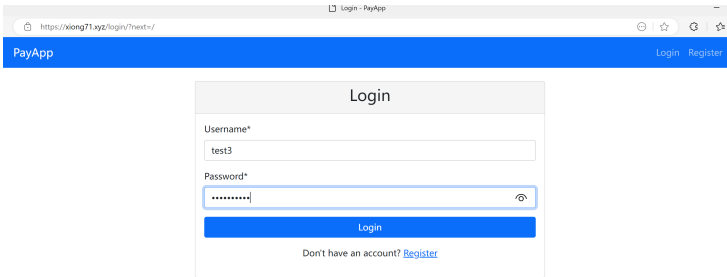


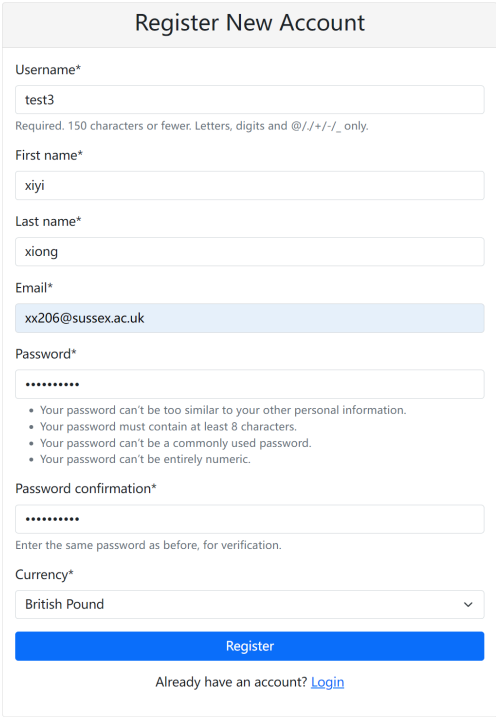
# WebPay2025: An Online Payment System Based on Django - Project Report

## 1. Presentation Layer Implementation

This system is built using the Django framework, providing users with an intuitive and efficient interface covering registration, login, dashboard display, transaction management, and notifications. The system utilizes the Bootstrap framework for responsive design, and all forms are implemented with Django form system and Crispy Forms components for validation and aesthetic enhancement.



The screenshot shows a web browser window with the URL `https://xiong71.xyz/login/?next=/`. The page has a blue header with the text "PayApp" on the left and "Login Register" on the right. The main content area is a light gray box titled "Login". It contains two input fields: "Username\*" with the value "test3" and "Password\*" with masked characters "\*\*\*\*\*". Below the password field is a blue "Login" button. At the bottom of the box, there is a link: "Don't have an account? [Register](#)".



The screenshot shows a web form titled "Register New Account". It contains several input fields: "Username\*" with the value "test3", "First name\*" with the value "xiyi", "Last name\*" with the value "xiong", "Email\*" with the value "xx206@sussex.ac.uk", "Password\*" with masked characters "\*\*\*\*\*", and "Password confirmation\*" with masked characters "\*\*\*\*\*". Below the password fields, there are four bullet points: "Your password can't be too similar to your other personal information.", "Your password must contain at least 8 characters.", "Your password can't be a commonly used password.", and "Your password can't be entirely numeric." Below these is a dropdown menu for "Currency\*" with the value "British Pound". At the bottom of the form is a blue "Register" button. Below the button, there is a link: "Already have an account? [Login](#)".

After logging in, users can access a feature-rich dashboard that clearly displays account balance, recent transactions, and received notifications. Payment functions include both direct payment and

payment request options, while the transaction history and notification system allow users to view and manage all financial activities at any time.

PayApp

DashboardTransaction HistorySend PaymentRequest PaymentNotifications

test

Account Information

750.00 GBP

Account Holder: xi yi xiong

Email: xx206@sussex.ac.uk

Send PaymentRequest Payment

Recent Transactions

No transaction records yet.

