

# Web Applications and Services Reports

## Contents

Presentation.....	2
What can the users do? .....	2
Users Viewing all their transactions.....	3
Users Making direct payments to other registered users. ....	3
Users request payments from other registered users.....	4
What can the administrator do?.....	4
Can administrators view user accounts? .....	4
Can administrators view all payment transactions?.....	4
Business logic layer .....	5
Data access layer.....	5
Security layer .....	6
Authentication Functionality .....	6
Registration .....	6
Login.....	8
Logout .....	9
Access control to restrict access to web pages to non-authorised users.....	10
Communication on top of HTTPS for every interaction with users and admins .....	10
Cross-site scripting (XSS), Cross-site request forgery (CSRF), SQL injection, and Clickjacking protection .....	10
Code Explanation .....	11
HTML Pages.....	11
Databases.....	11
Models .....	12
URLS .....	12
Views.....	13
Web services .....	13
Manual on how to use. ....	13
Step 1: Run the website using IntelliJ. ....	13
Step 2: Registration.....	15
Step 3: Login.....	16
Step 4: Website Navigation.....	16
Step 4.1: Home Page explanation.....	17

Step 5: Send Money .....	17
Step 5.1: View Transactions .....	17
Step 6: Request Money .....	18
Step 6.1: View Request .....	18
Step 7: Logout .....	19
Database .....	20
Admin .....	22
Figure 1 Viewing All transactions .....	3
Figure 2 Sending Money .....	4
Figure 3 Requesting Money .....	4
Figure 4 Admin Viewing all transactions .....	5
Figure 5 Register Part 1 Found Left on Lower Hand .....	7
Figure 6 Registration Page .....	8
Figure 7 What Can be seen when logging in .....	9
Figure 8 Logout button .....	9
Figure 9 What is seen when logout is pressed .....	10
Figure 10 Manual 1: User Homepage upon loading up IntelliJ .....	14
Figure 11 Manual 2: Registration Page .....	15
Figure 12 Manual 3: Login Page .....	16
Figure 13 Navigation bar .....	16
Figure 14 Manual 4: Send Money page .....	17
Figure 15 Manual 5: View Transactions Page .....	17
Figure 16 Manual 6: Request Money .....	18
Figure 17 Manual 7: Requests page when there are not requested to show. ....	18
Figure 18 Manual 8: Request page when there are requested to show. ....	19
Figure 19 Manual 9: Home Page .....	19
Figure 20 Auth User Database .....	20
Figure 21 Pay app Transfer .....	21
Figure 22 Register account database .....	22

## Presentation

### What can the users do?

The developed web application gives customers a platform to carry out fundamental banking operations, like sending and receiving money. To make use of the functions offered, users can register for an account on the platform and securely log in to their account.

A user can examine their account balance and transaction history after logging in. By sending money to another platform user, they can also start a new transaction. The system will perform the transaction, debiting the sender's account balance and crediting the recipient's account balance after the user enters the amount they wish to transfer.

The platform also allows users to view transaction requests sent to them by other users of the platform. If a user receives a transaction request, they can either accept or reject the request. If they accept the request, the system will process the transaction as described above. If they reject the request, no transaction will be processed.

The platform includes features to ensure the security of the user's accounts and transactions. For example, users are required to log in with their username and password, which is verified against the stored credentials in the database. The platform also uses Cross-Site Request Forgery (CSRF) protection to prevent unauthorized access to user accounts.

Users Viewing all their transactions.

The user can access the "Transactions" page by clicking on the link in the navigation bar and viewing all their transactions there. The transactions connected to the user's account will be listed on this page, together with information on each transaction's value, the user who initiated it, and the date it was made. To view previous transactions and review their specifics, the user can navigate through the list. The user can simply keep track of their transactions thanks to this feature, which gives them a clear overview of their financial behaviour on the web app.

WebApps Home Logout Welcome, kj265 Send Money Request Money View Transactions View Requests

## Transactions

### Incoming Transactions

- Amount: 50.00, Sender: ziad, Receiver: kj265, Date: May 2, 2023, 3:19 p.m.
- Amount: 6.00, Sender: ziad, Receiver: kj265, Date: May 3, 2023, 4:27 p.m.

### Outgoing Transactions

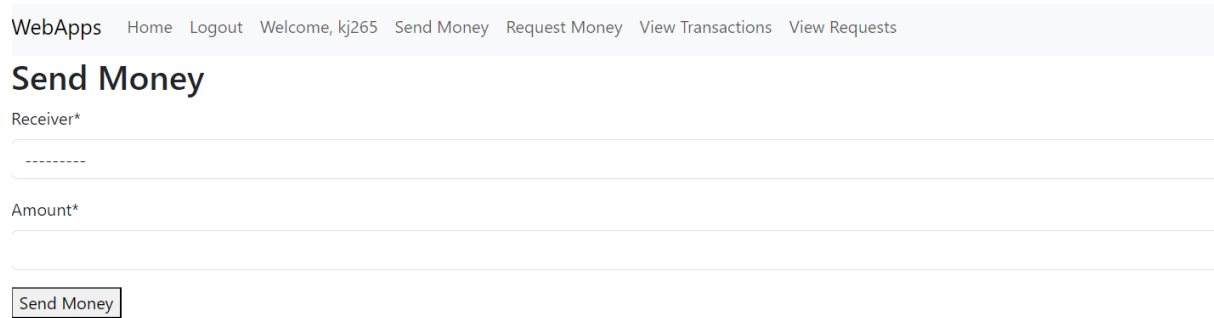
- Amount: 10.00, Sender: kj265, Receiver: imbatman, Date: May 2, 2023, 2:53 p.m.
- Amount: 25.00, Sender: kj265, Receiver: admin1, Date: May 2, 2023, 3:06 p.m.
- Amount: 25.00, Sender: kj265, Receiver: admin1, Date: May 2, 2023, 3:08 p.m.
- Amount: 50.00, Sender: kj265, Receiver: ziad, Date: May 3, 2023, 4:22 p.m.
- Amount: 9.00, Sender: kj265, Receiver: ziad, Date: May 3, 2023, 4:27 p.m.

Figure 1 Viewing All transactions.

Users Making direct payments to other registered users.

To make direct payments to other registered users in the web app, the user needs to navigate to the "Send Money" page. On this page, the user can select the recipient of the payment from a dropdown list of registered users. They can then enter the amount they wish to send and submit the form. If the form is valid, a new transaction model will be created and saved to the database, and the recipient will receive the payment. Both the sender and recipient can view this transaction on their "Transactions" page, which shows a history of all their transactions. This feature allows for secure and efficient peer-to-peer payments between registered users of the web app.

Candidate number: 268940



WebApps Home Logout Welcome, kj265 Send Money Request Money View Transactions View Requests

## Send Money

Receiver\*

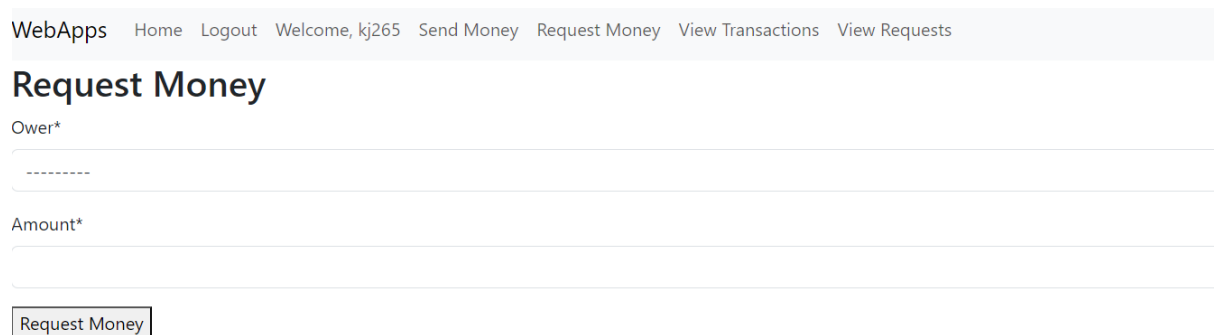
Amount\*

Send Money

Figure 2 Sending Money

Users request payments from other registered users.

