

Django-Based Payment Service System Documentation

1. Introduction

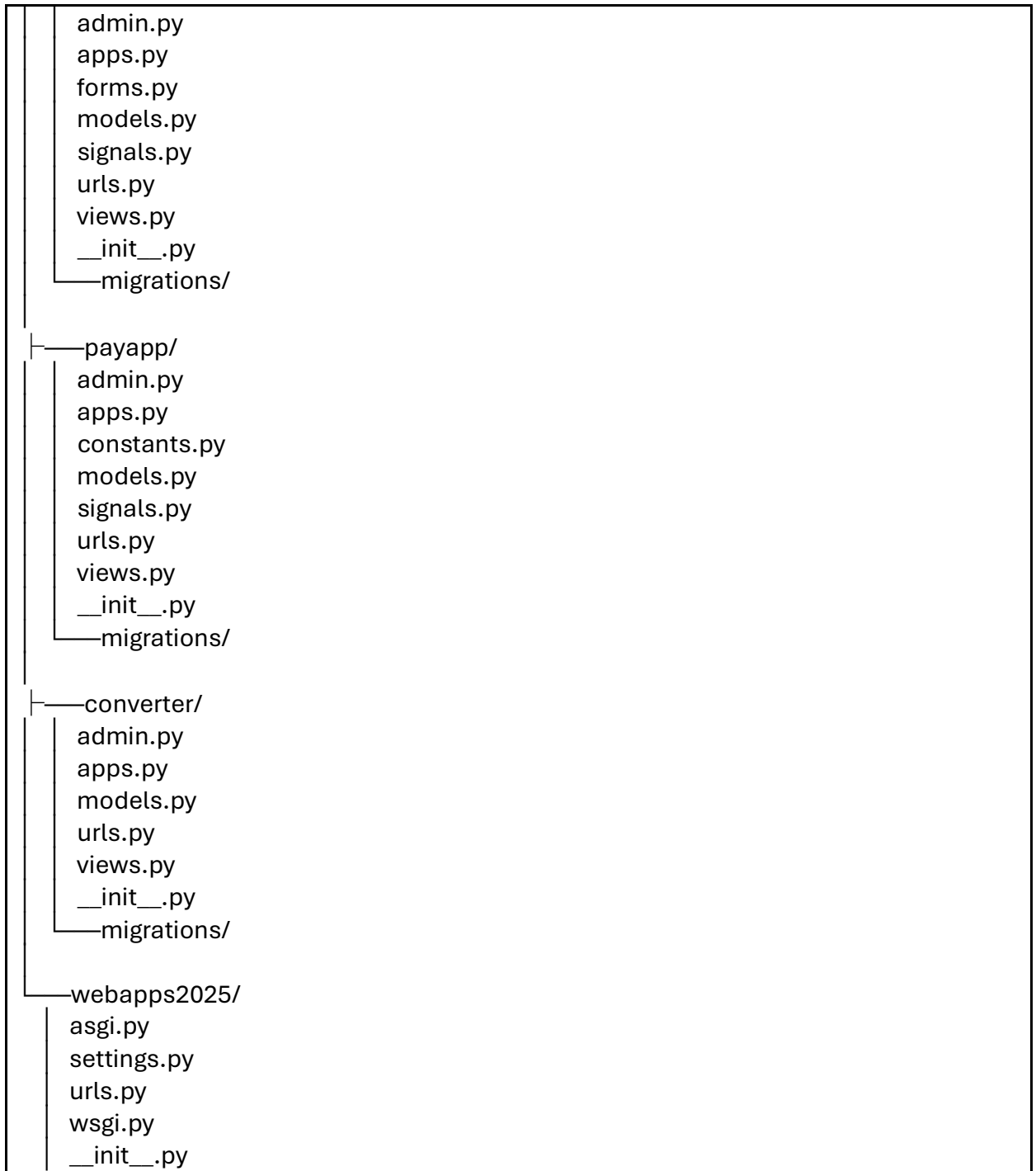
The WebApps2025 Payment System is a simplified PayPal-like web application built with Django that allows users to send and request money in multiple currencies (GBP, USD, EUR). The system includes user authentication, transaction processing, currency conversion, and administrative functions.

