International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Loan Analytics and Planning Application Using AI

Iswarya.S[1], Aasha.K[2], Abarna.S[3], Senthil Kumaran.G[4]

[1,2,3]Dept of Computer Science & Engineering, Meenakshi Ramaswamy Engineering College
[4]Associate Professor, Computer Science & Engineering, Meenakshi Ramaswamy Engineering College

Abstract
The Loan Analytics and Planning Application is a full-stack web application designed to assist financial institutions and individual users in analyzing loan application data using artificial intelligence. The system enables users to upload CSV files containing loan application records, which are then processed through an automated Databricks pipeline for AI-powered loan decision-making. The application provides a comprehensive analytics dashboard featuring key performance indicators (KPIs), interactive charts, and detailed data tables that present insights into loan approvals, rejections, and applicant demographics. The frontend of the application is developed using React.js with modern UI libraries including Ant Design and Material UI, providing responsive and intuitive user interface. The backend is built with Python Fast API, offering RESTful API endpoints for authentication, data retrieval, file upload, and AI-powered analysis. The system leverages Databricks as the primary data warehouse and compute engine for processing loan applications on a scale. For intelligent loan analysis, the application integrates the Groq AI API with the Llama 3.3 70B model to provide personalized improvement suggestions for rejected loan applicants, including CIBIL score improvement strategies, debt-to-income ratio recommendations, and optimal reapplication timelines.

Key features include secure user authentication with JWT tokens, realtime dashboard analytics with multiple visualization types (bar charts, pie charts, doughnut charts), searchable and sortable data grids, CSV file upload with automated Databricks job triggering and status polling, and AI-driven rejection analysis. The application demonstrates the practical application of modern web technologies, cloud-based data processing, and large language models in the financial technology domain.

Keywords: Loan Analytics, Artificial Intelligence, React.js, Fast API, Databricks, Groq AI, Dashboard, Financial Technology, Machine Learning, Data Visualization

1. Introduction

The financial sector has undergone a significant transformation with the adoption of digital technologies and artificial intelligence. Loan processing, which was traditionally a manual and time-consuming process, is now being revolutionized through automated systems that leverage data analytics and machine learning. The Loan Analytics and Planning Application is a comprehensive web-based system designed to streamline the loan analysis process by combining modern web technologies with AI-powered decision-making capabilities. This application serves as a bridge between raw loan application data and actionable insights, enabling financial analysts and loan officers to make informed decisions efficiently. The system provides an end-to-end solution that encompasses data upload,

automated processing, visual analytics, and AI-driven recommendations for loan applicants. The application follows a client-server architecture with a React.js-based single-page application (SPA) on the frontend and a Python Fast API server on the backend. The data processing pipeline leverages Databricks for scalable data warehousing and compute operations, while the Groq AI API with the Llama 3.3 70B model provides intelligent analysis and personalized recommendations for rejected loan applicants.

The primary objectives of this project are:

- To develop a user-friendly web application for uploading and processing loan application data through CSV files.
- To implement a comprehensive analytics dashboard with KPIs, interactive charts, and searchable data tables for loan portfolio analysis.
- To integrate an AI-powered analysis system that provides personalized improvement suggestions for rejected loan applicants, including CIBIL score improvement strategies and debt management recommendations.
- To leverage cloud-based data processing using Databricks for scalable and efficient loan data processing.

2. Literature Review

Several loan management and analytics systems currently exist in the financial technology domain:

1. Traditional Banking Loan Management Systems (LMS) Traditional banking systems such as those offered by FIS Global, Temenos, and Oracle Financial Services use monolithic architectures for loan origination and management. These systems handle the entire loan lifecycle from application to disbursement but typically lack advanced AI-powered analytics and real-time visualization capabilities. They are often expensive and require significant infrastructure investment.
2. Credit Scoring Platforms Platforms like CIBIL (TransUnion), Experian, and Equifax provide credit scoring services that financial institutions use to assess loan applicant creditworthiness. While these platforms are reliable for credit scoring, they do not provide comprehensive loan analytics dashboards or personalized improvement recommendations for applicants.
3. FinTech Lending Platforms Modern fintech companies such as Lending Club, Prosper, and Indian platforms like Lending kart and Capital Float use data-driven approaches for loan processing. These platforms incorporate some level of 7 automation and data analytics but typically operate as closed systems with proprietary algorithms.
4. Business Intelligence Tools like Tableau, Power BI, and Looker are used for financial data visualization and reporting. While powerful for analytics, they require significant configuration and do not provide integrated loan processing or AI-powered recommendation capabilities out of the box.

