains logs and historical data for an adequate period to satisfy audit requirements and operational reviews.
* **Application Audit Trails**:
  + **Activity Logging**: ALEP automatically logs access to sensitive data, including user interactions with the loan application system. Each access event records the user ID, timestamp, and details of the accessed data.
* **Verification Processes**:
  + **Data Validation**: When users input or update data, ALEP validates this against predefined rules and formats to ensure data integrity and correctness.
  + **Updates and Modifications**: Any addition, deletion, or modification of critical data within the system undergoes a verification process to ensure accuracy and prevent unauthorized changes.
* **Audit Information Traceability**:
  + **Detailed Logging**: The system's logs include detailed information such as user ID, the terminal or device used for access, the exact timestamp, and specifics of the data accessed or altered. This granularity supports effective audit trails and accountability.

## External Interfaces

Not Applicable.

### Interface Architecture

Not Applicable.

### Interface Detailed Design

Not Applicable.

Appendix A: Record of Changes

Table 1 - Record of Changes

| Version Number | Date | Author/Owner | Description of Change |
| --- | --- | --- | --- |
| 1.0 | 02/28/2024 | Awwal Ahmed  Sarah Freidel | Initial Document |
| 2.0 | 05/09/2024 | Awwal Ahmed  Sarah Freidel | Updated Documentation |

Appendix B: Acronyms

Table 2 - Acronyms

| Acronym | Literal Translation |
| --- | --- |
| ALEP | Automated Loan Eligibility Predictor |
| GLBA | Gram-Leach-Bliley Act |
| GUI | Graphical User Interface |
| LDM | Logical Data Model |
| ML | Machine Learning |
| PII | Personally Identifiable Information |
| RBAC | Role-Based Access Control |
| SDD | Software Design Document |
| TCP/IP | Transmission Control Protocol/ Internet Protocol |
| DBA | Database Administrator |
| DBMS | Database Management System |
| I/O | Input/Output |

Appendix C: Glossary

Table 3 - Glossary

| Term | Acronym | Definition |
| --- | --- | --- |
| Gramm-Leach-Bliley Act | GLBA | This is a federal law that protects consumer financial privacy and requires financial institutions to explain their information-sharing practices to their customers and to safeguard sensitive data. |
| Role-Based Access Control | RBAC | An access control method that assigns permissions to end-users based on their role within the organization. |

Appendix D: Referenced Documents

Table 4 - Referenced Documents

| Document Name | Document Location and/or URL | Issuance Date |
| --- | --- | --- |
| Software Architecture Diagram | *https://imgur.com/a/FHhVwbq* | 05/09/2024 |
| System Architecture Diagram | *https://imgur.com/a/9rv9B0h* | 05/09/2024 |
| Database Relational Schema | https://imgur.com/a/EQoZRUB | 05/09/2024 |
| UI Design Diagram | https://imgur.com/a/TXF1uan | 02/27/2024 |

Appendix E: Approvals

The undersigned acknowledge that they have reviewed the SDD and agree with the information presented within this document. Changes to this SDD will be coordinated with, and approved by, the undersigned, or their designated representatives.

Table 5 - Approvals

| Document Approved By | Date Approved |
| --- | --- |
| A black and white logo  Description automatically generatedName: Awwal Ahmed, ML Engineer – Loan Logic Solutions Inc. | Date |
| Name: Sarah Freidel, Software Engineer – Loan Logic Solutions Inc. | Date |