Send Payment

Recipient's Email\*

1249192949@qq.com

Amount\*

100

Send Payment

Back to Dashboard

Request Payment

Sender's Email\*

1249192949@qq.com

Amount\*

66

Request Payment

Back to Dashboard

PayApp

DashboardTransaction HistorySend PaymentRequest PaymentNotifications

test750.00 GBP (user account currency)Support

Notifications

Payment Request

You requested 66.00 GBP from 1249192949@qq.com

Status: Pending

Payment Notification

You sent 100.00 GBP to 1249192949@qq.com

Status: Completed

Transaction Details

Notification ID: 18

Date: 2025-04-06 19:39:54

Read Status: Unread

Transaction Information:

Transaction ID: 7

Type: REQUEST

Amount: 66.00 GBP

Status: PENDING

Sender: 1249192949@qq.com

Receiver: xx206@sussex.ac.uk

Date: 2025-04-06 19:39:54

Close

Mark All as Read

2025-04-06 19:39

View Details

2025-04-06 19:32

View Details

PayApp

Dashboard

Transaction History

Send Payment

Request Payment

Notifications

test1 (783.00 ( user:account:currency ))

Logout

Payment request accepted successfully.

Notifications

Mark All as Read

Payment Request

You accepted the payment request from xx206@sussex.ac.uk

2025-04-06 19:49

Completed

View Details

Payment Request

xx206@sussex.ac.uk requested 66.00 GBP from you

2025-04-06 19:39

Completed

View Details

Payment Notification

xx206@sussex.ac.uk sent you 100.00 GBP

2025-04-06 19:32

Completed

View Details

Payment Request

1341272263@qq.com rejected your payment request

2025-04-05 14:50

Rejected

View Details

Payment Request

You requested 500.00 GBP from 1341272263@qq.com

2025-04-05 14:48

Rejected

View Details

Payment Request

You accepted the payment request from 1341272263@qq.com

2025-04-05 14:47

Completed

View Details

Payment Request

1341272263@qq.com requested 50.00 GBP from you

2025-04-05 11:02

Completed

View Details

Payment Notification

1341272263@qq.com sent you 50.00 GBP

2025-04-05 11:02

Completed

View Details

The system provides administrators with a dedicated interface to implement global user and transaction management and monitoring, including viewing all user account information, monitoring all transactions in the system, and adding other administrator accounts.

PayApp

Dashboard

Transaction History

Send Payment

Request Payment

Notifications

Admin Dashboard

All Transactions

Register Admin

admin1 (750.00 ( user:account:currency ))

Logout

Total Users

8

Total Transactions

7

Admin Actions

Register New Admin

View All Transactions

User Accounts

Back to Admin Dashboard

Username	Email	Balance	Currency	Status
admin1	admin1@example.com	750.00	GBP	Active Admin
test_user	xx206@sussex.ac.uk	751.00	GBP	Active
test1	1249192949@qq.com	783.00	GBP	Active
test2	x71xxy@gmail.com	975.00	USD	Active
1	11@qq.com	877.50	EUR	Active
niuniububu	1341272263@qq.com	750.00	GBP	Active
admin2	123456@qq.com	975.00	USD	Active Admin
test3	xx206@sussex.ac.uk	716.00	GBP	Active

All TransactionsBack to Admin Dashboard

ID	Date	Type	Sender	Receiver	Amount	Status	Details
7	2025-04-06 19:39	Request	1249192949@qq.com	xx206@sussex.ac.uk	66.00 GBP	Completed	View Details
6	2025-04-06 19:32	Payment	xx206@sussex.ac.uk	1249192949@qq.com	100.00 GBP	Completed	View Details
5	2025-04-05 14:48	Request	1341272263@qq.com	1249192949@qq.com	500.00 GBP	Rejected	View Details
4	2025-04-05 11:02	Request	1249192949@qq.com	1341272263@qq.com	50.00 GBP	Completed	View Details
3	2025-04-05 11:02	Payment	1341272263@qq.com	1249192949@qq.com	50.00 GBP	Completed	View Details
2	2025-03-28 21:01	Request	xx206@sussex.ac.uk	1249192949@qq.com	1.00 GBP	Pending	View Details
1	2025-03-28 21:01	Payment	1249192949@qq.com	xx206@sussex.ac.uk	1.00 GBP	Completed	View Details

The system URL structure is designed with clarity and follows RESTful design principles, mainly including paths for home page, dashboard, user authentication, and transaction management.

```
urlpatterns = [
    path('', views.home_view, name='home'), path('dashboard/', views.dashboard_view, name='dashb
    path('register/', views.register_view, name='register'), path('login/', views.login_view, na
    path('logout/', views.logout_view, name='logout'), path('transactions/', views.transactions_
    path('send-payment/', views.send_payment_view, name='send_payment'),
    path('request-payment/', views.request_payment_view, name='request_payment'),
]
```

## 2. Business Logic Layer Implementation

The business logic layer is the core of the system, responsible for handling all business rules and data operations. All key logic is implemented using Django view functions, with transaction mechanisms ensuring data consistency.

