

DESIS ASCEND EDUCARE 2023-24

# PROJECT REPORT

UnityLend



## PRESENTED BY

Mentor

Mr. Purushottam Kumar

Mentees

Prathyusha Nimmagadda

Neha Kariya

Shelly Aggarwal

Surabhi Shreya

Vani Thapar

# TABLE OF CONTENTS

Executive Summary	3
Design and Architecture	4
Snippets	5 - 12
Functional Requirements	13
Tech Stacks	14
Flow Diagram for User Registration and Community Creation	15
Flow Diagram for Borrower raising a Borrow Request	16
Flow Diagram for creating Borrower EMI Schedule after Borrow Request is fulfilled	17
Flow Diagram for Lender lending for a Borrow Request	18
Flow Diagram for Repayment of EMI against a Borrow Request	19
ER Diagram	20
Class Diagram	21
Future Scope	22
Learnings	23
References	24

# EXECUTIVE SUMMARY

- Project Summary

UnityLend facilitates community lending, empowering users and fostering trust. It offers transparent borrowing processes, dynamic community creation, and efficient repayment management. Powered by PostgreSQL and Spring Boot, UnityLend promotes financial cooperation and mutual support.

- Goal

UnityLend powered by PostgreSQL and Spring Boot, UnityLend promotes financial cooperation and mutual support.

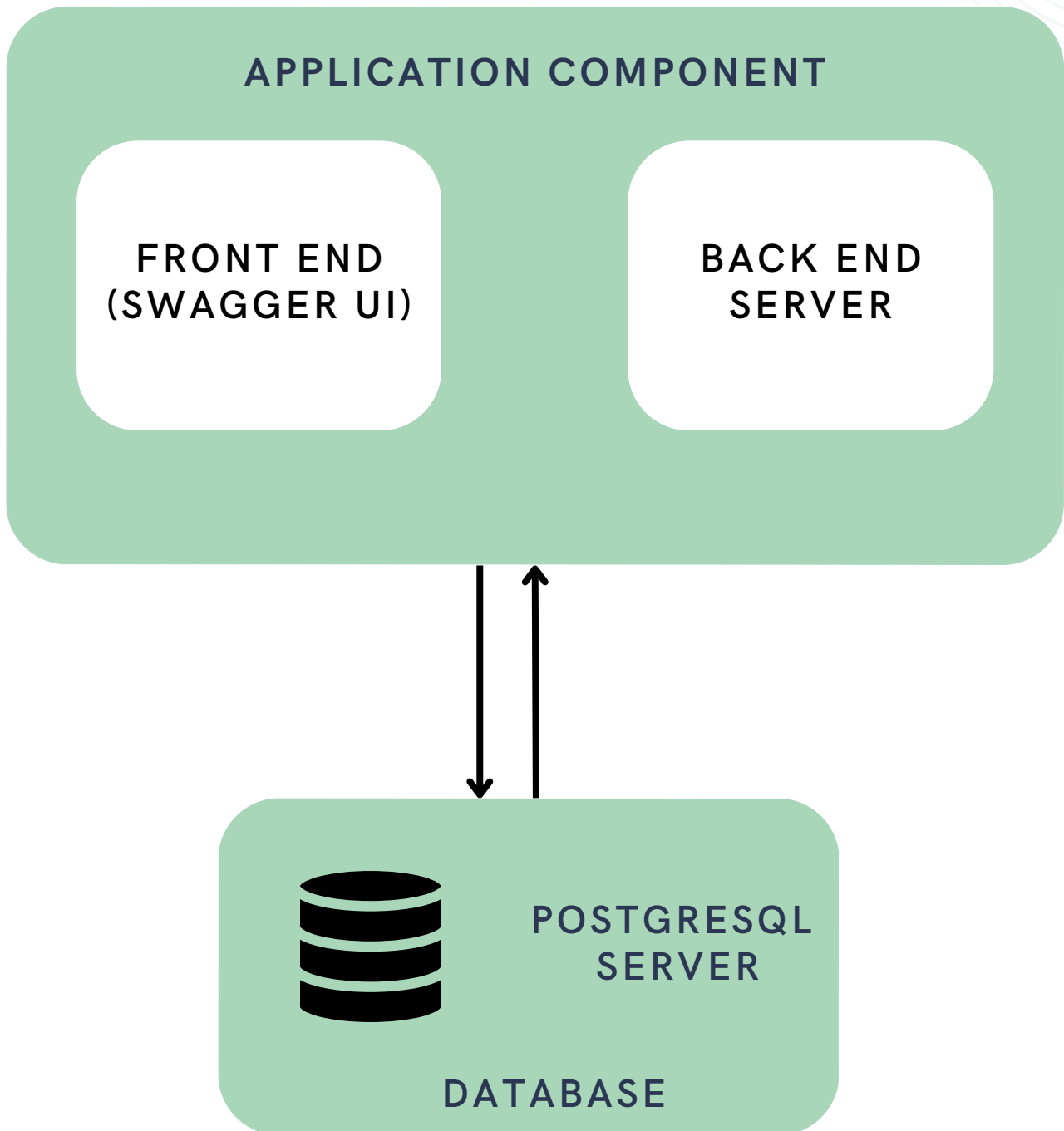
- Requirements

User registration with community tags, automated community formation, income-based borrowing limits, borrowing request validation, user wallet management, transparent repayment schedules, defaulted repayment handling, and monthly EMI options ensure efficient financial transactions.

- Utility

UnityLend empowers users to borrow and lend money within their communities, fostering financial cooperation and trust while addressing immediate financial needs.