The web app also allows users to request payments from other registered users. To request a payment, the user must navigate to the "Request Payment" page and enter the recipient's username, the amount they wish to receive, and a description of the payment. Once the user submits the request, it will be added to the list of pending requests that can be viewed on the "View Requests" page. Other users will be able to accept or reject the request, and the user will receive a notification informing them of the decision. If the request is accepted, the user will receive the payment, and the transaction will be recorded in the database. If the request is rejected, the user will be notified, and the request will be removed from the pending list.



WebApps Home Logout Welcome, kj265 Send Money Request Money View Transactions View Requests

## Request Money

Owner\*

Amount\*

Request Money

Figure 3 Requesting Money

What can the administrator do?

Can administrators view user accounts?

Can administrators view all payment transactions?

On the frontend from the administrator's point of view. The administrators can view all the transactions that are ongoing between the users. This includes payments requests both accepted

and pending. The administrator however cannot intervene and stop any of the transactions from taking place, they can only see the transactions which are currently ongoing.

WebApps Home Logout Welcome, admin1 Send Money Request Money View Transactions View Requests View All Transactions View All Requests

## All Transactions

- Amount: 10.00, Sender: kj265, Receiver: imbatman, Date: May 2, 2023, 2:53 p.m.
- Amount: 25.00, Sender: kj265, Receiver: admin1, Date: May 2, 2023, 3:06 p.m.
- Amount: 25.00, Sender: kj265, Receiver: admin1, Date: May 2, 2023, 3:08 p.m.
- Amount: 50.00, Sender: ziad, Receiver: kj265, Date: May 2, 2023, 3:19 p.m.
- Amount: 50.00, Sender: kj265, Receiver: ziad, Date: May 3, 2023, 4:22 p.m.
- Amount: 6.00, Sender: ziad, Receiver: kj265, Date: May 3, 2023, 4:27 p.m.
- Amount: 9.00, Sender: kj265, Receiver: ziad, Date: May 3, 2023, 4:27 p.m.

*Figure 4 Admin Viewing all transactions*

## Business logic layer

The website created follows the business logic layer as it separates the presentation layer from the model and database layer. The views contain the logic that accesses the model(s) and defers to the appropriate template(s), as they handle the business logic and interact with the database through the models.

The website also respects the transaction's ACID (Atomicity, Consistency, Isolation, Durability) properties. Database transactions are handled consistently and reliably thanks to ACID compliance. The transaction property ensures that all database updates will take place, or none at all. Because transactions can be rolled back or cancelled if a failure occurs, this property is crucial for data integrity.

The website's views support transactions, ensuring that data integrity is preserved. Any database transactions in your views are wrapped in a transaction atomic block, so they run as a single, indivisible unit of work. This ensures that the changes made to the database in one part of the view are either committed entirely or rolled back, which ensures data consistency.

## Data access layer

The website follows the recommended data access layer pattern. I have used Django's built-in ORM to define my models, which describe the essential fields and behaviours of the data being stored. I have used SQLite as my RDBMS, which simplifies deployment and configuration. SQLite is included in Python, so there is no need for additional installations or configurations.

After installing the web app, Django creates a database schema and a Python database-access API for accessing your objects. This makes it easy to interact with the database without writing raw SQL queries.

The views contain the business logic layer of the web app, which accesses the models and defers to the appropriate templates. The views support transactions, which helps to ensure data integrity. I have annotated your views with the appropriate transaction attributes to guarantee the ACID

properties. This ensures that your database transactions are atomic, consistent, isolated, and durable.

Overall, the web app follows the best practices recommended for the data access and business logic layers. By using Django's ORM and SQLite as the RDBMS.

## Security layer

### Authentication Functionality

The website contains a respectable authentication functionality. Following the authentication systems that used for another similar web application websites. The website requires the users to register before they can login. It provides a secure way for users to access their accounts and perform actions withing the application. The use of the Django authentication framework, along with the CSRF protect ensures that user data is protected, and data integrity is maintained.

### Registration

The registration process of a website is a crucial step in the user's journey. It is essential to provide a seamless and secure registration process to encourage users to sign up and use the website. On the website that has been created, the registration process is simple and straightforward, allowing new users to quickly create an account and start using the platform's features.

To register on the website, users must click on the 'Register' button on the homepage, which will direct them to a registration page. The registration page requires users to enter their username, email address, and password. Once the required information is provided, users must click on the 'Register' button to submit their registration request. After submitting their registration request, users will be redirected to a page indicating that their account has been created successfully.

To ensure the security of user data, the website uses Django's built-in authentication system, which provides secure storage and management of user account information. The authentication system stores user account information, including usernames and hashed passwords, in a database. The use of hashed passwords ensures that even if the database is compromised, user passwords remain secure.

In conclusion, the website's registration process is simple, secure, and efficient, allowing new users to quickly create an account and start using the platform's features.

# Login

Username\*

Password\*

[Register](#)

*Figure 5 Register Part 1 Found Left on Lower Hand*

WebApps [Login](#)

## User Registration

Username\*

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

Email\*

First name

Last name

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\*

Figure 6 Registration Page

### Login

The website's login feature is a crucial component that enables users to access their accounts and the services offered by the website. The website's login procedure is made to be simple, safe, and quick to use.

Users must go to the login page first, where they must enter their registered username and password, in order to log in. The user's credentials are verified by the website's authentication system after submission, and if they are accurate, access to the user's account is granted.

Most significantly, the database's password encryption keeps the login process secure. A user's password is hashed and safely kept in the database when they initially register on the website.



Because of this, even if a hacker manages to access the database, they won't be able to read the real passwords.

Overall, the login functionality of the website is a vital component of the user experience, ensuring that users can access their accounts securely and efficiently. By implementing security measures, such as password hashing, encryption, the website's login process ensures that user data is kept secure and protected from potential hackers.

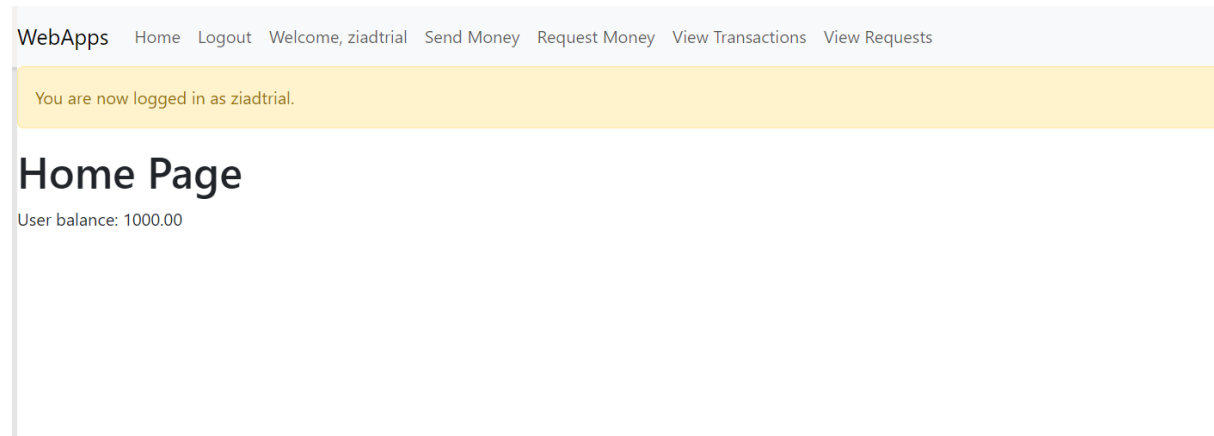


Figure 7 What Can be seen when logging in

## Logout

The logout feature in a website is essential for maintaining the security of the user's account. When a user logs out, it ends the session and ensures that no one else can access the user's account without proper authorization. The logout feature on the website that I have created is simple and intuitive for users.

