Smart FD

Secure Today, Stronger Tomorrow

Problem Statement :

The Digital Fixed Deposit System aims to provide a secure platform for customers to digitally create and manage Fixed Deposits (FDs). The system allows users to register/login, open FD accounts, view FDs with interest and maturity details, break FDs prematurely with penalties, and raise support tickets for FD-related queries.

The project is divided into five modules:

1. User Registration & Authentication: Secure registration/login with token-based authentication, user profile management.

2. Book & View Fixed Deposits: Book FDs with predefined interest/tenure, auto-calculate maturity and accrued interest, view FD list.

3. Interest Calculation & Status Management: Compute FD interest, update FD status (ACTIVE, MATURED, BROKEN), and provide interest details.

4. Premature Withdrawal & Penalty: Handle early FD break with revised payout computation (interest loss/penalty).

5. Support Ticket System: Allow users to raise and track FD-related queries via tickets.

Team :

Sarthak Kurothe

Shaik Mohammad Anees

Lakshmi Mrudula

Srijith

Rithika

1. Project Overview

Goal :

The goal of **Smart FD** is to provide a secure, transparent, and user-friendly digital platform for managing Fixed Deposits. By enabling users to register, book, monitor, and break deposits with real-time interest and penalty calculations, the system ensures financial safety today while empowering customers to build stronger financial growth for tomorrow.

Technologies Used :

- **Backend :**
  - SpringBoot
- **Frontend :**
  - Vue Js
- **Database :**
  - PostgreSQL
- **Testing :**
  - Junit and Mockito in Backend
  - Vite test in Frontend

2. System Architecture

Backend (Spring Boot) :

The backend of Smart FD is developed using Spring Boot and follows a layered architecture consisting of REST controllers, services, repositories, and entity models. This ensures clean separation of concerns, maintainability, and scalability.

- **REST API Endpoints :** The system exposes RESTful APIs to handle authentication, fixed deposit management, support ticket operations, and administrative tasks.

- **AuthController (/auth)**
  - **POST /auth/register** (Register a new user)
  - **POST /auth/login** (Authenticate user and issue token)
  - **GET /auth/me** (Fetch logged-in user profile)
- **FixedDepositController (`/fd`)**
  - **POST /fd/book** (Book a new FD)
  - **GET /fd/user/{userId}** (View FDs of a user)
  - **POST /fd/{fdId}/break** (Break FD prematurely)
  - **GET /fd/{fdId}/break-preview** (Preview payout and penalty for premature break)
  - **PUT /fd/{Id}/status** (Update FD status (ACTIVE, MATURED, BROKEN))
- **SupportTicketController (/support)**
  - **POST /support** (Create a support ticket)
  - **GET /support/user/{userId}** (Retrieve tickets by user)
- **AdminController (/admin)**
  - **GET /admin/fds** (View all FDs (admin dashboard))
  - **PUT /admin/fd/{id}** (Update FD status)
  - **GET /admin/tickets** (View all support tickets)
  - **POST /admin/tickets/{id}** (Update ticket status/response)

o **Data Models (Entities) :** The system uses JPA entities to model core concepts like **Users, Fixed Deposits, and Support Tickets**.

- **User.java**
  Represents a customer or admin in the system with attributes like user_id, name, email, password, age, role, and created_at.
- **FixedDeposit.java**
  Represents a booked FD, including details such as amount, interest_rate, tenure_months, start_date, maturity_date, status, and accrued_interest.
- **SupportTicket.java**
  Represents customer queries/issues linked to a specific FD or user. Contains fields like subject, description, status, response, and createdDate.

Each entity is mapped with **JPA annotations** (@Entity, @Table, @Id, @ManyToOne) and supports relational mappings like **User -> FixedDeposits** and **User ->SupportTickets**.

- **Services:** The service layer encapsulates the business logic of the application.

  - **UserService:** Manages user registration, authentication, and profile retrieval.
  - **UserDetailsService:** Provides user details for Spring Security integration.
  - **FixedDepositService:** Handles FD booking, maturity date/interest calculation, premature withdrawal, and status updates.
  - **AccruedInterestService:** Computes accrued interest for FDs.
  - **SupportTicketService**: Manages ticket creation, retrieval, and response handling.

