MONEY EXCHANGE WEBSITE

TABLE OF CONTENTS

|  |  |
| --- | --- |
| **TOPIC** | **PAGE NUMBER** |
| 1. SYSTEM OVERVIEW | 3 |
| 1.1 CONCEPT OF OPERATIONS | 4 |
| 1.2 DESIGN DETAILS | 8 |
| 1. SECURITY ARCHITECTURE | 11 |
| 2.1 DATA FLOW DIAGRAM | 13 |
| 1. THREAT MODELLING AND ATTACK SURFACE ANALYSIS | 15 |
| 3.1 ATTACK SURFACE ANALYSIS | 15 |
| 3.2 MITIGATION | 18 |
| 1. SECURITY REQUIREMENTS | 19 |
| 1. CODING STANDARDS | 21 |
| 1. SUPPLY CHAIN RISK MANAGEMENT | 26 |
| 1. INFORMATION, FUNCTIONAL, BEHAVIORAL MODELING | 27 |
| 1. TESTING STRATEGY | 34 |
| 1. DATABASE TABLE DESIGN | 36 |
| 1. UI MOCKUPS | 37 |

1. SYSTEM OVERVIEW

The application is a secure money exchange website that allows users to send money to each other and pay merchants for goods and services, either one time or subscription. It is designed with a security first mindset that includes threat modelling, architectural risk analysis, and security code reviews.

A basic overview of the system is as follows:

* The home page will give options to the user to login or sign up as a new user. There will also be a separate tab to login for the admin, but the user can’t register as an admin. The admin details will be saved in the database and only the person who have those details can sign in as an admin.
* The user once logged in can send money to a user or merchant (if they are also registered on the web site), the user will search for the user name of the person he/she wants to send money to and will complete the transaction. The user needs to add money to the wallet though to make sure he/she have enough money in their wallet to make sure they have enough funds. They can also transfer money from the wallet back to their account. The user can also update his/her profile.
* The admin can view the list of all the users, and the list of all the transactions made (sorted by date: from newest to oldest). The admin will also have the option to view the transactions made by a user. The admin can also update a user profile and can also block a user.
  1. CONCEPT OF OPERATIONS