Appendix F: Additional Appendices

# DATA DICTIONARY

**1- PROCESSES**

* + createUser:
    - Description: Process to create a new user record in the Users table.
  + retrieveUser:
    - Description: Process to retrieve user information based on userID.
  + updateUser:
    - Description: Process to update user information based on userID.
  + deleteUser:
    - Description: Process to delete a user record based on userID.
  + createLoanApplication:
    - Description: Process to create a new loan application record in the LoanApplications table.
  + retrieveLoanApplication:
    - Description: Process to retrieve loan application information based on applicationID.
  + updateLoanApplication:
    - Description: Process to update loan application information based on applicationID.
  + deleteLoanApplication:
    - Description: Process to delete a loan application record based on applicationID.
  + createLoanDecision:
    - Description: Process to create a new loan decision record in the LoanDecisions table.
  + retrieveLoanDecision:
    - Description: Process to retrieve loan decision information based on decisionID.
  + updateLoanDecision:
    - Description: Process to update loan decision information based on decisionID.
  + deleteLoanDecision:
    - Description: Process to delete a loan decision record based on decisionID.

**2- DATA FLOWS**

* User Data Flow:
  + Description: Flow of user-related data between the application and the Users table in the database.
* Loan Application Data Flow:
  + Description: Flow of loan application data between the application and the LoanApplications table in the database.
* Loan Decision Data Flow:
  + Description: Flow of loan decision data between the application and the LoanDecisions table in the database.

**3- DATA ELEMENTS**

* User Table:
  + userID (Primary key): Unique identifier for each user.
  + username: Name of the user.
  + email: Email address of the user.
  + password: Password for user authentication.
* LoanApplications Table:
  + applicationID (Primary key): Unique identifier for each loan application.
  + userID (Foreign key): Identifies the user who submitted the loan application.
  + loan\_amount: Amount of the loan requested.
  + credit\_score: Credit score of the applicant.
  + employment\_status: Status of employment of the applicant.
* LoanDecisions Table:
  + decisionID (Primary key): Unique identifier for each decision.
  + applicationID (Foreign key): Identifies the loan application associated with the decision.
  + answer: Decision on the loan application (e.g., approved, rejected).
  + reason: Explanation or reason for the decision.
  + decision\_data: Additional data related to the decision, if applicable.

**4- DATA STORES**

* Users Table:
  + Description: Stores information about users registered in the system.
* LoanApplications Table:
  + Description: Stores information about loan applications submitted by users.
* LoanDecisions Table:
  + Description: Stores information about decisions made on loan applications.

Appendix G: Database Design Document

# ASSUMPTIONS/CONSTRAINTS/RISKS

4.1. Assumptions

* **SQLite Usage**: The system assumes the use of SQLite as the database management system due to its lightweight nature and adequacy for small to medium-sized applications. This choice affects the database's scalability and concurrent access capabilities.
* **Operating System Independence**: The database design assumes no specific operating system dependencies. SQLite's cross-platform nature ensures that the ALEP system can operate on any major operating system without modification to the database operations.
* **Data Integrity**: It is assumed that data integrity checks, such as foreign key constraints and unique constraints, are handled within the application logic. SQLite supports these integrity checks, but they must be explicitly enabled and managed through the application code to ensure data consistency.
* **End-User Characteristics**: The database design assumes that end-users possess a basic understanding of web application interfaces but limited understanding of the underlying database structure. Thus, the user interface is designed to be intuitive and shields the user from the complexities of the database interactions.

4.2. Constraints

* The database design must adhere to regulatory compliance standards regarding the handling of personal and financial data.
* Constraints regarding data privacy and security necessitate the implementation of encryption mechanisms for sensitive data stored in the database.
* Constraints related to data consistency and integrity require the implementation of appropriate database constraints, such as foreign key constraints and unique constraints.
* Constraints on system performance and scalability require efficient indexing strategies and query optimization techniques.
* Budget constraints may limit the resources available for database scaling and maintenance.