The user dashboard view aggregates key information, including account balance, recent transactions, and unread notifications:

```
@login_required
def dashboard_view(request):
    account = request.user.account
    transactions = Transaction.objects.filter(Q(sender=request.user) | Q(receiver=request.user))
    notifications = Notification.objects.filter(user=request.user, is_read=False).order_by('-timestamp')
    return render(request, 'payapp/dashboard.html', {
        'account': account, 'transactions': transactions, 'notifications': notifications,
        'unread_count': notifications.count()
    })
```

Transaction processing involves multiple steps such as account balance verification, currency conversion, transaction record creation, and notification sending, all encapsulated within database transactions to ensure data consistency:

```
@login_required
def send_payment_view(request):
    with transaction.atomic():
        # Currency conversion (if necessary)
        if sender_account.currency != recipient_account.currency:
            converted_amount = convert_currency(sender_account.currency, recipient_account.currency, amount)
        # Create transaction and update balances
        transaction_obj = Transaction.objects.create(sender=request.user, receiver=recipient, amount=amount)
        sender_account.balance -= amount
        recipient_account.balance += converted_amount
        # Create notifications
        Notification.objects.create(user=recipient, transaction=transaction_obj)
        Notification.objects.create(user=request.user, transaction=transaction_obj)
```

The payment request function allows users to initiate payment requests to other users, who can choose to accept or reject. The system manages the entire lifecycle of requests through status tracking mechanisms, ensuring each request is appropriately handled.

Payment Request

xx206@sussex.ac.uk requested 66.00 GBP from you

Accept

Reject

Status: Pending

2025-04-06 19:39

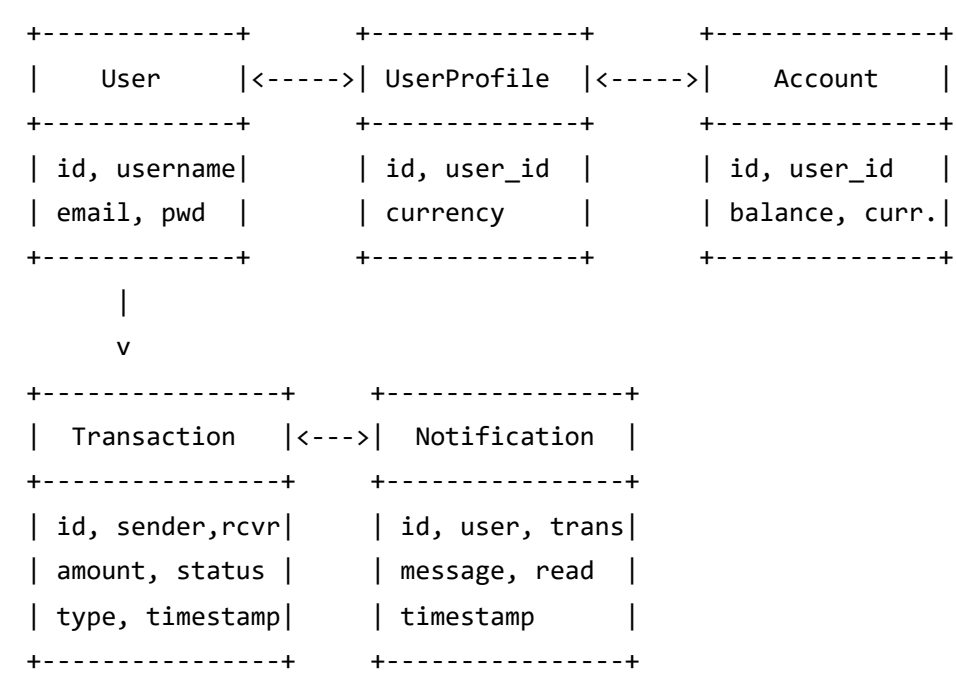
View Details

The administrator function module provides system management and monitoring capabilities, including user management, transaction monitoring, and system maintenance. The administrator dashboard aggregates system data, providing key metrics such as user count and transaction volume. The administrator registration function allows existing administrators to create new administrator accounts.