2. System Architecture

- The application is structured following Django's MVC paradigm:
- Presentation Layer: HTML templates with Bootstrap and Crispy Forms
- Business Logic Layer: Views and forms for handling application workflows
- Data Access Layer: Django models and ORM managing data interactions
- Security Layer: Authentication, authorization, CSRF, and other web security features
- Web Services Layer: REST API for currency conversion

2.1 Project Structure:

```
webapps2025/  
├── webapps.db  
├── manage.py  
├── setup.py  
├── templates/  
│   ├── payapp/  
│   │   ├── admin_dashboard.html  
│   │   ├── dashboard.html  
│   │   ├── make_payment.html  
│   │   └── request_payment.html  
│   ├── registration/  
│   │   ├── logged_out.html  
│   │   ├── login.html  
│   │   └── register.html  
└── register/
```



3. Presentation Layer (Implemented)

- Implemented HTML templates for all core pages:
 1. Home, Register, Login, Dashboard
 2. Send Payment, Request Payment
 3. Admin Dashboard, User List, Transaction List
- Navigation through Django's URL routing
- Bootstrap 5 and Crispy Forms used for UI and form formatting
- All pages validate inputs and are responsive

4. Business Logic Layer (Implemented)

- Django views handle:
 1. User and admin registration/login/logout
 2. Sending and requesting payments with balance validation
 3. Viewing notifications and transactions
- All transactions are atomic using @transaction.atomic
- Currency conversion during transactions
- Notifications sent to users for every financial action

5. Data Access Layer (Implemented)

- Models:
 1. UserProfile - stores currency, balance, linked to Django User
 2. Transaction - logs all payment and request transactions
 3. Notification - records of user activity logs
- Data stored using SQLite via Django ORM
- Admin interface registered for all models

6. Security Layer (Implemented)

- Authentication and role-based access control via decorators
- CSRF protection on all forms (enabled by default)
- Secure password storage using Django's hashers
- Session-based login, logout, access control

- Default middleware protects from XSS, SQL Injection, and Clickjacking

7. Web Services (Implemented)

- RESTful endpoint: /conversion/<currency1>/<currency2>/<amount>
- Returns JSON response with converted amount
- Uses static exchange rates (can be extended to use live APIs)
- Example:

GET <http://127.0.0.1:8000/conversion/GBP/USD/100/>

```
Response: {
  "converted_amount": "125.00",
  "rate": "1.250000",
  "from_currency": "GBP",
  "to_currency": "USD",
  "last_updated": "2025-04-10T20:46:32.367525+00:00"
}
```

8. Deployment of AWS (Not Implemented)

9. Setup Instructions

9.1 Installation

1. Create and activate virtual environment:

- `python -m venv venv`
- `source venv/bin/activate` # Linux/Mac
- `venv\Scripts\activate` # Windows