- **Repositories:** The repository layer provides database interaction using Spring Data JPA.

  - **UserRepository.java:** CRUD operations for users.

- **FixedDepositRepository.java**: Query and persistence operations for fixed deposits.
- **SupportTicketRepository.java:** Query and persistence operations for support tickets.

Frontend (Vue.js):

The frontend of **Smart FD** is developed using **Vue.js** with **Vue Router** for navigation and **Vuex** for state management. The UI is component-based, ensuring modularity, reusability, and maintainability. API integration is handled using **Axios** for secure communication with the Spring Boot backend.

o **Components:** The application is organized into reusable UI components for user-facing and admin-facing functionalities:

- **Core UI Components:**
    - **Navbar.vue, Sidebar.vue, Footer.vue, Header.vue, LoadingSpinner.vue** (Base layout and navigation elements.)
    - **FeatureCard.vue, FeaturesSection.vue, HeroSection.vue, SecurityCard.vue, SecuritySection.vue, TrustIndicator.vue** (Landing page and trust-building sections.)
- **FD & User Features:**
    - **FDCard.vue** (Displays individual fixed deposit details.)
    - **SchemeDropdown.vue** (Dropdown for selecting FD schemes.)
- **Charts & Dashboard Visuals:**

    - **BarChart.vue, PieChart.vue, ChartDonut.vue, TicketsPieChart.vue, DashboardCards.vue** (Data visualization for admin/user dashboards.)

- **Admin-Specific Components:**
    - **AdminInterface.vue, AdminSidebar.vue** (Base layout for admin dashboard.)

o Routing : Routing is managed via Vue Router (router/index.js), with role-based navigation and route guards:

- **Public Routes:**
    - **/ ->HomeView**
    - **/login -> Login**
    - **/register -> Register**
- **User Routes (/user)**
    - **/dashboard -> Dashboard**
    - **/my-fds -> MyFDs**
    - **/book-fd -> BookFD**
    - **/calculator -> Calculator**
    - **/support -> Support**
- **Admin Routes (/admin)**
    - **/fds -> AdminFixedDeposits**
    - **/tickets -> AdminSupportTickets**
    - **/ -> Admin** (Dashboard overview)

- **Guards & Redirects:**
    - **requiresAuth ->** Ensures only logged-in users can access protected routes.
    - **requiresAdmin ->** Restricts certain routes to admin role.
    - Authenticated users are redirected away from login/register/home to their respective dashboards.

o State Management : The system uses Vuex (store/index.js) for centralized state handling, with localStorage persistence for user and token.

- **State Variables:**
    - **user, token ->**Authentication data.
    - **fds ->** User’s fixed deposits.
    - **summary ->** Investment summary.
    - **dashboardInfo ->** Admin dashboard metrics.
    - **loading ->** Global loading flag.
- **Mutations:**
    - **setUser, setToken, clearAuth** (Authentication handling.)
    - **SET_FDS, UPDATE_FD_STATUS** (FD management.)
    - **SET_SUMMARY, SET_DASHBOARD_INFO** (Dashboard/summary updates.)
    - **SET_LOADING** (Loading state control.)
- **Actions:**
    - **login, register, logout** (Authentication workflows.)
    - **fetchFDs, breakFD, fetchBreakPreview** (FD operations.)

    - **fetchSummary, fetchDashboardInfo** (Investment and admin dashboard stats.
    - **setUserData** (Load logged-in user data via token.)
- **Getters:**
    - Provide easy access to authentication state, FD list, summary, and dashboard data.

o **API Integration:** The frontend integrates with backend APIs using Axios:

- **Authentication APIs: /auth/register, /auth/login, /auth/me**
- **FD APIs: /fd/book, /fd/user/{id}, /fd/{id}/break, /fd/{id}/break-preview, /fd/{id}/interest, /fd/{id}/status**
- **Support APIs: /support, /support/user/{id}**
- **Admin APIs: /admin/fds, /admin/tickets, /admin/dashboard/info**

🔹 **Admin dashboard views (Admin.vue, AdminFixedDeposits.vue, AdminSupportTickets.vue)** directly call respective APIs apart from Vuex store, ensuring real-time updates.