2.2 Related Work

Several research works have explored the intersection of artificial intelligence and loan processing: AI-Based Credit Risk Assessment: Research by Khandani et al. demonstrated the effectiveness of machine learning models in predicting consumer credit risk using transaction data. Their work showed that ensemble methods could significantly outperform traditional statistical models in predicting loan defaults. Deep Learning for Loan Default Prediction: Studies by researchers at various institutions have applied deep neural networks to loan default prediction, achieving high accuracy rates. These approaches consider multiple features including credit history, income, employment status, and demographic factors. Natural Language Processing in Finance: The advent of large language models (LLMs) has opened new possibilities for financial analysis. Models like GPT and Llama have been explored for generating financial reports, analyzing market trends, and providing personalized financial advice.

3.1 Feasibility Study

3.1.1 Technical Feasibility

The project is technically feasible as it utilizes well-established and widely adopted technologies:

1. React.js 19: A mature and stable JavaScript library for building user interfaces, backed by Meta with extensive community support and documentation.
2. Python Fast API: A modern, high-performance web framework for building APIs, with built-in support for asynchronous operations, automatic API documentation, and data validation.
3. Databricks: A unified analytics platform that provides reliable data warehousing and computing capabilities with SQL connector support.
4. Groq AI API: Provides high-speed inference for large language models with a simple REST API interface.
5. Chart.js and Ant Design Plots: Proven visualization libraries with extensive chart type support.

All technologies used in this project are open-source (except Databricks and Groq API services) and have comprehensive documentation, active communities, and regular updates.

3.1.2 Operational Feasibility

The application is designed with operational simplicity in mind:

1. The user interface follows modern design principles with intuitive navigation using a sidebar layout.
2. CSV file upload is a familiar process for loan officers who regularly work with spreadsheet data.
3. The dashboard presents data in easily understandable formats (KPI cards, charts, tables).
4. AI-generated recommendations are presented in plain language that both financial professionals and applicants can understand.
5. The system requires minimal training for end users due to its straightforward workflow.

3.1.3 Economic Feasibility

The project uses predominantly open-source technologies, significantly reducing development costs:

1. Frontend: React.js, Ant Design, Material UI, Chart.js – all open-source and free.
2. Backend: Python, Fast API, uvicorn – all open-source and free.
3. Cloud Services: Databricks and Groq API offer free tiers suitable for development and small-scale deployment.
4. Development Tools: VS Code (free), Git (free), npm (free), pip (free).

The primary costs are limited to cloud service usage beyond free tiers and development effort.

3.2 Requirement Analysis

3.2.1 Functional Requirements

Req. ID	Requirement Description
FR-01	The system shall allow new users to register with name, email, and password
FR-02	The system shall authenticate users using email and password and issue JWT tokens
FR-03	The system shall restrict access to dashboard and planner pages to authenticated users only
FR-04	The system displays aggregate KPIs: total loans, approved count, rejected count, average income, and average credit score
FR-05	The system shall render interactive bar charts showing loan type by approval/rejection status
FR-06	The system shall render pie charts showing approved vs. rejected loan distribution
FR-07	The system shall render doughnut charts showing loan type distribution
FR-08	The system shall display a searchable, sortable data grid of all loan records
FR-09	The system shall allow users to upload CSV files containing loan application data
FR-10	The system shall transfer uploaded CSV files to Databricks Volumes for storage
FR-11	The system shall trigger Databricks processing jobs upon CSV upload
FR-12	The system shall poll and display Databricks job status in real-time
FR-13	The system shall provide AI-powered analysis for individual rejected loan applications
FR-14	The system shall display AI-generated improvement suggestions including CIBIL score tips and reapplication timelines