Auxiliary functions such as the currency conversion service support cross-currency transaction capabilities through an internal API, supporting conversions between the three main currencies: British Pounds, US Dollars, and Euros.

### 3. Data Access Layer Implementation

The system's data access layer is implemented based on Django ORM, adopting a clear model structure and relationship design. Core data models include the User model, UserProfile model, Account model, Transaction model, and Notification model.



The UserProfile model extends Django's built-in User model, adding currency preference settings to support users' selection of one of the three main currencies (GBP, USD, or EUR) as their default currency. The Account model corresponds one-to-one with users, storing the user's account balance and current currency type. The implementation code for these two models is as follows:

```

class UserProfile(models.Model):
    CURRENCY_CHOICES = [('GBP', 'British Pound'), ('USD', 'US Dollar'), ('EUR', 'Euro')]
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='profile')
    currency = models.CharField(max_length=3, choices=CURRENCY_CHOICES, default='GBP')

class Account(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='account')
    balance = models.DecimalField(max_digits=10, decimal_places=2)
    currency = models.CharField(max_length=3)

```

The Transaction model records detailed information of all transaction operations, while the Notification model implements the system's internal messaging mechanism. The system uses Django signal mechanisms to automatically create initial accounts and implements multiple query optimization measures, including indexes, queryset method chains, and pagination mechanisms.

## 4. Web Services Implementation

The system implements a complete RESTful API service, primarily used for currency conversion functionality. The API design follows REST principles, transmitting parameters required for conversion through URL path parameters. The API endpoint format is

/conversion/{currency1}/{currency2}/{amount} , for example, /conversion/GBP/USD/100 will return the result of converting 100 British Pounds to US Dollars.



The API implementation combines Django view functions with HTTP method decorators, ensuring only GET requests are accepted. The function internally implements currency conversion calculations through a hardcoded exchange rate dictionary, supporting conversions between the three main currencies (GBP, USD, and EUR) in the system. The implementation code is as follows:

```

@csrf_exempt
@require_http_methods(["GET"])
def conversion_view(request, currency1, currency2, amount):
    # Exchange rates dictionary
    exchange_rates = {'GBP_USD': Decimal('1.30'), 'GBP_EUR': Decimal('1.17'), 'USD_GBP': Decimal('0.77')}
    # Conversion logic
    rate_key = f"{currency1}_{currency2}"
    rate = exchange_rates.get(rate_key)
    converted_amount = Decimal(amount) * rate
    return JsonResponse({
        'from_currency': currency1, 'to_currency': currency2, 'from_amount': amount,
        'converted_amount': str(converted_amount.quantize(Decimal('0.01'))), 'exchange_rate': str(rate)
    })

```

The business logic layer calls the conversion API through client functions, integrating API calls, response parsing, and error handling logic to provide a stable and reliable service interface.

## 5. Security Implementation

The system's security implementation adopts a multi-layered defense strategy, including three main aspects: authentication authorization, security configuration, and Web attack defense.

The authentication system is based on Django's built-in framework, implementing secure login verification and session management. All passwords are stored after processing with the PBKDF2 algorithm and SHA256 hash function, using random salt values to enhance security. Access control implements permission checks through the decorator mechanism:

```

@login_required
def admin_dashboard_view(request):
    if not request.user.is_staff: return redirect('dashboard')
    # Administrator function implementation...

```

In terms of security configuration, the system enables multiple security middlewares, including HTTPS redirection, XSS protection, and CSRF protection. Production environment security settings include:

```

SECURE_SSL_REDIRECT = True           # Force HTTPS
SESSION_COOKIE_SECURE = True         # Secure Cookie
CSRF_COOKIE_SECURE = True           # Secure CSRF Token
X_FRAME_OPTIONS = 'DENY'            # Prevent clickjacking

```

Password policy is enforced through Django's password validators, requiring users to create strong passwords and preventing the use of common passwords:

```
AUTH_PASSWORD_VALIDATORS = [
    {'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator'},
    {'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator', 'OPTIONS': {'min_
    {'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator'},
    {'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator'},
]
```