# DESIGN AND ARCHITECTURE



# SNIPPETS - User, Wallet, Community Controllers

## user-controller

PUT /user/update-user-details

POST /user/create-user

GET /user

GET /user/get-user-by-user-id/{userId}

GET /user/get-all-users

DELETE /user/delete-user-by-user-id/{userId}

## wallet-controller

GET /wallet

GET /wallet/get-wallet-for-wallet-id/{walletId}

GET /wallet/get-wallet-for-user-id/{userId}

GET /wallet/get-wallet-balance/{walletId}

GET /wallet/get-all-wallets

GET /wallet/deduct-amount/{walletId}

GET /wallet/add-amount/{walletId}

## community-controller

GET /community

GET /community/get-communities-by-user-id/{userId}

GET /community/get-all-communities

# SNIPPETS - Borrow Request, Borrow Request - Community Map, Transaction Controllers

## borrow-request-controller ^

**POST** /borrow-request/get-borrow-request-in-community-by-amount/{communityId} v

**POST** /borrow-request/create-borrow-request v

**GET** /borrow-request v


**GET** /borrow-request/get-borrow-request-in-community-by-interest/{communityId}/{interest} v

**GET** /borrow-request/get-borrow-request-by-user-id/{userId} v

**GET** /borrow-request/get-borrow-request-by-community-id/{communityId} v

**GET** /borrow-request/get-all-borrow-requests v

## borrow-request-community-map-controller ^


**GET** /borrow-request-community-map/get-requests/{communityId} v 

**GET** /borrow-request-community-map/get-communities/{BorrowRequestId} v

## transaction-controller ^

**GET** /transaction v

**GET** /transaction/get-transaction/{senderId}/{receiverId} v

**GET** /transaction/get-transaction-by-sender/{senderId} v 

**GET** /transaction/get-transaction-by-date/{senderId} v

**GET** /transaction/get-all-transactions v

# SNIPPETS - Lend Transaction, Repayment Controllers

## lend-transaction-controller

POST /lend-transaction/lend

GET /lend-transaction

GET /lend-transaction/get-lend-transaction-info/{lendTransactionId}

GET /lend-transaction/get-all-lend-transactions

GET /lend-transaction/get-all-lend-transactions-by-user/{userId}

## repayment-transaction-controller

POST /repayment-transaction/repay

POST /repayment-transaction/repay-default-emi

GET /repayment-transaction

GET /repayment-transaction/get-repayment-transaction-info/{repaymentTransactionId}

GET /repayment-transaction/get-all-repayment-transactions

GET /repayment-transaction/get-all-repayment-transactions-by-user/{userId}

# SNIPPETS - User Creation

## Responses

### Curl

```
curl -X 'POST' \
  'http://localhost:8085/user/create-user' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "password": "pass@987",
    "name": "Borrower",
    "email": "Borrower1@example.com",
    "contactNo": "8960258574",
    "dob": "1982-03-22",
    "income": 80000,
    "communityDetails": {
      "university": "ISB Hyderabad",
      "office": "D.E.Shaw"
    },
    "communityDetailsJson": "{\\\"university\\\": \\\"ISB Hyderabad\\\", \\\"office\\\": \\\"D.E.Shaw\\\"}"
  }'
```

### Request URL

http://localhost:8085/user/create-user

### Server response

Code	Details
------	---------

200	
-----	--

#### Response body

User created successfully

#### Response headers

connection: keep-alive  
content-length: 25  
content-type: text/plain; charset=UTF-8  
date: Sat, 23 Mar 2024 15:27:45 GMT  
keep-alive: timeout=60



# SNIPPETS - User Creation

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8085/user/create-user' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "password": "pass@876",
    "name": "Lender",
    "email": "lender@example.com",
    "contactNo": "8960259264",
    "dob": "1976-03-22",
    "income": 90000,
    "communityDetails": {
      "school": "Sri Chaitanya School",
      "office": "D.E.Shaw"
    }
  },
  "communityDetailsJson": "{\\"school\\": \\"Sri Chaitanya School\\", \\"office\\": \\"D.E.Shaw\\"}"
}'
```

Request URL

`http://localhost:8085/user/create-user`

Server response

Code	Details
200	<p>Response body</p> <p>User created successfully</p> <p>Response headers</p> <p>connection: keep-alive content-length: 25 content-type: text/plain; charset=UTF-8 date: Sat, 23 Mar 2024 15:30:00 GMT keep-alive: timeout=60</p>

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8085/user/create-user' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "password": "pass@875",
    "name": "Lender2",
    "email": "lender2@example.com",
    "contactNo": "8060259963",
    "dob": "1983-03-22",
    "income": 60000,
    "communityDetails": {
      "university": "IIM Ahmedabad",
      "office": "D.E.Shaw"
    }
  },
  "communityDetailsJson": "{\\"university\\": \\"IIM Ahmedabad\\", \\"office\\": \\"D.E.Shaw\\"}"
}'
```

Request URL

`http://localhost:8085/user/create-user`

Server response

Code	Details
200	<p>Response body</p> <p>User created successfully</p> <p>Response headers</p> <p>connection: keep-alive content-length: 25 content-type: text/plain; charset=UTF-8 date: Sat, 23 Mar 2024 15:40:26 GMT keep-alive: timeout=60</p>

# SNIPPETS - Borrow Request Creation

## Responses

### Curl

```
curl -X 'POST' \
  'http://localhost:8085/borrow-request/create-borrow-request' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "borrower": {
      "userId": "f5cbe277-0668-44e8-a82a-c60a691854ec",
      "password": "pass@987",
      "income": 80000,
      "communityDetails": {"university": "ISB Hyderabad", "office": "D.E.Shaw"}
    },
    "returnPeriodMonths": 2,
    "monthlyInterestRate": 20,
    "requestedAmount": 5000.00,
    "communityIds": ["60610fce-b283-4c83-9e61-079dce089095"]
  }'
```