3.2.2 Non-Functional Requirements

Req. ID	Requirement Description
NFR-01	The system shall respond to API requests within 3 seconds under normal load
NFR-02	The system shall support concurrent users without degradation
NFR-03	The system shall use password hashing (bcrypt) for storing user credentials
NFR-04	The system shall use JWT tokens with expiration for session management
NFR-05	The system shall provide a responsive UI that works on desktop and tablet screens
NFR-06	The system should handle CSV files up to 10 MB in size

Req. ID	Requirement Description
NFR-07	The system shall be maintainable with modular code organization

3.3 Hardware and Software Requirements

Table 3.1: Hardware Requirements

Component	Minimum Requirement
Processor	Intel Core i5 or equivalent (64-bit)
RAM	8 GB
Hard Disk	256 GB SSD
Internet	Broadband connection (for Databricks and Groq API access)
Display	1366 x 768 resolution or higher

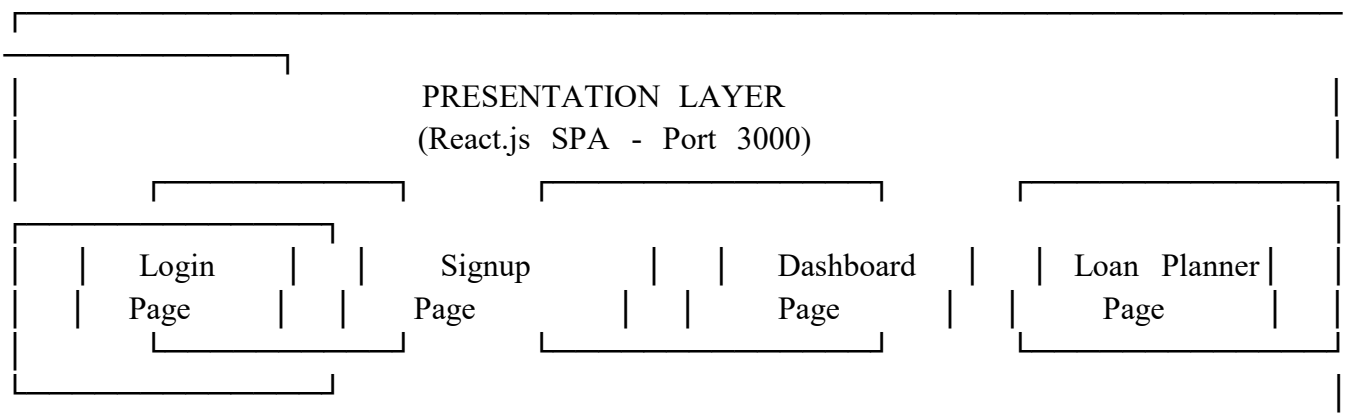
Table 3.2: Software Requirements

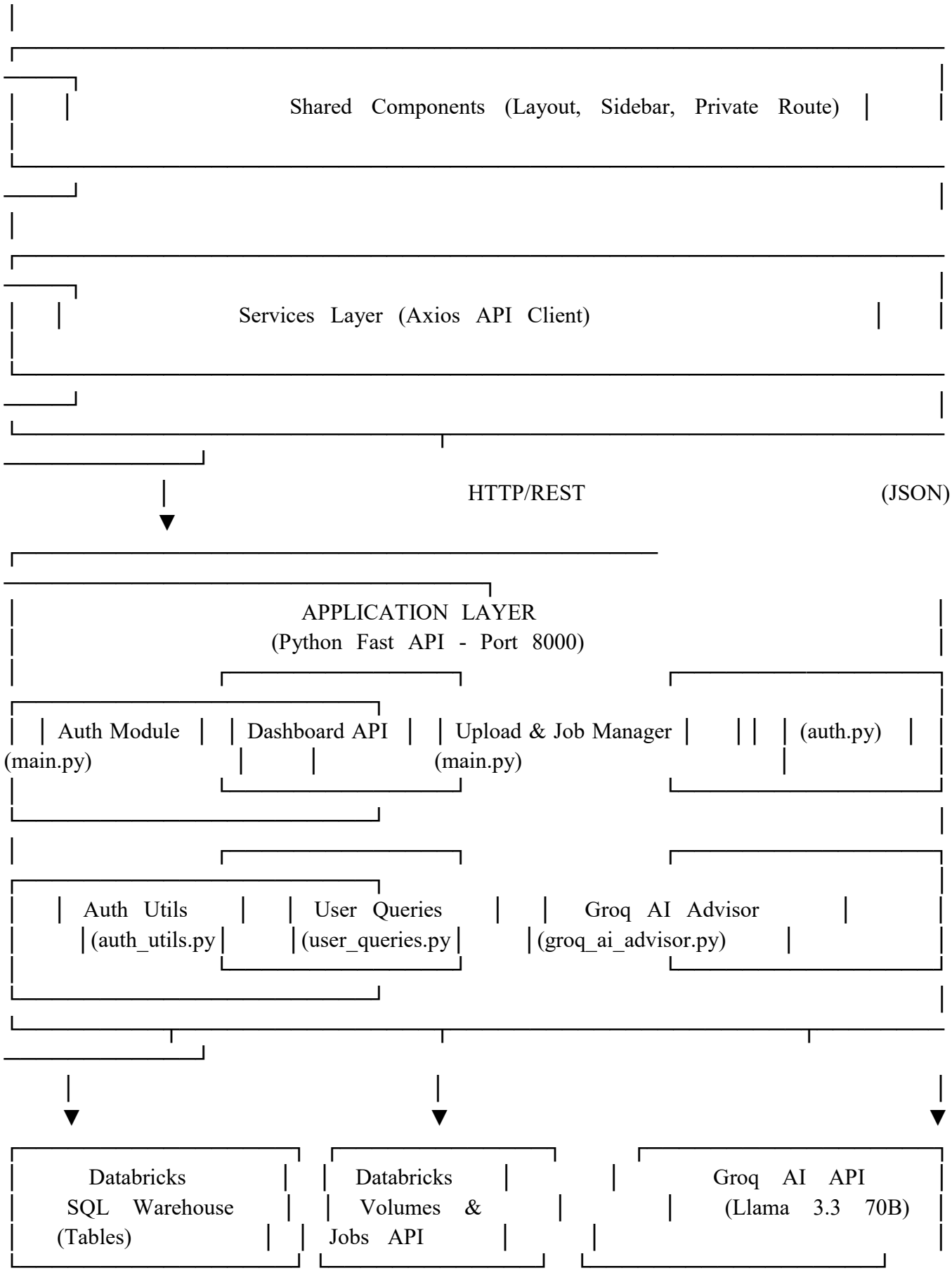
Component	Requirement
Operating System	Windows 10/11, macOS 12+, or Ubuntu 20.04+
Node.js	v18.0 or higher
npm	v9.0 or higher
Python	3.9 or higher
Web Browser	Google Chrome 100+, Firefox 100+, or Edge 100+
IDE	Visual Studio Code (recommended)
Version Control	Git 2.30+
Backend Framework	Fast API 0.100+
Frontend Framework	React.js 19
Database	Databricks SQL Warehouse
AI Service	Groq API (Llama 3.3 70B)

4.1 System Architecture

The Loan Analytics and Planning Application follows a three-tier client-server architecture consisting of the Presentation Layer (Frontend), Application Layer (Backend), and Data Layer (Databricks + AI Service).