When a user wants to log out, they can simply click on the "Logout" button on the navigation bar. The logout button is only visible when a user is logged in, and it disappears once the user logs out. Once the user clicks on the logout button, it redirects them to the logout view. The logout view clears the user's session, and it redirects them to the home page.

Messages are also incorporated into the website's logout mechanism to give users feedback. A notification indicating that the user has successfully logged out is sent to them once they log out. This notice appears on the home page for a brief period of time before disappearing. Users will have a great user experience thanks to this feedback function, which makes sure they are aware that they have been logged out.

Overall, the logout feature on the website I have created is easy to use and intuitive. It ensures the security of the user's account and provides feedback to the user to enhance their experience.

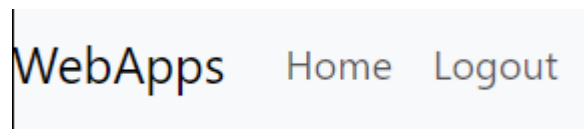


Figure 8 Logout button

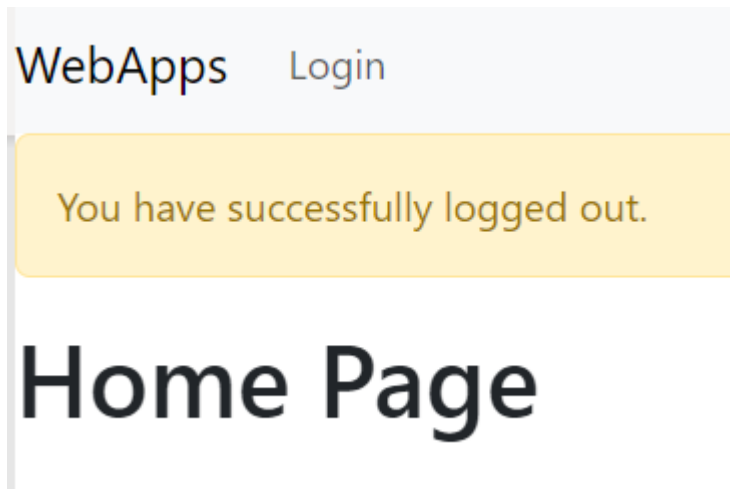


Figure 9 What is seen when logout is pressed

### Access control to restrict access to web pages to non-authorised users.

Access control is a safety measure that restricts access to internet resources to users who have been given authorization. In other words, it prevents unauthorised users from accessing certain parts of the website. This is accomplished using user authentication, which requires users to submit a valid login and password in order to access restricted pages. If the user is not authenticated, the login page will appear. After properly authenticating, the user is granted access to the restricted pages they are authorised to view. This ensures that only individuals with the appropriate authorisation can read it and helps protect sensitive information.

### Communication on top of HTTPS for every interaction with users and admins

Unfortunately, the website doesn't use HTTPS for secure communication with users and administrators, making private information like login information and financial transactions susceptible to hacking and interception. Without HTTPS, hackers may be able to view or change the data being transmitted, which might result in identity theft, financial loss, and unauthorised access to sensitive data. Even though the website is protected by other security measures, the absence of HTTPS leaves it open to network-based attacks. To guarantee the privacy, authenticity, and integrity of data transferred over the internet, HTTPS must be used in all communications with users and administrators.

### Cross-site scripting (XSS), Cross-site request forgery (CSRF), SQL injection, and Clickjacking protection

The website is protected from various assaults that can be harmful to users or the website itself.

- Cross-site request forgery (CSRF): This attack involves deceiving a user into sending a form without their knowledge or consent. To avoid this, the website has put protections in place. It is placed in many sections of the code. Below is an example of the CRSF token being used.

```
{% extends 'base.html' %}
{% load crispy_forms_filters %}

{% block content %}
    <h2>Login</h2>
    {% if form.errors %}
        <p>Your username and password didn't match. Please try again.</p>
    {% endif %}
    <form method="post">
        {% csrf_token %}
        {{ login_user | crispy | safe }}
        <button type="submit">Login</button>
        <a href="{% url 'register_user' %}">Register</a>
    </form>
{% endblock %}
```

- SQL injection: This attack involves the introduction of malicious code into a website's SQL database, which can be exploited to retrieve sensitive data afterwards. To avoid this, the website has put protections in place.
- Communication of HTTPS: Unfortunately HTTPS was not implemented
- XSS: Partially implemented.

Overall, these safeguards contribute to the security and safety of the website and its visitors.

## Code Explanation

### HTML Pages

The user interface of the web application is provided by the HTML pages in the code. The visual design, text, photos, and interactive components that a user interacts with through a web browser are all contained in them.

Based on the information and reasoning offered by the views and models in the application's backend, the HTML pages are created dynamically. HTML pages that may display dynamic content, such as user data or payment transactions, are created using the Django template engine.

Each HTML page has a specific purpose and functionality, such as allowing the user to register or login, displaying payment transactions, or allowing the user to send money to another user. They are designed to be easy to read and use, and to provide a seamless and intuitive user experience.

A template file called "base.html" provides the framework for subsequent web pages. It includes the primary HTML code that is shared by all pages on the website, including the header, navigation bar, and footer. Other web pages can inherit the structure of 'base.html' by utilising it as a starting point, then change or add content as necessary. When building many pages with similar design, this helps to assure consistency and save time.

Overall, the HTML pages are essential to the web application's operation and usability and are a fundamental element in creating a satisfying user experience.

### Databases

Databases are used in the web application to store data and information for a variety of purposes. The web application makes use of several databases, including Django's default SQLite database, which holds data about user accounts, transactions, and requests. Fields that specify the data to be saved, such as the sender and recipient of a transaction, the amount being transferred, and the date of the transaction, are included in the models used to generate database tables. These fields make sure that all pertinent data is present and maintain the data's integrity.

The web application makes use of the databases to carry out several operations, including fetching user account data and showing transaction history. Additionally, authentication data, including usernames, passwords, and other sensitive data, are stored and managed in the database. The web application interacts with the database using SQL queries to get data, add new data entries, and modify or remove existing data.

Overall, the web application's databases are essential for organising, storing, and retrieving the data that is necessary for the programme to work properly. Transactional data, user account data, and other crucial data are among the items kept in the databases. To make sure that the online application runs smoothly and safely, the databases are utilised in conjunction with SQL queries and other database management strategies.

## Models

Models are the foundation of database-driven web applications in the Django framework. Classes called models are used to communicate with database tables. The fields of the table are described, along with their names, data types, and connections to other tables. Developers may quickly change the data in the database using Python code by using models.

The `models.py` file located in the `payapp` folder defines the models used in the online application. The app uses the `Request` and `Transaction` models.

The amount, the requester, the owner, and the timestamp are just a few of the columns that the `Request` model specifies for the request table in the database. Additionally, it specifies how to store the request object in the database.

The `Transaction` model defines the fields for the transaction table in the database, including the sender, the receiver, the amount, and the date. It also defines a method to save the transaction object to the database.

By using these models, developers can easily add, modify, or delete data in the database by writing Python code. The Django ORM (Object-Relational Mapping) handles the low-level database operations such as creating tables, inserting data, updating data, and querying data. This makes it easier for developers to work with the database and to keep their code organized.