### Request URL

```
http://localhost:8085/borrow-request/create-borrow-request
```

### Server response

Code	Details
------	---------

200	
-----	--

#### Response body

```
true
```

#### Response headers

```
connection: keep-alive
content-type: application/json
date: Sat, 23 Mar 2024 15:33:00 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

# SNIPPETS - Fetching Lending Transaction Details By Lender Id

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8085/lend-transaction/lend?lenderId=16a52635-dd0e-4394-9322-20b70f17334f&borrowRequestId=b8151f93-6df0-4c4a-bc96-ef0dd2054e6e&amount=2500' \
  -H 'accept: */*' \
  -d ''
```

Request URL

```
http://localhost:8085/lend-transaction/lend?lenderId=16a52635-dd0e-4394-9322-20b70f17334f&borrowRequestId=b8151f93-6df0-4c4a-bc96-ef0dd2054e6e&amount=2500
```

Server response

Code	Details
200	<p>Response body</p> <pre>Lend Transaction is successful</pre> <p>Response headers</p> <pre>connection: keep-alive content-length: 30 content-type: text/plain;charset=UTF-8 date: Sat, 23 Mar 2024 15:50:23 GMT keep-alive: timeout=60</pre>

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8085/lend-transaction/lend?lenderId=6ed0ac78-6cbe-4eab-94fd-7e6e91c07204&borrowRequestId=b8151f93-6df0-4c4a-bc96-ef0dd2054e6e&amount=2500' \
  -H 'accept: */*' \
  -d ''
```

Request URL

```
http://localhost:8085/lend-transaction/lend?lenderId=6ed0ac78-6cbe-4eab-94fd-7e6e91c07204&borrowRequestId=b8151f93-6df0-4c4a-bc96-ef0dd2054e6e&amount=2500
```

Server response

Code	Details
200	<p>Response body</p> <pre>Lend Transaction is successful</pre> <p>Response headers</p> <pre>connection: keep-alive content-length: 30 content-type: text/plain;charset=UTF-8 date: Sat, 23 Mar 2024 16:14:08 GMT keep-alive: timeout=60</pre>

# SNIPPETS - Repayment By Borrow

## Request Id

**Responses**

**Curl**

```
curl -X 'POST' \
  'http://localhost:8085/repayment-transaction/repay?borrowRequestId=b8151f93-6df0-4c4a-bc96-ef0dd2054e6e' \
  -H 'accept: */*' \
  -d ''
```

**Request URL**

