Software Design Description

for

Automate Credit Scoring System using ML

Version 1.2

Prepared by

Group Name: Team 2

Agnishwar Raychaudhuri	BT22GDS056	agnishwar.raychaudhuri22@st.niituniversity.in
Gautam Sharma	BT22GCS187	gautam.sharma22@st.niituniversity.in
Samarth Tanay	BT22GCS192	samarth.tanay22@st.niituniversity.in
Sarabjoth Bhatia	BT22GCS233	sarabjoth.bhatia22@st.niituniversity.in
Yuvraj Chawla	BT22GCS375	yuvraj.chawla22@st.niituniversity.in

Instructor:	***Manish Hurkat***
Course:	**CS 301 – Capstone I**
Lab Section:	***C2 C3 C6***
Date:	**20th February 2025**

Table of Contents

1. Introduction

Credit Score Prediction is the process of assessing an individual's creditworthiness (likelihood of repaying debts) using data-driven methods.It assigns a numerical score based on factors like HIstorical credit behaviour, Finance profile, Demographic information like age, employment status.
Traditional credit scoring methods often struggle with bias, limited adaptability, and an inability to effectively assess individuals with thin or nonexistent credit histories. To address these shortcomings, we propose "Automated Credit Scoring," a web-based application leveraging the power of machine learning (ML). This system aims to provide a more equitable and efficient creditworthiness evaluation process.

1.1 Purpose

The purpose of this document is to provide a comprehensive Software Design Description (SDD) for the Automating Credit Scoring System using ML. This document will serve as a blueprint for developers, financial institutions and other stakeholders to understand the architecture,data flow, system components and various other components involved in the systems development.

This document will provide an in-depth look at the system’s architecture and the tech stack,ensuring a well-structured and scalable implementation.

1.2 Scope

The Automating Credit Scoring System is designed to be a fully automated, ML-driven credit assessment tool that will evaluate individuals based on financial transactions, income patterns, and other relevant factors.

1. The system will Collect and preprocess structured/unstructured data (e.g., credit history, income, employment, and alternative data sources).
2. The system will Train, validate, and deploy ML models (e.g., Random Forest, Gradient Boosting, or Neural Networks) to predict creditworthiness.
3. Provide interpretable results (Credit Score) to comply with regulatory standards and user trust.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
MERN	MongoDB, Express.js, React.js, Node.js
ML	Machine Learning
API	Application Programming Interface
UI	User Interface
SDD	Software Design Description
CI/CD	Continuous Integration and Continuous Deployment

1.4 References

1. https://medium.com/@byanalytixlabs/how-to-develop-a-credit-scoring-model-with-machine-learning-41e95f76a7ec
2. https://www.slideshare.net/slideshow/sdd-software-des-sample/34428403
3. https://www.mongodb.com/docs/
4. https://svitla.com/blog/machine-learning-for-credit-scoring/
5. https://www.tutorialspoint.com/expressjs/index.htm

2. System Overview

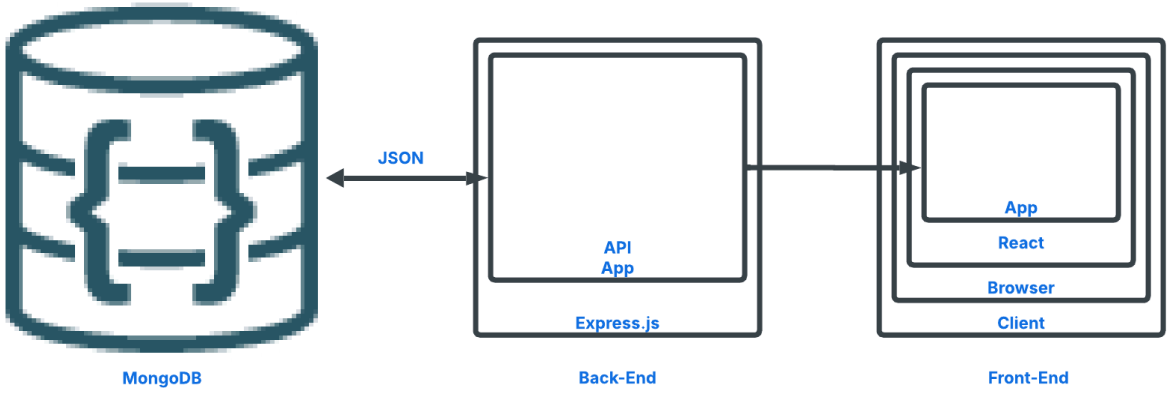
2.1 System Description

The Automating Credit Scoring System is a web-based application that utilizes Machine Learning (ML) to evaluate the creditworthiness of individuals. The system aims to streamline the credit evaluation process for financial institutions, lenders, and individuals by automating credit score predictions based on financial and behavioral data..