2. Install requirements:

- `pip install django crispy-bootstrap5 requests`

3. Run migrations:

- `python manage.py makemigrations`
- `python manage.py migrate`

4. Create initial data:

- `python setup.py`

9.2 Running the Application

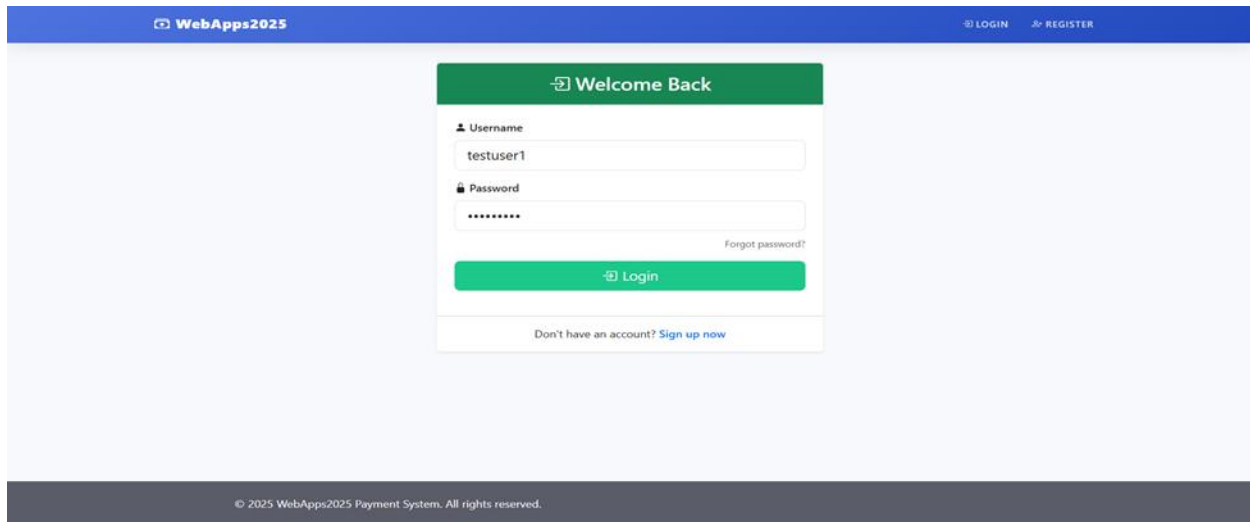
- `python manage.py runserver`

Access the application at: <http://localhost:8000>

Admin login: admin1

Admin Password: admin1

10. Screenshots and Demo



Login Form for User1

WebApps2025

LOGINREGISTER

Create Your Account

Username

testuser2

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email Address

test2@gmail.com

We'll never share your email with anyone else.

Password

Confirm Password

Enter the same password as before, for verification.

Account Currency

US Dollar (USD)

Register

OR

Already have an account?

Login Here

© 2025 WebApps2025 Payment System. All rights reserved.

Registration form for user 2

WebApps2025

DASHBOARDSEND MONEYPAY REQUEST MONEY

TESTUSER1

Registration successful! You are now logged in.