## URLS

In a Django web application, URLs are used to map the requests from the user's web browser to the appropriate view function that should handle the request. In other words, URLs act as a router that determines which view should be called based on the URL that the user requested.

In the context of a Django application, URLs are defined in the `urls.py` file of each app. In this file, URL patterns are defined using regular expressions to match the requested URL to a particular view function. When a request is made, Django looks for the matching URL pattern in the `urls.py` file and calls the corresponding view function to generate a response.

The URL routing process starts at the project's `urls.py` file, which contains the base URL patterns for the entire project. These base URL patterns then include the URL patterns of each app in the project, which are defined in the respective `urls.py` files of each app.

URLs in a Django web application can also include parameters, which are passed to the corresponding view function as arguments. These parameters can be used to identify specific objects or to provide additional context to the view.

In summary, URLs in a Django web application are used to map the user's request to the appropriate view function that can handle the request. They are defined using regular expressions and can include parameters that are passed to the view function as arguments.

## Views

Views are the bridge between the models and the templates in a web application. They handle requests from the user and decide what data to show in the response. Views also contain the logic to access the model data and make it available to the template. For example, if a user wants to see their account details, they will send a request to the view associated with the account page. The view would then retrieve the necessary account data from the model and pass it to the account template, which would display it to the user.

In short, views control what data is presented to the user and how it is presented. They are the main controllers of the web application's functionality.

## Web services

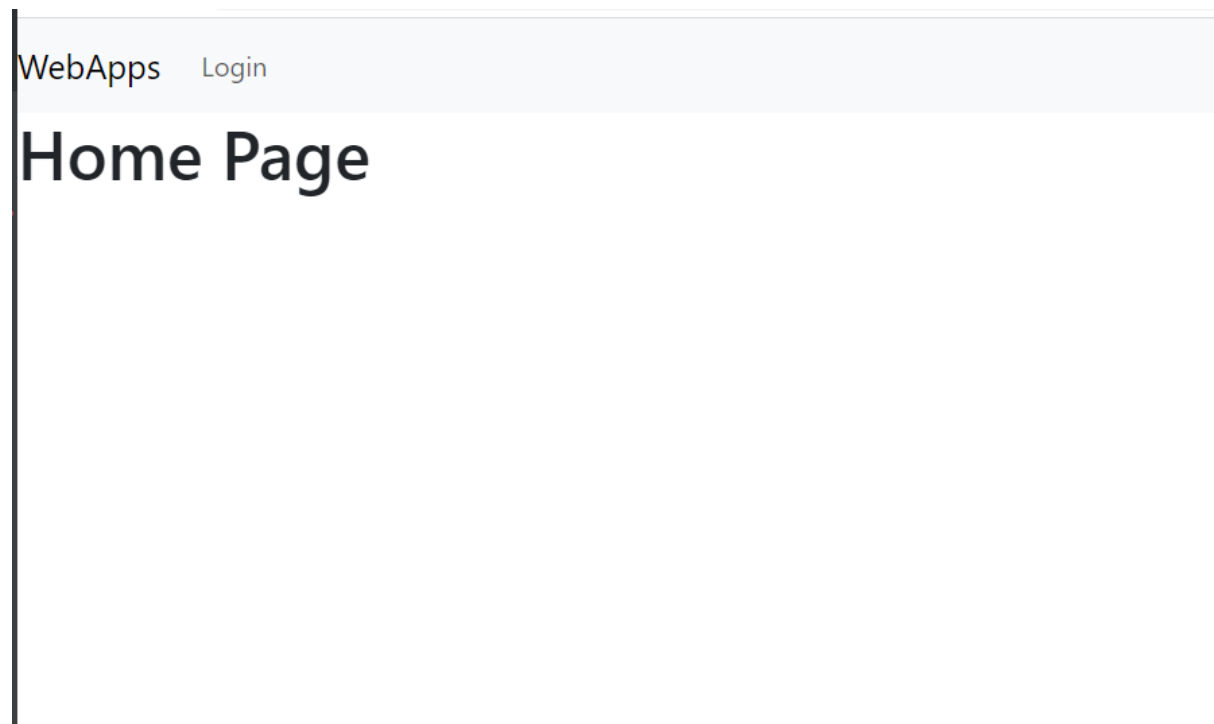
### Manual on how to use.

Below you will find a manual on how an average user can use the website that is provided.

#### Step 1: Run the website using IntelliJ.

Upon running the website using IntelliJ the user will be provided with a link: <http://127.0.0.1:8000/>

This will direct the user to the homepage.



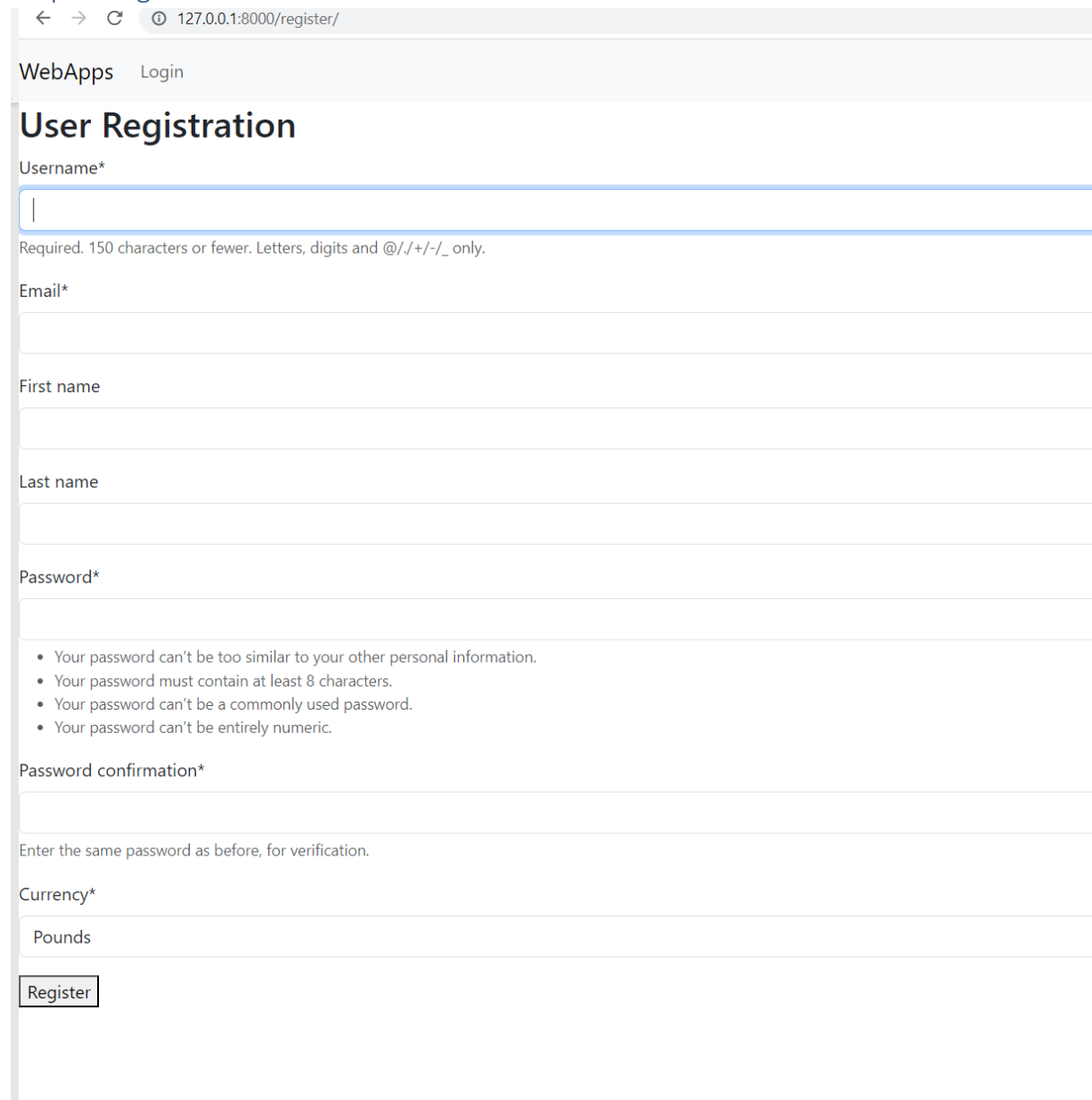
*Figure 10 Manual 1: User Homepage upon loading up IntelliJ*

IntelliJ must stay running in order to access the website and use its features.

