Project Abstract: PayEasy - A Paytm-Like Payment App Using Spring Boot Microservices

The objective of this project is to design and develop **PayEasy**, a digital payment platform similar to Paytm, leveraging the power of **Spring Boot Microservices** architecture. The application will enable users to perform seamless mobile-based financial transactions, including money transfers, bill payments, recharges, and online shopping, all within a secure and scalable ecosystem.

Key Features:

- **User Authentication & Authorization**: Secure login and account creation with multi-factor authentication (MFA) to safeguard user data and transactions.
- **Wallet Management**: Users can load money into their digital wallets, view wallet balances, and transfer funds between PayEasy users.
- **Bill Payments & Recharges**: Payment for utilities like electricity, water, phone, and broadband services, as well as mobile/DTH recharges.
- **Bank Account Integration**: The app will allow users to link their bank accounts for direct money transfers or UPI-based payments.
- **E-Commerce**: Integration with merchants for online shopping, where users can browse products and make payments via the app.
- **Transaction History**: Users can track all past transactions, ensuring transparency and trust.

Architecture:

The application is built using **Spring Boot Microservices** for modularity, scalability, and maintainability. Each microservice performs a specific task:

- **User Service**: Handles user registration, profile management, and authentication.
- **Payment Service**: Manages wallet functionalities, including balance management, and fund transfers.
- **Billing Service**: Manages bill payments, recharges, and other payment gateways.
- **Merchant Service**: Facilitates merchant registration and integration for e-commerce transactions.
- **Notification Service**: Sends notifications, updates, and transaction confirmations via email/SMS.
- **API Gateway**: Acts as a single entry point for client requests, routing them to the appropriate microservices.
- **Database Services**: Ensures efficient data storage and retrieval with the use of relational (MySQL) and non-relational (MongoDB) databases for different types of data.

Technology Stack:

- **Spring Boot 3.3.2** for microservice architecture and REST APIs.
- **Spring Security** for authentication and authorization.
- **MySQL/MongoDB** for relational and non-relational data storage.
- **Spring Cloud** for service discovery, load balancing, and resilience.

User Stories

1. User Authentication & Authorization

- US101: As a new user, I want to register using my mobile number and email, so that I can create an account on PayEasy.
- US102: As a registered user, I want to log in securely using OTP and password, so that my account is protected from unauthorized access.
- US103: As a user, I want to enable multi-factor authentication, so that my transactions remain secure even if my password is compromised.

---

2. Wallet Management

- US201: As a user, I want to add money to my PayEasy wallet using my linked bank account, so that I can make quick payments.
- US202: As a user, I want to check my current wallet balance, so that I know how much I can spend.
- US203: As a user, I want to transfer money to other PayEasy users, so that I can send money to friends or family easily.

---

3. Bill Payments & Recharges

- US301: As a user, I want to pay my electricity and water bills, so that I don’t miss payment deadlines.
- US302: As a user, I want to recharge my mobile or DTH service, so that I can continue using them without interruption.
- US303: As a user, I want to save frequent billers, so that future payments become faster.

---

4. Bank Account Integration

- US401: As a user, I want to link my bank account using UPI, so that I can make direct payments from my account.
- US402: As a user, I want to view linked bank account details (last 4 digits only), so that I can confirm which account I'm using.

---

5. E-Commerce Integration

- US501: As a user, I want to browse and search merchant products, so that I can shop through the app.

- US502: As a user, I want to pay merchants using my wallet or linked account, so that I have flexibility in payment.
- US503: As a merchant, I want to register and list products, so that users can buy from me via PayEasy.

---

6. Transaction History

- US601: As a user, I want to view my complete transaction history, so that I can track all my spending and payments.
- US602: As a user, I want to filter transactions by date or type, so that I can find specific transactions easily.

---

7. Notifications

- US701: As a user, I want to receive an SMS/email notification after each transaction, so that I have real-time updates on my payments.
- US702: As a user, I want to receive notifications for low balance and failed payments, so that I can take timely action.

---

8. Technical Stories (For Developers/DevOps)

- US801: As a developer, I want to implement API Gateway routing, so that client requests are directed to appropriate microservices.
- US802: As a developer, I want to enable service discovery using Spring Cloud, so that all services can locate each other dynamically.
- US803: As a DevOps engineer, I want to use centralized config management, so that all services get consistent configuration.

◆

Frontend Layer (OJET)

- A single-page application (SPA) built with **Oracle JET components** (tables, charts, forms, navigation).
- Communicates with the backend via **REST APIs exposed by the API Gateway**.
- Secure login UI integrated with JWT tokens from Spring Security.
- Rich visualization for **wallet balance, transaction history, bills, and shopping cart**.

Backend Layer (Spring Boot Microservices)

- Same as your definition: User, Payment, Billing, Merchant, Notification services.
- Spring Cloud (Eureka/Consul) for service discovery, Config Server, Resilience4j for fault tolerance.

Database Layer

- MySQL → Structured data (users, merchants, transactions, bills).
- MongoDB → Unstructured data (notifications, logs, product catalog).

◆ OJET Integration Points (UI Requirements)

Here's how OJET will fit for each feature:

1. User Authentication & Authorization

- **OJET Screens**:
  - Login/Signup form (oj-input-text, oj-button, oj-form-layout).
  - MFA with OTP input.
- **Backend APIs**: Auth Service → `/auth/register`, `/auth/login`, `/auth/mfa/verify`.

---

2. Wallet Management

- **OJET Screens**:
  - Wallet dashboard with **oj-chart** (donut/pie) showing balance distribution.
  - Transfer money form (oj-input-number, oj-combobox for PayEasy user selection).
- **Backend APIs**: Payment Service → `/wallet/add`, `/wallet/balance`, `/wallet/transfer`.

---

3. Bill Payments & Recharges

- **OJET Screens**:
  - Billers list with **oj-table**.
  - Payment/recharge wizard with step-by-step forms.
  - Frequently saved billers displayed in a card layout.
- **Backend APIs**: Billing Service → `/bills/pay`, `/bills/save`, `/bills/list`.

---

4. Bank Account Integration

- **OJET Screens**:
  - Bank linking form with masked account display (`****1234`).
  - Account overview card component.
- **Backend APIs**: Bank Service → `/bank/link`, `/bank/list`.

---

5. E-Commerce Integration

- **OJET Screens**:
  - Product catalog grid with **oj-list-view / oj-data-grid**.
  - Cart and checkout page with wallet/bank payment options.
- **Backend APIs**: Merchant Service → `/merchant/register`, `/merchant/products`, `/merchant/order`.

---

6. Transaction History

- **OJET Screens**:
  - Filterable **oj-table** with transaction history.
  - Date-range picker and type filter.
  - oj-chart line graph for spending trends.
- **Backend APIs**: Payment Service → `/transactions/history`, `/transactions/filter`.

---

7. Notifications

- **OJET Screens**:
  - Notifications bell icon with dropdown (`oj-messages`).
  - Low balance alert banners.
- **Backend APIs**: Notification Service → `/notifications/recent`.

◆ Extra Technical Stories (Frontend – OJET)

- **US901**: As a user, I want a responsive dashboard in OJET, so that I can view my balance and recent transactions in one place.
- **US902**: As a user, I want charts for spending trends, so that I can visualize where my money goes.
- **US903**: As a developer, I want OJET to consume all APIs through the Spring Cloud API Gateway, so that routing and security remain consistent.
- **US904**: As a developer, I want OJET role-based access (user vs merchant), so that UI elements are shown/hidden accordingly.