```
http://localhost:8085/repayment-transaction/repay?borrowRequestId=b8151f93-6df0-4c4a-bc96-ef0dd2054e6e
```

**Server response**

Code	Details
200	<p><b>Response body</b></p> <pre>Repayment Transaction is successful</pre> <p><b>Response headers</b><pre>connection: keep-alive content-length: 35 content-type: text/plain;charset=UTF-8 date: Sat,23 Mar 2024 16:55:14 GMT keep-alive: timeout=60</pre></p>

# FUNCTIONAL REQUIREMENTS

- User registration entails providing basic details along with community tags such as place of residence, school, college, workplace, etc.
- The application automatically organizes communities based on user tags.
- Users can list borrowing requests within their community and offer lending options.
- The user wallet management feature includes pre-defined virtual currency with options for lending and borrowing from wallet funds.
- Borrowers are presented with a repayment schedule detailing dates and amounts, including interest.
- Lenders are provided with a repayment schedule indicating dates and amounts to be received.
- Borrowers have access to a list of overdue payment records.
- Lenders can view a list of overdue payments to be received.
- Borrowers can review their past repayment history.
- Lenders can review their past received payments.
- Overdue payments incur fines as well.

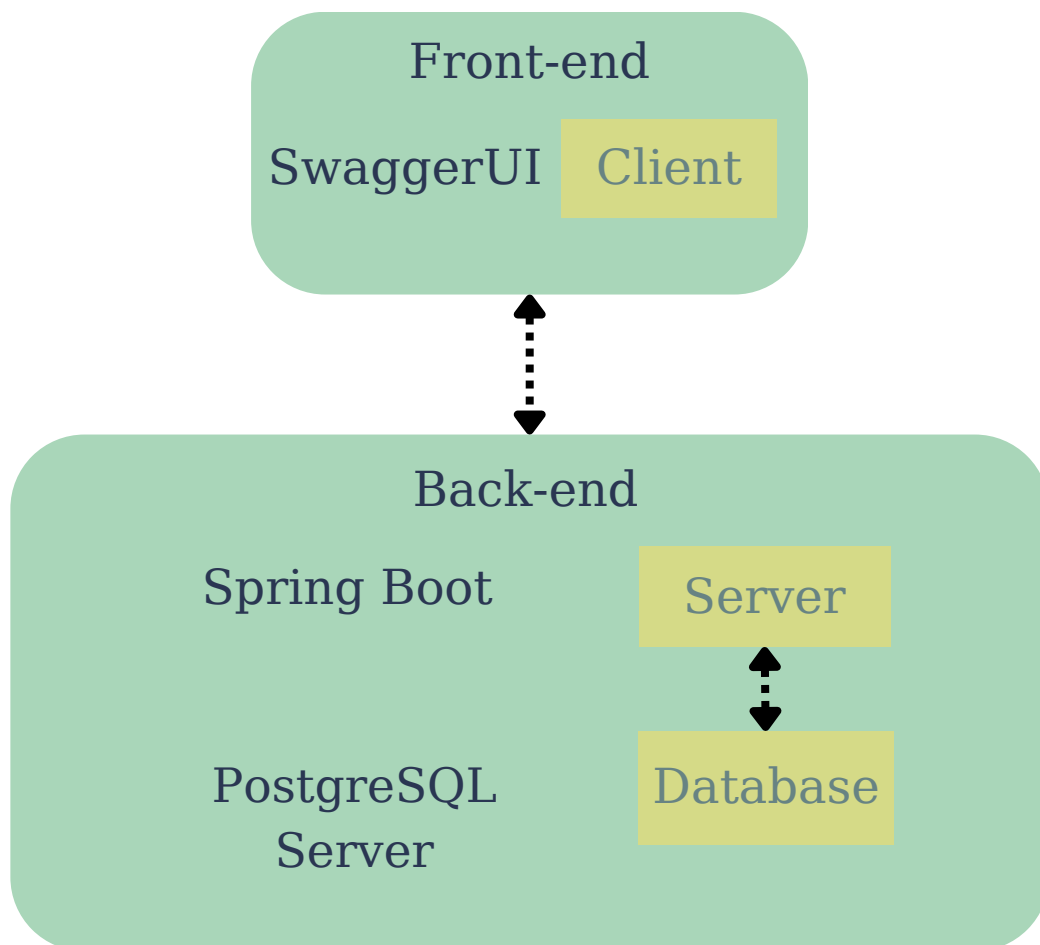
# TECH STACKS

Front-end:

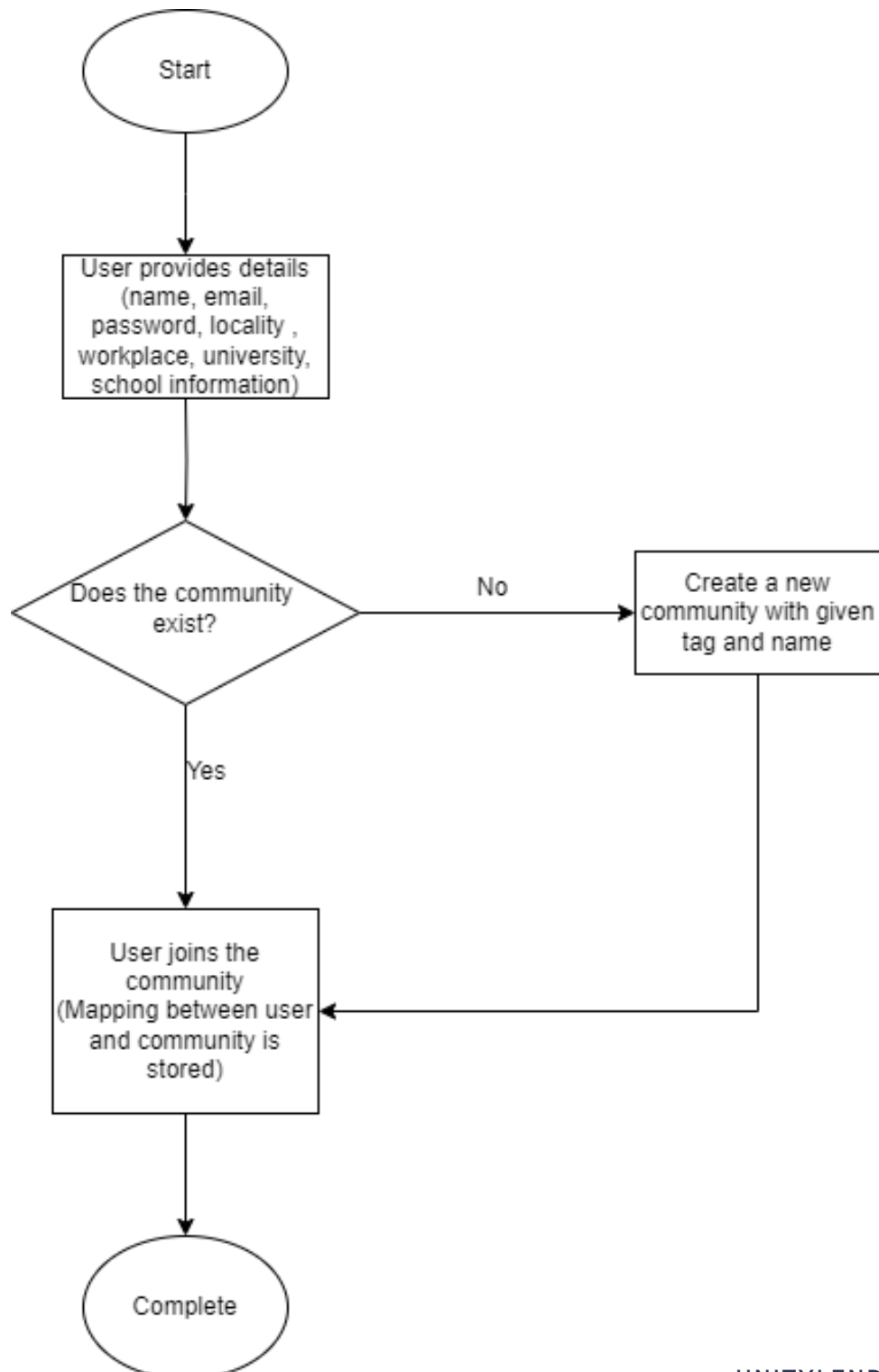
- Swagger

Back-end:

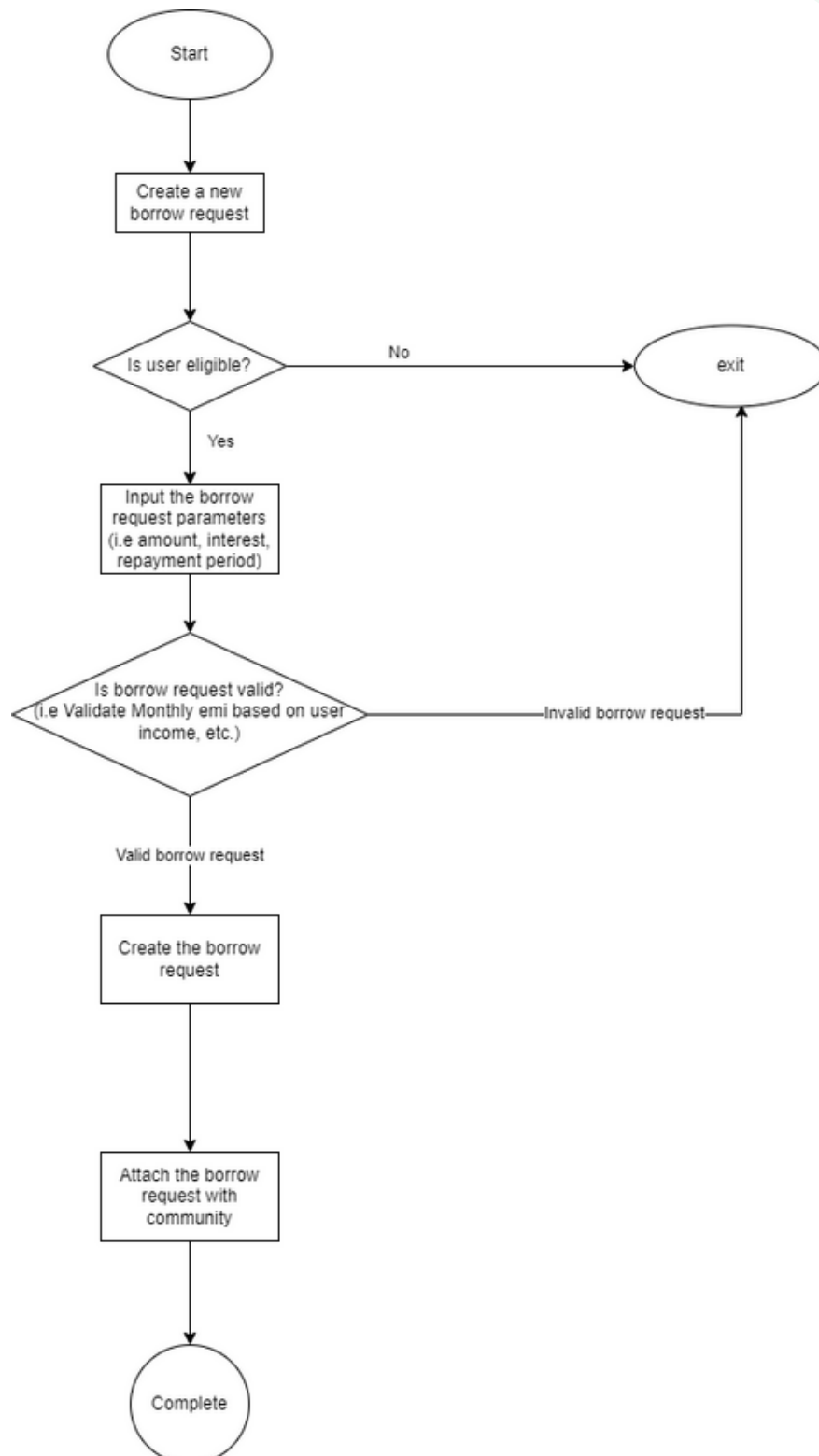
- Tool - Spring Boot using MyBatis
- Database - Postgres



# Flow Diagram for User Registration and Community Generation

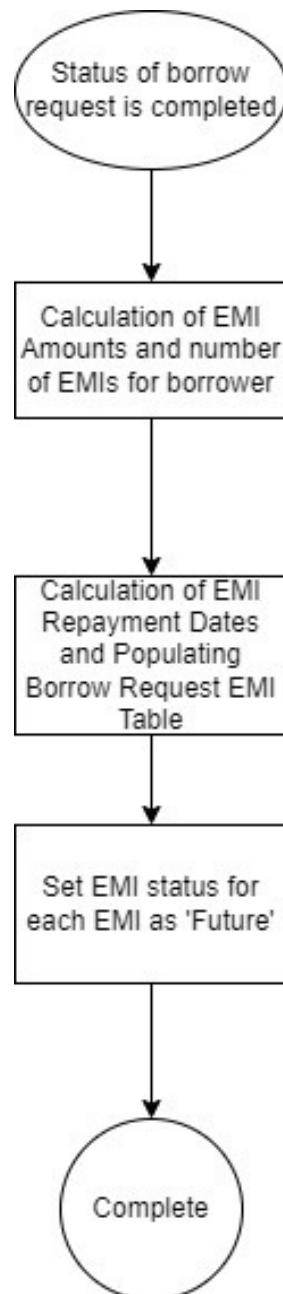


# Flow Diagram for Borrower raising a Borrow Request

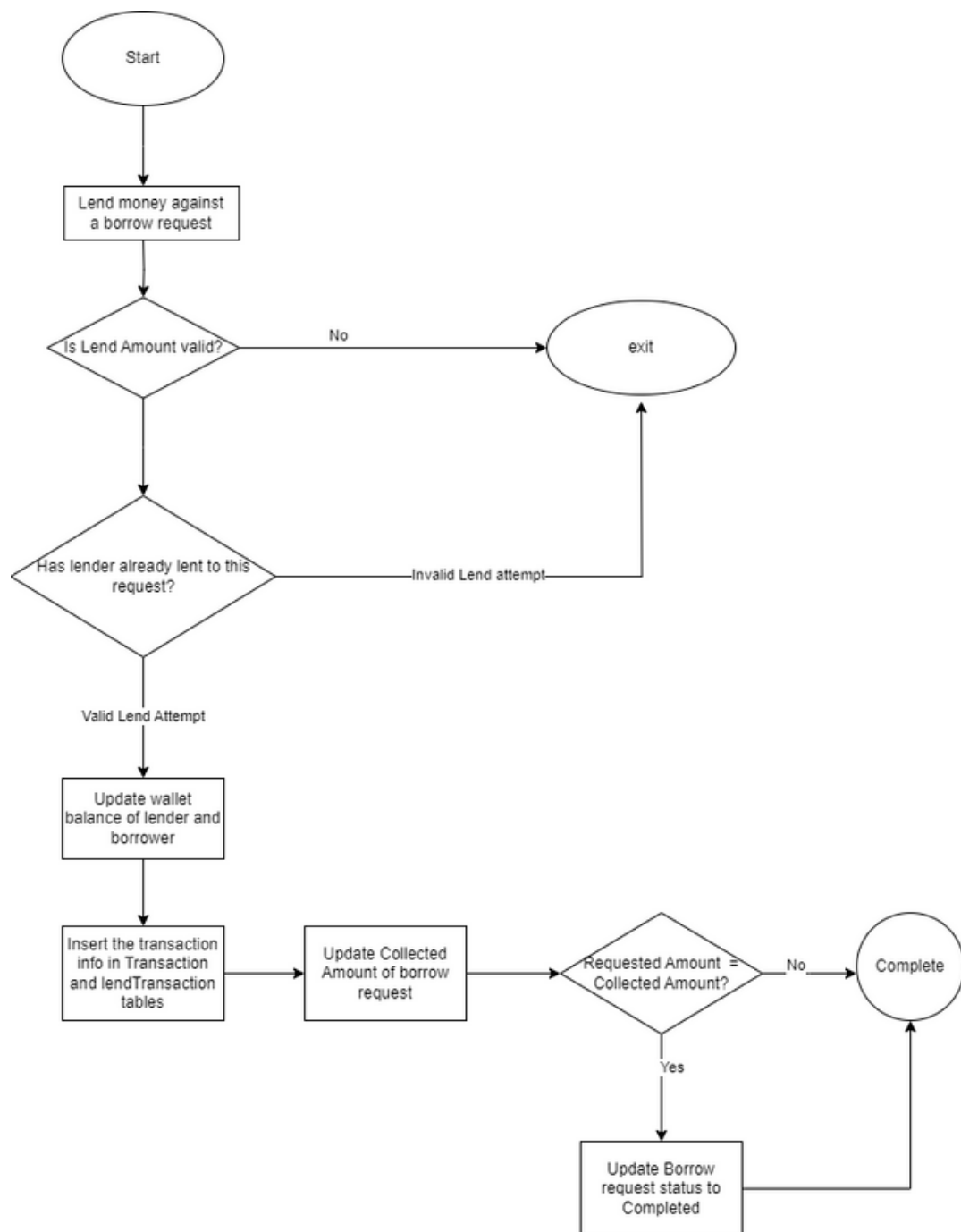