Upon running the website, the user will be directed to the image found at Figure 10:

This is the home page upon pressing the Login Button found at the navigation bar on the top, they will be directed to the login/registration page.

## Step 2: Registration



← → ↻ ⓘ 127.0.0.1:8000/register/

WebApps Login

# User Registration

Username\*

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

Email\*

First name

Last name

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\*

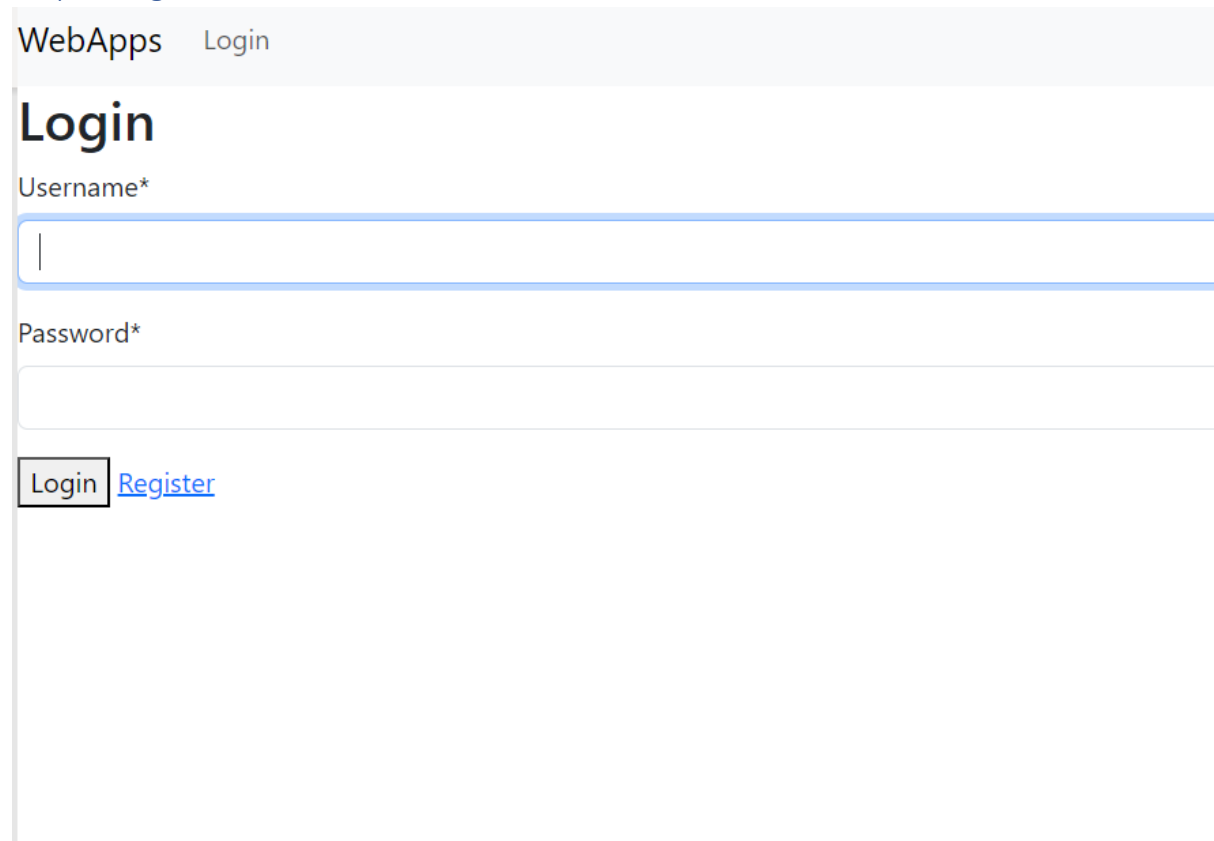
Pounds

Register

Figure 11 Manual 2: Registration Page

Now that the user has found them selves on the login page as can be seen on Figure 12. They will need to press the register button; there the user needs to follow the instructions provided on the registration page as can be seen in Figure 11. They then select a currency that they wish to use, in which case they will then be logged and redirected to the login page where they must login again for security reasons.

### Step 3: Login



WebApps Login

# Login

Username\*

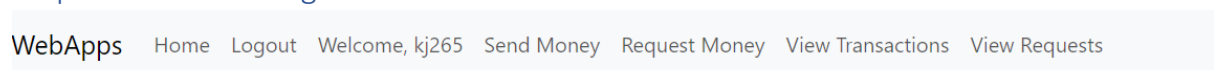
Password\*

[Login](#) [Register](#)

Figure 12 Manual 3: Login Page

Once redirected back to the login page. The users must enter the details that they placed earlier in the registration page. Once this complete, they will be directed into the homepage of the application after having been logged in.

### Step 4: Website Navigation



WebApps Home Logout Welcome, kj265 Send Money Request Money View Transactions View Requests

Figure 13 Navigation bar

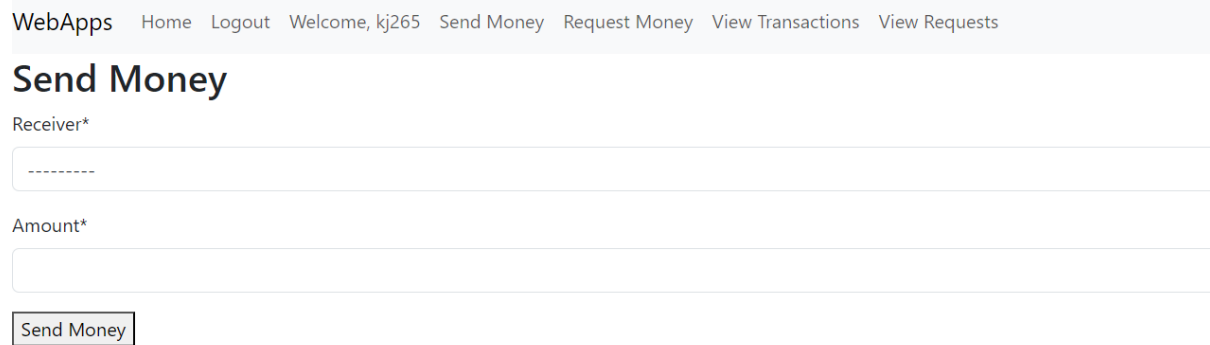
after entering the home page. The navigation bar will be viewed as what can be seen in figure 13. The user can then move on to navigate to the pages they want from here. They will be able to view whichever pages they choose thanks to this.