|  |  |
| --- | --- |
| **Requirement ID** | **User Functions** |
| **R1** | **Register** |
| R1.1 | The user shall select register function to create an account. |
| R1.2 | The user shall enter **First Name, Last Name, Gender, Mobile No, Email Address, Username and Password to register/create an account.** |
| R1.3 | The system shall return prompt message to confirm successful registration of the user. |
| R1.4 | The user shall redirect to login function page. |
|  |  |
| **R2** | **Login** |
| R2.1 | The user shall select Login function from the list of functions. |
| R2.2 | The user shall enter registered **Username and Password.** |
| R2.3 | The user shall redirect to their functions list. |
| R2.4 | The user shall now select any other function. |
| R2.4.2 | Once logged in User shall perform **View Profile, Update Profile, Wallet, Add money, Send money, Confirm amount, Confirm payment** and **Transfer to bank.** |
|  |  |
| **R3** | **Logout** |
| R3.1 | The user shall select Logout function from the list of functions. |
| R3.2 | The system shall return a prompt message confirming that the user has been successfully logged out. |
| R3.3 | The user shall be redirected to main function list and shall have the option to login or register new user. |
|  |  |
| **R4** | **View Profile** |
| R4.1 | The user shall select View Profile function from the list of functions. |
| R4.2 | The user shall see editable profile attributes **First Name, Last Name, Gender, Mobile Number, Email Address and Password.** |
| R4.3 | The user shall view or edit profile attributes. |
| R4.4 | The user shall now select Update Profile function or Logout function or return to the list of functions. |
|  |  |
| **R5** | **Update Profile** |
| R5.1 | The user shall select Update Profile function from View Profile function result page. |
| R5.2 | The user shall update profile attributes **First Name, Last Name, Gender, Mobile Number, Email Address and Password.** |
| R5.3.1 | The system shall display all the changes made by the user on the same page. |
| R5.3.2 | The system shall return prompt message confirming successfully updating of user profile. |
| R5.4 | The user shall now select Update Profile function (again) or Logout function or return to the list of functions. |
|  |  |
| **R6** | **Wallet** |
| R6.1 | The user shall select Wallet function from list of functions. |
| R6.2 | The system shall display the wallet details like current balance and the user can further add money or send money. |
| R6.3 | The user can select a function from the list of functions or return to list of functions on the user home screen. |
|  |  |
| **R7** | **Add money** |
| R7.1 | The user shall select Add money function from Wallet function result page. |
| R7.2 | The system shall redirect the user to the payment page. |
| R7.3 | The user shall confirm the amount. |
| R7.4 | The user can select a function from the list of functions on the Wallet home screen or return to list of functions on the user home screen. |
|  |  |
| **R8** | **Send money** |
| R8.1 | The user shall select Send money function from Wallet function result page. |
| R8.2 | The system shall redirect the user to the search username home screen. |
| R8.3 | The user shall confirm the payment. |
| R8.4 | The user can select a function from the list of functions on the Wallet home screen or return to list of functions on the user home screen. |
|  |  |
| **R9** | **Transfer to bank** |
| R9.1 | The user shall select Transfer to bank function from Wallet function result page. |
| R9.2 | The system shall redirect the user to the amount screen. |
| R9.3 | The user shall confirm the amount. |
| R9.4 | The user can select a function from the list of functions on the Wallet home screen or return to list of functions on the user home screen. |
|  |  |
| **R10** | **Confirm amount** |
| R10.1 | The user shall select Confirm Amount function from Amount screen result page. |
| R10.2 | The user shall confirm the amount entered by the user. |
| R10.3 | The system shall return a prompt message confirming the transfer. |
| R10.4 | The user shall now select any function from the function list on the Wallet home screen. |
|  |  |
| **R11** | **Confirm payment** |
| R11.1 | The user shall select Confirm Payment function after searching and confirming the username on Recipient Username screen result page. |
| R11.2 | The user shall confirm the payment entered by the user. |
| R11.3 | The system shall return a prompt message confirming the transfer. |
| R11.4 | The user shall now select any function from the function list on the Wallet home screen. |

|  |  |
| --- | --- |
| **R12** | **View User** |
| R12.1 | Admin shall select View User function from function list. |
| R12.2 | The system shall return selectable list of all registered users; sorted alphabetically by First Name. |
| R12.3 | Admin shall select a **First Name** to view user detail. |
| R12.4 | Admin shall now select View User Detail function or Logout function or return to the list of functions. |
|  |  |
| **R13** | **View User Detail** |
| R13.1 | Admin shall select View User Detail function after selecting First Name from generated list. |
| R13.2 | The system shall display editable profile attributes- **First Name, Last Name, Gender, Mobile No, Email Address.** |
| R13.3 | Admin shall now view or edit user profile attributes. |
| R13.4 | Admin shall now select Update User Profile function or Block User function or View User Transactions or Logout function or return to the list of functions. |
|  |  |
| **R14** | **Update User Profile** |
| R14.1 | Admin shall select Update User Profile function from View User Detail function result page. |
| R14.2 | Admin shall update - **First Name, Last Name, Gender, Mobile No, Email Address.** Note: Admin shall not update user role and password for any user. |
| R14.3 | Admin shall now select Block User function or Logout function or return to the function list. |
|  |  |
| **R15** | **Block User** |
| R15.1 | Admin shall select Block function from View User Detail function result page. |
| R15.2 | The system shall return a prompt message confirming blocking the user. |
| R15.3 | Admin shall redirect to the generated list of all registered users. |
|  |  |
| **R16** | **View User Transactions** |
| R16.1 | The admin shall select View User Transactions function from the list of functions. |
| R16.2 | The system shall return a list of transactions made by the user; sorted by date(from newest to oldest). |
| R16.3 | The manager shall select a Username from the generated list. |
| R16.4 | The manager shall now return to the list of functions on User Detail screen. |
|  |  |