Database

o Schema Design

The database schema for Smart FD is designed to support both user-facing and admin-facing functionalities, ensuring secure storage of customer data, fixed deposits, transactions, and support interactions. The schema follows relational design principles with proper normalization, foreign key relationships, and indexing for optimized queries.

1. User Table (users)

Stores customer and admin information.

Column Name	Data Type	Constraints	Description
user_id	BIGINT (PK)	AUTO_INCREMENT, NOT NULL	Unique identifier for each user.
name	VARCHAR(100)	NOT NULL	Full name of the user.
email	VARCHAR(150)	UNIQUE, NOT NULL	Email address used for login.
password	VARCHAR(255)	NOT NULL	Encrypted password.
phone	VARCHAR(15)	UNIQUE, NOT NULL	Contact number of the user.
role	ENUM('USER','ADMIN')	DEFAULT 'USER'	Defines whether user is admin or customer.
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Registration date.
date_of_birth	Date	NOT NULL	Date of birth of the user.

2. Fixed Deposit Table (fixed_deposits)

Captures all FD-related transactions for users.

Column Name	Data Type	Constraints	Description
fd_id	BIGINT (PK)	AUTO_INCREMENT, NOT NULL	Unique FD identifier.

user_id	BIGINT (FK)	REFERENCES users(user_id)	FD holder (user).
scheme_name	VARCHAR(100)	NOT NULL	Type of FD scheme chosen.
amount	DECIMAL(12,2)	NOT NULL	Investment amount.
interest_rate	DECIMAL(5,2)	NOT NULL	Applicable interest rate (%).
start_dae	DATE	NOT NULL	FD booking date.
maturity_date	DATE	NOT NULL	FD maturity date.
status	ENUM('ACTIVE','BROKEN','MATURED')	DEFAULT 'ACTIVE'	Current status of the FD.
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Record creation time .

3. Support Tickets Table (support_tickets)

Manages queries/issues raised by users.