Candidate number: 268940

#### Step 4.1: Home Page explanation

#### Step 5: Send Money



WebApps Home Logout Welcome, kj265 Send Money Request Money View Transactions View Requests

## Send Money

Receiver\*

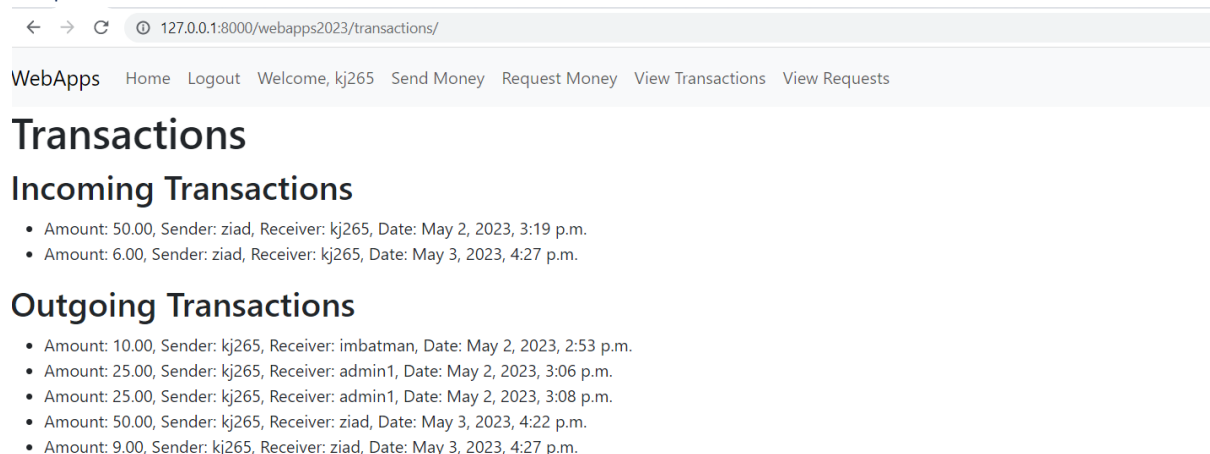
Amount\*

Send Money

Figure 14 Manual 4: Send Money page

This segment is a crucial part of the website. With this segment users can send money to other users for what ever reasons that they may need. All they must do is select the user from the drop-down menu that is present in the form and select which user they want to send the money to, from a list of registered users.

#### Step 5.1: View Transactions



← → 127.0.0.1:8000/webapps2023/transactions/

WebApps Home Logout Welcome, kj265 Send Money Request Money View Transactions View Requests

## Transactions

### Incoming Transactions

- Amount: 50.00, Sender: ziad, Receiver: kj265, Date: May 2, 2023, 3:19 p.m.
- Amount: 6.00, Sender: ziad, Receiver: kj265, Date: May 3, 2023, 4:27 p.m.

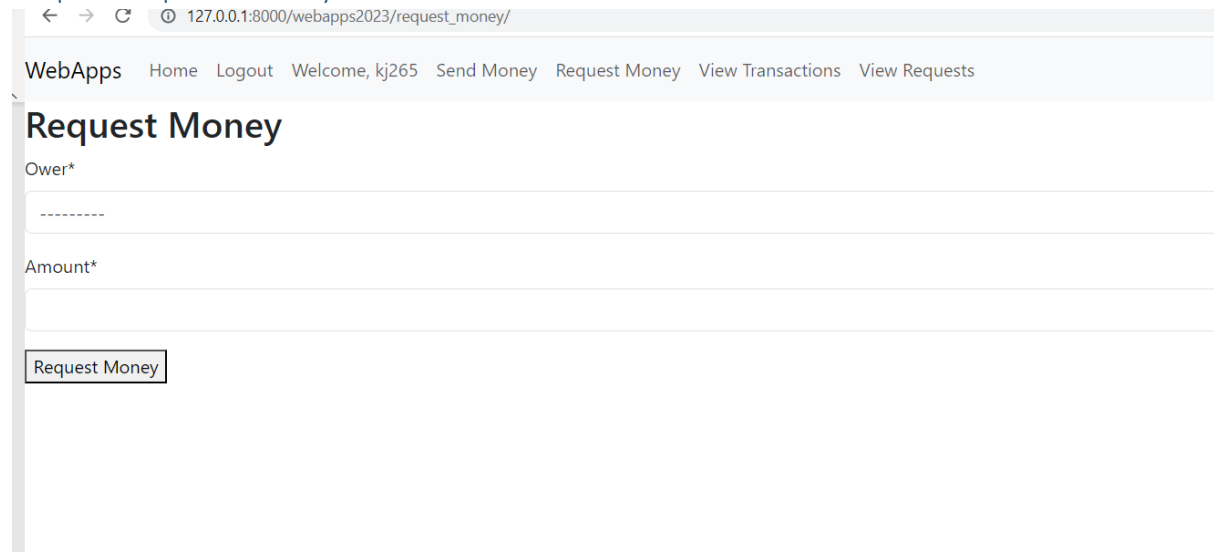
### Outgoing Transactions

- Amount: 10.00, Sender: kj265, Receiver: imbatman, Date: May 2, 2023, 2:53 p.m.
- Amount: 25.00, Sender: kj265, Receiver: admin1, Date: May 2, 2023, 3:06 p.m.
- Amount: 25.00, Sender: kj265, Receiver: admin1, Date: May 2, 2023, 3:08 p.m.
- Amount: 50.00, Sender: kj265, Receiver: ziad, Date: May 3, 2023, 4:22 p.m.
- Amount: 9.00, Sender: kj265, Receiver: ziad, Date: May 3, 2023, 4:27 p.m.

Figure 15 Manual 5: View Transactions Page

Additionally, the users can also view all their incoming and outgoing transactions. On the view transactions page, as can be seen in Figure 15.

## Step 6: Request Money



WebApps Home Logout Welcome, kj265 Send Money Request Money View Transactions View Requests

### Request Money

Owner\*

-----

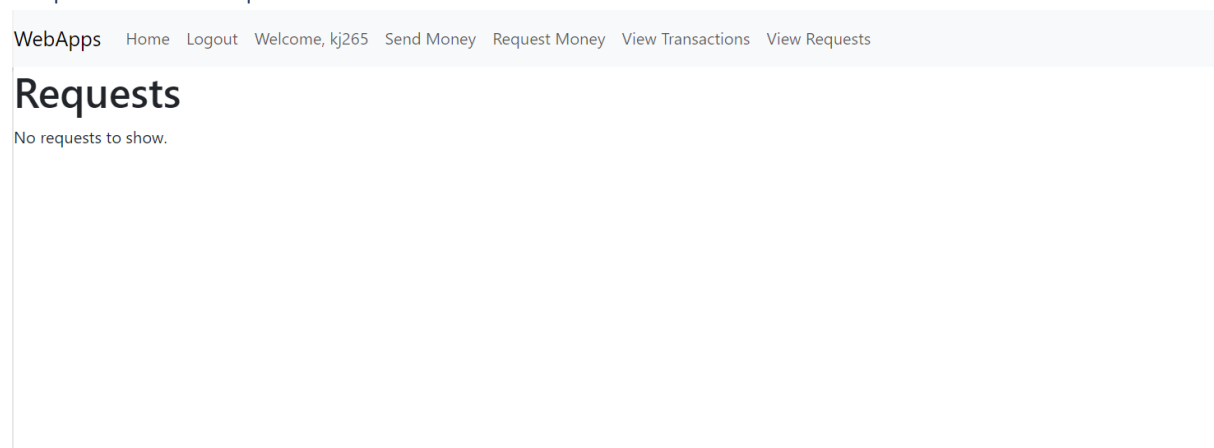
Amount\*

Request Money

Figure 16 Manual 6: Request Money

Users can then go over to the request money page, as can be seen in the Figure 16. They can then select from a list of registered users who they want to request money from, and a request will then be sent to that user as can be seen in figure 18. Figure 17 is what the users see before the request are sent through.

### Step 6.1: View Request



WebApps Home Logout Welcome, kj265 Send Money Request Money View Transactions View Requests

### Requests

No requests to show.

Figure 17 Manual 7: Requests page when there are not requested to show.