Account Balance

£750.00

Send MoneyRequest

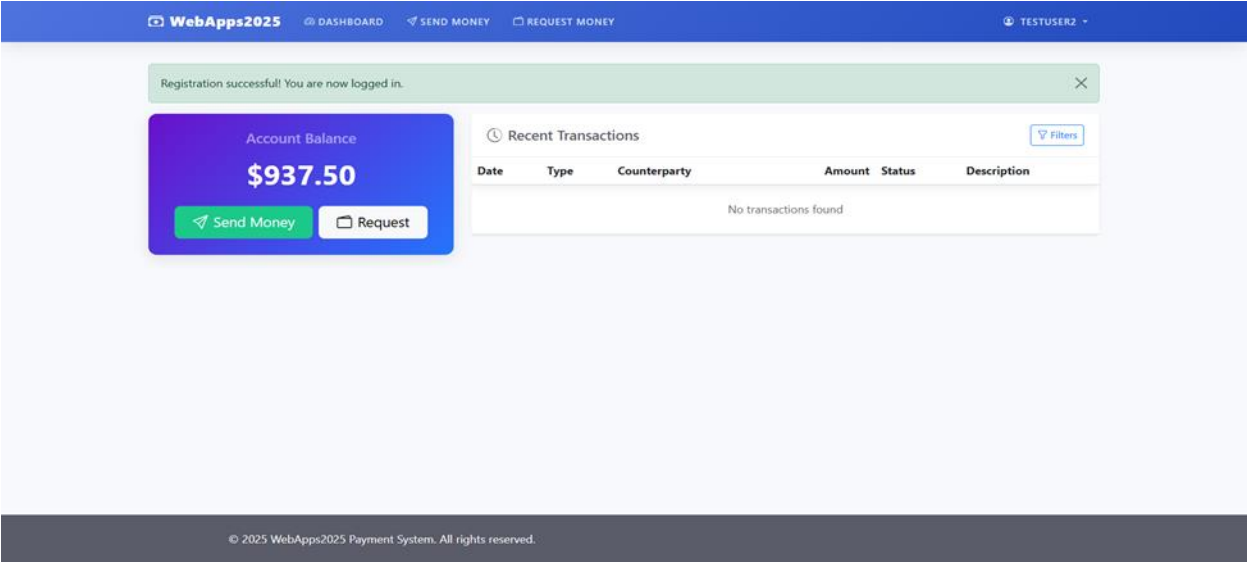
Recent Transactions

Filters

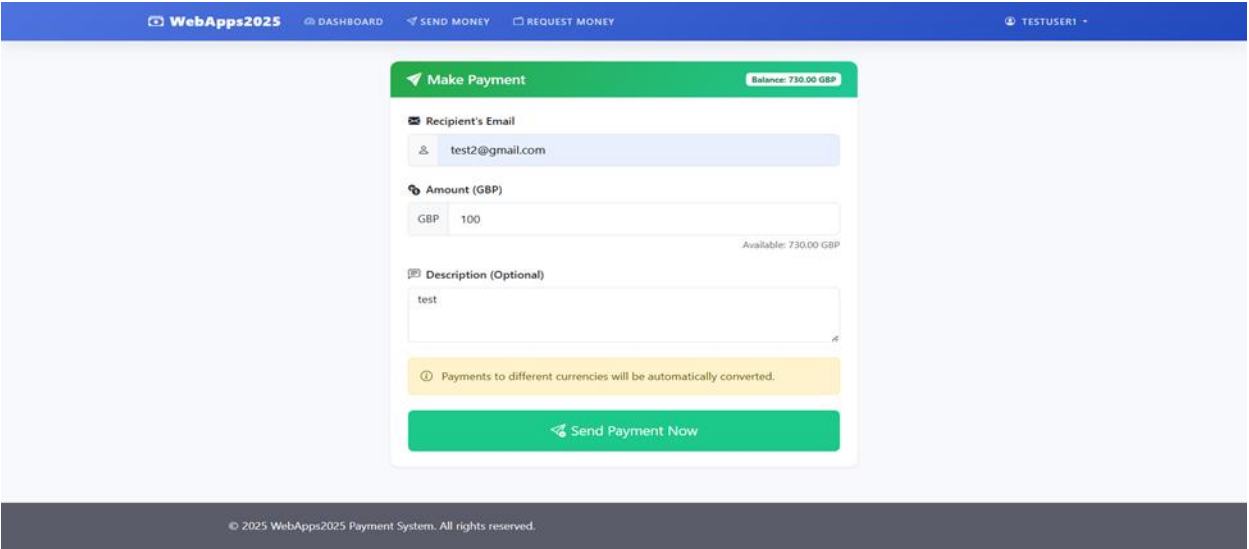
Date	Type	Counterparty	Amount	Status	Description
No transactions found					

© 2025 WebApps2025 Payment System. All rights reserved.

GBP User Dashboard for user1



USD user2 Dashboard for user2



Make payment from user1 to user2

WebApps2025

DASHBOARDSEND MONEYREQUEST MONEY

TESTUSER2

Payment of 100 USD sent successfully!

Account Balance

\$837.50

Send Money

Request

Recent Transactions

Date	Type	Counterparty	Amount	Status	Description
Apr 11, 12:02	Payment	To: testuser1	-\$100.00	Completed	test

© 2025 WebApps2025 Payment System. All rights reserved.