The system has also configured Content Security Policy (CSP), limiting the sources of executable scripts and loaded resources, effectively preventing XSS and other injection attacks:

```
# CSP configuration
CSP_DEFAULT_SRC = ('self',)
CSP_STYLE_SRC = ('self', 'https://stackpath.bootstrapcdn.com') # Bootstrap CDN
CSP_SCRIPT_SRC = ('self', 'https://code.jquery.com') # jQuery CDN
CSP_FONT_SRC = ('self', 'https://stackpath.bootstrapcdn.com') # Font CDN
CSP_IMG_SRC = ('self', 'data:') # Allow data URI images
```

## 5.3 Web Attack Defense Measures

The system takes multiple measures to defend against common Web attacks:

**CSRF Protection:** All forms use CSRF tokens for protection, preventing cross-site request forgery attacks. Each POST request must include a valid CSRF token to be processed:

```
<form method="post" action="{% url 'send_payment' %}">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Send Payment</button>
</form>
```

**SQL Injection Defense:** The system relies entirely on Django ORM and parameterized queries, avoiding the risk of direct SQL string concatenation:

```
# Secure: user = User.objects.get(email=email) # Parameterized query
# Insecure: cursor.execute("SELECT * FROM users WHERE email = '" + email + "'") # SQL injection
```



**XSS Defense:** Django's template system automatically escapes all variable outputs, ensuring user input is appropriately processed before rendering:

```
<!-- Variables automatically escaped: <p>{{ user_message }}</p> -->
<!-- Only safe content: <div>{{ safe_html|safe }}</div> -->
```

**Clickjacking Protection:** Prevents the website from being embedded in iframes of malicious sites by setting appropriate X-Frame-Options headers.

**Sensitive Data Protection:** All transaction data and user information are encrypted via HTTPS during transmission, ensuring data transmission security.

**Transaction Integrity:** All operations involving data changes are encapsulated in database transactions, ensuring the atomicity of operations and data integrity, preventing data inconsistency due to partial operation failures:

```
with transaction.atomic():
    transaction_obj = Transaction.objects.create(...)    # Create transaction record
    sender_account.balance -= amount; sender_account.save()    # Update sender
    receiver_account.balance += converted_amount; receiver_account.save()    # Update receiver
    Notification.objects.create(...)    # Create notification
```

Through the comprehensive application of these security measures, the system effectively defends against common Web application security threats, safeguarding user data security and transaction reliability.

## 6. AWS Deployment Implementation

The system has been successfully deployed on the AWS cloud platform, adopting a standard web application deployment architecture. Core components include EC2 instances, Nginx reverse proxy, Gunicorn WSGI server, and SQLite database, with HTTPS communication implemented through SSL certificates obtained from Let's Encrypt.

Instance summary for i-0e586274d50258f80 [Info](#)

Updated less than a minute ago

Instance ID

i-0e586274d50258f80

IPv6 address

–

Hostname type

IP name: ip-172-31-84-158.ec2.internal

Answer private resource DNS name

IPv4 (A)

Auto-assigned IP address

–

IAM Role

–

IMDSv2

Required

Operator

–

Public IPv4 address

3.226.139.61 | [open address](#)

Instance state

Running

Private IP DNS name (IPv4 only)

ip-172-31-84-158.ec2.internal

Instance type

t2.micro

VPC ID

vpc-09ca9eaa741b6341b | [open address](#)

Subnet ID

subnet-0389d6c42ea6c6b41 | [open address](#)

Instance ARN

arn:aws:ec2:us-east-1:730335632087:instance/i-0e586274d50258f80

Private IPv4 addresses

172.31.84.158

Public IPv4 DNS

ec2-3-226-139-61.compute-1.amazonaws.com | [open address](#)

Elastic IP addresses

3.226.139.61 [Public IP]

AWS Compute Optimizer finding

[Opt-in to AWS Compute Optimizer for recommendations.](#) | [Learn more](#)

Auto Scaling Group name

–

Managed

false

Last login: Sat Apr 5 14:38:44 2025 from 31.205.226.8

ubuntu@ip-172-31-84-158:~\$ sudo systemctl status unicorn

• unicorn.service - unicorn daemon for webapps2025

Loaded: loaded (/etc/systemd/system/unicorn.service; enabled; preset: enabled)

Active: active (running) since Sun 2025-04-06 19:02:37 UTC; 4min 15s ago

Main PID: 519 (unicorn)

