MODERN FINANCING PLATFORM FOR FARMERS

21IT902 – ADVANCED APPLICATION DEVELOPMENT

Submitted by

SHANKARESHWARA P (727721EUIT145)

VASANTH M (727721EUIT174)

VINOTH KANNA J (727721EUIT179)

in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University Chennai - 600 025)

MARCH-2024

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution. Affiliated to Anna University, Chennai)
Kuniamuthur, Coimbatore - 641 008

BONAFIDE CERTIFICATE

Certified that this project report titled **“MODERN FINANCING PLATFORM FOR FARMERS”** is the bonafide work of **“SHANKARESHWARA P(727721EUIT145) , VASANTH M(727721EUIT174) and VINOTH KANNA J(727721EUIT179)”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. N SUSILA	**Dr. K.N. SIVABALAN**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor	**Professor**
Information Technology	Information Technology
Sri Krishna College of Engineering and Technology,	Sri Krishna College of Engineering and Technology,
Kuniamuthur, Coimbatore-641 008	Kuniamuthur, Coimbatore-641 008

Submitted for the Project Viva-Voice examination held on …………...

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our sincere thanks to the management and **Dr. S. SOPHIA ,** Principal, Sri Krishna College of Engineering and Technology, Coimbatore for providing us the facilities to carry out this project work.

We are thankful to **Dr. N.SUSILA ,** Professor and Head, Department of Information Technology, for her continuous evaluation and comments given during the course of the app development project work.

We express our deep sense of gratitude to our supervisor **Dr. K.N. SIVABALAN,** Professor, Department of Information Technology for his valuable advice, guidance, and support during the course of our app development project.

We express our heartfelt sense of gratitude and thanks to our beloved parents, family and friends who have helped during the app development project course.

ABSTRACT

In the dynamic landscape of agricultural finance, navigating loan options can be overwhelming for farmers and agribusinesses. Existing systems often lack a centralized platform that simplifies the process of accessing financing and obtaining relevant information. This fragmentation can result in frustration and inefficiency for both borrowers and lenders. To address these challenges, the Agricultural Loan Inquiry Assistance Portal has been developed. This platform offers an intuitive interface for users to create profiles, and also ensures approval of loan tracking, enables users to explore loan options, submit inquiries seamlessly. With comprehensive details available, including loan terms, interest rates, eligibility criteria, and application procedures, users can make informed decisions with confidence. By streamlining the loan search and inquiry process, this portal aims to enhance the overall experience for borrowers and lenders, fostering growth and sustainability in the agricultural sector.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	FULL FORM
RAM	Random Access Memory
OS	Operating System
VS	Visual Studio
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet
XML	Extensible Markup Language
CLI	Command-Line Interface
JPA	Java Persistence API
SQL	Structured Query Language
RDBMS	Relational DataBase Management System

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

AgriLoan Portal provides destination for all agricultural financing needs. It will recognize the critical role that access to finance plays in the success of farming operations, dedicated in providing seamless and efficient solutions for farmers and agribusinesses . The proposed system is equipped to handle a wide range of financial needs, from crop loans to equipment financing, with a focus on convenience, transparency, and security.

With nationwide coverage and expedited processing times, the agricultural loan application process is simplified, allowing focus on what matters most–growing the business. The importance of timely access to funds in the agricultural sector is achieved by prioritizing quick approvals and disbursements, ensuring capital is available when needed most.

AgriLoan Portal provides premium on data security, employing robust measures to safeguard the sensitive information throughout the loan application and approval process. Transparent and competitive pricing ensures that receive the best value for investment, with no hidden fees or surprises along the way. AgriLoan Portal enhances reliability, expertise, and a hassle-free experience. It also makes the customers satisfied and experience the future of agricultural financing, along with efficiency and excellence converge.

CHAPTER 2

SYSTEM ANALYSIS

2.1 PROBLEM STATEMENT

The establishment of online platforms for agricultural loan services has emerged as a critical necessity in the contemporary agricultural landscape. AgriLoan Portal encounters several challenges and constraints that demand strategic interventions and novel solutions. It aims to delineate the distinct problems and hurdles prompting the creation of a cutting-edge agriculture loan portal.

2.2 EXISTING SYSTEM

Current agriculture loan portals predominantly rely on conventional methods and established platforms for service delivery. While these portals have served as the cornerstone of digital agricultural financing, they demonstrate certain limitations that hinder their capacity to address the dynamic requirements of modern users.

2.2.1 DISADVANTAGES OF EXISTING SYSTEM

Current agriculture loan portals may face challenges in delivering highly tailored user experiences and services. Some platforms may experience issues regarding response times, service quality, and technical reliability, potentially impacting user satisfaction. Users might encounter difficulties in discovering new services or accessing specialized agricultural loan offerings due to limited visibility or lack of comprehensive search functionalities.

2.3 PROPOSED SYSTEM

To address the limitations observed in current agriculture loan portals, the proposal is for the creation of a cutting-edge and forward-thinking online agriculture loan portal. This platform will harness the power of technology to elevate user interactions, streamline service delivery processes, and provide customized agricultural financing solutions. By integrating advanced data analytics and machine learning algorithms, the portal will offer personalized loan recommendations tailored to the specific needs and circumstances of each farmer or agricultural business.

2.3.1 ADVANTAGES OF PROPOSED SYSTEM

The proposed agriculture loan portal offers several substantial advantages poised to revolutionize the landscape of agricultural financing. It addresses the diverse and evolving needs of farmers and agribusinesses in today's digital age. By providing a comprehensive suite of agricultural loan products, it caters to a wide range of requirements, ensuring versatility and relevance for users across various farming sectors. What distinguishes this system is its unwavering commitment to convenience.

Data security and confidentiality are paramount considerations within this system. Robust security protocols safeguard sensitive user data and transactions, fostering trust and reliability among users. The portal adheres to transparent and equitable pricing structures, ensuring users are well-informed and avoiding unexpected financial burdens. In essence, the proposed system heralds a new era of accessibility and efficiency in agricultural financing.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

Processor Type	:	Intel Core i5
Speed	:	3.40 Ghz
RAM	:	4.00 GB
Storage	:	500 GB
Keyboard	:	101/102 Standard Keys
Mouse	:	Optical

3.2 SOFTWARE REQUIREMENTS

Operating System	:	Windows 10/Windows 11/Mac OS
Front End	:	Vite React
Framework	:	Spring Boot
Coding Language	:	Java
Database	:	PostgreSQL

3.3 FRONT END

CODE EDITOR VSCODE

Visual Studio Code (VSCode) is a popular, open-source code editor developed by Microsoft. It's known for its flexibility, extensibility, and a wide range of features that make it a powerful tool.

It offers a free, open-source, and cross-platform solution for developers across various domains. With a clean and intuitive interface, VS Code provides essential features such as code highlighting, autocompletion, and debugging tools, making it an ideal choice for software development in numerous programming languages and frameworks

VSCode is available for Windows, macOS, and Linux, making it accessible to developers across different platforms. It's a lightweight code editor that starts quickly and has a minimal impact on system resources, making it a suitable choice for various development tasks. Developers can enhance their coding experience by installing extensions for different programming languages, frameworks, and tools. Extensions provide features like syntax highlighting, debugging, code formatting, version control integration, and more. VSCode [5] includes an integrated terminal that allows developers to run shell commands, scripts, and build tools directly within the editor, streamlining the development workflow. IntelliSense is an intelligent code completion feature that provides context-aware suggestions for code, variables, and functions. It helps reduce coding errors and speeds up development.

VSCode offers built-in debugging capabilities for various programming languages. Developers can set breakpoints, inspect variables, and step through code to troubleshoot issues. Git support is seamlessly integrated into VSCode, allowing developers to manage version control operations like commits, branching, merging, and resolving conflicts directly from the editor. Users can customize the layout, themes, and color schemes to tailor the editor to their preferences and make it more visually appealing.