4.3. Risks

* Risk of data breaches or unauthorized access to sensitive information stored in the database.
* Mitigation: Implement robust authentication mechanisms, role-based access control (RBAC), and encryption for data at rest and in transit.
* Risk of data corruption or loss due to system failures or hardware malfunctions.
* Mitigation: Implement regular backups, redundancy, and failover mechanisms to ensure data durability and availability.
* Risk of performance degradation under high load or concurrent user access.
* Mitigation: Perform thorough performance testing, optimize database queries, and scale resources as needed to handle increased demand.
* Risk of compatibility issues or conflicts with third-party software components or libraries.
* Mitigation: Conduct thorough compatibility testing, maintain up-to-date documentation, and stay informed about software updates and patches.

# DESIGN DECISIONS

5.1. Key Factors Influencing Design

* ALEP's database design is influenced by critical requirements related to performance, availability, security, and privacy.
* Performance: The database design aims to optimize query performance and ensure responsiveness under varying workloads.
* Availability: High availability is ensured through redundant database instances, automated failover mechanisms, and regular backups.
* Security: The database will enforce stringent access controls, encryption of sensitive data, and logging mechanisms to monitor and detect unauthorized access.
* Privacy: Compliance with data privacy regulations such as GDPR will be ensured through anonymization of personal data and restricted access to sensitive information.

## 5.2. Functional Design Decisions

* **Inputs and Outputs:** The database will accept loan applications submitted through the web interface and provide eligibility decisions as outputs.
* **Processing:** Sequential processing is favored for inserts and updates to maintain data consistency, while random access is optimized for queries to retrieve application details and decision outcomes.

5.3. Database Management System Decisions

* **SQLite**, version 3.35, is selected as the initial DBMS for ALEP due to its lightweight architecture, ease of setup and maintenance, and cost-effectiveness. It provides the necessary features for local storage without the overhead of a server-based system.
* The database design incorporates flexibility to adapt to changing requirements through schema migrations, which facilitate modifications and enhancements to the database structure without impacting existing data integrity. This ensures the system remains adaptable to evolving business needs.

5.4. Security and Privacy Design Decisions

In designing the security and privacy features of the ALEP system's database, several key decisions were made to ensure the protection of data and the privacy of users:

* **Authentication**: The system implements robust authentication mechanisms to verify user identity. This involves a combination of username and password verification. The Flask backend handles authentication, ensuring that only registered users can access their respective loan application data.
* **Authorization**: Authorization levels are strictly defined within the ALEP system. Users are only able to access and modify their own loan applications.
* **Data Encryption**: Sensitive data, such as user personal information and loan details, is encrypted using industry-standard encryption protocols. This ensures that data stored in the SQLite database is protected against unauthorized access and breaches.
* **Privacy Controls**: Privacy design adheres to principles of minimal data retention and access limitation. Personal information is only retained as long as necessary for the processing of loan applications and is accessible only to authorized personnel.

5.5. Performance and Maintenance Design Decisions

* **Backup and Restoration:** Regular backups will be scheduled and automated to safeguard against data loss and ensure business continuity. Automated processes for data restoration will minimize downtime in the event of a system failure or data corruption, thereby enhancing data integrity and reliability.
* **Concurrency Control:** To manage concurrent access and minimize contention, ALEP will implement data partitioning strategies. By partitioning data into smaller segments based on defined criteria (e.g., geographical region, loan type), the system can reduce locking conflicts and enhance parallelism, thereby improving throughput and reducing latency.

# DETAILED DATABASE DESIGN

### 6.1. Data Software Objects and Resultant Data Structures

In the ALEP system, data management is centralized around the use of both a relational database management system (RDBMS) and various in-memory data structures that facilitate rapid processing and analysis. Below is a detailed description of the key data software objects and resultant data structures utilized in the ALEP system.

**Database Objects**

**1. Users Table:**

* **Description:** Stores all user-related information.
* **Fields:** id (Integer, Primary Key), firstname (String), lastname (String), username (String, Unique), email (String, Unique), password\_hash (String).
* **Functions:**
  + - 1. - set\_password(password): Hashes the password for secure storage.
      2. **-** Input: password (String)
      3. **-** Output: None (Updates password\_hash)

**2. LoanApplications Table:**

* + **Description:** Contains details of each loan application submitted by users.
  + **Fields:** id (Integer, Primary Key), user\_id (Integer, Foreign Key), gender (String), married (String), dependents (String), education (String), self\_employed (String), applicant\_income (Integer), coapplicant\_income (Integer), loan\_amount (Float), loan\_term (Integer), credit\_history (Integer), property\_area (String), purpose (String).

