Loan Management System

1.0 Problem statement

The Loan Management System manages the process of originating loan orders, modifying/updating, and cancelling loan related information. The loan information accounts for the loan number, loan amount, loan term, borrower information, property information, status, loan management fees, origination date, origination account, lien information, legal documents, and loan history.

The main components that make up the loan are as follows:

a) Borrower information: details of the person or party borrowing the loan amount
b) Property information: details of the property for which loan is originated
c) Legal documents: legal documents related to the property and ownership related information

Scope of the System

The scope of the system is explained through its modules as follows

- User Login and Registration – used by customers to enter their personal information into the system. The system stores the details of the customer in the system along with the account details.

  A user should be able to login with a User Id and Password that exists in the database. On clicking logout, the session should be invalidated, and the login page must be displayed.

  Capture information such as Name, Username, Password, Address, State, Country, and Email Address.

- Apply Loan - registered customers will use this option to apply for a loan in the system. The system stores the loan details in the system along with the account details.

  Capture fields like Loan Type, Loan Amount, Date, Loan Number, Property Address

  The user can add related documents by using the Upload option.

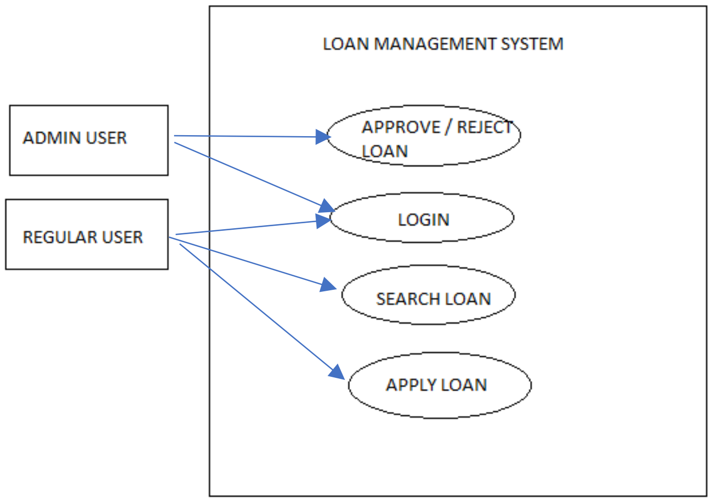
  Loan application number to be displayed to the user

- Search Loan – will be used to search for the loans applied by the user and check the status

When the user logs in, they should be able to search for loans and check their status

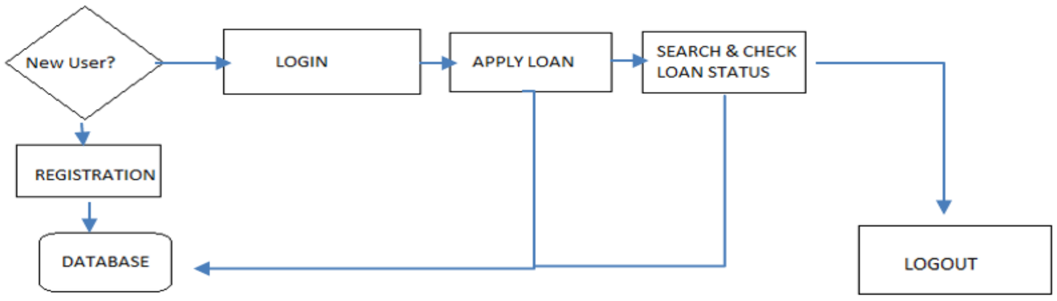
- User roles: There should be two kinds of users (admin and regular). User registration creates a regular user. An Admin user should be created only from the back end.
- Approve / Reject Loan – An admin user should be able to view loans and approve or reject them.

2.0 Use Case Diagram

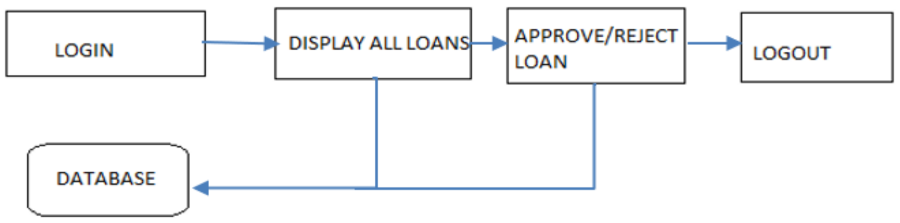


Flow Diagram

REGULAR USER



ADMIN USER



3.0 Project Development Guidelines

The project to be developed based on the below design considerations

Backend Development	• Use Rest APIs (Springboot/ASP.Net Core WebAPI to develop the services • Use Java/C# latest features • Use ORM with database • Use Swagger to invoke APIs • Implement API Versioning • Implement security to allow/disallow CRUD operations • Message input/output format should be in JSON (Read the values from the property/input files, wherever applicable). Input/output format can be designed as per the discretion of the participant. • Any error message or exception should be logged and should be user-readable (not technical) • Database connections and web service URLs should be configurable • Implement Unit Test Project for testing the API • Follow Coding Standards
Frontend Development	• Use Angular/React to develop the UI • Implement Forms, databinding, validations • Implement Routing and navigations • Use JavaScript to enhance functionalities • Implement External and Custom JavaScript files • Implement Typescript for Functions, Operators. • Any error message or exception should be logged and should be user-readable (and not technical) • Follow coding standards • Follow Standard project structure

4.0 Good to have implementation features

- Generate a SonarQube report and fix the required vulnerability
- Use the Moq framework as applicable
- Create a Docker image for the frontend and backend of the application
- Implement OAuth Security
- Implement Logging
- Implement design patterns
- Use JWT for authentication in SpringBoot/WebApi. A Token must be generated using JWT. Tokens must expire after a definite time interval, and authorization must be handled accordingly based on token expiry
- Deploy the docker image in AWS EC2 or Azure VM
- Build the application using the AWS/Azure CI/CD pipeline. Trigger a CI/CD pipeline when code is checked-in to GIT. The check-in process should trigger unit tests with mocked dependencies
- Use AWS RDS or Azure SQL DB to store the data