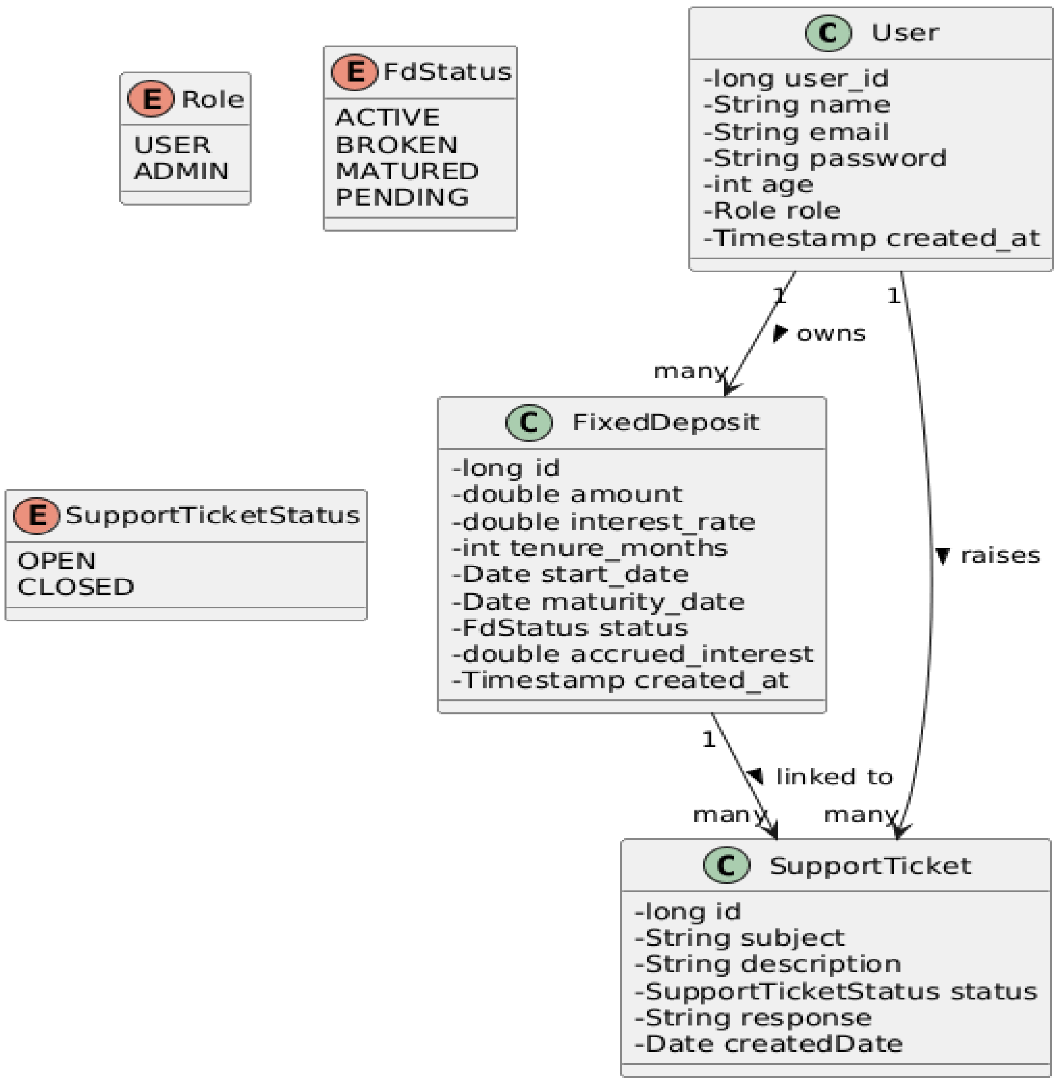
Column Name	Data Type	Constraints	Description
ticket_id	BIGINT (PK)	AUTO_INCREMENT, NOT NULL	Unique support ticket ID.
user_id	BIGINT (FK)	REFERENCES users(user_id)	User who raised the ticket.

subject	VARCHAR(200)	NOT NULL	Short description of issue.
description	TEXT	NOT NULL	Detailed explanation of issue.
status	ENUM('OPEN','IN_PROGRESS','RESOLVED','CLOSED')	DEFAULT 'OPEN'	Current state of the ticket.
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	When the ticket was created.
updated_at	TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP	Last updated time.

Relationships in Schema:

- **One-to-Many**: users → fixed_deposits (a user can have multiple FDs).
- **One-to-Many**: users → support_tickets (a user can raise multiple tickets).
- **Admin users** (role = 'ADMIN') have read-only access to fixed_deposits and support_tickets for monitoring.

UML:



Test Plan

○ Unit Tests :

BookFD.spec.js

- Validate **minimum deposit amount** logic.
- Ensure **maturity date computation** is correct based on selected scheme.
- Validate **FD return calculations** using utility (simple & compound interest).

- Ensure **tenure > 0** validation.

BreakFD.spec.js

- Render **loading state** while preview data fetches.
- Display **preview details** correctly when data is fetched.
- Show **error message** when API fetch fails.
- Handle **confirmation dialog** interactions:
  - Confirm Break → emits fdBroken.
  - Cancel → emits close.

Calculator.spec.js

- Validate **simple interest** calculation.
- Validate **compound interest** calculation.
- Ensure **senior citizen rate bump** is applied correctly.
- Confirm **senior benefits message** is displayed for age ≥ 60.

Other Component Specs (Login, Register, MyFDs, Support)

- Validate input field bindings and error handling.
- Ensure store actions are dispatched correctly.
- Confirm toast notifications on success/failure events.

Backend (JUnit + Mockito)

Controller Tests

- **AdminControllerTest** – Verify admin-level operations such as user and FD management.
- **AuthControllerTest** – Validate login, registration, and token generation.
- **UserControllerTest** – Verify user-related operations (profile fetch/update).
- **FixedControllerTest** – Validate booking, breaking, and fetching FD records.