Complete transaction of 100\$ Payment from user2 to user1

WebApps2025

DASHBOARDSEND MONEYREQUEST MONEY

TESTUSER1

Account Balance

£830.00

Send Money

Request

Recent Transactions

Date	Type	Counterparty	Amount	Status	Description
Apr 11, 12:02	Payment	From: testuser2	+\$100.00	Completed	test

© 2025 WebApps2025 Payment System. All rights reserved.

Recived payment from user1 and automatically convert into GBP.

WebApps2025

DASHBOARD

SEND MONEY

REQUEST MONEY

TESTUSER2

Request Payment

Balance: 962.50 USD

Recipient's Email

test1@gmail.com

Amount (USD)

USD100

Available: 962.50 USD

Description (Optional)

test

Send Payment Request

© 2025 WebApps2025 Payment System. All rights reserved.

Request payment from user2 to user1

WebApps2025

DASHBOARD

SEND MONEY

REQUEST MONEY

TESTUSER1

Account Balance

£830.00

Send Money

Request

Pending Requests

\$100.00

0 minutes ago

testuser2 is requesting payment from you

test

Decline

Accept

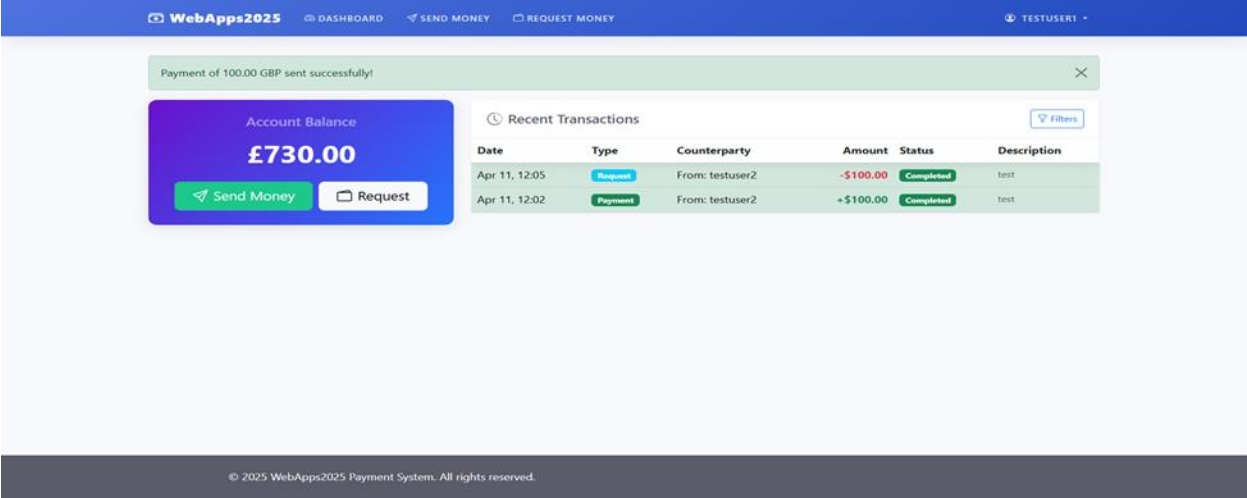
Recent Transactions

Filters

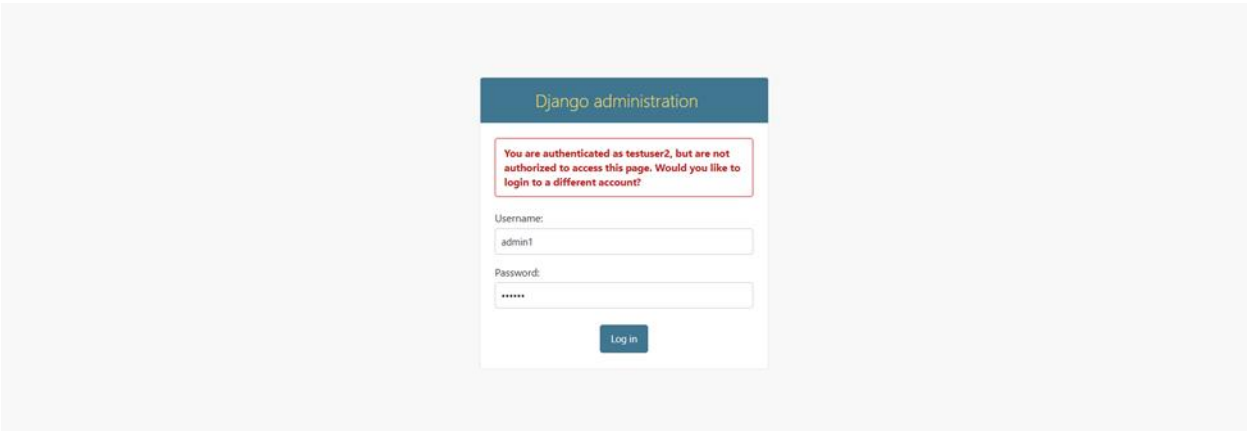
Date	Type	Counterparty	Amount	Status	Description
Apr 11, 12:05	Request	From: testuser2	\$100.00	Pending	test
Apr 11, 12:02	Payment	From: testuser2	+\$100.00	Completed	test

© 2025 WebApps2025 Payment System. All rights reserved.

User2 Dashboard as receiving payment request pending from user2



Successful Transaction



Admin Login Form

Django administration

WELCOME, ADMIN1. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Payapp > Transactions

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

CONVERTER

Conversion Rates + Add

PAYAPP

Transactions + Add

User profiles + Add

REGISTER

Registration requests + Add

Select transaction to change

Q