**3. LoanDecisions Table:**

* + **Description:** Stores the outcome of the loan evaluation process for each application.
  + **Fields:** id (Integer, Primary Key), application\_id (Integer, Foreign Key), answer (String), reason (String), decision\_date (DateTime).

**Major Data Structures**

* **Loan Application Data Structure:**
  + **Usage:** Passed between the front-end where loan applications are captured and the back-end where they are processed.
  + **Structure:** JSON object containing keys corresponding to the fields in the LoanApplications table.
* **Evaluation Result Data Structure:**
  + **Usage:** Used to return the results of the loan eligibility evaluation from the back-end to the front-end.
  + **Structure:** JSON object with fields such as status (Approved or Rejected) and reasons (list of reasons for rejection).

**Non-DBMS Files**

**1. Random Forest Model (**random\_forest\_model.joblib**):**

* **Description:** Serialized file containing the trained Random Forest classifier.
* **Usage:** Loaded into the application at runtime to evaluate new loan applications.
* **File Type:** Permanent.
* **Modules Accessing:** Accessed by the model loading routine in the back-end during application startup and whenever a loan application is evaluated.

**2. Scaler File (**scaler.joblib**):**

* **Description:** Contains the MinMaxScaler used during the preprocessing of training data, which ensures that new data inputs are scaled identically for model evaluation.
* **Usage:** Used both for processing incoming loan application data and during initial model training.
* **File Type:** Permanent.
* **Modules Accessing:** Utilized in the data preprocessing step for new loan applications and during the model training phase.
* Functions:
  + get\_eligibility\_decision: Retrieves eligibility decision based on applicationID.
  + update\_loan\_application\_status: Updates the status of a loan application based on decisionID.

### 6.2. Database Management System Files

Physical Database Design

* The database management system (DBMS) files will be structured according to the SQLite relational database model.
* Objects created to support access methods, such as indexed access for efficient data retrieval and manipulation.
* Distribution, partitioning, or other compartmentalization of the data will be implemented to optimize performance and scalability.
* The estimated DBMS file size and volume of data within the file will be monitored and adjusted as needed to accommodate data growth and optimize performance.
* Definition of the update frequency of the database tables, views, files, areas, records, sets, and data pages will be determined based on transaction volume and system requirements.

These updates ensure that the database functions and design decisions align with the requirements and objectives of the ALEP system.

# DATABASE ADMINISTRATION AND MONITORING

### 7.1. Roles and Responsibilities

* **Database Administrator:** Responsible for overall management and maintenance of the database system, including schema design, performance tuning, backup, and recovery procedures.
* **System Administrator:** Responsible for managing the hardware and software infrastructure supporting the database system, ensuring optimal system performance and availability.
* **Security Administrator:** Responsible for implementing and maintaining security measures to protect the integrity and confidentiality of the database, including access control policies and encryption mechanisms.

### 7.2. System Information

The ALEP system is assumed to run effectively on hardware configurations meeting the following minimum requirements:

* CPU: Intel Core i5 or higher, or equivalent
* RAM: 8GB or higher
* Storage: At least 100GB of available disk space

The system is assumed to be compatible with the following operating systems:

* Windows 10 or higher
* macOS Catalina (version 10.15) or higher
* Linux 20.04 LTS or higher

Since the ALEP system is currently in the development stage, the PostgreSQL database will be stored in a local development environment, primarily on developers' machines. This environment will be utilized for testing, debugging, and local development purposes. This hardware configuration and usage of local development environments deviate from the organization's standard architecture. However, as of 2/28/2024, a waiver has been granted for these deviations.