System Architecture & Technology Stack

1. MERN Stack Implementation
    1.1. MongoDB (NoSQL Database): Stores user profiles, transaction histories, and credit reports.
    1.2. Express.js & Node.js (Backend): Provides a RESTful API that facilitates communication between the frontend and ML model.
    1.3. React.js (Frontend): A user-friendly dashboard where individuals can register, submit financial data, and view credit scores.
2. Machine Learning Integration with Python
    2.1. The ML model is developed in Python
    2.2. The ML model will process financial data, analyze key factors like income, spending habits, outstanding loans, and payment history, and generate a credit score.
    2.3. The trained model will be utilized via a REST API (Flask or FastAPI) and integrated into the MERN-based web application.
3. Data Preprocessing & Model Training
    3.1. Conduct Data Cleaning and Normalization followed by feature engineering
    3.2. Conduct Model Training and Evaluation
4. Security & Compliance
    4.1. Implements encryption to secure sensitive financial data.
    4.2. Include authentication to secure user sessions
5. Cloud & Containerization
    5.1. The entire system will be containerized using Docker
6. CI/CD Pipeline
    6.1. Integrates Continuous Integration/Continuous Deployment (CI/CD) using Jenkins or GitLab

2.2 System Architecture



The diagram represents the architecture of a MERN stack-based application, illustrating the interaction between the database, backend, and frontend components. At the foundation of the system is MongoDB, a NoSQL database that stores all relevant user data, including financial records, credit history, and transactional details. The database communicates with the backend using JSON format, allowing for seamless data retrieval and storage.

The backend, built using Node.js and Express.js, serves as the intermediary between the frontend and the database. It processes user requests, applies business logic, and interacts with MongoDB to fetch or update data. The backend also hosts an API (Application Programming Interface), which enables smooth communication between different parts of the system.

On the other side, the frontend is developed using React.js and runs within a web browser, forming the user interface of the application. The frontend interacts with the backend through API calls, sending requests for data and receiving responses in JSON format. When a user submits financial details or requests a credit score, the frontend sends this data to the backend, which processes it and retrieves the relevant information from MongoDB. The backend then returns a response, which the frontend uses to update the UI dynamically, displaying the necessary insights to the user.

2.3 Users

The **Automating Credit Scoring System** is designed to serve multiple stakeholders, each with specific roles and responsibilities. The system ensures secure access, personalized interactions, and role-based functionalities to cater to different user needs.

The key users of the application will be

a. Customers (Loan Applicants) - Individuals applying for loans, credit cards, or other financial products. Users looking to assess and improve their creditworthiness.
b. Financial Institutions (Banks, Credit Agencies, FinTech Firms) - Banks, credit unions, and lenders that need automated credit scoring for loan approvals. Credit bureaus that assess risk profiles for individuals.

3. Data Flow and Processing

The Automating Credit Scoring System follows a structured data pipeline, ensuring efficient collection, processing, transformation, and analysis of financial data. The data flow is designed to handle real-time processing, ensuring accurate and up-to-date credit score predictions.

3.1 Data Sources

The system aims to integrate multiple data sources to build a robust dataset for credit scoring. The data sources that we aim to include are

a. User-Submitted Data - This would include, personal data, financial details, loan history and bank statements
b. Transactional Data (Banking & Financial Records) - This would include account transaction history, credit card payment history, loan repayment records and defaults(if any)

3.2 Data Processing

Once data is collected from multiple sources, it will undergo preprocessing to ensure accuracy, consistency, and reliability before being fed into the Machine Learning Model.

The steps involved would be

a. Data Cleaning and Handling Missing Values - This will be done to remove duplicate and inconsistencies from financial data through methods such as mean imputation, median

imputation or regression-based filtering. To detect outliers we aim to employ statistical methods like Z-Score and IQR-based filtering

b. Data Normalization & Transformation - This will be done through feature scaling to bring all values within a uniform range. One-hot encoding will be done for categorical variables. Finally, we aim to use log transformation for highly skewed numerical data
c. Preprocessed data is stored in MongoDB (NoSQL Database) for fast retrieval.