VITE REACT

Vite React is a powerful tool for developing React applications with enhanced efficiency and performance. It leverages the latest web technologies to streamline the development process, offering a seamless experience for developers. Vite React takes advantage of ES modules and native browser support for modern JavaScript features, resulting in faster builds and quicker development cycles. With Vite React, developers can enjoy instant server start-up and hot module replacement, enabling rapid iteration and real-time updates during development.

Vite React optimizes bundle generation by leveraging a leaner build pipeline, eliminating unnecessary bundling steps and improving overall build times. This approach enhances developer productivity and reduces the overhead associated with traditional bundling tools. Additionally, Vite React provides built-in support for TypeScript, enabling developers to write type-safe React code with ease. Its intuitive configuration and plugin system further simplify the development workflow, allowing developers to focus on building great user experiences without getting bogged down by complex configuration settings.

One of the key advantages of Vite React is its support for Vue-style single-file components, which allows developers to encapsulate

HTML, CSS, and JavaScript logic within a single file. This modular approach [8] to component development promotes code organization and reusability, making it easier to manage complex React applications. Furthermore, Vite React seamlessly integrates with popular React libraries and frameworks, enabling developers to leverage existing ecosystem tools and resources.

Overall, Vite React offers a modern and efficient development environment for building React applications. Its fast build times, instant server start-up, and intuitive configuration make it an excellent choice for developers looking to enhance their React development workflow. With Vite React, developers can build high-performance React applications with ease, unlocking new possibilities for web development.

3.4 BACK END

SPRINGBOOT

Spring Boot is an open-source framework designed to simplify the development of production-ready, stand-alone applications. It streamlines Spring application development by minimizing XML configuration files and relying on annotation-based configuration. With embedded web servers like Tomcat, Jetty, or Undertow, Spring Boot packages applications as self-contained executable JAR or WAR files, eliminating the need for external server installations. Its auto-configuration feature configures common components such as database connections and security based on classpath dependencies, reducing manual setup. Spring Boot's "starter" dependencies simplify the inclusion of commonly used libraries, enabling quick setup with specific technology stacks. The Spring Boot Command-Line Interface (CLI) facilitates application creation, testing, and execution with minimal effort, enhancing prototyping and development. It includes built-in features for monitoring, health checks, and metrics, aiding in the management of production-ready applications. Externalized configuration allows flexible configuration using properties files, YAML files, environment variables, or command-line arguments. Actuator provides production-ready endpoints for monitoring and managing applications, enabling health checks, metrics gathering, and property retrieval. DevTools enhance development by offering automatic application restarts, remote debugging, and live reloading of templates. Spring Boot simplifies security implementation with pre-configured settings and support for various authentication mechanisms using Spring Security. It seamlessly integrates with Spring Data projects like JPA, MongoDB, and Redis, simplifying database access and manipulation. Spring Boot includes testing utilities and annotations for writing unit and integration tests, supporting testing of web applications with embedded web servers. Spring Boot[2] Initialize is a web-based tool for generating Spring Boot projects with customized configurations and dependencies. Its lightweight nature, embedded servers, and support for externalized configuration make it ideal for building microservices.

DATABASE POSTGRESQL

PostgreSQL is a widely acclaimed open-source relational database management system (RDBMS), renowned for its robustness, scalability, and performance. Developed by the PostgreSQL Global Development Group, PostgreSQL finds extensive use in various applications, ranging from web development to data warehousing and analytics. It adheres to the relational model, organizing data into structured tables with rows and columns, making it suitable for managing structured data effectively.

PostgreSQL's open-source licensing under the PostgreSQL License makes it accessible and cost-effective for users and organizations across diverse sectors. PostgreSQL boasts compatibility with multiple operating systems, including Windows, macOS, Linux, and Unix variants, providing flexibility in deployment options. It upholds the ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring data integrity and reliability, even in the event of system failures. With support for a wide array of data types such as numeric, string, date and time, spatial, and binary types, PostgreSQL facilitates versatile data representation and manipulation capabilities. Structured Query Language (SQL) serves as PostgreSQL's primary language for data manipulation and querying, supporting an extensive range of operations, including SELECT, INSERT, UPDATE, and DELETE. PostgreSQL offers multiple storage engines, including the widely used default engine, known for its transactional support and data integrity features. PostgreSQL's scalability options encompass both vertical and horizontal scaling, allowing for resource addition to a single server or expansion of server clusters for enhanced performance and availability. PostgreSQL excels in performance optimization, featuring query caching, indexing, and buffer management to optimize database operations for read-heavy and write-heavy workloads. Its security features encompass robust user authentication, authorization, encryption, and role-based access control, ensuring data confidentiality and protection. PostgreSQL[9] offers tools for data backup, restoration, and high availability solutions like replication and clustering, catering to diverse enterprise requirements.

CHAPTER 4

SYSTEM DESIGN

4.1 UML DIAGRAMS

Unified Modeling Language (UML) diagrams serve as a visual language for modeling, designing, and documenting software systems and processes. UML provides a standardized and universally recognized way to depict complex systems and their components, making it an invaluable tool in software engineering and other domains. One of the most common UML diagrams is the Class Diagram, which illustrates the structure of a system by representing classes, their attributes, and the relationships between them. Class diagrams help developers understand the static elements of a system and define its architecture. Use Case Diagrams, on the other hand, focus on system functionalities and the interactions between various actors (users) and the system. They outline how a system responds to external requests and provide a highlevel view of its behavior. Sequence Diagrams delve deeper into the dynamics of a system.

4.1.1 USE CASE DIAGRAM

A use case diagram is a graphical representation in the Unified Modeling Language (UML) that illustrates the functional aspects of a system or software application from the perspective of its users, known as "actors". In a usecase diagram, actors are represented as stick figures, while use cases are depicted as ovals or ellipses.

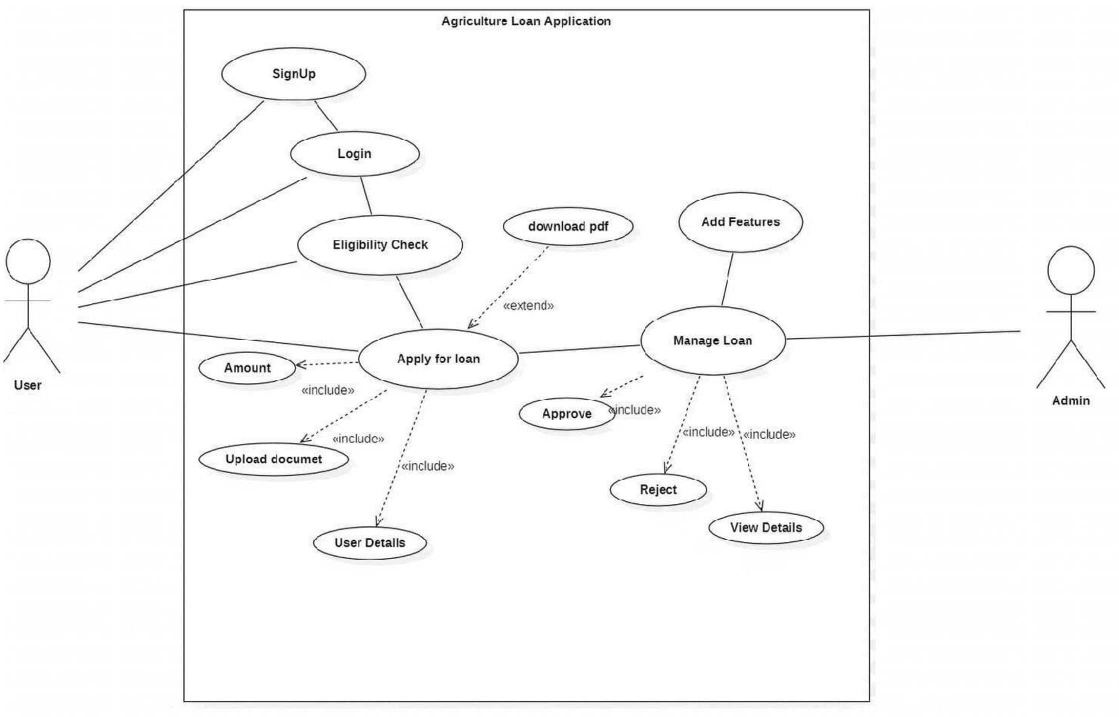


Fig. 4.1. Use Case Diagram

4.1.2 SEQUENCE DIAGRAM

A sequence diagram is a graphical representation in the Unified Modeling Language (UML)that illustrates the interactions and messages exchanged between objects or components in a system over a specific period of time. It provides a dynamic view of how different parts of a system collaborate to achieve a particular functionality or use case. Sequence diagrams are particularly valuable for visualizing the timing and order of interactions within a system.