### 7.2.1. Database Management System Configuration

The ALEP system uses SQLite version 3.35 as its DBMS due to its serverless, zero-configuration setup, which is suitable for environments requiring minimal maintenance. The system is designed to operate effectively on hardware with an Intel Core i5 processor or equivalent, 8GB of RAM, and at least 50GB of SSD storage. This setup ensures optimal performance and scalability to manage the database and application logs efficiently.

## 7.2.2. Database Support Software

Database support software for SQLite includes several essential tools to facilitate database management, maintenance, and performance optimization. Key utilities include the SQLite command line shell for executing SQL commands and managing databases directly. For database backup and restoration, SQLite supports online backup API, allowing hot backups to be performed seamlessly. Performance monitoring can be accomplished using the SQLite Analyzer, which provides detailed analysis of database files to optimize performance. Official SQLite documentation offers comprehensive guidance on utilizing these tools, detailing command syntax, usage examples, and configuration options.

#### 7.2.3. Security and Privacy

Security and privacy measures for SQLite are maintained through the implementation of access controls and encryption methods. SQLite supports database-level access controls and the use of SQL-based permissions to manage access to schemas, tables, and individual records. SQLite does not natively support role-based access control (RBAC) or built-in encryption; however, these can be implemented via third-party extensions such as SQLite Encryption Extension (SEE) for encryption at rest. For encryption in transit, application-level security measures are advised. Comprehensive documentation on integrating these security features and best practices is available in the official SQLite documentation, which offers guidance on enhancing security configurations to meet specific operational and compliance requirements.

#### 7.3. Performance Monitoring and Database Efficiency

Performance monitoring in the ALEP system is managed collaboratively by the database administrator (DBA), system administrator, and application development team. The DBA is tasked with overseeing database-specific metrics such as space utilization, query performance, and index efficiency. System administrators focus on broader system resources like CPU usage, memory utilization, and disk I/O. Together with the application development team, they work to pinpoint and resolve performance bottlenecks, particularly in application code and database interactions.

For effective performance monitoring, ALEP employs a combination of native and third-party tools. Native SQLite tools like the command-line shell provide basic metrics, while enhanced monitoring capabilities are achieved through third-party solutions like Datadog, New Relic, and Prometheus. These tools offer extensive visibility into both system and application performance.

Interfacing with external systems, such as credit scoring services, machine learning model inference, and data enrichment platforms, also requires rigorous monitoring. This ensures that integrations do not adversely affect ALEP's performance. Moreover, data exchanges with partner systems or external databases need careful monitoring to guarantee data synchronization and system efficiency. The interface between the database and the front-end, particularly crucial for performance in data-intensive operations, is monitored to prevent issues related to data retrieval and UI rendering.

#### 7.3.1. Operational Implications

No operational implications have been identified at this stage.

#### 7.3.2. Data Transfer Requirements

No specific data transfer requirements have been identified for the ALEP system.

#### 7.3.3. Data Formats

Data formats will adhere to standard conventions for interoperability and compatibility with other system components. No specific formats have been identified as of now.

### 7.4. Backup and Recovery

Backup and recovery strategies for the ALEP system will include the following:

* Regular Backups: Periodic backups of the database will be scheduled to ensure data integrity and availability.
* Differential Backups: Incremental or differential backups will be implemented to minimize data loss and optimize backup storage usage.
* Recovery Procedures: Procedures for restoring the database from backups will be documented and tested regularly to ensure effectiveness in case of data loss or system failures.
* Backup Storage: Backup data will be stored securely in offsite locations to mitigate risks associated with hardware failures or disasters.
* Backup Retention: Policies for backup retention will be defined to manage storage space effectively while ensuring compliance with data retention requirements.