3.3 Feature Selection

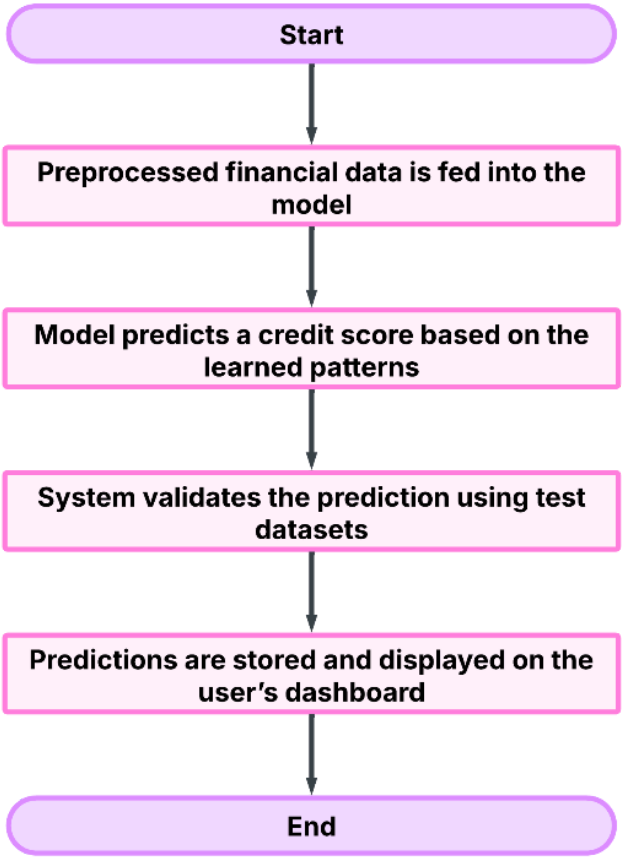
The selected features for the Credit Scoring Model are given below

Feature Type	Selected Features
Demographic Features	Includes employment type and assets
Income & Expense Features	Includes the financial records
Loan History Features	Includes previous loan approvals, outstanding loans and repayment history
Credit Behavior Features	Includes credit usage, credit age and missed payments
Transaction-Based Features	Includes spending patterns and major transactions

3.4 ML Model

The Machine Learning model is responsible for predicting the credit score based on the selected features. The model is trained using supervised learning algorithms.

The workflow of the ML Model is as such



The Algorithms that will be used for Credit Scoring are

a. Logistic Regression - Baseline model for binary classification
b. Random Forest - To improve the accuracy using multiple decision trees
c. XGBoost - To boost the performance of the model

Model Deployment

a. The trained ML model is deployed as a REST API using Flask or FastAPI.
b. The MERN stack backend (Node.js/Express.js) calls this API to fetch real-time credit score predictions.

4. Implementation Details

4.1 Technology Stack

The Automating Credit Scoring System will be built using a MERN (MongoDB, Express.js, React.js, Node.js) stack for web application development and Python-based Machine Learning models for credit scoring

Frontend (User Interface)

- **React.js** – Used for building an interactive and responsive UI.
- **Bootstrap** – Used for styling the application.

Backend (API & Server Logic)

- **Node.js** – Provides a runtime environment for executing JavaScript on the server.
- **Express.js** – A lightweight framework for building RESTful APIs and handling HTTP requests.

Database (Data Storage and Retrieval)

- **MongoDB (NoSQL Database)** – Stores user profiles, transaction histories, and credit scores.

Machine Learning Model (Credit Scoring Algorithm)

- **Python** – Used for training ML models.
- **Flask / FastAPI** – Creates an API to serve the trained ML model to the web application.
- **Pandas & NumPy** – Used for data preprocessing and feature extraction.

Security & Compliance (Tentative)

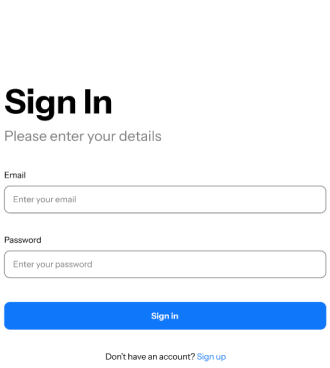
- **Google Authentication** – Provides a secure login experience using OAuth.

