Project: Loan Management System (LMS)

1. Introduction

The Loan Management System (LMS) is a financial information system that allows users to apply for loans. This system is designed to be implemented using ASP.NET Core Web API and Express.js, with different features allocated to each technology.

2. System Architecture

The LMS follows a **three-tier architecture**:

- **Presentation Layer**: Web-based UI
- **Business Logic Layer**: RESTful APIs implemented using both ASP.NET Core and Express.js
- **Data Layer**: SQL-based relational/NoSQL database

3. Functional Requirements

3.1 User Management (Implemented in ASP.NET Core)

- User Registration and Authentication (JWT-based authorization)
- Role-based access control (Admin, Borrower, Loan Officer)

3.2 Loan Management (Implemented in Express.js)

- Loan application submission
- Loan approval/rejection by Loan Officer

4. API Endpoints

4.1 Authentication & User Management (ASP.NET Core)

Method	Endpoint	Description
POST	/api/auth/register	Register new user
POST	/api/auth/login	User login
GET	/api/users	List all users (Admin)

4.2 Loan Management (Express.js)

Method	Endpoint	Description
POST	/api/loans/apply	Apply for a loan
GET	/api/loans/{id}	Get loan details
PUT	/api/loans/{id}/approve	Approve loan (Loan Officer)
PUT	/api/loans/{id}/reject	Reject loan (Loan Officer)

5. Database Design

5.1 Users Table

Column	Type	Constraints
id	UUID	PRIMARY KEY
name	VARCHAR(100)	NOT NULL
email	VARCHAR(255)	UNIQUE, NOT NULL
password	VARCHAR(255)	NOT NULL
role	ENUM	['Admin', 'Borrower', 'Loan Officer']

5.2 Loans Table

Column	Type	Constraints
id	UUID	PRIMARY KEY
user_id	UUID	FOREIGN KEY -> Users(id)
amount	DECIMAL(10,2)	NOT NULL
status	ENUM	['Pending', 'Approved', 'Rejected', 'Disbursed']
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

6. Technology Stack

- **Back-end**: ASP.NET Core Web API (User Management) and Express.js (Loan Management)
- **Database**: SQL Server/MongoDB
- **Authentication**: JWT-based authentication

- **Hosting**: Azure

7. Implementation Plan

Phase 1: Project Setup and Architecture (10 Hours)

- Project Overview and Goals (2 Hours)
- Choosing the Right Architecture (2 Hours)
- Setting Up Development Environment (2 Hours)
- Project Folder Structure (2 Hours)
- Version Control Setup (2 Hours)

Phase 2: Development Phase (16 Hours)

- Backend Development with .NET Core (User Management) (8 Hours)
- Backend Development with Node.js (Loan Management) (8 Hours)

Phase 3: Frontend Integration (4 Hours)

- Designing User Interface (2 Hours)
- Connecting Frontend to Backend (2 Hours)

Phase 4: Testing and Quality Assurance (4 Hours)

- Writing Unit Tests (2 Hours)
- Performing Integration Testing (2 Hours)

Phase 5: Deployment and Monitoring (4 Hours)

- Dockerizing Applications (2 Hours)
- Deploying to Azure (2 Hours)

Phase 6: Project Presentation and Assessment (4 Hours)

- Preparing Presentation (2 Hours)
- Conducting Assessment (2 Hours)

8. Security Considerations

- Encrypt sensitive data (passwords)
- Implement role-based access control (RBAC)
- Use HTTPS and secure headers for API communication

9. Conclusion

This LLD provides a structured approach to developing a Loan Management System that is implementable using both ASP.NET Core Web API and Express.js, with distinct feature allocation to each technology.