* 1. DESIGN DETAILS

The high-level functioning of the system is described below.

|  |  |
| --- | --- |
| **Use Case** | **Description** |
| UC 1: Register | **TUCBW** The User selects Register function from the main screen. |
| **TUCEW** The User sees login screen with a successful registration confirmation message. |
| UC 2: Login | **TUCBW** The User enter their username and password and then selects Login function from their login screen. |
| **TUCEW** The User sees functions list on their home screen. |
| UC 3: Logout | **TUCBW** The User selects the Logout function from their function list. |
| **TUCEW** The User sees the login screen. |
| UC 4: View Profile | **TUCBW** The User selects View Profile function from their home screen. |
| **TUCEW** The User sees editable profile attributes on profile screen. |
| UC 5: Update Profile | **TUCBW** The User selects Update Profile function from profile screen after editing profile attributes. |
| **TUCEW** The User sees updated profile attributes on profile screen. |
| UC 6: Wallet | **TUCBW** The User selects Wallet function from their homesreen. |
| **TUCEW** The User sees wallet details e.g. current balance and recent transaction and also a list of functions. |
| UC 7: Add Money | **TUCBW** The User selects Add money function from Wallet screen. |
| **TUCEW** The User sees the Confirm amount function on the screen. |
| UC 8: Send Money | **TUCBW** The User selects Send Money function from Wallet screen. |
| **TUCEW** The User sees the Confirm amount function on the screen. |
| UC 9: Transfer to Bank | **TUCBW** The User selects Transfer to Bank function from Wallet screen. |
| **TUCEW** The User sees the Confirm amount function on the screen. |
| UC 10: Confirm Amount | **TUCBW** The User selects the Confirm Amount function from their screen. |
| **TUCEW** The User sees the money transfer has been made. |
| UC 11: Confirm payment | **TUCBW** The User selects the Confirm Payment function from their screen. |
| **TUCEW** The User sees the money has been transferred to the recipient. |
| UC 12: View User | **TUCBW** The Admin selects View User function from their home screen. |
| **TUCEW** The Admin sees selectable list of first names of all registered users on users list screen. |
| UC 13: View User Detail | **TUCBW** The Admin selects View User Detail function from users list screen after selecting a first name from the generated list. |
| **TUCEW** The Admin sees profile attributes of selected user on user detail screen. |
| UC 14: Update User Profile | **TUCBW** The Admin selects Update User Profile function from user detail screen after editing profile attributes. |
| **TUCEW** The Admin sees the updated profile attributes on user detail screen. |
| UC 15: Block User | **TUCBW** The Admin selects Block User function from user detail screen. |
| **TUCEW** The Admin sees a successful prompt message and redirects to user detail screen. |
| UC 16: View User Transactions | **TUCBW** The Admin selects View User Transactions function from the User Detail Home screen. |
| **TUCEW** The Admin sees all the transactions made by that user. |