Figure 4.1: System Architecture Diagram




Shared Components (Layout, Sidebar, Private Route)
Services Layer (Axios API Client)
HTTP/REST
(JSON)
APPLICATION LAYER
(Python Fast API - Port 8000)
Auth Module
(main.py)
Dashboard API
Upload & Job Manager
(main.py)
(auth.py)
Auth Utils
(auth_utils.py)
User Queries
(user_queries.py)
Groq AI Advisor
(groq_ai_advisor.py)
Databricks
SQL Warehouse
(Tables)
Databricks
Volumes &
Jobs API
Groq AI API
(Llama 3.3 70B)

Architecture Description:

Presentation Layer: The React.js SPA handles all user interactions, routing, and rendering. It communicates with the backend exclusively through RESTful API calls using Axios.

Application Layer: The Fast API backend serves as the middleware between the frontend and data services. It handles authentication, data aggregation, file management, and AI service integration.

Data Layer: Databricks provides SQL warehouse capabilities for storing and querying loan data, Volumes for file storage, and Jobs API for compute operations. The Groq AI API provides LLM-based analysis capabilities.

4.2 Module Description

The application is organized into the following modules:

Table 4.1: Module Description

Module	Description	Key Files
Authentication Module	Handles user registration, login, password hashing using bcrypt, JWT token generation and validation	auth.py, auth_utils.py, Login.jsx, Signup.jsx, PrivateRoute.jsx
Dashboard Module	Provides analytics dashboard with KPI cards, interactive charts (bar, pie, doughnut), and data tables with search/sort/filter	Dashboard.jsx, Dashboard.css, main.py (dashboard endpoints)
Loan Planner Module	Enables CSV file upload, file transfer to Databricks Volumes, job triggering, and real-time status polling	LoanPlanner.jsx, main.py (upload endpoints)
AI Analysis Module	Integrates Groq AI for personalized loan rejection analysis and improvement recommendations	groq_ai_advisor.py, fetch_ai_insights.py
Navigation Module	Provides sidebar navigation, layout management, and user profile/logout functionality	Layout.jsx, Sidebar.jsx, Sidebar.css
Data Service Module	Manages API communication between frontend and backend using Axios	api.js, authService.js
Database Module	Handles Databricks SQL connections, user CRUD operations, and data queries	databricks_connection.py, user_queries.py

Detailed Module Descriptions:

1. Authentication Module

The authentication module provides secure user management through: - User Registration: New users provide name, email, and password. The password is hashed using bcrypt before storage in the Databricks user's table. A UUID is generated for each user. - User Login: Credentials are validated against the database. Upon successful authentication, a JWT token is generated with an expiration time and returned to the client. - Route Protection: The Private Route component checks for a valid token in local Storage before rendering protected pages.

2. Dashboard Module

The dashboard module consists of: - KPI Cards: Display aggregate statistics — total loans, approved count, rejected count, average monthly income, and average credit score. - Bar Chart: Visualizes loan applications grouped by loan type (e.g., personal, home, auto) with approval/rejection status using Chart.js. - Pie Chart: Shows the overall distribution of approved vs. rejected applications. - Doughnut Chart: Displays the distribution of applications across different loan types. - Data Grid: A Material UI DataGrid component displaying all loan records with search, sort, and filter capabilities. - Rejection Reasons Table: Lists rejected applications with reasons for rejection.

3. Loan Planner Module

The loan planner module enables: - CSV Upload: Users select a CSV file through a file input. The file is sent to the backend via a multipart form-data POST request. - Databricks Volume Transfer: The backend uses the Databricks API to upload the file to

/Volumes/workspace/loan_analytics/csv_data/. - Job Triggering: After successful upload, the backend triggers Databricks Job ID 87388874042203 to process the uploaded data. - Status Polling: The frontend polls the /job-status/{run_id} endpoint every 3 seconds to track processing progress.

4. AI Analysis Module

The AI analysis module provides: - Per-User Analysis: When a user clicks "View AI" on a loan record, the system retrieves that applicant's latest loan data and sends it to the Groq AI API. - LLM Prompt Engineering: The system constructs a detailed prompt including loan amount, income, credit score, loan type, and rejection reason, requesting structured improvement advice. - Response Parsing: The AI response is parsed and displayed in a dialog showing CIBIL improvement tips, debt-to-income ratio advice, recommended waiting period, and actionable steps.