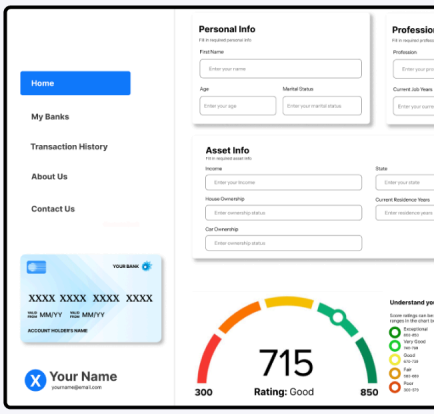
Deployment & DevOps

- **Docker** – Containerizes the application for better scalability and portability.
- **CI/CD Pipelines (GitHub Actions / Jenkins)** – Automates testing and deployment.

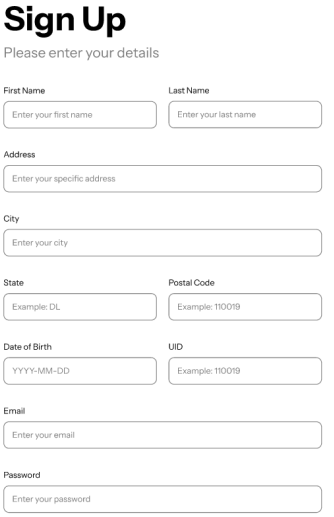
4.2 GUI (Graphical User Interface)