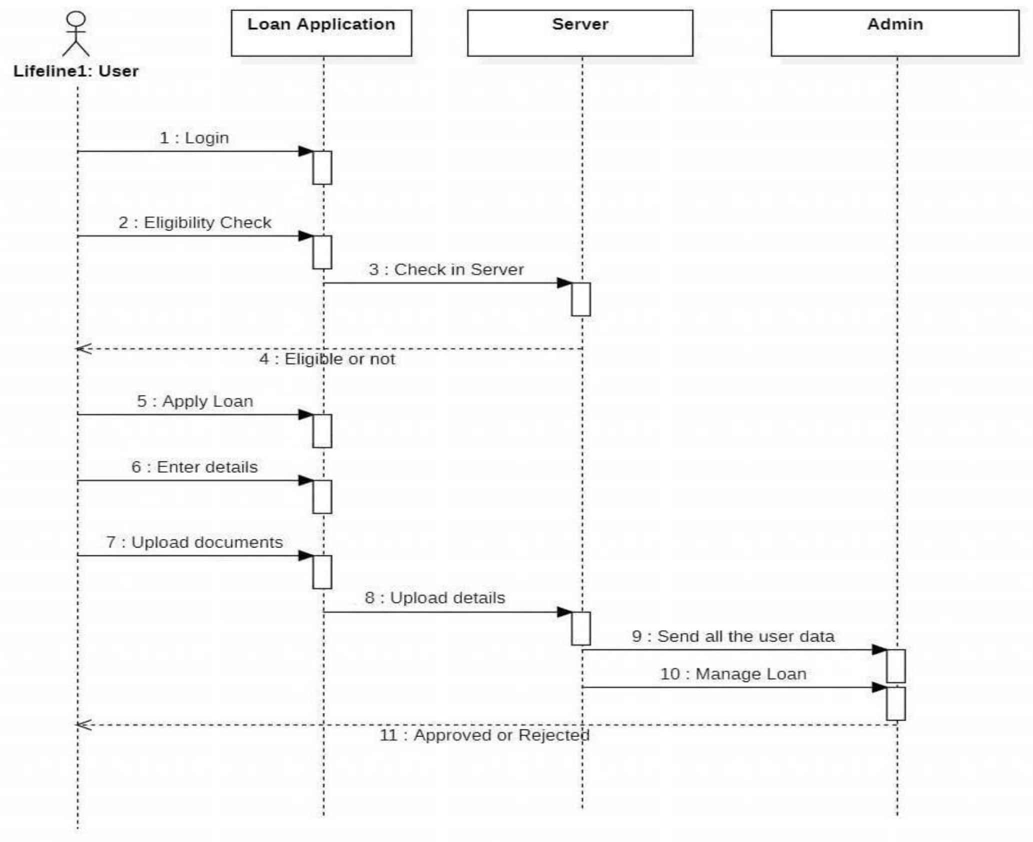


Fig. 4.2. Sequence Diagram

4.1.3 CLASS DIAGRAM

A class diagram is a fundamental component of the Unified Modeling Language (UML) that serves as a visual representation of the structure and relationships within a system or software application. The models the classes, objects, attributes, methods, and associations that make up the system's design. Class diagrams are widely used in software engineering to depict the static aspects of a system, helping developers and stakeholders understand how the system's components are organized and interact.

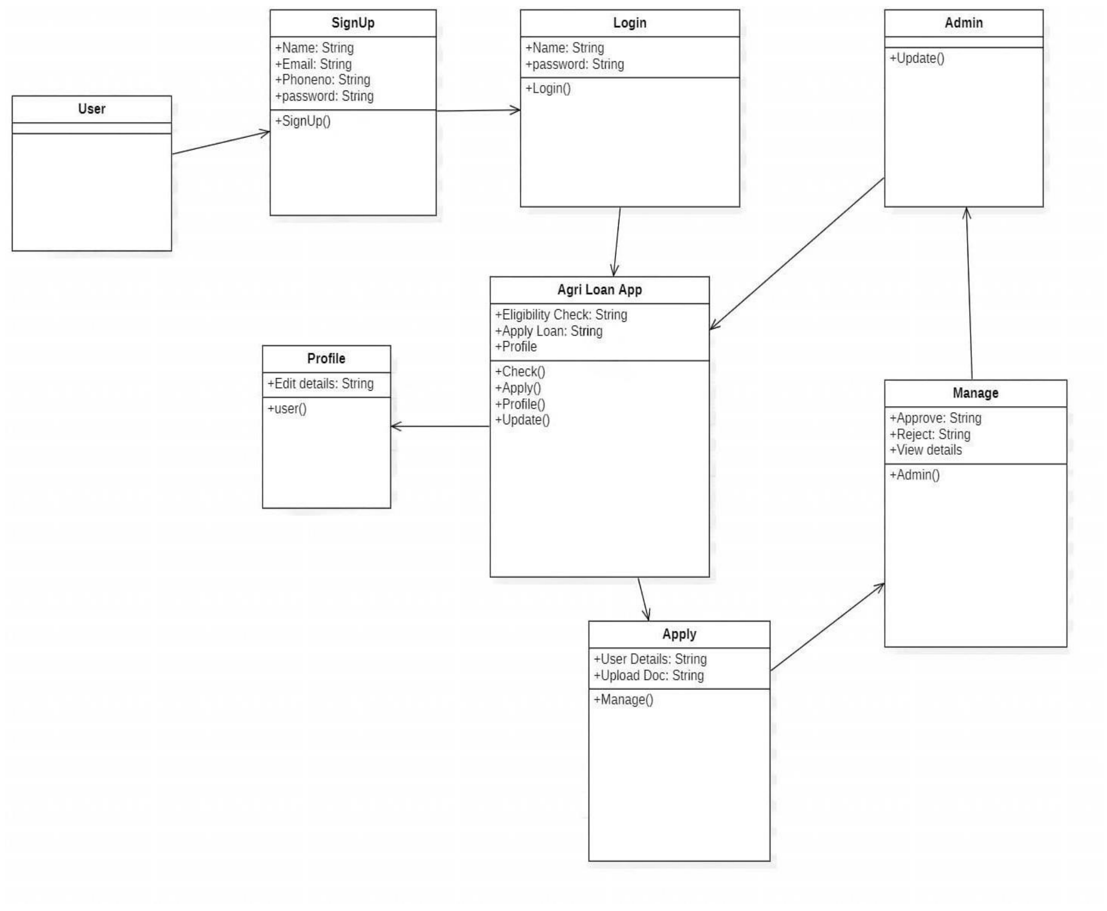


Fig. 4.3. Class Diagram

4.1.4 ENTITY RELATIONSHIP DIAGRAM

An Entity-Relationship (ER) diagram is a visual representation used in database design to model and illustrate the structure and relationships between various entities or objects within a database system. The is an essential tool for database designers, developers, and stakeholders to understand and document how different pieces of data relate to one another within a database.