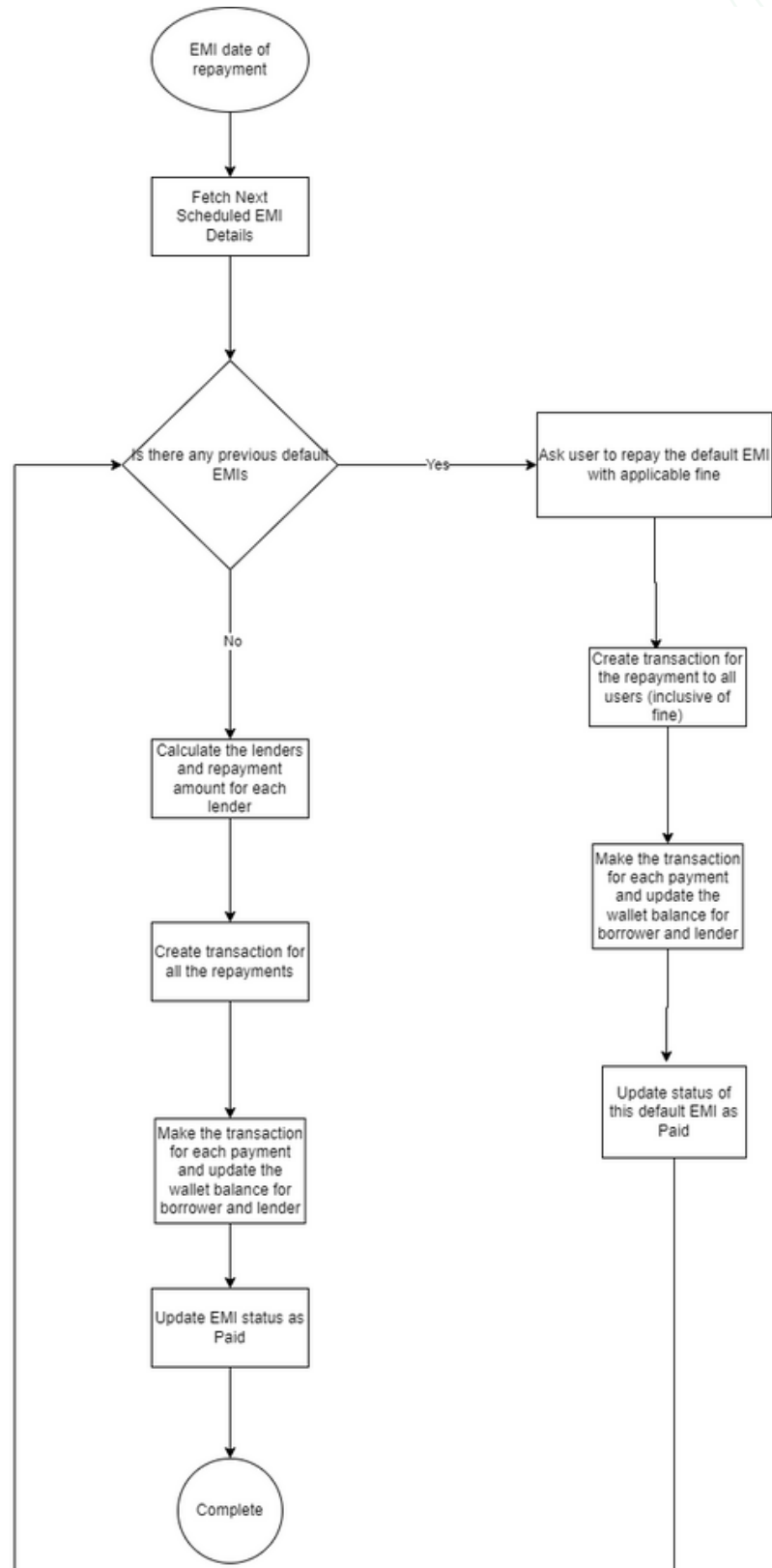
## Flow Diagram for creating Borrower EMI schedule after Borrow Request is fulfilled



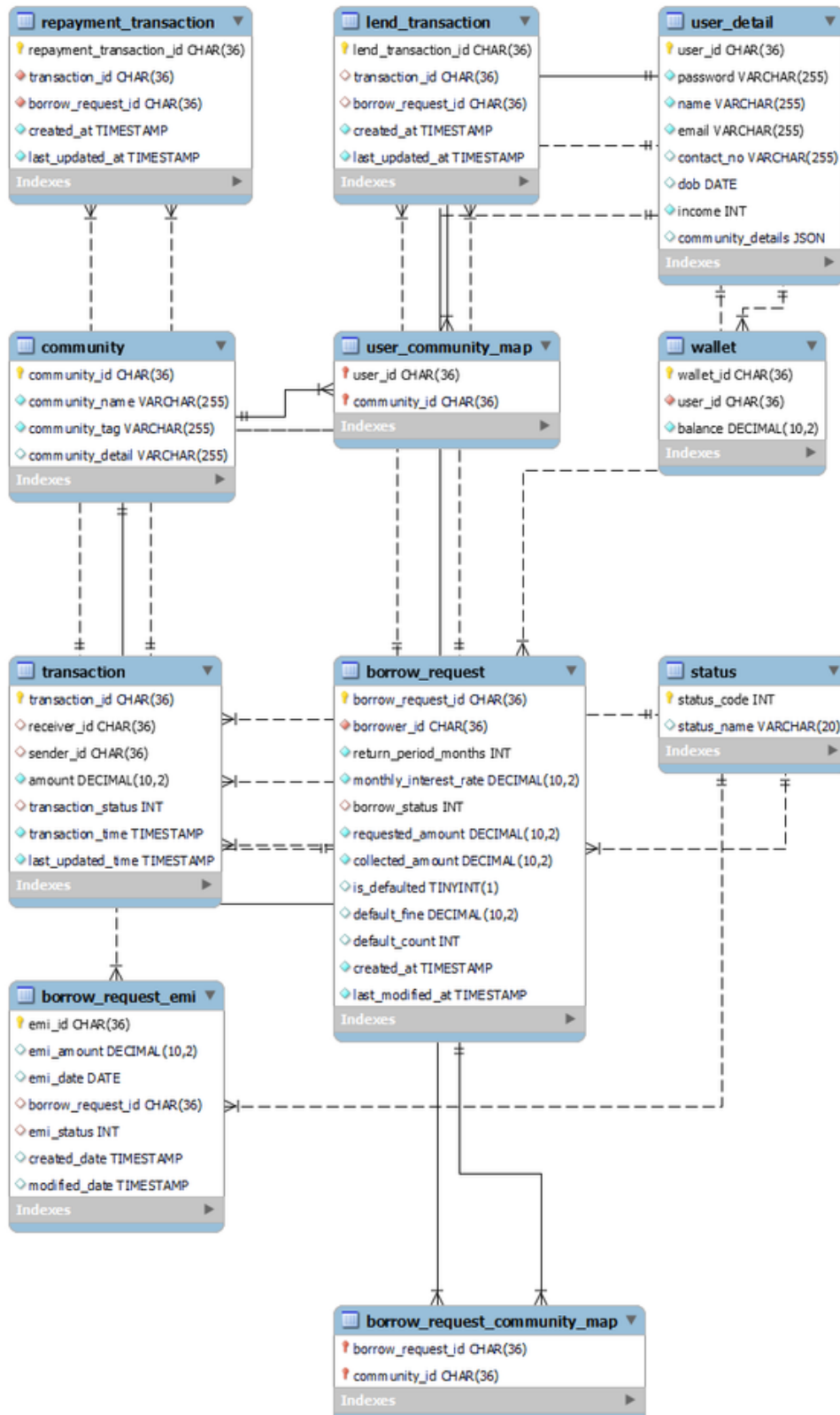
# Flow Diagram for Lender lending for a Borrow Request



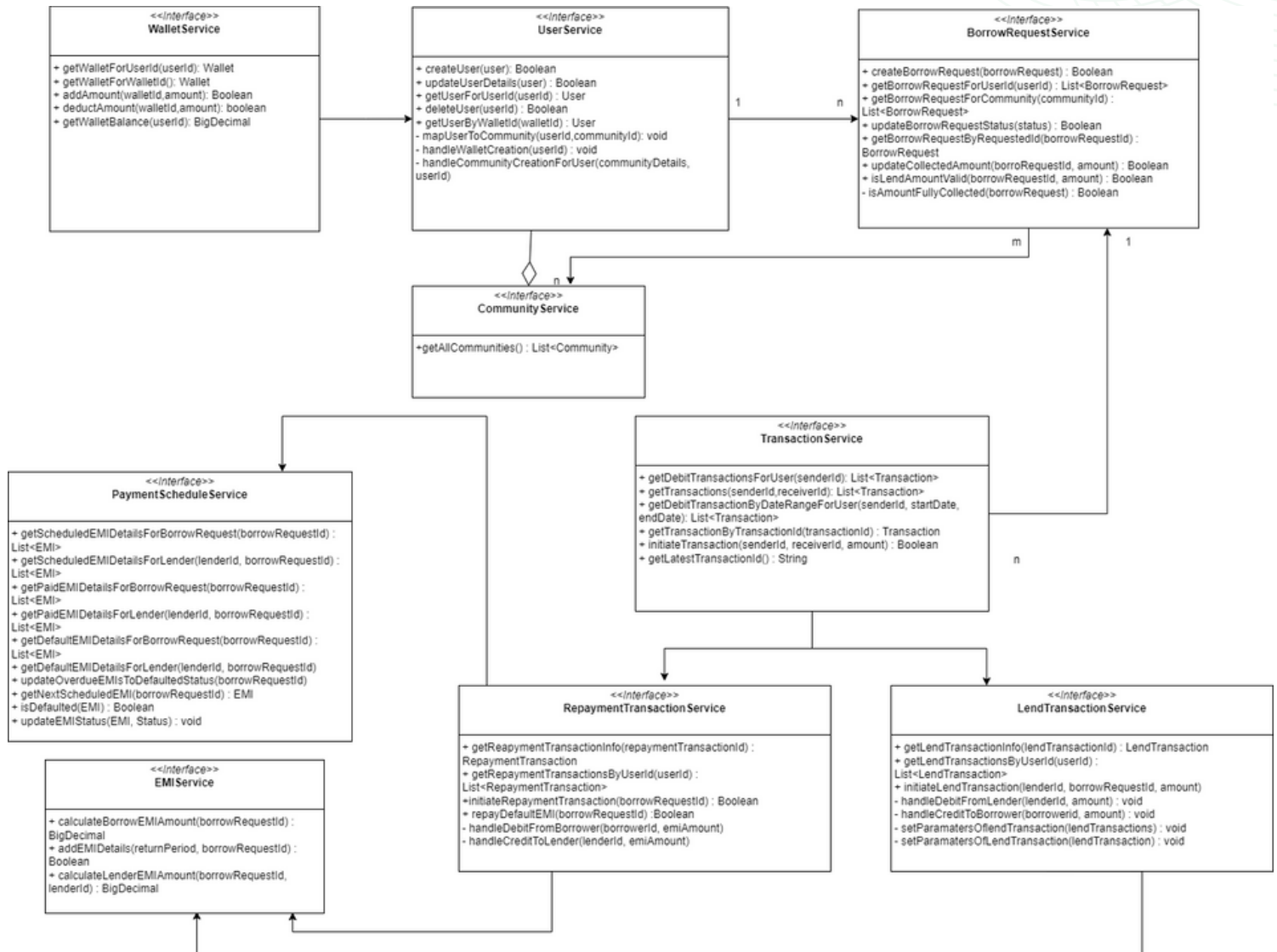
# Flow Diagram for Repayment of EMI against a Borrow Request



# ER Diagram



# Class Diagram



# FUTURE SCOPE

- User rating management through community member feedback and loan repayment records.
- Overall community feedback/rating based on the loan repayment schedule miss
- Provide mechanisms to handle cases where borrow requested amount is not fully collected
- Loan recovery and penalizing mechanisms.
- Handle the lifetime of requests
- Build an interactive User Interface to provide all the functionalities
- Integrate with the actual money wallet or bank accounts.
- Enhance user verification and income source.