Tasks: 4 (limit: 1129)

Memory: 99.0M (peak: 100.1M)

CPU: 811ms

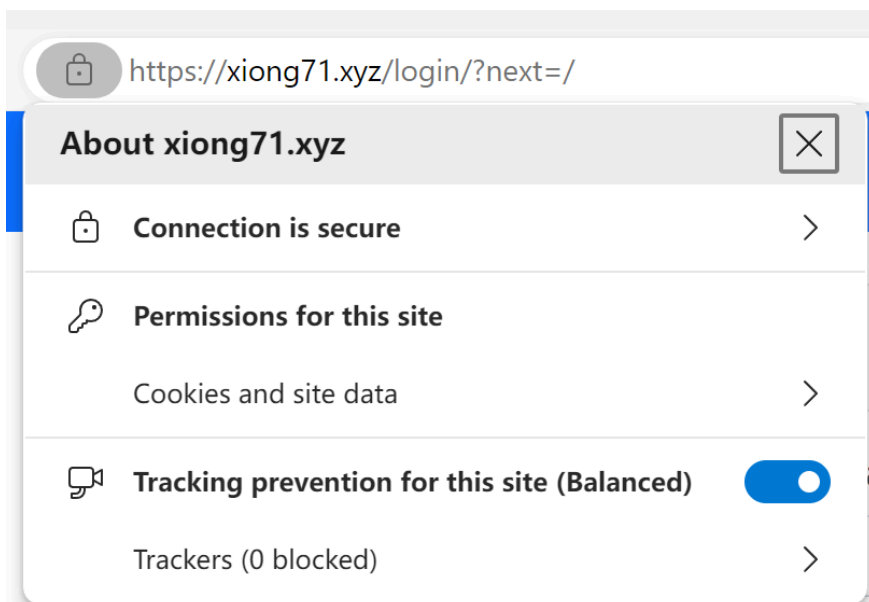
CGroup: /system.slice/unicorn.service

└─519 /home/ubuntu/webapps2025/venv/bin/python3 /home/ubuntu/webapps2025/venv/bin/unicorn --access-logfile /home/ubuntu/webapps2025/unicorn-access.log --error-logfile /home/ubuntu/webapps2025/unicorn-error.log

└─775 /home/ubuntu/webapps2025/venv/bin/python3 /home/ubuntu/webapps2025/venv/bin/unicorn --access-logfile /home/ubuntu/webapps2025/unicorn-access.log --error-logfile /home/ubuntu/webapps2025/unicorn-error.log

└─777 /home/ubuntu/webapps2025/venv/bin/python3 /home/ubuntu/webapps2025/venv/bin/unicorn --access-logfile /home/ubuntu/webapps2025/unicorn-access.log --error-logfile /home/ubuntu/webapps2025/unicorn-error.log

└─779 /home/ubuntu/webapps2025/venv/bin/python3 /home/ubuntu/webapps2025/venv/bin/unicorn --access-logfile /home/ubuntu/webapps2025/unicorn-access.log --error-logfile /home/ubuntu/webapps2025/unicorn-error.log



The deployment process includes four main steps: EC2 instance creation, environment configuration, application deployment, and web service configuration. The system chooses Ubuntu Server 20.04 LTS as the operating system, configures security groups to open ports 22/80/443, and assigns an elastic IP (3.226.139.61) pointing to the domain xiong71.xyz. Key configurations include:

1. Gunicorn as the WSGI server, communicating with Nginx through a socket
2. Nginx handling static files and forwarding dynamic requests to Gunicorn
3. Let's Encrypt providing SSL certificates for HTTPS access
4. System service configuration ensuring automatic application startup and error recovery

# 7. User Guide

## 7.1 Regular User Function Guide

### 7.1.1 Registration and Login

#### New Account Registration:

1. Visit the website homepage <https://xiong71.xyz>
2. Click the "Register" link
3. Fill in username, password, and currency selection (GBP/USD/EUR)
4. Click the "Register" button to submit
5. After successful registration, the system will automatically create your profile and account, with an initial balance of 750 GBP or the equivalent in your selected currency

Register New Account

Username\*

test3

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

First name\*

xiyi

Last name\*

xiong

Email\*

xx206@sussex.ac.uk

Password\*

.....

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation\*

.....

Enter the same password as before, for verification.