Service Tests

- **AccruedInterestServiceTest** – Validate interest accrual calculations.
- **FixedDepositServiceTest** – Validate FD booking, maturity, and break logic.
- **FixedDepositServiceStatusTest** – Validate status transitions (active → closed).
- **PasswordHashingTest** – Ensure password hashing and verification work correctly.
- **SupportTicketServiceTest** – Verify ticket creation, updates, and status changes.
- **TokenValidationTest** – Validate JWT token creation and parsing.
- **UserServiceTest** – Verify user CRUD operations and business rules.

3. Setup and Configuration

Backend Setup

○ Prerequisites

- Java Development Kit (JDK) 17 or above
- Maven (for dependency management and build)
- PostgreSQL installed and running
- Git installed (to clone the project repository)
- IDE such as IntelliJ IDEA / Eclipse

○ Steps to run the backend:

1. Clone the repository

```
git clone https://github.com/sarthakkurothe/Digital-Fixed-Deposit-System.git
cd digital-fixed-deposit-system/backend
```

2. Configure PostgreSQL database credentials in 'application.properties'

```
spring.datasource.url=jdbc:postgresql://localhost:5432/fdms
spring.datasource.username=postgres
spring.datasource.password=your_password
```

3. Build the project using Maven

```
mvn clean install
```

4. Run the Spring Boot application

The backend will start at: http://localhost:8080

Frontend Setup

○ Prerequisites

- Node.js (version 18 or above)
- npm (comes with Node.js)
- Vue.js (installed via npm when setting up the project)
- Vite (bundler for Vue project, already configured in the repo)
- Tailwind CSS (for styling, already configured in the repo)

○ Steps to run the frontend:

1. Navigate to the frontend folder

```
cd digital-fixed-deposit-system/frontend
```

2. Install dependencies

```
npm install
```

3. Run the development server

```
npm run dev
```

The frontend will start at: http://localhost:5173 (default Vite port)

Database Setup

- #### Prerequisites

  - PostgreSQL installed and running
  - pgAdmin (optional, for GUI-based management)

- #### Steps to configure the database:

  1. Create a new database : CREATE DATABASE fdms;
  2. Ensure the username and password match those provided in the backend ‘application.properties’.
  3. Spring Boot with JPA will automatically create tables based on entities during the first run.

4. API Documentation

1. AuthController

- **Register**

  Endpoint: /auth/register

  Method: POST

  Example Request: http://localhost:8080/auth/register

  Example Response: Status - 201 Created

  Request Body:

  ```
  {
    "name": "John Doe",
    "email": "john@gmail.com",
    "password": "Password*123",
    "dateOfBirth": “1994-09-12”
  }
  ```

  Response Body:

  User added successfully!

- **Login**

  Endpoint: /auth/login

  Method: POST

  Example Request: http://localhost:8080/auth/login

  Example Response: Status - 200 OK

  Request Body:

  ```
  {
    "email": "john@gmail.com",
    "password":"Password*123"
  }
  ```

  Response Body:

  ```
  {
     "accessToken": "eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJqb2huMUBnbWFpb…",
     "refreshToken": "eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJqb2huMUBnbWFpb…"
  }
  ```

- **Get Current User**

  Endpoint: /auth/me

  Method: GET

  Example Request: http://localhost:8080/auth/login

  Example Response: Status - 202 Accepted

  Request Body: *None*

  Response Body:

  ```
  {
     "id": 1,
     "name": "John Doe",
     "email": "john@gmail.com",
     "age": 30,
     "role": "ROLE_USER"
  }
  ```

2. AdminController

- **Get All Fixed Deposits**

  Endpoint: /admin/fds

  Method: GET

  Example Request: http://localhost:8080/admin/fds

  Example Response: Status - 200 OK

  Request Body: *None*

  Response Body:

```
[
   {
      "fdId": 1,
      "name": "John Doe",
      "email": "john@gmail.com",
      "amount": 1000.0,
      "interest_rate": 6.0,
      "mature_date": "2026-03-28T18:30:00.000+00:00",
      "fdStatus": "ACTIVE"
   },
   {
      "fdId": 2,
      "name": "John Doe",
      "email": "john@gmail.com",
      "amount": 100000.0,
      "interest_rate": 7.5,
      "mature_date": "2028-09-28T18:30:00.000+00:00",
      "fdStatus": "BROKEN"
   }
]
```

