**Payment\_Backend**

1. **Problem Statement:**

Objective of this use case is to build the APIs that can capture the payment information and store them in DB and read from the DB and show payment list in API response.

Payment Information to be collected:

* From Account 🡪
  + Account from which we make payment [Debited Account]
  + Account Name or Account Number to be used to uniquely identify the account.
  + List of accounts to be maintained in DB.
* To Account 🡪
  + it can be individual accounts, payee account to which we credit the amount.
  + Payee Account or Payee name to used to identify
  + Individuals or Payees like Mobile providers, Internet providers, Credit cards etc. - to be maintained in DB.
* Payment Date 🡪
  + Date on which payment scheduled, it can be today or future date.
  + Date format DD/MM/YYYY
  + Validate the date that it cannot be submitted with past date.
* Payment Amount 🡪
  + Amount in Rupees
  + Amount should be formatted with Rupees symbol while listing.
* Fee Amount 🡪
  + Fee amount to be calculated by the system based on the selected amount.
  + Amount should be formatted with Rupees symbol.
  + Calculation based on the fee table.
  + Should be calculated based on the amount Entered

|  |  |  |
| --- | --- | --- |
| Minimum Amount | Maximum Amount | Fee |
| 0 | 99 | 10 |
| 100 | 999 | 25 |
| 1000 | 9999 | 50 |
| 10,000 | 99999 | 100 |
| more |  | 500 |

* Memo 🡪
  + Optional field.
  + Comments about the payment.
  + Should be less than 100 characters.
* List of APIs to be build
  + GET 🡪 /api/payment [list of all payments]
  + GET 🡪 /api/{payment-id}/payment
  + POST 🡪 /api/payment
  + PUT 🡪 /api/payment
  + DELETE 🡪 /api/payment
* POST and PUT Payments should create confirmation with Payment ID.
* Integrate the APIs with Swagger UI [Mock given below] or Create a Project in Postman with the given API.
* Code coverage should be 80% or above.

1. **Technology Stack:**

* Java/Python/C#
* MySQL/Postgres

1. **Features/Requirements:**

|  |  |  |
| --- | --- | --- |
| Feature/requirement | Feature /Requirement Description with Business rules and design constraints | Expected Duration to complete |
| Create CRUD Operation of all API | GET, POST, PUT, DELETE | 4 hours |
| Validate the information | Mandatory, date calendar | 1 hour |
| List down the From and to accounts/Payees | Store the Account payee, fee and payment information in DB | 2 hours |
| API integration with DB |  | 1 hour |
| Code Coverage | 80% or above | 1 hour |
| Swagger Integrations |  | 1 hour |

1. **Key Points to be followed:**

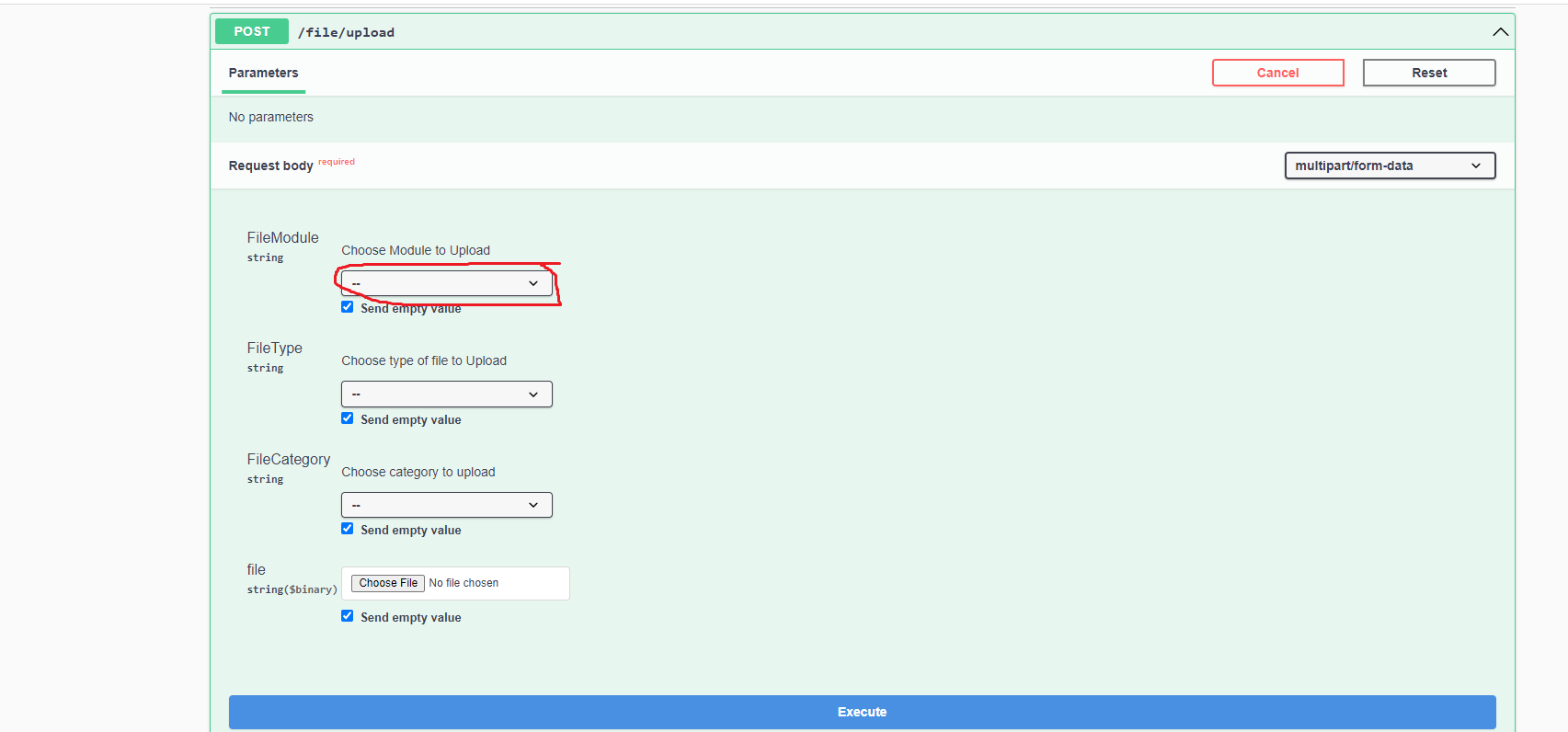
**UI/UX: Mock Swagger UI**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

API With dropdown [List of accounts] – This is only Mock



1. **Table Structure:**

|  |  |
| --- | --- |
| TABLE NAME | COLUMNS |
| Account | * account\_id, * account\_number, * account\_name, * account\_balance, * account\_status, * updated\_datetime. |
| Payee | * payee\_id, * payee\_number, * payee\_name, * amount\_due, * due\_date, * updated\_datetime |
| Fee | * fee\_id, * fee\_amount, * amount\_min, * amount\_max, * updated\_datetime |
| Payment | * payment\_id, * account\_id, * payee\_id, * fee\_id, * updated\_datetime. |

1. **Output to be generated using GitHub Copilot:**
2. Requirement Document
3. Database Design Document
4. User Stories with acceptance criteria
5. Database scripts – Scripts to create tables and Stored Procedures
6. Scripts to load tables
7. Working code.
8. DOCUMENT 🡪 Document the copilot commands used foe code generation.
9. Test Plan
10. Unit test case
11. Deployment Script
12. **UI Details with sample snapshots**
13. **Additional Things :-**
14. Integrate Jacoco into the Springboot Project
15. Create a sample Junit test case[Account Controller]
16. Run the code coverage and see the percentage.