Currency\*

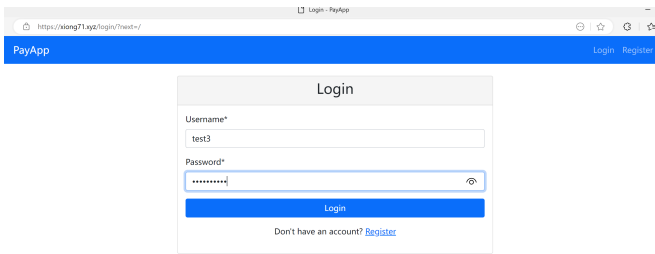
British Pound

Register

Already have an account? [Login](#)

#### User Login:

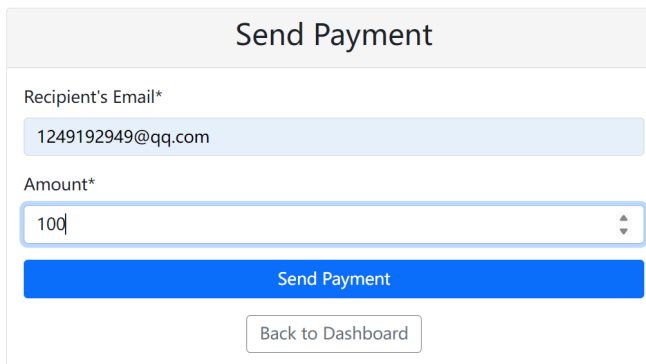
1. Visit the homepage, click the "Login" link
2. Enter your username and password, click the "Login" button
3. After successful login, you will enter the user dashboard



## 7.1.2 Core Function Operations

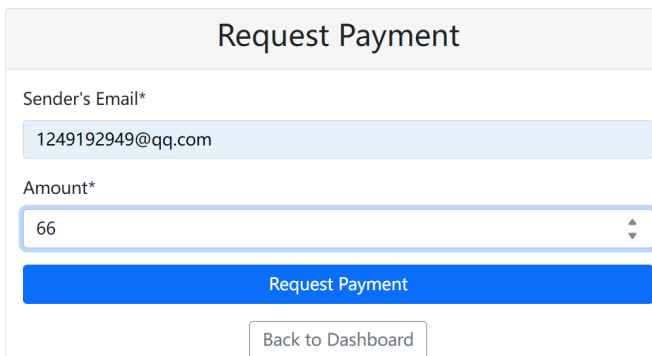
### Send Payment:

1. Click "Send Payment", enter the recipient's username and amount
2. The system will verify the recipient exists and your balance is sufficient
3. Click the "Send" button to confirm, after the transaction is completed, both parties' balances will be updated and notifications sent



### Request Payment:

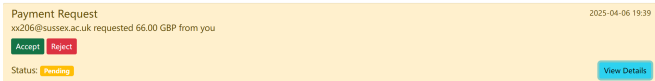
1. Click "Request Payment", enter the payer's username and amount
2. Click "Request" to submit, the system creates a transaction record with "PENDING" status
3. The system sends a notification to the payer



### Process Payment Requests:

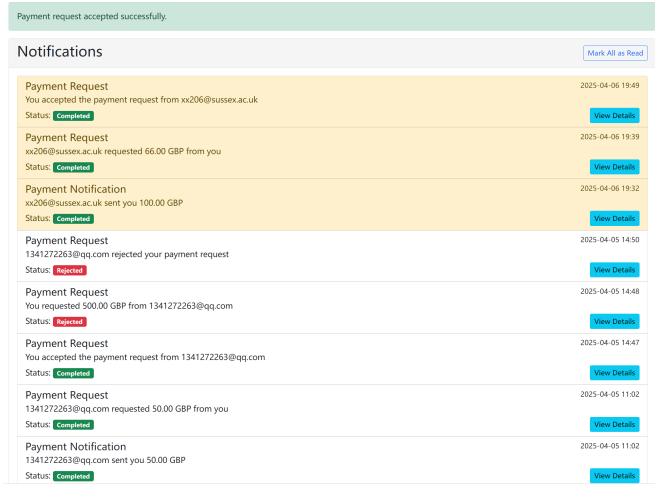
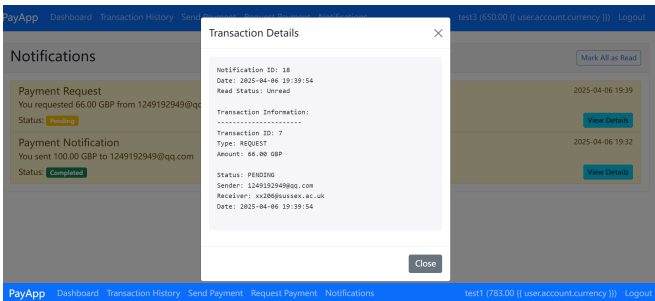
1. View received requests in notifications or transaction history