- **Update FD Status**

  Endpoint: /admin/fd/{id}

  Method: PUT

  Example Request: http://localhost:8080/admin/fd/2

  Example Response: Status - 200 OK

  Request Body:

  ```
  {
    "status": "ACTIVE"
  }
  ```

  Response Body: *None*

- **Get All Support Tickets**

  Endpoint: /admin/tickets

  Method: GET

  Example Request: http://localhost:8080/admin/tickets

  Example Response: Status - 200 OK

  Request Body: *None*

  Response Body:

  ```
  [
      {
          "id": 1,
          "subject": "Break Fixed Deposit",
          "fd": {
              "id": 1,
              "user": {
                  "name": "John Doe",
                  "email": "john@gmail.com",
                  "password": "$2a$10$DuT8aNc2mu3XjhUUkdj5sOtLPRNclcHHq…",
                  "dateOfBirth": "1994-09-12",
  ```

```
                "role": "ROLE_USER",
                "created_at": "2025-09-29T11:54:43.077+00:00",
                "id": 1
            },
            "amount": 1000.0,
            "interest_rate": 6.0,
            "tenure_months": 6,
            "start_date": "2025-09-28T18:30:00.000+00:00",
            "maturity_date": "2026-03-28T18:30:00.000+00:00",
            "status": "ACTIVE",
            "accrued_interest": 0.0,
            "created_at": "2025-09-29T11:55:22.954+00:00"
        },
        "description": "Premature Withdrawal",
        "status": "CLOSED",
        "createdDate": "2025-09-29",
        "name": "User",
        "email": "user@gmail.com"
    }
]
```

- **Close Support Ticket**

  Endpoint: /admin/tickets/{id}

  Method: POST

  Example Request: http://localhost:8080/admin/tickets/2

  Example Response: Status - 200 OK

  Request Body:

  ```
    "response": "Issue resolved"
  ```

  Response Body: *None*

3. SupportTicketController

- **Create Support Ticket**

  Endpoint: /support

  Method: POST

  Example Request: http://localhost:8080/support

  Example Response: Status - 201 Created

  Request Body:

```
{
  "userId":1,
  "fdId":1,
  "subject":"Issue with fixed deposit",
  "description":"FD status didn't update after maturity"
}
```

  Response Body: *None*

- **Get Support Tickets by User**

  Endpoint: /support/user/{userId}

  Method: GET

  Example Request: http://localhost:8080/support/user/1

  Example Response: Status - 202 Accepted

  Request Body: *None*

  Response Body:

```
[
   {
       "id": 1,
       "fdId": 2,
       "subject": "Break Fixed Deposit",
       "description": "Premature Withdrawal",
       "status": "CLOSED",
       "response": "Broke FD",
       "createdDate": "2025-09-29"
   },
   {
       "id": 2,
       "fdId": 1,
       "subject": "Issue with fixed deposit",
       "description": "FD status didn't update after maturity",
       "status": "OPEN",
       "response": null,
       "createdDate": "2025-09-30"
   }
]
```

4. FixedDepositController

- **Book Fixed Deposit**

  Endpoint: /fd/book

  Method: POST

  Example Request: http://localhost:8080/fd/book

  Example Response: Status - 201 Created

  Request Body:

```
{
  "user_id": 1,
  "amount": 50000,
  "interest_rate": 6.5,
  "tenure_months": 12
}
```

  Response Body: *None*

- **View FDs by User**

  Endpoint: /fd/user/{userId}

  Method: GET

  Example Request: http://localhost:8080/fd/user/1

  Example Response: Status - 200 OK

  Request Body: *None*

  Response Body:

```
[
   {
      "id": 1,
      "user": {
         "name": "John Doe",
         "email": "john@gmail.com",
         "password": "$2a$10$DuT8aNc2mu3XjhUUkdj5sOtLPRNclcHHq…",
         "dateOfBirth": "1994-09-12",
         "role": "ROLE_USER",
         "created_at": "2025-09-29T11:54:43.077+00:00",
         "id": 1
      },
      "amount": 1000.0,
      "interest_rate": 6.0,
      "tenure_months": 6,
      "start_date": "2025-09-28T18:30:00.000+00:00",
      "maturity_date": "2026-03-28T18:30:00.000+00:00",
      "status": "ACTIVE",
      "accrued_interest": 0.0,
      "created_at": "2025-09-29T11:55:22.954+00:00"
   }
]
```