Candidate number: 268940

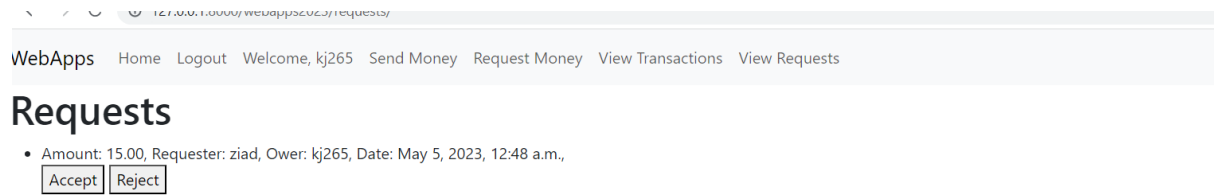
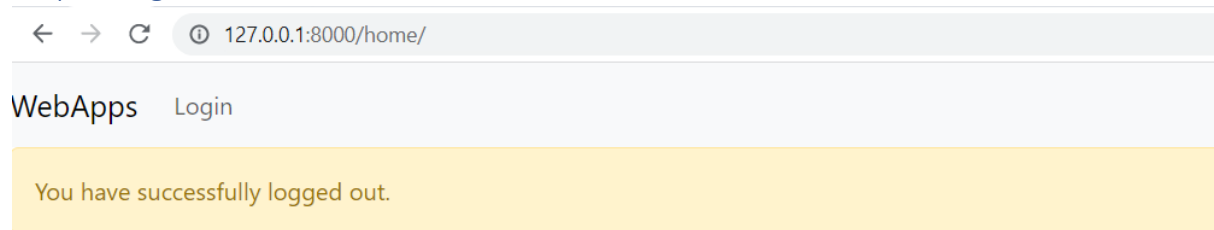


Figure 18 Manual 8: Request page when there are requested to show.

### Step 7: Logout



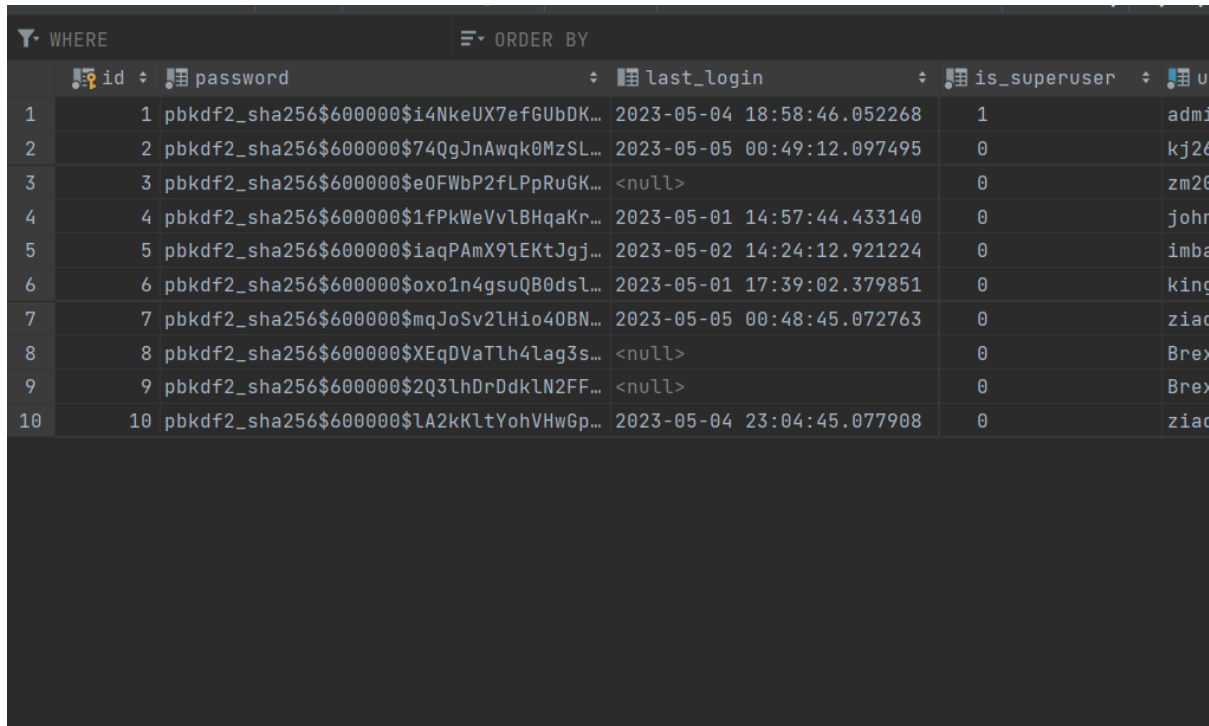
## Home Page

Figure 19 Manual 9: Home Page

Once the users decides that they have used the system enough they can navigate to the logout button, and it will then proceed to log them out of the system. The popup will appear as can be seen in Figure 19, hence confirming they're logout.

Candidate number: 268940

## Database



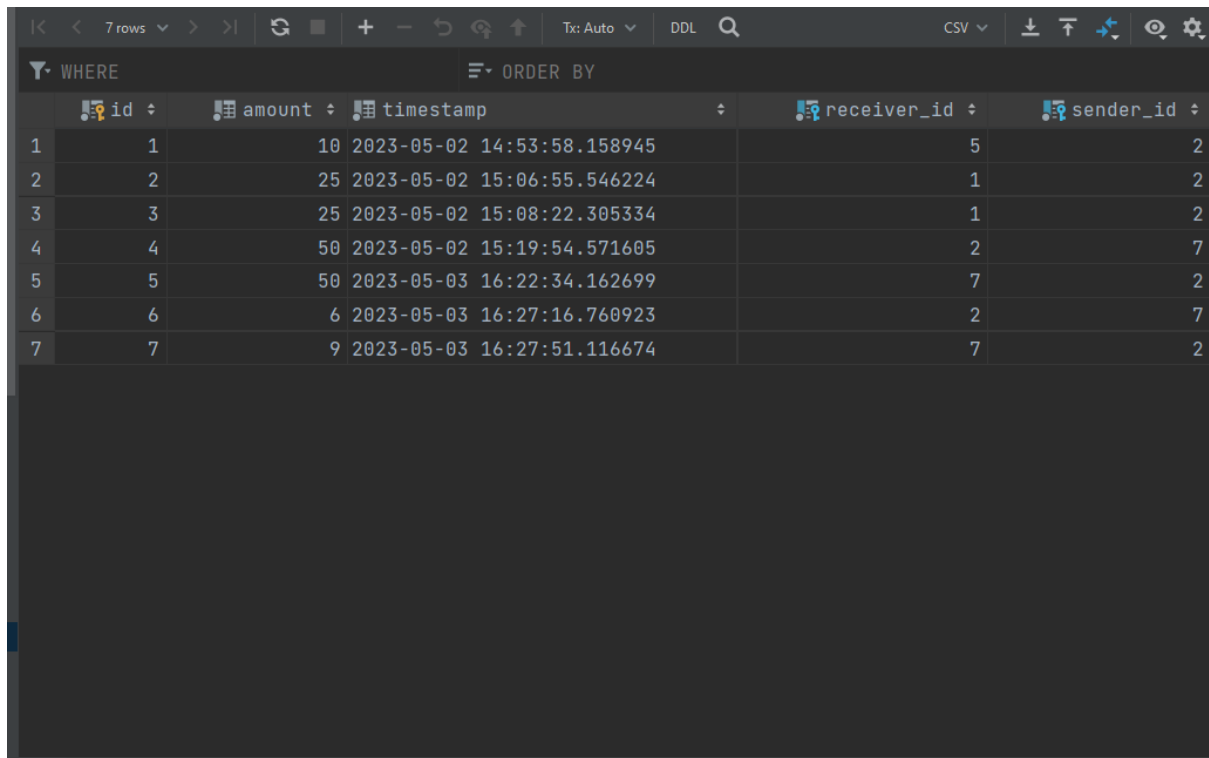
The screenshot shows a database query interface with a table of user data. The table has columns for id, password, last\_login, is\_superuser, and username. The data is sorted by id. The passwords are encrypted using pbkdf2-sha256. The last\_login column shows the date and time of the last login, with some entries being null.