- 2. Click "Accept" to accept or "Reject" to decline
- 3. If accepted, the system automatically completes the fund transfer; if rejected, only the status is updated



**View Notifications and Transaction History:**

- 1. Click "Notifications" to view all notifications, which can be marked as read
- 2. Click "Transaction History" to view complete transaction records
- 3. Click "View Details" to see details of any transaction



## 7.2 Administrator Function Guide

Administrators have higher permissions and can access additional functions by logging into the system with an administrator account:

**User Management:**

- 1. Access the administrator dashboard to view a list of all users
- 2. Information includes username, email, account balance, currency type, etc.

PayApp

Dashboard

Transaction History

Send Payment

Request Payment

Notifications

Admin Dashboard

All Transactions

Register Admin

admin1 (750.00 { user:account:currency })

Logout

Total Users

8

Total Transactions

7

Admin Actions

Register New Admin

View All Transactions

User Accounts				
Username	Email	Balance	Currency	Status
admin1	admin1@example.com	750.00	GBP	<span>Active</span> <span>Admin</span>
test_user	xx206@sussex.ac.uk	751.00	GBP	<span>Active</span>
test1	1249192949@qq.com	783.00	GBP	<span>Active</span>
test2	x71xxy@gmail.com	975.00	USD	<span>Active</span>
1	11@qq.com	877.50	EUR	<span>Active</span>
niuniububu	1341272263@qq.com	750.00	GBP	<span>Active</span>
admin2	123456@qq.com	975.00	USD	<span>Active</span> <span>Admin</span>
test3	xx206@sussex.ac.uk	716.00	GBP	<span>Active</span>

## Transaction Monitoring:

1. View all system transaction records
2. Each record contains transaction ID, both parties' users, amount, type, status, etc.

All Transactions

Back to Admin Dashboard

ID	Date	Type	Sender	Receiver	Amount	Status	Details
7	2025-04-06 19:39	<span>Request</span>	1249192949@qq.com	xx206@sussex.ac.uk	66.00 GBP	<span>Completed</span>	<span>View Details</span>
6	2025-04-06 19:32	<span>Payment</span>	xx206@sussex.ac.uk	1249192949@qq.com	100.00 GBP	<span>Completed</span>	<span>View Details</span>
5	2025-04-05 14:48	<span>Request</span>	1341272263@qq.com	1249192949@qq.com	500.00 GBP	<span>Rejected</span>	<span>View Details</span>
4	2025-04-05 11:02	<span>Request</span>	1249192949@qq.com	1341272263@qq.com	50.00 GBP	<span>Completed</span>	<span>View Details</span>
3	2025-04-05 11:02	<span>Payment</span>	1341272263@qq.com	1249192949@qq.com	50.00 GBP	<span>Completed</span>	<span>View Details</span>
2	2025-03-28 21:01	<span>Request</span>	xx206@sussex.ac.uk	1249192949@qq.com	1.00 GBP	<span>Pending</span>	<span>View Details</span>
1	2025-03-28 21:01	<span>Payment</span>	1249192949@qq.com	xx206@sussex.ac.uk	1.00 GBP	<span>Completed</span>	<span>View Details</span>

## Administrator Registration:

1. Select the "Register Admin" option
2. Fill in new administrator information, click "Create Admin"
3. The system automatically assigns administrator permissions and initial balance

Register New Admin

Username\*

admin3

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

First name\*

Last name\*

Email\*

Password\*

Your password can't be too similar to your other personal information.

Your password must contain at least 8 characters.

Your password can't be a commonly used password.

Your password can't be entirely numeric.

Password confirmation\*

Enter the same password as before, for verification.

Currency\*

US Dollar

Register Admin

Back to Admin Dashboard