Search

2025

April 10

April 11

Action:

Go

0 of 4 selected

ID	SENDER	RECIPIENT	AMOUNT	CURRENCY	TRANSACTION TYPE	STATUS	CREATED AT
4	testuser2 (test2@gmail.com)	testuser1 (test1@gmail.com)	100.00 USD	US Dollar (\$)	Request	Completed	April 11, 2025, 12:05 p.m.
3	testuser2 (test2@gmail.com)	testuser1 (test1@gmail.com)	100.00 USD	US Dollar (\$)	Payment	Completed	April 11, 2025, 12:02 p.m.
2	user2 (user2@test.com)	user3 (user3@test.com)	50.00 USD	US Dollar (\$)	Request	Pending	April 10, 2025, 8:46 p.m.
1	user1 (user1@test.com)	user2 (user2@test.com)	100.00 GBP	British Pound (£)	Payment	Completed	April 10, 2025, 8:46 p.m.

4 transactions

FILTER

Show counts

By transaction type

All

Payment

Request

By status

All

Pending

Completed

Declined

By currency

All

British Pound (£)

US Dollar (\$)

Euro (€)

By created at

Any date

Today

Past 7 days

This month

This year

Complete Transaction Details from the Admin

← ↻ ⓘ 127.0.0.1:8000/conversion/GBP/USD/100/

Pretty-print ✓

```
{
  "converted_amount": "125.00",
  "rate": "1.250000",
  "from_currency": "GBP",
  "to_currency": "USD",
  "last_updated": "2025-04-10T20:46:32.367525+00:00"
}
```

Example of the REST API Currency Conversion

11. Implementation Summary Table

Section	Status	Comments
Presentation Layer	Fully Implemented	All user/admin templates connected, Bootstrap styling for clean interface.
Business Logic Layer	Fully Implemented	Validations, workflows, atomic transactions
Data Access Layer	Fully Implemented	SQLite models, relationships defined, Data integrity maintained.
Security Layer	Fully Implemented	Login, CSRF/XSS, role-based access, protections
Web Services	Fully Implemented	REST conversion service with static exchange rates
AWS Deployment		