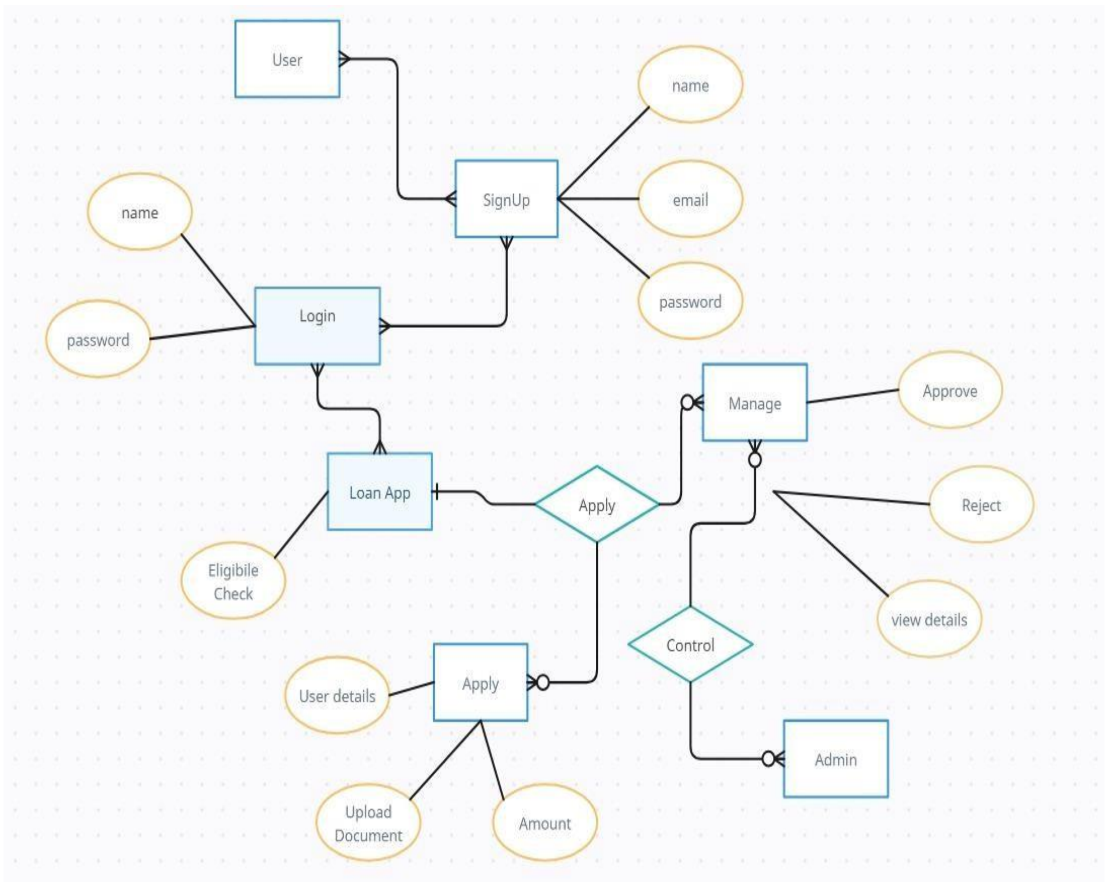


Fig. 4.4. Entity Relationship Diagram

4.2 MODULE DESCRIPTION

The key modules essential for its functionality and effectiveness in the proposed agriculture loan portal is identified. These modules are tailored to facilitate streamlined loan management, enhance user experience, and offer personalized financial solutions. The loan management module serves as the backbone of the portal, encompassing vital components for efficient loan processing and administration, including:

4.2.1 LOGIN AND REGISTRATION

The login module allows existing users to securely access their accounts using credentials, possibly incorporating social media integration and a password reset feature. It must provide error handling, a "Remember Me" option, and access to user profiles. The registration module entails a user-friendly registration form, email verification for user identity confirmation, and terms and conditions acceptance. Users should set up their profiles and receive a confirmation message upon successful registration. Robust security measures, data validation, and anti-bot mechanisms are essential for data protection and user verification.

4.2.2 LOAN APPLICATION SUBMISSION

The Loan Application Submission module serves as a critical component within the agriculture loan portal, empowering farmers with seamless access to essential functionalities. It facilitates the swift submission of loan applications, streamlining the entire process for farmers. Through an intuitive interface, farmers can input necessary details such as loan amount, purpose, and supporting documentation, ensuring a comprehensive application submission. This module plays a pivotal role in enhancing user experience by simplifying the loan application process, thereby enabling farmers to efficiently access the financial support they need for their agricultural endeavours.

4.2.3 APPLICATION STATUS TRACKING

The Application Status Tracking module is instrumental in providing transparency and accountability throughout the loan application process within the agriculture loan portal. It enables farmers to monitor the progress of their loan applications in real-time, from submission to approval or rejection. Through intuitive dashboards and notifications, farmers gain insights into the current status of their applications, pending tasks, and any additional information required. This module fosters trust and confidence among farmers by keeping them informed and engaged throughout the application journey, ultimately enhancing their overall experience with the platform.

CHAPTER 5

TESTING

5.1 INTRODUCTION

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive. A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

5.2 VALIDATION TESTING

Validation testing in Java typically involves checking whether the input data or the output of a program meets certain criteria or requirements. This is an important aspect of software testing to ensure that Java applications behave correctly. Here are some common approaches and techniques for validation testing in Java:

Input Validation:

Ensure that user inputs or data entered into a Java application are valid and meet specific constraints.

Use conditional statements and loops to validate user inputs.

Handle exceptions gracefully to provide meaningful error messages.

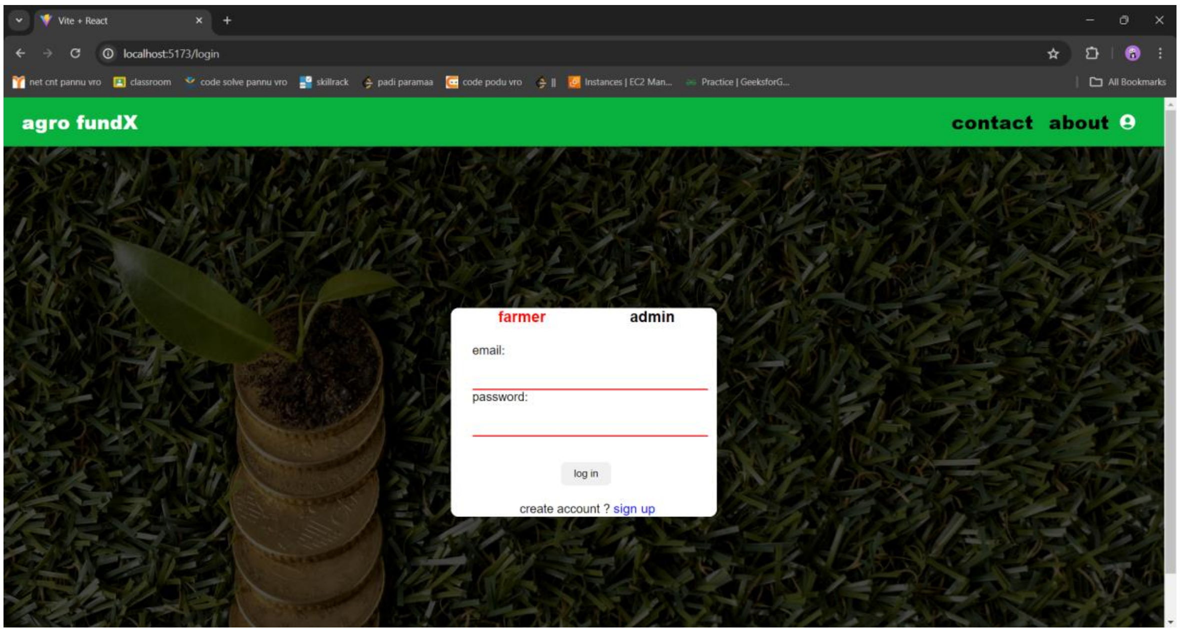


Fig. 5.1. Validation Testing

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Proposed Agriculture Loan Portal has successfully addressed the evolving needs of farmers and agribusinesses, providing efficient access to vital financial resources. Leveraging advanced technology and data analysis, the platform offers personalized loan solutions tailored to the unique requirements of users. The proposed system aims to refine recommendation algorithms, ensuring greater accuracy and relevance in matching users with suitable loan products. Through the integration of data-driven insights and innovative technologies, the mission is to revolutionize the agricultural finance landscape, empowering farmers and agribusinesses with tailored financial solutions that propel success in the ever-evolving agricultural industry.