- Implement User authentication and authorisation

# LEARNINGS

- Object Oriented Programming (OOPS) - Unity Lend employs object-oriented programming principles by representing entities like User, Transaction, and BorrowRequest as classes. These classes encapsulate data and behavior, offering methods for actions like user registration, transaction processing, and borrow request management.
- Java - UnityLend leverages Java's Spring Boot framework for its backend, facilitating rapid development and deployment of RESTful APIs. The OOPS principles followed in Java enabled us to design and implement the UnityLend application in a structured and organized manner.
- DBMS - The fundamental principles of relational databases, including concepts like tables, rows, columns, keys, and relationships helped in designing the database schema for UnityLend, determining how data should be organized and structured to support the application's functionality.
- Git and GitHub - Git offered a robust framework for source code management, enabling streamlined collaboration among team members. Leveraging branching, merging, and pull request capabilities, we concurrently developed various features while upholding code integrity. GitHub acted as a centralized platform for hosting our project repository, simplifying collaboration, code review, and issue tracking.

# REFERENCES

- PostgreSQL - <https://www.postgresql.org/>
- Spring Boot - <https://spring.io/projects/spring-boot>
- MyBatis - <https://www.baeldung.com/mybatis>
- General Doubts and Queries - <https://stackoverflow.com/>
- Understanding UML Class Diagrams - <https://creately.com/guides/class-diagram-relationships/>
- MySQL ER Diagram Generation - <https://www.mysql.com/products/workbench/design/>