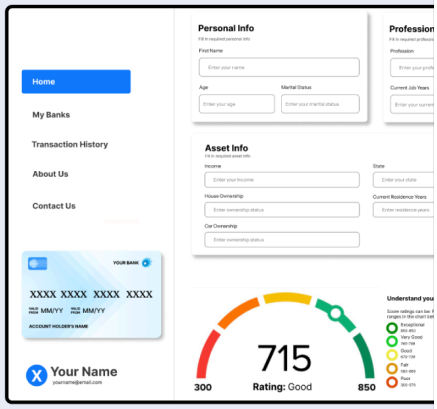
4.2.1 Sign In page



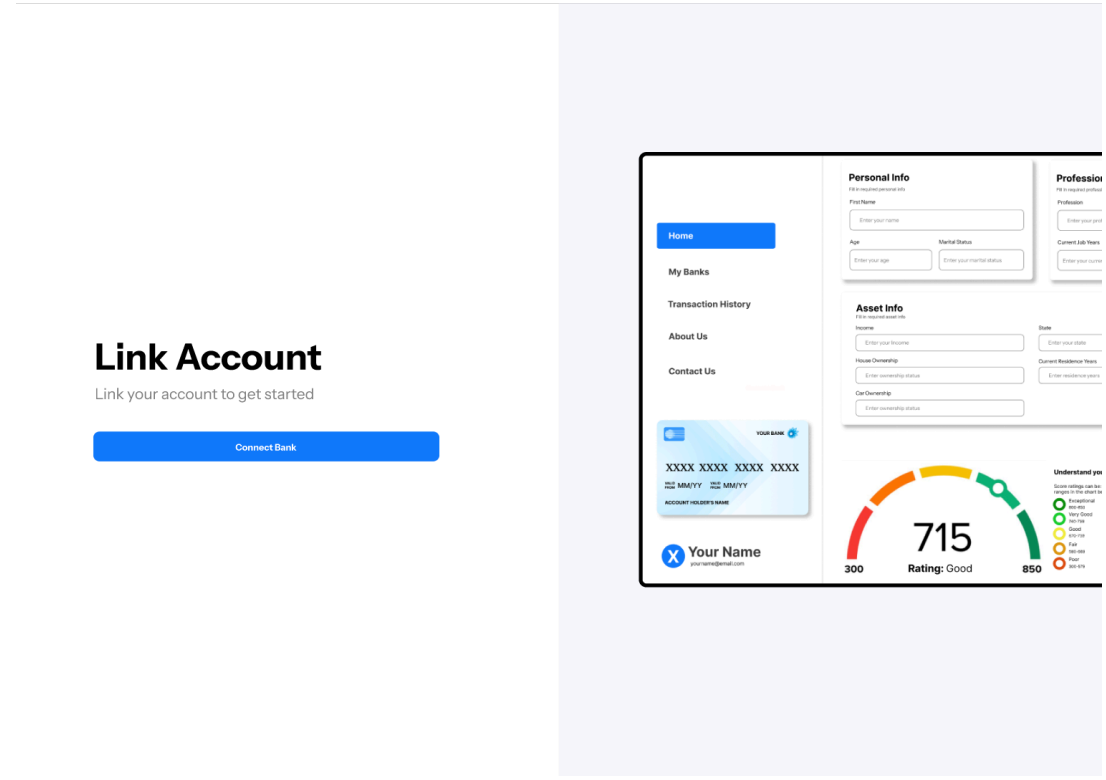


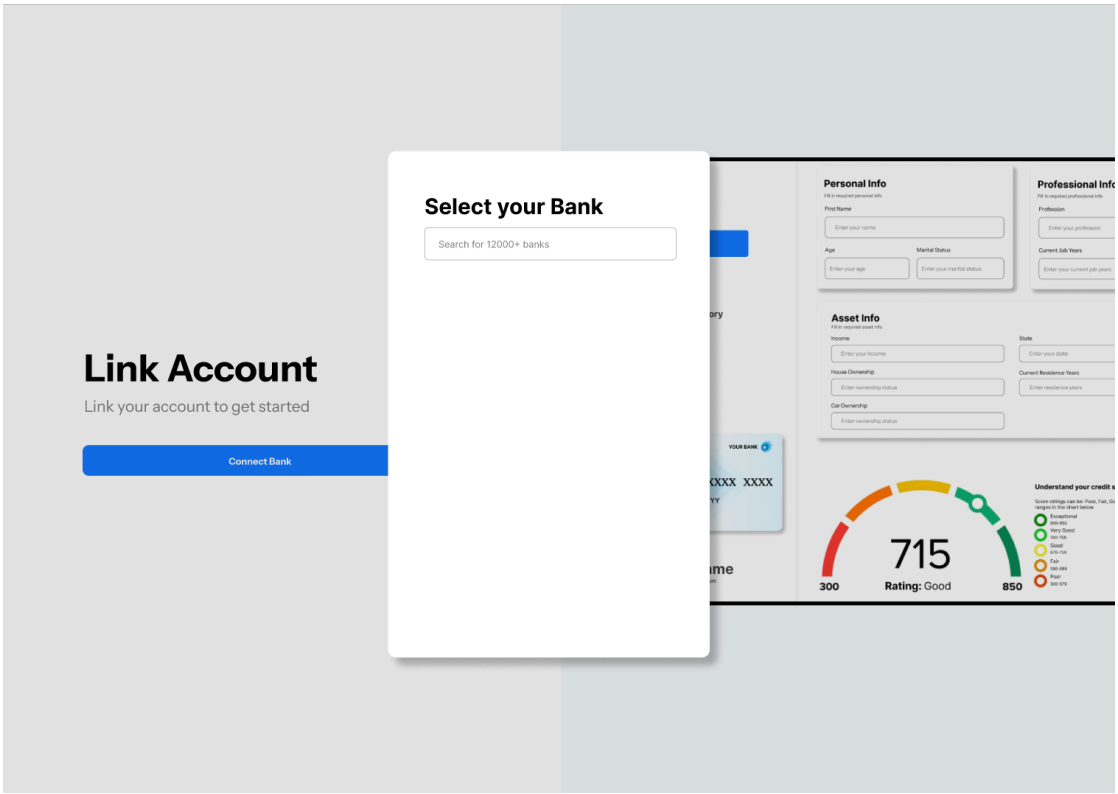
4.2.2 Sign Up page



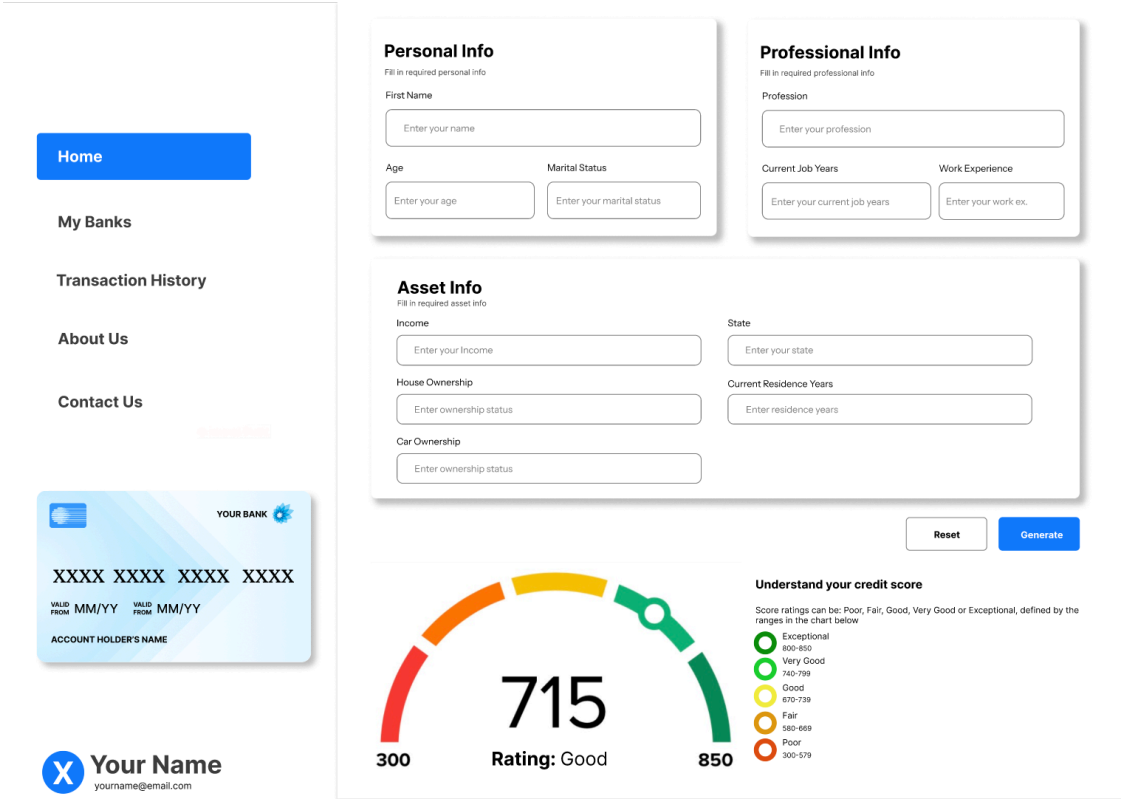


4.2.3 Link bank account page





4.2.4 Home page



5. Security Aspects

Ensuring data security and regulatory compliance is critical in financial applications like the Automating Credit Scoring System.

5.1 Security Measures (Tentative)

The system aims to utilize Google Authentication to provide a secure OAuth-based login mechanism. By leveraging a trusted third-party authentication provider, it enhances security, protects user credentials, and reduces the risk of unauthorized access. This approach ensures robust access control while simplifying the login process for users.

6. CI/CD & Deployment Strategy

We plan to implement Continuous Integration (CI) and Continuous Deployment (CD) to keep the Automating Credit Scoring System reliable, scalable, and easy to update. This setup will help us automatically test, review, and deploy code changes, reducing downtime and minimizing errors in production. While we haven't set it up yet, integrating a CI/CD pipeline will ensure smoother development and faster updates in the future.

6.1 CI/CD Proposal (Tentative)

We plan to set up a version control and code management system using Git, with repositories hosted on GitHub or GitLab. We will follow a structured branching strategy with feature branches, a development branch, and a main branch to keep the project stable and organized. Before merging any changes, pull requests will go through automated checks and manual code reviews to ensure quality.

For Continuous Integration (CI), we plan to automate testing to ensure that new code changes do not break existing functionality. This will include running unit tests, integration tests, and security scans.

When it comes to Continuous Deployment (CD), we plan to automate the build and deployment process using GitHub Actions or Jenkins

7. Conclusion

The Automating Credit Scoring System aims to provide a modern, data-driven approach to credit assessment by leveraging Machine Learning and the MERN stack. With planned features like real-time ML processing and Google Authentication, the system will ensure secure and accurate credit scoring. A CI/CD pipeline will be implemented to automate testing, updates, and deployments without downtime. Designed with a user-friendly interface and explainable AI insights, the platform will offer intelligent financial recommendations, making it a strong alternative to traditional credit scoring methods. Future improvements will focus on advanced ML techniques, enhanced authentication mechanisms, and optimized deployment strategies to further improve security, performance, and scalability.