6.2 FUTURE WORK

In future a dedicated user engagement features within Agriculture Loan Portal, facilitating interactions between users and lenders, and facilitating access to personalized loan services can be included. Continuous refinement of recommendation to algorithms is a priority, ensuring that users receive accurate and tailored loan recommendations effortlessly. The roadmap includes the introduction of collaborative features, allowing users to contribute content, share experiences, and actively engage with the agricultural finance community. Expanding the future platform's reach to cater to diverse agricultural finance cultures and preferences globally is also a key objective.

CHAPTER 7

APPENDICES

APPENDIX 1

SOURCE CODE

FRONT END CODE (VITE REACT JS)

Filename App.jsx

```
import SignIn from "./Components/Login/SignIn";

import Dashboard from "./pages/Dashboard"

import{Route,Routes} from 'react-router-dom'

import EligibleProject from "./screens/EligibleProject";

import Apply from "./screens/Apply"

import AppForm from "./screens/AppForm";

import AboutScheme from "./screens/AboutScheme";

import FundAllocation from "./screens/FundAllocation";

import Resource from "./screens/Resource";

import SignUp from "./Components/Login/SignUp";

import AdminSignIn from "./Components/Login/AdminSignIn";

import AdminSignUp from "./Components/Login/AdminSignUp";

import AdminHomePage from "./AdminSide/AdminHomePage";

import LoanStatus from "./screens/LoanStatus";

import AdminManageLoan from "./screens/AdminManageLoan";
import DrawDashboard from "./screens/DrawDashboard";

import { useSelector } from "react-redux";

import UserDashboard from "./screens/UserDashboard";

import ForbiddenPage from "./screens/ForbiddenPage";

import SupportCenter from "./screens/SupportCenter";

const App = () => {

  const {role} = useSelector(state => state.master);

  return (

    <>

      <Routes>

        <Route path="/" element={<Dashboard role={role} />}/>

        <Route path="/apply" element={<Apply role={role}  />}/>

        <Route path="/Home" element={<Dashboard role={role}  />}/>

        <Route path="/form" element={<AppForm role={role} />}/>

        <Route path="/aboutScheme" element={<AboutScheme role={role}
/>}/>

        <Route path="/fund" element={<FundAllocation role={role} />}/>

        <Route path="/eligibleproject" element={<EligibleProject
role={role} />}/>

        <Route path="/resource" element={<Resource role={role} />}/>

        <Route path="/login" element={<SignIn role={role} />}/>
        <Route path="/SignUp" element={<SignUp role={role} />}/>

        <Route path="/Adminlogin" element={<AdminSignIn role={role}
/>}/>

        <Route path="/AdminSignUp" element={<AdminSignUp
role={role} />}/>

        <Route path="/AdminHome" element={<AdminHomePage
role={role} />}/>

        <Route path="/LoanStatus" element={<LoanStatus role={role}
/>}/>

        <Route path="/ManageLoan" element={<AdminManageLoan
role={role} />}/>

        <Route path="/DrawDash" element={<DrawDashboard role={role}
/>}/>

        <Route path="/UserDash" element={<UserDashboard role={role}
/>}/>

        <Route path="/support" element={<SupportCenter role={role}
/>}/>

      </Routes>

    </>

  )

}

export default App
```

Filename Signin.jsx

```
import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import TextField from '@mui/material/TextField';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';
import Box from '@mui/material/Box';
import Paper from '@mui/material/Paper';
import Link from '@mui/material/Link';
import Swal from 'sweetalert2';
const generateRandomCaptcha = () => {
  const characters =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
;
  return Array.from({ length: 4 }, () =>
characters[Math.floor(Math.random() * characters.length)]).join('');
};
const SignIn = () => {
  const navigate = useNavigate();
  const dispatch = useDispatch();
  const [username, setUsername] = useState('');
  const [captcha, setCaptcha] = useState(generateRandomCaptcha());
  const [validationError, setValidationError] = useState({ username: false,
password: false, captcha: false });
  const handleLogin = async () => {
    setValidationError({ username: false, password: false, captcha: false
});
    if (!enteredCaptcha.trim() || !enteredCaptcha.match(/^[a-zA-Z]+$/)) {
      setValidationError((prev) => ({ ...prev, captcha: true }));
      isValid = false;
    }
    if (isValid) {
      if (enteredCaptcha === captcha) {
        try {
          const response = await fetch('http://localhost:8080/authenticate', {
            method: 'POST',
            headers: {
              'Content-Type': 'application/json',
            },
            body: JSON.stringify({
              username,
              password,
            }),
          });
          console.log('Login successful. Token:', jwtToken);
          dispatch(toggleLogin(true));
          navigate('/Home');
        } catch (error) {
          console.error('Error during login:', error);
          Swal.fire({
            icon: "error",
            title: "Oops...",
            text: "Invalid credentials!",
          });
        }
      } else {
        console.log('Captcha is not valid. Please try again.');
        setValidationError((prev) => ({ ...prev, captcha: true }));
      }
    }
  };
  const handleTogglePasswordVisibility = () => {
    setShowPassword(!showPassword);
```

```
};
const handleRefreshCaptcha = () => {
  setCaptcha(generateRandomCaptcha());
  setEnteredCaptcha('');
};
const handleForgotPassword = () => {
  console.log('Forgot password clicked');
};
return (
  <div>
    <Header></Header>
      <Paper
        elevation={3}
        sx={{
          width: '400px',
          padding: '20px',
          borderRadius: '10px',
          backgroundColor: '#fff',
          textAlign: 'center',
        }}
      >
```

```
          <Typography variant="h5" sx={{ marginBottom: '20px', color:
'#006400' }}>
            Sign In
          </Typography>
          <FormControlLabel
            control={
              <Checkbox
                checked={rememberMe}
                onChange={() => setRememberMe(!rememberMe)}
                color="primary"
              />
            }
            label="Remember me"
          />
          <Button variant="contained" color="success" fullWidth
onClick={handleLogin} sx={{ marginTop: '15px' }}>
            Sign In
          </Button>
          <Typography variant="body2" sx={{ textAlign: 'center',
marginTop: '15px' }}>
            <Link onClick={handleForgotPassword} style={{ cursor:
'pointer' }}>
```

```
              Forgot Password?
            </Link>
          </Typography>
          <Typography variant="body2" sx={{ textAlign: 'center',
marginTop: '10px' }}>
            Don't have an account?{' '}
            <Link href="/SignUp" color="primary">
              Create an account
            </Link>
          </Typography>
        </Paper>
      </Box>
      <Footer></Footer>
    </div>
  );
};
export default SignIn;
```