- **Break FD**

  Endpoint: /fd/{fdId}/break

  Method: POST

  Example Request: http://localhost:8080/fd/1/break

  Example Response: Status - 201 Created

  Request Body: *None*

  Response Body: *None*

- **Break Preview**

  Endpoint: /fd/{fdId}/break-preview

  Method: GET

  Example Request: http://localhost:8080/fd/2/break-preview

  Example Response: Status - 200 OK

  Request Body: *None*

  Response Body:

  ```
  {
     "fdId": 2,
     "principalAmount": 100000.0,
     "accruedInterest": 3000.0,
     "startDate": "2025-03-28T18:30:00.000+00:00",
     "maturityDate": "2026-03-28T18:30:00.000+00:00",
     "penalty": 500.0,
     "payout": 102500.0,
     "interestRate": 6.0,
     "tenure": 12.0,
     "timeElapsed": 6
  }
  ```

- **View Accrued Interest**

  Endpoint: /fd/{fdId}/interest

  Method: GET

Example Request: http://localhost:8080/fd/2/interest

Example Response: Status - 200 OK

Request Body: *None*

Response Body:

3000.0

- **Update FD Status**

  Endpoint: /fd/{Id}/status

  Method: PUT

  Example Request: http://localhost:8080/fd/1/status?status=MATURED

  Example Response: Status - 200 OK

  Request Body: *None*

  Response Body: *None*

5. Deployment

Deployment Environment:

- Backend runs on localhost:8080
- Frontend runs on localhost:5173

6. Future Enhancements

The current implementation of the Digital Fixed Deposit (FD) System provides core functionalities such as secure registration/login, FD booking, interest computation, premature withdrawal with penalties, and support ticket management. To enhance scalability, customer experience, and integration with real-world banking systems, the following improvements can be considered in the future:

Enhanced Security

- **Two-Factor Authentication (2FA)** for login and high-value FD transactions.
- Advanced **fraud detection and anomaly monitoring**.
- **Audit trail logs** for all critical actions, ensuring compliance with banking regulations.

Payment Gateway & Bank Integration

- Direct **integration with banking APIs/UPI/Net banking** for deposit and withdrawal transactions.
- Real-time **account balance updates** after FD booking/break.
- Integration with **core banking systems (CBS)** for seamless operations.

Notifications & Alerts

- Automated **email/SMS/app notifications** for FD maturity reminders, break confirmations, or support ticket updates.
- Configurable **alert preferences** (daily/weekly/monthly summaries).
- Push notifications for upcoming interest payouts or offers.

Regulatory & Compliance Enhancements

- **KYC/AML integration** for identity verification.
- Ensure compliance with **RBI guidelines and other financial regulations**.
- Support for **audit and compliance reporting**.

7. Team and Roles

Team Name: Zeta Dreamers

Team Members :

SNo	Name	Role
1	Sarthak Kurothe	Team Lead - Support Ticket Module
2	Anees	Team Member - User Registration and Authentication
3	Lakshmi Mrudula	Team Member - Book and View Fixed Deposits
4	Srijith	Team Member - Interest Calculation and FD Status Management
5	Rithika	Team Member - Premature Withdrawal and Penalty Computation

8. Appendix

- **https://www.hdfcbank.com/personal/resources/rates#/fixed-deposit-interest-rate-less-than-5-cr**
- **https://groww.in/calculators/fd-calculator**
- **https://www.hdfcbank.com/personal/useful-links/important-messages/changes-in-premature-withdrawal-terms-and-conditions**
- **https://www.paisabazaar.com/fixed-deposit/premature-withdrawal-of-fixed-deposit/**