Appendix H: Interface Control Document

### Interface Overview

As the ALEP system does not interact with any third-party service or external interface, there are no external systems to describe in this section. The ALEP system operates as a standalone application within the financial institution’s infrastructure. The system is a self-contained web-based application designed to automate the loan eligibility process for financial institutions. It incorporates a ML model to analyze consumer financial data and predict loan eligibility. The system architecture primarily consists of internal components hosted within the financial institution’s network.

The ALEP system provides loan officers with a user-friendly interface to input consumer financial data directly into the application. Key functionalities include:

* Data Input: Loan officers can input financial data using the application’s user interface.
* Data Analysis: The application processes the entered financial data using the ML model to predict loan eligibility.
* Result Display: The predicted loan eligibility results are displayed within the application’s home page for loan officers to review and make informed lending decisions.

The architecture of the ALEP system is to be primarily comprised of internal components hosted within the financial institution’s infrastructure, including:

* Frontend: Developed using React.js framework, providing an intuitive and interactive user interface for loan officers.
* Backend: Implemented using Python with Flask framework, responsible for server-side processing, data validation, loan eligibility prediction, and database interactions.
* Database: Utilizes SQLite for storing consumer financial data and prediction results.

No external interfaces or third-party interactions are involved in the ALEP architecture.

#### Interface Controls

Table 6 - OSI Application Layer

| Interface Type | Interface From | Interface To | Description of Interface | Other Information |
| --- | --- | --- | --- | --- |
| User Interface | User Input | Application | Enables users to input financial data | N/A |
| Prediction Engine | Application | Application | Analyzes financial data to predict loan eligibility | Involves ML algorithm model |
| Reporting Dashboard | Application | User Output | Displays loan eligibility results and statistics | May include interactive charts and graphs |

Table 7 - OSI Presentation Layer

| Interface Type | Interface From | Interface To | Description of Interface | Other Information |
| --- | --- | --- | --- | --- |
| Data Formatting | Application | Application | Formats data for consistency and compatibility | Ensures proper display of information |

### Functional Allocation

End User Interaction:

* End-users, such as loan officers, interact directly with the ALEP system through a user-friendly web interface. They input consumer financial data into the system by filling out an application form provided by the ALEP app. This data includes information such as income, credit history, and other relevant financial details.

Data Processing:

* Upon submission of the application form, the ALEP system processes the inputted data using its loan eligibility prediction engine. This engine analyzes the financial data provided by the end-user to determine the consumer’s eligibility for a loan. The prediction engine considers factors such as credit score, income, and other relevant metrics.

Eligibility Determination:

* Based on the analysis performed by the prediction engine, the ALEP system generates loan eligibility predictions. These predictions are then presented to the end-user through the web interface. The end-user can view the results, including whether the consumer is approved for a loan.

Reporting:

* The ALEP system has reporting capabilities to generate reports on loan approval statistics and trends. These reports provide insights into the lending process and help financial institutions make informed decisions regarding loan approvals.

### Data Transfer

In the ALEP system, data movement among components is primarily facilitated through internal data processing and communication within the application itself. As the system is self-contained, there are no external component systems involved in data transfer. The ALEP system operates as a cohesive unit, where data is processed internally within the application’s architecture. Upon submission of the loan application form by the loan officer, the data is directly received and processed within the ALEP system’s backend. The loan eligibility prediction engine within the ALEP system analyzes the received data to determine loan eligibility and generate predictions. The eligibility results and any associated data are then presented to the loan officer through the web interface. The reporting features of the ALEP system also generate reports on loan approval statistics and trends based on the processed data within the application.

Connectivity among the components within the ALEP application is implemented using internal communication protocols and mechanisms. The web-based interface allows for seamless interaction between the loan officers and the ALEP system, facilitating data input, retrieval of eligibility results, and access to reporting features. Data transfer within the ALEP system utilizes standard web communication protocols such as HTTP(S) for transmitting data between the frontend and backend components. The type of messaging or packaging of data used within the ALEP system is based on industry-standard practices for web-based applications, ensuring secure and efficient data transfer.