	id	password	last_login	is_superuser	username
1	1	pbkdf2_sha256\$600000\$i4NkeUX7efGUbDK...	2023-05-04 18:58:46.052268	1	admin
2	2	pbkdf2_sha256\$600000\$74QgJnAwqk0MzSL...	2023-05-05 00:49:12.097495	0	kj26
3	3	pbkdf2_sha256\$600000\$e0FWbP2fLPpRuGK...	<null>	0	zm26
4	4	pbkdf2_sha256\$600000\$1fPkWeVvLBHqaKr...	2023-05-01 14:57:44.433140	0	johr
5	5	pbkdf2_sha256\$600000\$iaqPAmX9LEKtJgj...	2023-05-02 14:24:12.921224	0	imba
6	6	pbkdf2_sha256\$600000\$oxo1n4gsuQB0dsl...	2023-05-01 17:39:02.379851	0	king
7	7	pbkdf2_sha256\$600000\$mQJoSv2lHio40BN...	2023-05-05 00:48:45.072763	0	ziac
8	8	pbkdf2_sha256\$600000\$XEqDVaTlh4lag3s...	<null>	0	BreX
9	9	pbkdf2_sha256\$600000\$2Q3lhDrDdkLN2FF...	<null>	0	BreX
10	10	pbkdf2_sha256\$600000\$1A2kKltYohVHwGp...	2023-05-04 23:04:45.077908	0	ziac

Figure 20 Auth User Database

The database that can be viewed in Figure 20 is where the created users are stored. This database is used to when requiring seeing who the users are and how many. As can be seen the password is encrypted.

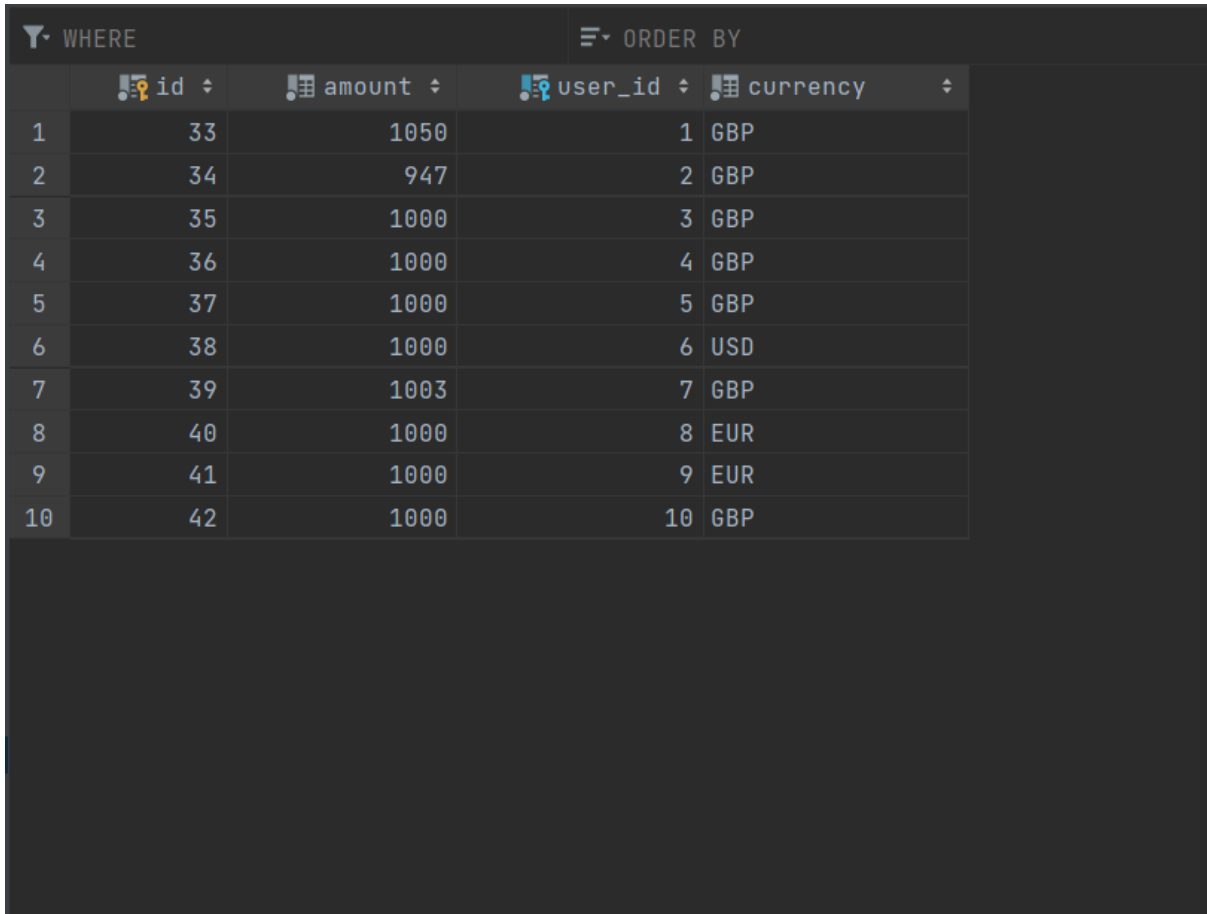
Candidate number: 268940



The screenshot shows a database query interface with a table of 7 rows. The table has five columns: id, amount, timestamp, receiver\_id, and sender\_id. The data represents transactions between different users (ids) over time (timestamps). The interface includes a toolbar with various icons for navigation and editing, and a search bar at the top.

	id	amount	timestamp	receiver_id	sender_id
1	1	10	2023-05-02 14:53:58.158945	5	2
2	2	25	2023-05-02 15:06:55.546224	1	2
3	3	25	2023-05-02 15:08:22.305334	1	2
4	4	50	2023-05-02 15:19:54.571605	2	7
5	5	50	2023-05-03 16:22:34.162699	7	2
6	6	6	2023-05-03 16:27:16.760923	2	7
7	7	9	2023-05-03 16:27:51.116674	7	2

Figure 21 Pay app Transfer



The screenshot shows a database query interface with a table of account transactions. The table has four columns: 'id', 'amount', 'user\_id', and 'currency'. The data is sorted by 'user\_id' in ascending order. The interface includes a 'WHERE' clause section on the left and an 'ORDER BY' section on the right.

	id	amount	user_id	currency
1	33	1050	1	GBP
2	34	947	2	GBP
3	35	1000	3	GBP
4	36	1000	4	GBP
5	37	1000	5	GBP
6	38	1000	6	USD
7	39	1003	7	GBP
8	40	1000	8	EUR
9	41	1000	9	EUR
10	42	1000	10	GBP

Figure 22 Register account database.

In the database seen in figure 22. Whenever a user adds a new account. This account goes into this database table here. With that the selected currency is put in place, as well as the user ID. The user ID is then used as well when transferring and receiving money between the accounts.

## Admin

The admin function when it comes to using the website is quite different to regular users. That is because the admin can view all the ongoings of the website and to see what transactions have take place.

It is important to note that the admin is also labelled on the program as a “Superuser”.

Admin username: admin1

Admin password: admin1

During the mode the user can inspect all the transactions and all the requests from all the users.