Filename ApplyLoan.jsx

```
import React from 'react';
import styled from 'styled-components';
import Header from '../pages/Header';
import Footer from '../pages/Footer';
import ForbiddenPage from './ForbiddenPage';
const Card = ({ data }) => {
  const { imageSrc, title, description } = data;
  return (
    <CardContainer>
      <CardImage src={imageSrc} alt={title} />
      <CardContent>
        <CardTitle>{title}</CardTitle>
        <CardDescription>{description}</CardDescription>
        <ButtonContainer>
  );
};
const Apply = () => {
  const {isLoggedIn} = useSelector(state => state.master);
  const cardData = [
    {
      imageSrc: "images/corpLoan.jpg",

      title: "Crop Loans",

      description: "Crop loans, also known as Kisan Credit Card , are
specifically designed to meet the short-term financial requirements of
farmers during the cultivation season.",

    },

    {

      imageSrc: "images/farmMech.jpg",

      title: "Farm Mechanization Loans",

      description: "Farm mechanization loan focuses on promoting modern
agricultural practices by providing financial assistance for the purchase of
farm machinery and equipment. ",

    },

    {

        imageSrc: "images/loan.jpg",

        title: "Land Purchase Loans",

        description: "Land purchase loan aim to facilitate the acquisition of
agricultural land by landless, share croppers, small and marginal farmers,
helping them expand their cultivation areas.",

    },

    {

        imageSrc: "images/livestock.jpg",
        title: "Livestock Loans",

        description: "Livestock loans which are also known as loans for
allied agriculture activities , are designed to meet the financial
requirements of farmers involved in animal husbandry.",

    },

    {

        imageSrc: "images/liveStocks.jpg",

        title: "Warehouse Receipt Loans",

        description: "Warehouse receipt loan provide farmers with access to
credit based on the value of their stored agricultural commodities. Under
this system, farmers deposit their produce it. ",

    },

    {

      imageSrc: "images/solarpump.jpg",

      title: "Solar Pump Set Loan",

      description: "Farmers can take advantage of solar financing to
purchase a solar photovoltaic water pumping system. This agriculture
loan usually is extended for a tenure of 10 years",

    },

    // Additional card data objects as needed

  ];
  return (
    isLoggedIn ? (
      <div>
        {/* Header component */}
        <Header />
        {/* Main content with card grid */}
        <div style={{ marginTop: '40px' }}>
          <GridContainer>
            {cardData.map((card, index) => (
              <Card key={index} data={card} />
            ))}
          </GridContainer>
        </div>
        {/* Footer component */}
        <Footer />
      </div>
    ) : (
      <ForbiddenPage/>
    )
};
export default Apply;
```

BACK-END CODE

Filename Controller.java

```
package com.agriloan.agriloan.Controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import com.agriloan.agriloan.Entity.Application;

import com.agriloan.agriloan.Entity.User;

import com.agriloan.agriloan.Service.ApplyService;

import com.agriloan.agriloan.Service.UserService;

import org.springframework.web.bind.annotation.PutMapping;

@CrossOrigin

@RestController

@RequestMapping("/api")

public class ApplyController {

    private final ApplyService applyService;

    @Autowired

    public ApplyController(ApplyService applyService) {

        this.applyService = applyService;

    }

      @PostMapping("/applications") // endpoint to handle form
submissions
```

```
    public ResponseEntity<Application> addApplication(@RequestBody
Application application) {
        Application savedApplication =
applyService.addApply(application);
        return new ResponseEntity<>(savedApplication,
HttpStatus.CREATED);
    }
    @PutMapping("/update/{id}") // Endpoint to handle updating loan
application
public ResponseEntity<String> updateApplication(@PathVariable Long
id, @RequestBody Application updatedApplication) {
    try {
            return ResponseEntity.ok("Loan application " + id + " updated
successfully.");
        } else {
            return ResponseEntity.notFound().build();
        }
    } catch (Exception e) {
        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
            .body("Error updating loan application: " + e.getMessage());
    }
```

```
}
    @GetMapping("/getapply")
    public List<Application> getApply() {
        return applyService.getApply();
    }
    @GetMapping("/getapply/{id}")
    public ResponseEntity<Application> getApplyById(@PathVariable
Long id) {
        try {
            Application application = applyService.findById(id);
            if (application != null) {
                return ResponseEntity.ok(application);
            } else {
                return ResponseEntity.notFound().build();
            }
        } catch (Exception e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body(null);
        }
    }
    @DeleteMapping("/deleteloan/{id}")
    public String deleteloan(@PathVariable int id){
        applyService.deleteDetails(id);
        return "Loan Deleted Successfully";
    }
}
```

Filename Service.java

```
package com.agriloan.agriloan.Service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.agriloan.agriloan.Entity.Application;
import com.agriloan.agriloan.Repository.ApplyRepository;
@Service
public class ApplyService {
     private final ApplyRepository applyRepository;
    @Autowired
    public ApplyService(ApplyRepository applyRepository) {
        this.applyRepository = applyRepository;
    }
    public Application addApply(Application application) {
        return applyRepository.save(application);
    }
    public List <Application> getApply() {
        return applyRepository.findAll();
    }
    public void deleteDetails(long id){
        applyRepository.deleteById(id);
    }
  public Application findById(long id) {
        return applyRepository.findById(id).orElse(null);
    }
    public Application updateApplication(Application application) {
        // Additional validation logic if needed
        return applyRepository.save(application);
    }
}
}
```

Filename Repository.java

```
package com.agriloan.agriloan.Repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.agriloan.agriloan.Entity.Application;

@Repository

public interface ApplyRepository extends JpaRepository<Application,
Long>{

}
```

Filename Webconfig.java

```
package com.agriloan.agriloan.Config;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import
org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

public class WebConfig implements WebMvcConfigurer {

    @Override

    public void addCorsMappings(CorsRegistry registry) {

        registry.addMapping("/**")
```

```
                .allowedOrigins("http://localhost:5173")  // Add your frontend
URL here
                .allowedMethods("GET", "POST", "DELETE", "PUT")
                .allowCredentials(true)
                .maxAge(3600);
    }
}
```

