COMSATS University Islamabad

Department of Computer Science

WEB TECHNOLOGIES

Assignment no. 3

Group members:
Ammar Rauf (FA23-BCS-059)
Kaab Bhinder (FA23-BCS-068)
Syeda Rubab (FA23-BCS-096)

Submitted to: Ma'am Sadia Maqbool

Class: BCS-5B **Due Date: 11-12-25**

Project Overview

ShareIt is a community-based borrow & lend system designed to reduce waste and unnecessary spending by facilitating the safe temporary sharing of physical items (tools, books, appliances) within a local network. The platform manages the entire lifecycle, from item listing and user authentication to the borrowing workflow, payment handling via a Wallet, and dispute resolution. The backend is built on REST principles and designed for implementation using FastAPI.

The system includes:

- **Lender:** Uploads and manages items for lending.
- **Borrower:** Requests, borrows, and returns items.
- **Admin:** Manages user disputes and system oversight.

The backend exposes REST APIs for:

- User authentication
- Item listing
- Borrowing workflow
- Wallet & payments
- Dispute management

Modules / Entities Overview

1. Users

- Authentication (register, login)
- Role management (borrower/lender/admin)

2. Items

- Add item
- Update item
- View item
- Delete/disable item

3. Bookings

- Borrow requests
- Accept / Reject
- Pickup & return confirmation

4.Wallet & Transactions

- Wallet balance
- Deposit locking
- Penalty / refund

5. Disputes

- File dispute
- Admin resolution

API Endpoints Specification

1. Authorization module

- **Register User**

Method: POST
Route: /authorization/register
Description: Create a new user account.

Request Body:

```
{
"full_name": "string",
"email": "string",
"password": "string",
"phone": "string",
"address": "string",
"role": "borrower"
 }
```

Response:

```
{
 "user_id": 1,
 "full_name": "string",
 "email": "string",
 "role": "borrower"
}
```

Errors:

- 400 Missing fields
- 409 Email already exists

- **Login**

Method: POST
Route: /auth/login

<u>Request:</u>

```
{
 "email": "string",
 "password": "string"
 }
```

<u>Response:</u>

```
{
 "access_token": "jwt-token",
 "token_type": "bearer"
}
```

2. <u>Item module</u>

1. Create Item

Method: POST
Route: /items/
Auth: Lender

<u>Request:</u>

```
{
  "title": "Drill Machine",
  "description": "working fine",
  "condition": "good",
  "estimated_price": 4000,
  "min_days": 1,
  "max_days": 7,
  "daily_deposit": 150,
  "images": ["url1","url2"],
  "location": "Johar Town"
}
```

<u>Response (201):</u>

```
{
  "item_id": 10,
  "title": "Drill Machine",
  "lender_id": 3
}
```

2. Get All Items

Method: GET
Route: /items/

Response:

```
[
  {
    "item_id": 10,
    "title": "Drill Machine",
    "condition": "good",
    "images": ["url1"]
  }
]
```

3. Get Single Item

GET /items/{item_id}

4. Update Item

PATCH /items/{item_id}
Authorization: Lender owns item

5. Delete (Disable) Item

DELETE /items/{item_id}

3. Booking module

1. Create Borrow Request

Method: POST
Route: /bookings/
Auth: Borrower

Request:

```
{
  "item_id": 10,
  "start_date": "2025-12-10",
  "end_date": "2025-12-12",
  "reason": "Need for small repair work"
}
```

Response (201):

```
{
  "booking_id": 21,
  "status": "pending"
}
```

2. Lender Accept / Reject

PATCH /bookings/{booking_id}/decision

Request:

```
{
 "status": "accepted"
}
```

Response:

```
{ "message": "Booking accepted" }
```

3. Confirm Pickup

PATCH /bookings/{id}/pickup

Response:

```
{
  "status": "pickup"
}
```

4. Confirm Return

PATCH /bookings/{id}/return

Response:

```
{
  "status": "returned"
}
```

4. Wallet & transactions module

1. View Wallet Balance

GET /wallet/

Response:

```
{
  "balance": 1200
}
```

2. Add Money

POST /wallet/add

Request:

```
{ "amount": 500 }
```

3. View All Transactions

GET /transactions/

5. Dispute module

1. File Dispute

POST /disputes/
Auth: Lender

Request:

```
{
  "booking_id": 21,
  "description": "Item returned with broken handle",
  "estimated_cost": 500
}
```

2. Admin Resolve

PATCH /disputes/{id}/resolve

Request:

```
{
  "status": "resolved"
}
```

4.Authentication Requirements

Module	Role Required
Register/Login	Public
Create Item	Lender
Update/Delete Item	Item owner
Create Booking	Borrower
Accept/Reject Booking	Lender
Wallet	Owner
Create Dispute	Lender
Resolve Dispute	Admin

Authentication is done using **JWT Bearer Token**.

5. Error Handling Structure

Every error returns:

```
{
  "detail": "Error message"
}
```

Code	Meaning
400	Invalid input
401	Unauthorized (no token)
403	Forbidden (wrong role)
404	Resource not found
409	Conflict (duplicate email etc.)