### Transactions

Data Submission Transaction:

* Type: One-way transaction
* Description: This transaction involves the submission of consumer’s loan application data by the loan officer through the web-based interface.
* Process: The loan officer fills out the loan application form with relevant data and submits the form to the ALEP system for analysis.
* Interface: The data submission transaction occurs between the frontend web interface and the backend server of the ALEP system.

Data Processing Transaction:

* Type: Internal Transaction
* Description: This transaction involves the internal processing of submitted loan application data within the ALEP system.
* Process: Upon receiving the loan application data, the ALEP system’s backend processes the data using the loan eligibility prediction engine.
* Interface: The data processing transaction occurs entirely within the backend server of the ALEP system.

Eligibility Prediction Transaction:

* One-way transaction
* Description: This transaction involves the generation of loan eligibility predictions based on the processed application data.
* Process: The ALEP system’s eligibility prediction engine analyzes the submitted data to determine the consumer’s eligibility for the loan.
* Interface: The eligibility prediction transaction occurs within the backend server of the ALEP system and results are communicated to the frontend web interface for display to the loan officer.

Reporting Transaction:

* Type: One-way transaction
* Description: This transaction involves the generation of reports on loan approval statistics and trends based on the processed data.
* Process: The ALEP system’s reporting features generate comprehensive reports summarizing loan approval statistics, trends, and other relevant information.
* Interface: The reporting transaction occurs within the backend server of the ALEP system and results are presented to the loan officer through the frontend web interface.

### Security and Integrity

Security and integrity measures are implemented to ensure the confidentiality, authenticity, and integrity of data transmission within the system. Given that the ALEP system operates within a secure environment, with no external interfaces or connections to external networks, the interface between its components does not require elaborate security measures beyond standard internal protocols. Security and integrity measure for the ALEP system include:

Access Security:

* Access to the ALEP system is restricted to authorized users only, such as loan officers, who are authenticated through secure login credentials.
* RBAC is implemented to ensure that users can only access data and functionalities relevant to their roles and responsibilities within the system.

Data Transmission Security:

* As the ALEP is self-contained, data transmission occurs within the internal network infrastructure.
* The transmission medium used for data transfer within the system is a secure local network, ensuring that data remains within the controlled environment of the financial institution’s premises.
* Although data encryption is not mandated for internal data transmission, data integrity is ensured through standard network security measures and protocols.

Data Protection and Integrity:

* Data integrity is maintained through internal data validation and verification mechanisms within the ALEP system.
* Data protection measures are implemented to safeguard sensitive information stored within the system’s databases, including encryption of stored data and regular data backups.

Authentication and Auditing:

* User actions within the ALEP system are audited and logged to track system activities and ensure accountability.
* User authentication and session management mechanisms are employed to verify the identity of users accessing the system and track their interactions with the application.

### Detailed Interface Requirements

Given that the ALEP system is a standalone, self-contained, web-based application, there are no requirements for interfacing with an external system. As a result, no specific interface specifications, technical requirements, processing steps, message formats, communication methods, or security measures are needed for interfacing with external systems.

### Requirements for Interface with an External System

Not Applicable.

#### Assumptions

1. Not Applicable.

#### Technical Interface Requirements

Not Applicable.

#### General Processing Steps

Not Applicable.

#### Interface Processing Time Requirements

Not Applicable.

#### Message Format (or Record Layout) and Required Protocols

Not Applicable.

##### File Layout

Not Applicable.

##### Data Assembly Characteristics

1. Not Applicable.

##### Field/Element Definition

Not Applicable.

#### Communication Methods

Not Applicable.

##### Interface Initiation

Not Applicable.

##### Flow Control

Not Applicable.

#### Security Requirements

Not Applicable.

### Quality Assurance

Not Applicable.