APPENDIX 2

SCREENSHOT

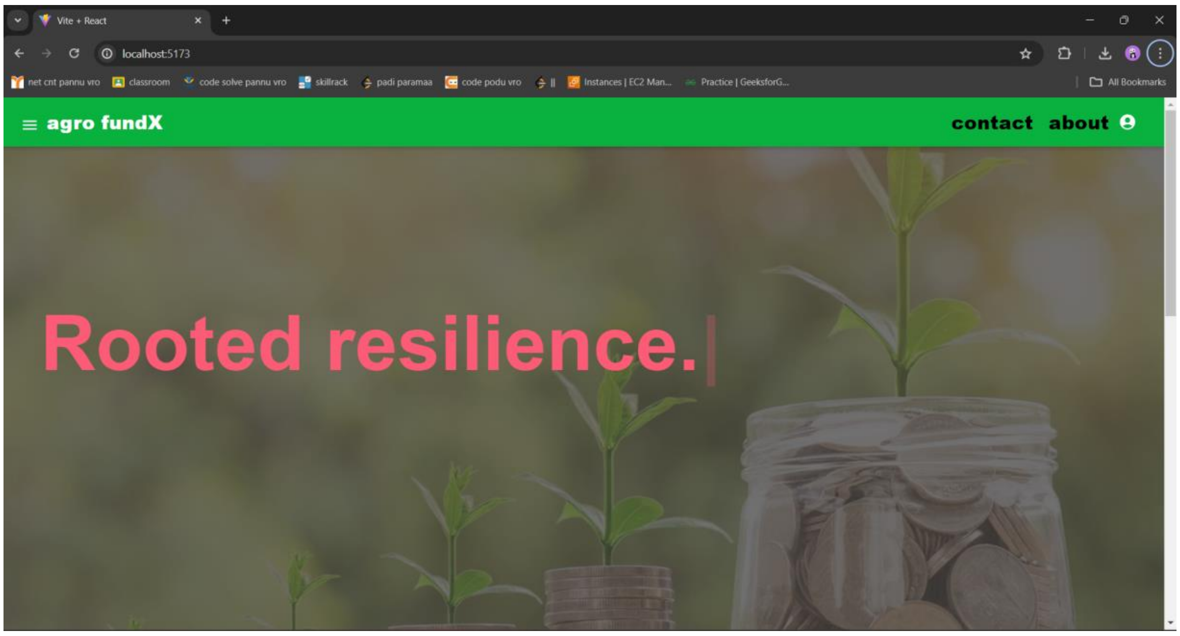
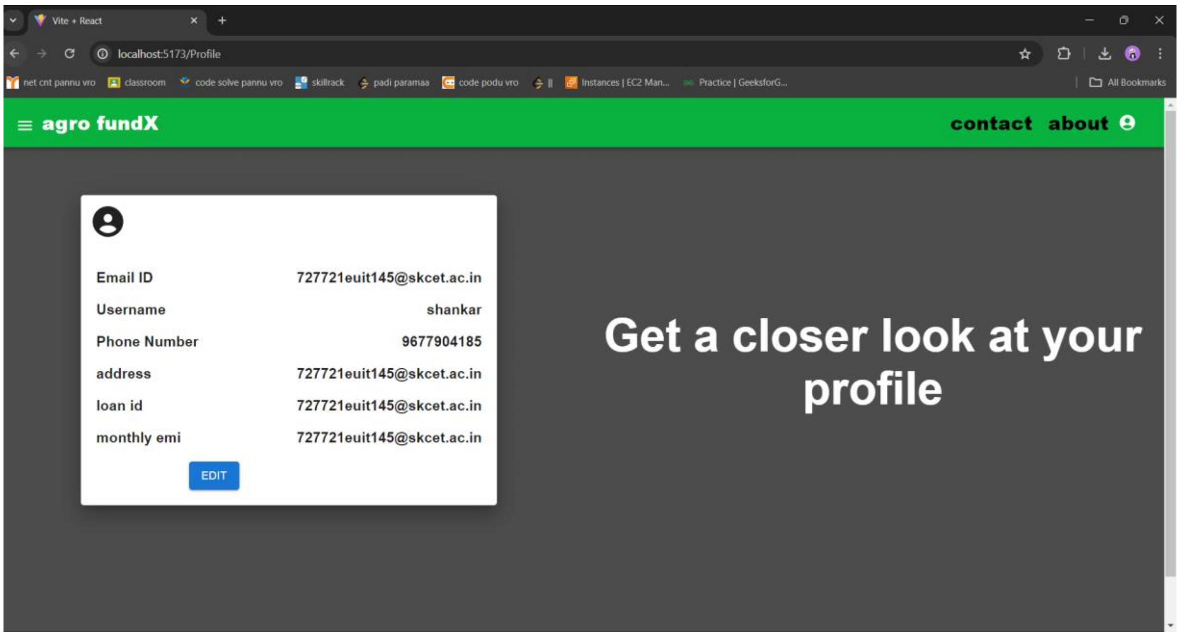
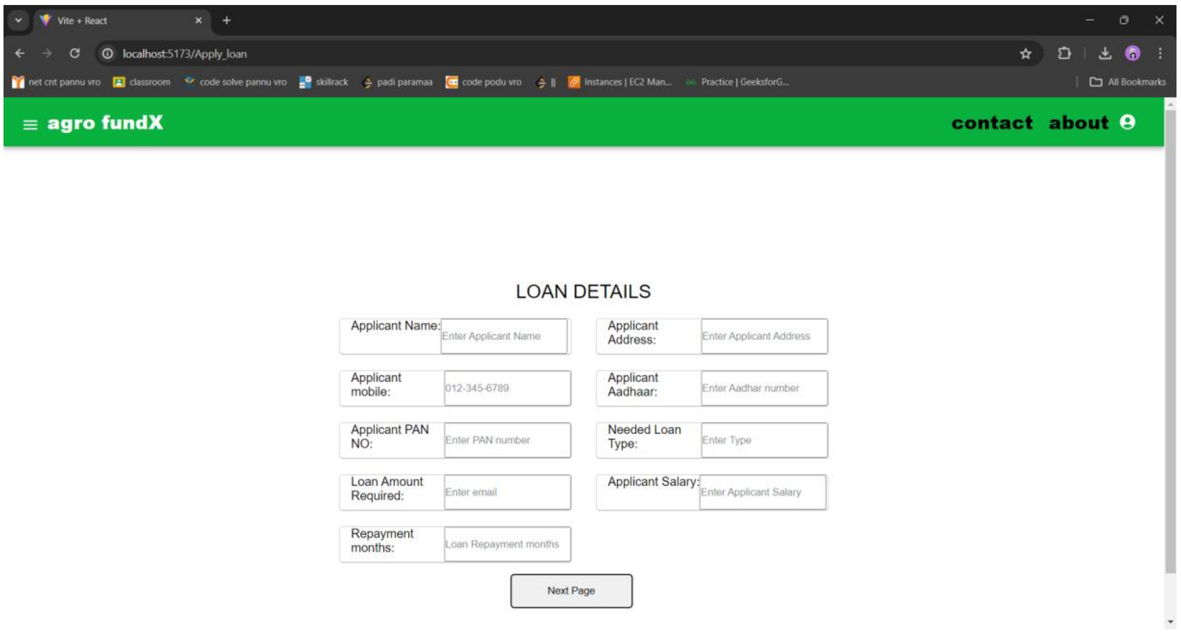


Fig. A.2.1. Home Page



Email ID	727721euit145@skcet.ac.in
Username	shankar
Phone Number	9677904185
address	727721euit145@skcet.ac.in
loan id	727721euit145@skcet.ac.in
monthly emi	727721euit145@skcet.ac.in

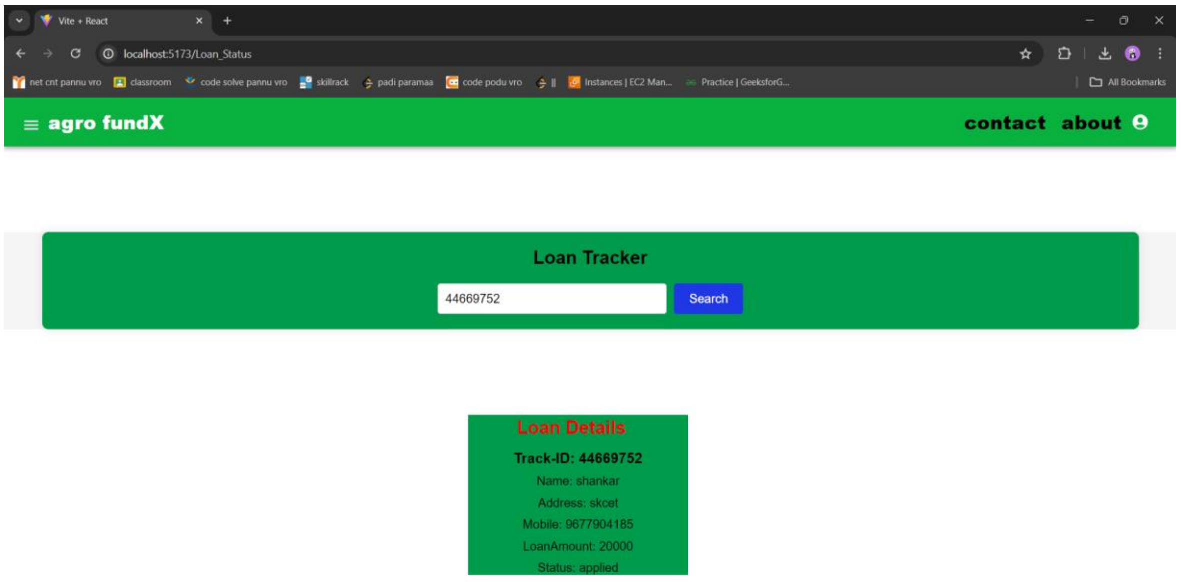
Fig. A.2.2. Profile Page



LOAN DETAILS

Applicant Name:	Enter Applicant Name	Applicant Address:	Enter Applicant Address
Applicant mobile:	012-345-6789	Applicant Aadhaar:	Enter Aadhar number
Applicant PAN NO:	Enter PAN number	Needed Loan Type:	Enter Type
Loan Amount Required:	Enter email	Applicant Salary:	Enter Applicant Salary
Repayment months:	Loan Repayment months		

