# Setup Instructions

### Run Spring Boot application:

1. Make sure you are in the “*backend”* folder (“cd Backend”)
2. **./mvnw spring-boot:run**
   1. Or just click run on the IDE for FairSplitApplication.java
   2. When you run the Spring Boot application, it automatically creates the necessary tables/columns for you if you wrote the code properly.
   3. Should have this line:

“Started FairsplitApplication in 1.752 seconds (process running for 1.893)”

1. It should continue running until you exit, if it exits by itself, then there’s an error

### Connect database to workbench:

(or you can use mysql cli), same thing

1. Hostname: database-201.cx8g6s2wav7t.us-west-1.rds.amazonaws.com
2. Port: 3306
3. Username (no quotes): “admin”
4. Password (no quotes): “CSCI201Project”

\*Contact Minyu if issues with connecting to database

# Endpoints

BaseUrl: <http://localhost:8080/> (for local, or whatever endpoint we deploy on)

Implemented

In-progress

Not yet

| **Action** | **HTTP** | **Endpoint** | **Body** |
| --- | --- | --- | --- |
| **User** |  |  |  |
| Register User | POST | /api/users/register | {fullname, username, password} |
| Update User | PUT | /api/users/{userId} | {fullname, username, password} |
| Get User by id | GET | /api/users/{userId} |  |
| Get all users | GET | /api/users |  |
| Update user balance | PUT | /api/users/{userId}/balance | Double (e.g., 100.0) |
| Add/minus to user balance | PUT | /api/users/{userId}/add-balance | Double (e.g., 100.0)   * **+:** add, **-**: subtract |
| Delete User | DELETE | /api/users/{userId} |  |
| Get users in group | GET | /api/users/group/{groupId} |  |
| **Group** |  |  |  |
| Get Groups: get all groups | GET | /api/groups | N/A |
| Get Group by id | GET | /api/groups/{groupId} | N/A |
| Add Group | POST | /api/groups | {name, users, activeStatus} |
| Update Group | PUT | /api/groups/{groupId} | {name, activeStatus} |
| Delete Group | DELETE | /api/groups/{groupId} |  |
| Add user to group | POST | /api/groups/{groupId}/users/{userId} |  |
| Remove user to group | DELETE | /api/groups/{groupId}/users/{userId} |  |
| **Expense** |  |  |  |
| Get all expenses | GET | /api/expenses |  |
| Get expense by id | GET | /api/expenses/{expenseId} |  |
| Create expense | POST | /api/expense | {"name": "Dinner",  "amount": 100.0,  "payer": {"id": 1},  "group": {"id": 1}, "category": "Food", "assignedUsers":  [{"id": 1}, {"id": 2}]} |
| Edit expense | PUT | /api/expense/{expenseId} | Any fields that need change |
| Delete expense | DELETE | /api/expense/{expenseId} |  |
| **Request** |  |  |  |
| Get all requests | GET | /api/requests |  |
| Get request by id | GET | /api/requests/{requestId} |  |
| Edit request | PUT | /api/requests/{requestId} |  |
| Accept request (triggers payment) | PUT | /api/requests/{requestId}/accept |  |
| Delete request | DELETE | /api/requests/{requestId} |  |
| **Payment** |  |  |  |
| Get all payments | GET | /api/payments |  |
| Get payment by id | GET | /api/payments/{requestId} |  |
| Create payment | POST | /api/payments | **With request**  {"name": "Dinner payment", "amount": 50.0,  "debtor": {"id": 2},  "debtee": {"id": 1},  "group": {"id": 1},  "request": {"id": 1}}  **Standalone w/o request** {"name": "Direct payment", "amount": 25.0,  "debtor": {"id": 2},  "debtee": {"id": 1},  "group": {"id": 1}} |
| Delete payment | DELETE | /api/payments/{requestId} |  |
| Get payments by debtor (**borrower**) | GET | /api/payments/debtor/{userId} |  |
| Get payments by debtee (**lender**) | GET | /api/payments/debtee/{userId} |  |
| Get payments by group | GET | /api/payments/group/{groupId} |  |

# General Structure / Examples (actual project may differ)

**🌱 Spring Boot Naming Convention for FairSplit**

**📁 Package Structure (Standard Convention)**

com.fairsplit

│

├── controller → **REST** **endpoints** (that frontend fetches from)

├── service → Business logic (functions called by endpoints)

├── repository → Data access layer (JPA/JDBC)

├── model → Entity classes (User, Group, etc.)

├── config → Config files (e.g., DB, CORS)

└── FairSplitApplication.java (main application)

**Classes**

**Model Layer (Entities - in model)**

User.java

Group.java

Expense.java

Request.java

Payment.java

**Repository Layer (in repository)**

UserRepository.java

GroupRepository.java

ExpenseRepository.java

RequestRepository.java

PaymentRepository.java

**Service Layer (in service)**

XService.java

**Controller Layer (in controller)**

XController.java