5.1 Frontend Implementation

5.1.1 Project Structure

The front end is built using React.js 19, initialized with Create React App. The project follows a standard React application structure:

```
src/
├── App.js                    # Main application with route definitions
├── index.js                  # React entry point (renders App)
├── components/               # Reusable UI components
```

```
│   ├── Layout.jsx          # Page layout wrapper (sidebar + content)
│   ├── PrivateRoute.jsx    # Authentication guard component
│   └── Sidebar.jsx         # Navigation sidebar component
├── context/
│   └── AuthContext.jsx     # Authentication context provider
├── pages/                  # Page-level components
│   ├── Login.jsx           # Login page
│   ├── Signup.jsx          # Registration page
│   ├── Dashboard.jsx       # Analytics dashboard
│   └── LoanPlanner.jsx     # CSV upload and processing
├── services/               # API communication layer
│   ├── api.js              # Axios instance and API functions
│   └── authService.js      # Authentication API calls
└── styles/                 # CSS stylesheets
    ├── Global.css          # Global and form styles
    ├── Dashboard.css       # Dashboard-specific styles
    └── Sidebar.css         # Sidebar and layout styles
```

Key dependencies include: - antd (v6): Ant Design component library for forms, buttons, cards, modals - @mui/material (v7): Material UI for data grid and additional components - chart.js & react-chartjs-2: Chart rendering for bar, pie, and doughnut charts - @ant-design/plots: Ant Design charting components - react-router-dom (v7): Client-side routing - axios: HTTP client for API communication - react-toastify: Toast notification system

5.1.2 Authentication Module

Login Page (Login.jsx)

The login page uses Ant Design's Form component with email and password fields. Upon submission, it calls the /login endpoint and stores the returned JWT token in localStorage.

```
// Login.jsx - Key Implementation
const handleLogin = async (values) => {
  setLoading(true);
  try {
    const res = await loginUser(values);
    localStorage.setItem("token", res.data.access_token);
    toast.success("Login successful");
    navigate("/");
  } catch (err) {
    toast.error("Invalid credentials");
  } finally {
    setLoading(false);
  }
};
```

Signup Page (Signup.jsx)

The signup page collects name, email, and password with confirmation. It calls the /signup endpoint and redirects to the login page upon successful registration.

```
// Signup.jsx - Key Implementation
const handleSignup = async (values) => {
  setLoading(true);
  try {
    await signupUser({
      name: values.name,
      email: values.email,
      password: values.password,
    });
    toast.success("Account created successfully");
    navigate("/login");
  } catch (err) {
    toast.error("Signup failed");
  } finally {
    setLoading(false);
  }
};
```

Private Route (PrivateRoute.jsx)

The PrivateRoute component protects authenticated routes by checking for a token in localStorage:

```
// PrivateRoute.jsx
const PrivateRoute = ({ children }) => {
  const token = localStorage.getItem("token");
  return token ? children : <Navigate to="/login" />;
};
```

5.1.3 Dashboard Module

Dashboard Page (Dashboard.jsx)

The dashboard is the primary analytics interface, divided into several sections:

KPI Cards Section:

```
// Dashboard.jsx - KPI Fetching
useEffect(() => {
  const fetchStats = async () => {
    try {
      const res = await axios.get("http://127.0.0.1:8000/dashboard-stats");
      setStats(res.data);
    } catch (err) {
```

```
      console.error("Error fetching stats:", err);
    }
  };
  fetchStats();
}, []);
```

The KPI section displays five key metrics in Ant Design Card components: - Total Loan Applications - Approved Applications - Rejected Applications - Average Monthly Income - Average Credit Score

Charts Section:

The dashboard renders three types of charts:

Bar Chart – Loan applications grouped by loan type with approval/rejection status:

```
// Bar chart configuration
const barData = {
  labels: loanTypes,
  datasets: [
    {
      label: "Approved",
      data: approvedByType,
      backgroundColor: "rgba(75, 192, 192, 0.6)",
    },
    {
      label: "Rejected",
      data: rejectedByType,
      backgroundColor: "rgba(255, 99, 132, 0.6)",
    },
  ],
};
```