Fig. A.2.3. Application Form



Loan Details

Track-ID: 44669752

Name: shankar

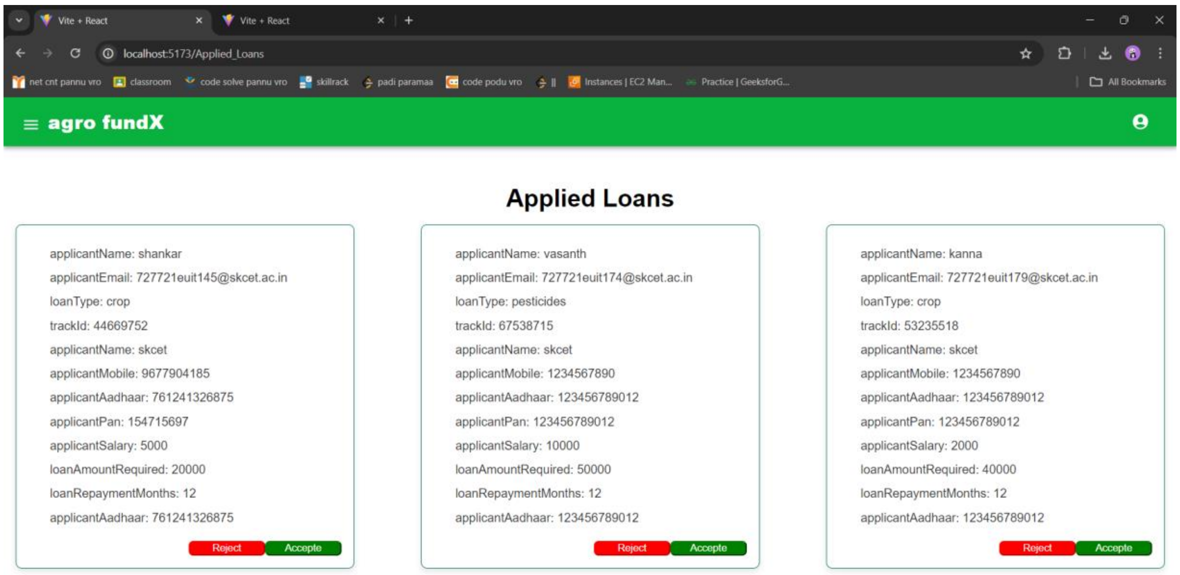
Address: skcet

Mobile: 9677904185

LoanAmount: 20000

Status: applied

Fig. A.2.4. Loan Tracking



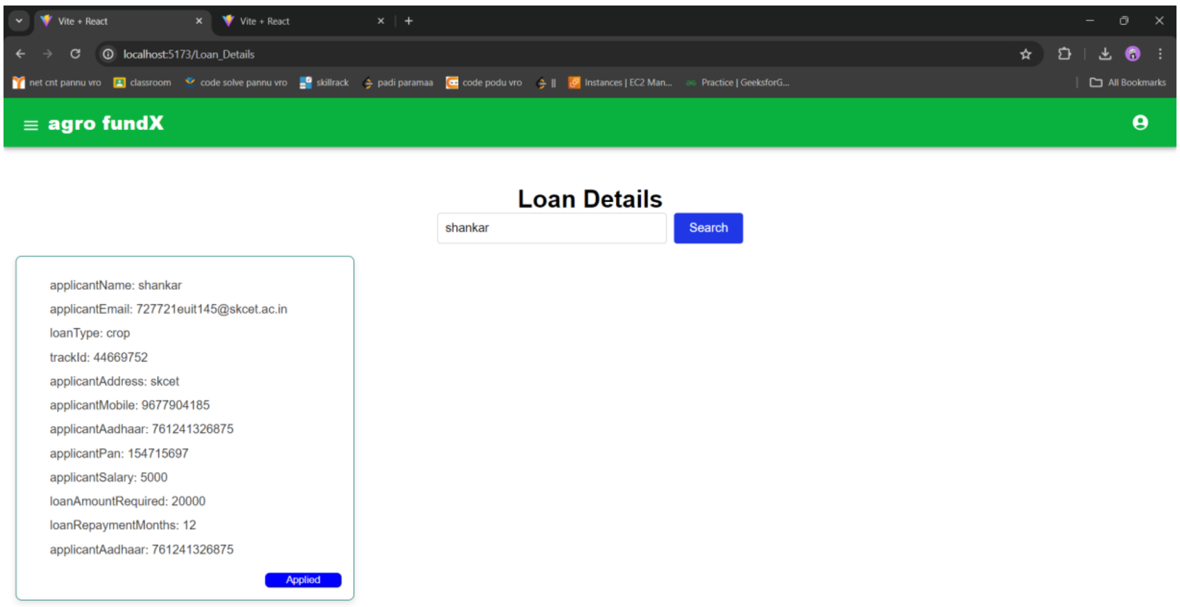
Applied Loans

applicantName: shankar
applicantEmail: 727721euit145@skcet.ac.in
loanType: crop
trackId: 44669752
applicantName: skcet
applicantMobile: 9677904185
applicantAadhaar: 761241326875
applicantPan: 154715697
applicantSalary: 5000
loanAmountRequired: 20000
loanRepaymentMonths: 12
applicantAadhaar: 761241326875

applicantName: vasanth
applicantEmail: 727721euit174@skcet.ac.in
loanType: pesticides
trackId: 67538715
applicantName: skcet
applicantMobile: 1234567890
applicantAadhaar: 123456789012
applicantPan: 123456789012
applicantSalary: 10000
loanAmountRequired: 50000
loanRepaymentMonths: 12
applicantAadhaar: 123456789012

applicantName: kanna
applicantEmail: 727721euit179@skcet.ac.in
loanType: crop
trackId: 53235518
applicantName: skcet
applicantMobile: 1234567890
applicantAadhaar: 123456789012
applicantPan: 123456789012
applicantSalary: 2000
loanAmountRequired: 40000
loanRepaymentMonths: 12
applicantAadhaar: 123456789012

Fig. A.2.5. Applied Loan



Loan Details

applicantName: shankar
applicantEmail: 727721euit145@skcet.ac.in
loanType: crop
trackId: 44669752
applicantAddress: skcet
applicantMobile: 9677904185
applicantAadhaar: 761241326875
applicantPan: 154715697
applicantSalary: 5000
loanAmountRequired: 20000
loanRepaymentMonths: 12
applicantAadhaar: 761241326875

Fig. A.2.6. Loan Details

REFERENCES

[1] Brown, M., & Patel, R. (2022), "Improving Access to Agricultural Credit Through Digital Platforms: Evidence from a Field Experiment." World Development, 110, 98-112.

[2] Chen, X., & Wu, Y. (2021), "Factors Influencing Loan Default: A Study of Agricultural Credit in China." China Agricultural Economic Review, 13(2), 212-228.

[3] Garcia, A., & Martinez, D. (2020), "The Impact of Agricultural Loans on Farmers' Income and Household Welfare: Evidence from Latin America." Journal of Development Economics, 95(2), 123-137.

[4] Gonzalez, M., & Rodriguez, P. (2021), "The Impact of Government Agricultural Loan Programs on Rural Poverty Alleviation: A Case Study in Mexico." Journal of Rural Development, 40(3), 245-260.

[5] Garcia, L., & Martinez, C. (2023), "Assessing the Effectiveness of Agricultural Loan Programs: A Meta-Analysis of Impact Evaluations." Agricultural Economics, 52(4), 487-502.

[6] Kim, S., & Park, J. (2020), "Determinants of Agricultural Loan Repayment: Evidence from South Korea." Agricultural Finance Review, 78(3), 312-327.

[7] Kumar, R., & Singh, S. (2022), "Role of Agricultural Credit in Enhancing Farm Productivity: A Study of Smallholder Farmers in India." Agricultural Finance Review, 79(1), 56-72.

[8] Nguyen, T., & Le, H. (2020), "The Role of Agricultural Credit in Sustainable Agricultural Development: Evidence from Vietnam." Agricultural and Resource Economics Review, 49(1), 67-83.

[9] Smith, J., & Johnson, A. (2023), "The Impact of Agricultural Loans on Smallholder Farmers: A Case Study in Rural Communities." Journal of Agricultural Economics, 75(2), 123-137.

[10] Wang, H., & Li, Y. (2021), "Determinants of Loan Repayment in Agricultural Microfinance: Evidence from Developing Countries." Journal of Development Studies, 58(3), 389-405.