

# Fintech Application API Documentation & Testing Guide

## Table of Contents

1. [Application Architecture](#)

2. [UI Access](#)

3. [Authentication Flow](#)

4. [API Endpoints](#)

5. [Testing Workflow](#)

6. [Postman Setup](#)

7. [Sample Test Data](#)

8. [Error Handling](#)

---

## Application Architecture

The fintech application is built on a microservices architecture with 5 independent services:

Service	Port	Purpose
Auth Service	3001	User authentication and authorization
Accounts Service	3002	Account management and balance operations
Transfer Service	3003	Fund transfers and transactions
Ledger Service	3004	Double-entry bookkeeping records
Consumer Service	N/A	Kafka event consumer for analytics

---

## UI Access

**Admin UI:** <http://localhost:8081>

Features available:

- Toggle mock scenarios (success, slow, error)
  - View audit logs
  - Check system health
-

## Authentication Flow

All services except the Auth Service require JWT token-based authentication.

### Step 1: Register a User

```
POST http://localhost:3001/auth/register
```

```
Content-Type: application/json
```

```
{
  "email": "test@example.com",
  "password": "password123"
}
```

### Step 2: Login to Get Tokens

```
POST http://localhost:3001/auth/login
```

```
Content-Type: application/json
```

```
{
  "email": "test@example.com",
  "password": "password123"
}
```

### Response:

```
json
{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": 1,
    "email": "test@example.com",
    "roles": ["user"]
  }
}
```

### Step 3: Use Access Token

Add the access token to the Authorization header for all authenticated requests:

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

## API Endpoints

### Auth Service (<http://localhost:3001>)

#### POST /auth/register

Register a new user

#### Request:

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

#### POST /auth/login

User login and token generation

#### Request:

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

**Response:** Returns accessToken, refreshToken, and user details

#### POST /auth/refresh

Refresh expired access token

#### Request:

```
json
```

```
{  
  "refreshToken": "string"  
}
```

## GET /health

Health check endpoint

---

## Accounts Service (<http://localhost:3002>)

### GET /accounts

List all user accounts

**Authentication:** Required **Response:** Array of user accounts

---

### GET /accounts/:id/balance

Get balance for specific account

**Authentication:** Required **Path Parameters:**

- `[id]`: Account ID

**Response:** Account balance information

---

### POST /accounts/:id/deposit

Deposit funds into account

**Authentication:** Required **Path Parameters:**

- `[id]`: Account ID

**Request:**

```
json
```

```
{  
  "amount": "number",  
  "description": "string"  
}
```

---

## POST /accounts/:id/withdraw

Withdraw funds from account

### Authentication: Required Path Parameters:

- `[id]`: Account ID

### Request:

```
json  
{  
  "amount": "number",  
  "description": "string"  
}
```

---

## GET /health

Health check endpoint

---

## Transfer Service (<http://localhost:3003>)

### POST /transfer

Initiate a fund transfer

### Authentication: Required Headers:

- `[Idempotency-Key]`: Unique identifier to ensure idempotent transfers

### Request:

```
json
```

```
{  
  "fromAccountId": "number",  
  "toAccountId": "number",  
  "amount": "number"  
}
```

---

## GET /transfer/:id/status

Get transfer status

### Authentication: Required Path Parameters:

- `[id]`: Transfer ID
- 

## GET /transactions

Get paginated transaction history

### Authentication: Required Query Parameters:

- `[page]`: Page number (optional)
  - `[size]`: Page size (optional)
- 

## POST /documents/upload

Upload supporting documents

### Authentication: Required Content-Type: Form-data Form Parameters:

- `[document]`: File to upload
- 

## POST /webhook/payment

Payment webhook endpoint

### Headers:

- `[X-Signature]`: Signature string for verification
-

## **GET /external/otp/verify**

Internal OTP verification endpoint

---

## **POST /external/payment/process**

Internal payment processing endpoint

---

## **GET /admin/migrations**

Check database migration status

**Authentication:** Required

---

## **GET /admin/audit-logs**

View audit logs

**Authentication:** Required

---

## **GET /health**

Health check endpoint

---

## **Ledger Service (<http://localhost:3004>)**

### **GET /ledger/accounts/:accountId**

Get ledger entries for account

**Authentication:** Required **Path Parameters:**

- `accountId`: Account ID

**Query Parameters:**

- `page`: Page number (optional)
  - `size`: Page size (optional)
- 

### **GET /ledger/accounts/:accountId/transactions**

Get transactions for account

### Authentication: Required Path Parameters:

- `accountId`: Account ID

### Query Parameters:

- `page`: Page number (optional)
  - `size`: Page size (optional)
- 

## GET /health

Health check endpoint

---

## Testing Workflow

### Step 1: Start Services

```
bash
cd infra
docker-compose up
```

### Step 2: Run Database Setup

```
bash
cd ../scripts
npm run migrate
npm run seed
```

### Step 3: Test Authentication

1. Register: `POST http://localhost:3001/auth/register`
2. Login: `POST http://localhost:3001/auth/login`
3. Copy the `accessToken` from the response

### Step 4: Test Account Operations

1. Get accounts: `GET http://localhost:3002/accounts` (add Authorization header)
2. Check balance: `GET http://localhost:3002/accounts/1/balance` (add Authorization header)

## Step 5: Test Transfer

1. Initiate transfer: `POST http://localhost:3003/transfer` (add Authorization and Idempotency-Key headers)
  2. Check status: `GET http://localhost:3003/transfer/1/status` (add Authorization header)
- 

## Postman Setup

### Environment Variables

Create a Postman environment with the following variables:

```
base_url_auth: http://localhost:3001
base_url_accounts: http://localhost:3002
base_url_transfer: http://localhost:3003
base_url_ledger: http://localhost:3004
access_token: (fill after login)
refresh_token: (fill after login)
```

### Authorization

1. Go to the **Authorization** tab in Postman
  2. Select **Bearer Token**
  3. Enter `{{access_token}}`
- 

## Sample Test Data

After running database seeding, use these credentials:

### Admin User

- Email: `admin@example.com`
- Password: `admin123`

### Regular Users

- Email: `user1@example.com` / Password: `admin123`
- Email: `user2@example.com` / Password: `admin123`

## Sample Accounts

Account	Owner	Balance
ACC001	<a href="mailto:user1@example.com">user1@example.com</a>	\$1,000.00
ACC002	<a href="mailto:user2@example.com">user2@example.com</a>	\$2,500.00
ACC003	<a href="mailto:user1@example.com">user1@example.com</a>	\$5,000.00

## Error Handling

### HTTP Status Codes

Code	Meaning	Description
200	OK	Successful request
201	Created	Resource successfully created
400	Bad Request	Validation errors or malformed request
401	Unauthorized	Missing or invalid authentication token
404	Not Found	Resource does not exist
409	Conflict	Duplicate resource or conflict detected
429	Too Many Requests	Rate limit exceeded
500	Internal Server Error	Server-side error occurred

### Common Error Response Format

```
json

{
  "error": "Error message",
  "code": "ERROR_CODE",
  "status": 400
}
```