Pie Chart – Overall approved vs. rejected distribution

Doughnut Chart – Distribution across loan types

Data Grid Section:

The loan data is displayed in a Material UI DataGrid with the following features: - Column sorting (ascending/descending) - Text search across all fields - Pagination - “View AI” action button per row for triggering AI analysis

AI Analysis Dialog:

When a user clicks “View AI” on a rejected loan, a dialog opens displaying AI-generated recommendations:

```
// AI Analysis trigger
const handleViewAI = async (userId) => {
```

```
 try {
   const res = await axios.get(
     `http://127.0.0.1:8000/user-ai-analysis/${userId}`
   );
   setAiResult(res.data);
   setDialogOpen(true);
 } catch (err) {
   toast.error("Failed to fetch AI analysis");
 }
};
```

5.1.4 Loan Planner Module

CSV Upload Page (LoanPlanner.jsx)

The Loan Planner page provides CSV file upload functionality with the following workflow:
File Selection: Users select a CSV file using a file input component.
Upload: The file is sent to the backend via a multipart form-data POST request.
Job Monitoring: After uploading, the frontend polls the job status every 3 seconds.

```
// LoanPlanner.jsx - Upload and Job Monitoring
const handleUpload = async () => {
 const formData = new FormData();
 formData.append("file", file);

 try {
   const res = await axios.post("http://127.0.0.1:8000/upload", formData);
   setRunId(res.data.run_id);
   toast.success("File uploaded! Processing started.");

   // Start polling job status
   const interval = setInterval(async () => {
     const statusRes = await axios.get(
       `http://127.0.0.1:8000/job-status/${res.data.run_id}`
     );
     setJobStatus(statusRes.data.status);

     if (statusRes.data.status === "TERMINATED" ||
         statusRes.data.status === "SUCCESS") {
       clearInterval(interval);
       toast.success("Processing completed!");
     }
   }, 3000);
 } catch (err) {
   toast.error("Upload failed");
```

```
  }
};
```

5.1.5 Navigation and Layout

Sidebar Component (Sidebar.jsx)

The sidebar provides persistent navigation with: - Application logo/branding - Navigation links: Dashboard (home icon) and Upload CSV (upload icon) - Active route highlighting - User profile section at the bottom - Logout functionality that clears the token and redirects to login

Layout Component (Layout.jsx)

The Layout component wraps page content with the sidebar:

```
// Layout.jsx
const Layout = ({ children }) => {
  return (
<div className="layout-container">
<Sidebar />
<div className="main-content">{children}</div>
</div>
  );
};
```

5.2 Backend Implementation

5.2.1 FastAPI Application Setup

The backend is built with Python FastAPI with CORS middleware enabled:

```
# main.py - Application Setup
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

Table 5.1: API Endpoints Summary

Method	Endpoint	Description
POST	/signup	Create a new user account
POST	/login	Authenticate user and return JWT token
GET	/dashboard-stats	Get aggregate KPI statistics
GET	/all-loans	Get all loan decision records (limit 1000)
GET	/loan/{user_id}	Get loan data for a specific user
GET	/loans-by-type/{loan_type}	Get loans filtered by type
GET	/users	Get list of distinct user IDs
GET	/user-ai-analysis/{user_id}	Get AI analysis for a user's latest loan
GET	/loan-ai-insights	Get pre-computed AI insights by email
POST	/upload	Upload CSV file to Databricks and trigger job
GET	/job-status/{run_id}	Get Databricks job run status
GET	/	Health check endpoint

7.1 Conclusion

The Loan Analytics and Planning Application has been successfully designed, developed, and tested as a comprehensive full-stack web application for loan data analysis and AI-powered decision support. The project demonstrates the practical integration of modern web technologies, cloud-based data processing, and large language models in the financial technology domain.

The key accomplishments of this project include:

Functional Web Application: A complete full-stack application with React.js frontend and Python Fast API backend, providing an intuitive and responsive user interface.

Comprehensive Analytics Dashboard: An interactive dashboard with KPI cards, multiple chart types (bar, pie, doughnut), and a searchable data grid that enables efficient loan portfolio analysis.

Automated Data Processing Pipeline: Integration with Databricks for scalable CSV data processing, file storage through Volumes, and compute operations through Jobs API.

AI-Powered Analysis: Integration with Groq AI (Llama 3.3 70B) to provide personalized improvement recommendations for rejected loan applicants, adding significant value beyond traditional loan management systems.

Secure Authentication: Implementation of JWT-based authentication with bcrypt password hashing for secure user management.

Real-Time Monitoring: Job status polling mechanism that provides real-time feedback on data processing pipeline status.

The application addresses the identified gaps in existing systems by combining data upload, automated processing, visual analytics, and AI-driven recommendations into a single, unified platform.

7.2 Future Enhancements

The following improvements are proposed for future development:

Role-Based Access Control (RBAC): Implement different user roles (admin, analyst, viewer) with varying levels of access to features and data.

Email Notifications: Add email notification capabilities for loan decision updates and AI analysis reports.

Advanced Filtering and Reporting: Implement date range filters, custom report generation, and PDF/Excel export capabilities for dashboard data.

Real-Time Data Streaming: Replace polling-based job monitoring with WebSocket connections for real-time data streaming.

Mobile Responsive Design: Enhance the UI to fully support mobile devices and smaller screens.

Multi-Language Support: Add internationalization (i18n) support for serving users in different languages.

Predictive Analytics: Integrate machine learning models for predicting loan approval likelihood before submission.

Audit Trail: Implement comprehensive logging and audit trail for all user actions and system events.

API Rate Limiting and Security Hardening: Add rate limiting, input sanitization, parameterized queries, and backend JWT middleware for production readiness.

Automated Testing Suite: Develop comprehensive unit and integration test suites using Jest, React Testing Library, and pytest.

CI/CD Pipeline: Set up continuous integration and deployment pipelines for automated testing and deployment.

Containerization: Dockerize both frontend and backend for consistent deployment across environments.

REFERENCES

1. Abadi, M. et al. (2016). "TensorFlow: A System for Large-Scale Machine Learning." Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp. 265–283.
2. Ant Design (2024). "Ant Design – A UI Design Language and React UI Library." Available at: https://ant.design/
3. Chart.js (2024). "Chart.js – Simple yet Flexible JavaScript Charting." Available at: https://www.chartjs.org/
4. Databricks (2024). "Databricks Documentation – SQL Warehouses and REST API." Available at: https://docs.databricks.com/
5. Facebook Inc. (2024). "React – A JavaScript Library for Building User Interfaces." Available at: https://react.dev/
6. Fast API (2024). "Fast API – Modern, Fast Web Framework for Building APIs with Python." Available at: https://fastapi.tiangolo.com/
7. Groq Inc. (2024). "Groq API Documentation – High-Speed LLM Inference." Available at: https://groq.com/
8. Khandani, A. E., Kim, A. J. and Lo, A. W. (2010). "Consumer Credit-Risk Models via Machine-Learning Algorithms." Journal of Banking & Finance, Vol. 34, No. 11, pp. 2767–2787.
9. Material UI (2024). "MUI – The React Component Library." Available at: https://mui.com/
10. Meta AI (2024). "Llama 3.3: Open Foundation and Fine-Tuned Chat Models." Meta Platforms Inc.
11. Node.js Foundation (2024). "Node.js – JavaScript Runtime." Available at: https://nodejs.org/
12. Python Software Foundation (2024). "Python Programming Language." Available at: https://www.python.org/
13. React Router (2024). "React Router – Declarative Routing for React." Available at: https://reactrouter.com/
14. Touvron, H. et al. (2023). "Llama 2: Open Foundation and Fine-Tuned Chat Models." arXiv preprint arXiv:2307.09288.
15. TransUnion CIBIL (2024). "Understanding Your CIBIL Score." Available at: https://www.cibil.com/