1. SECURITY ARCHITECTURE

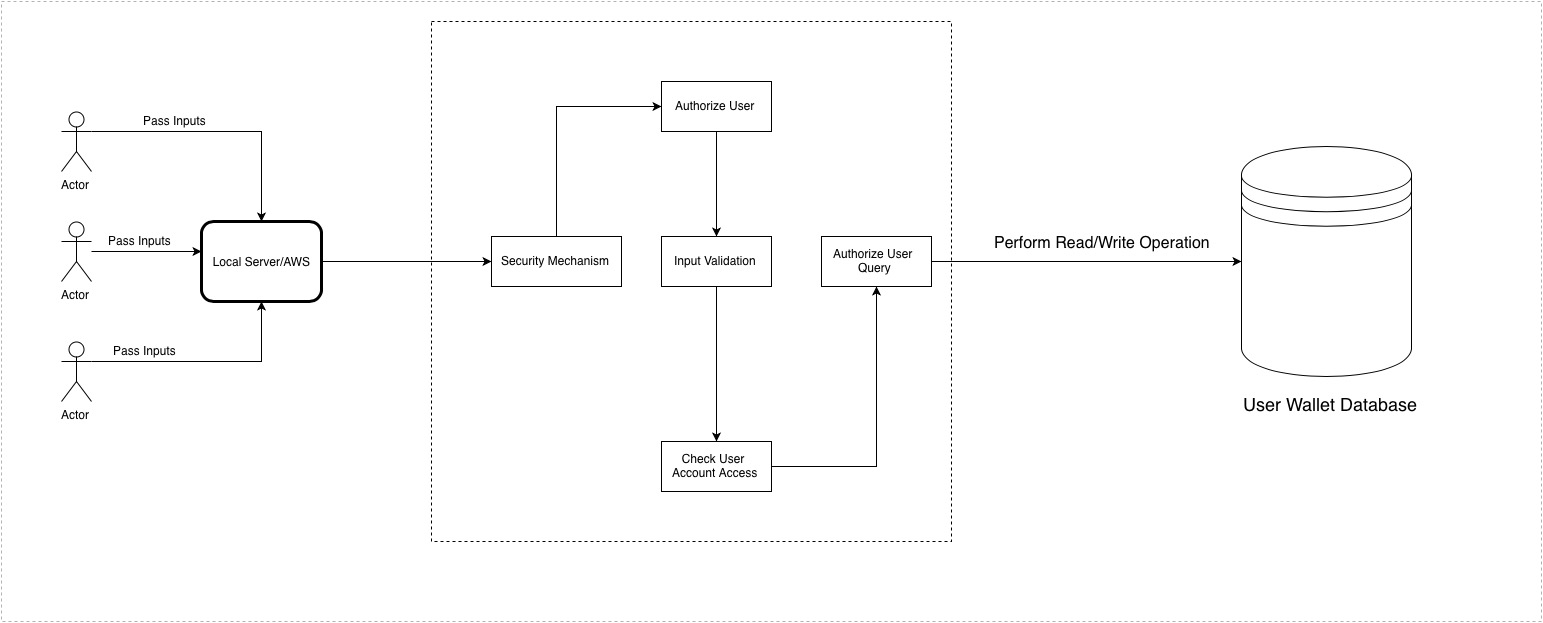
The project is a secure online portal for keeping digital money, send/receive funds, add money to the wallet. Since, the project involves money and we know that everything that involves money is something that has to be protected with the highest level of security for the money, user information and the system that is managing the digital money on behalf of the user.

The utmost important step in ensuring security of the data is to keep the system safe and foolproof with the user inputs. OWASP has identified “the user inputs” as the most common attack vector by the hackers. Any input that goes without filtering in the system is harmful and as we have seen in history has resulted in hacking and unauthorized users gain access to the system and then the sensitive information of its users and their data.

The system implemented here involves security mechanism that involves: checking user authorization to perform an action, verifying the identity of the user, sanity checking the input of the user, checking account access and privileges of the user to perform that action. When an action by the user goes through and pass all the security mechanism is then subjected to execution. Such sequence of security steps helps ensure maintain the user access, user control and right access by all the users.

Security mechanisms helps keep the attackers or the users with malicious intent away from the system. Security mechanism are implemented to keep the system sane and actions by the user viable to a farther point. Also, as an additional step for the security mechanism, the project encrypts user data and only store the hash of the data which in-turn makes sure that even if an attacker get access to the data, the data they have gotten access to do doesn’t make much sense at all. Efforts of such kind and many more are put in place to ensure system integrity and system.

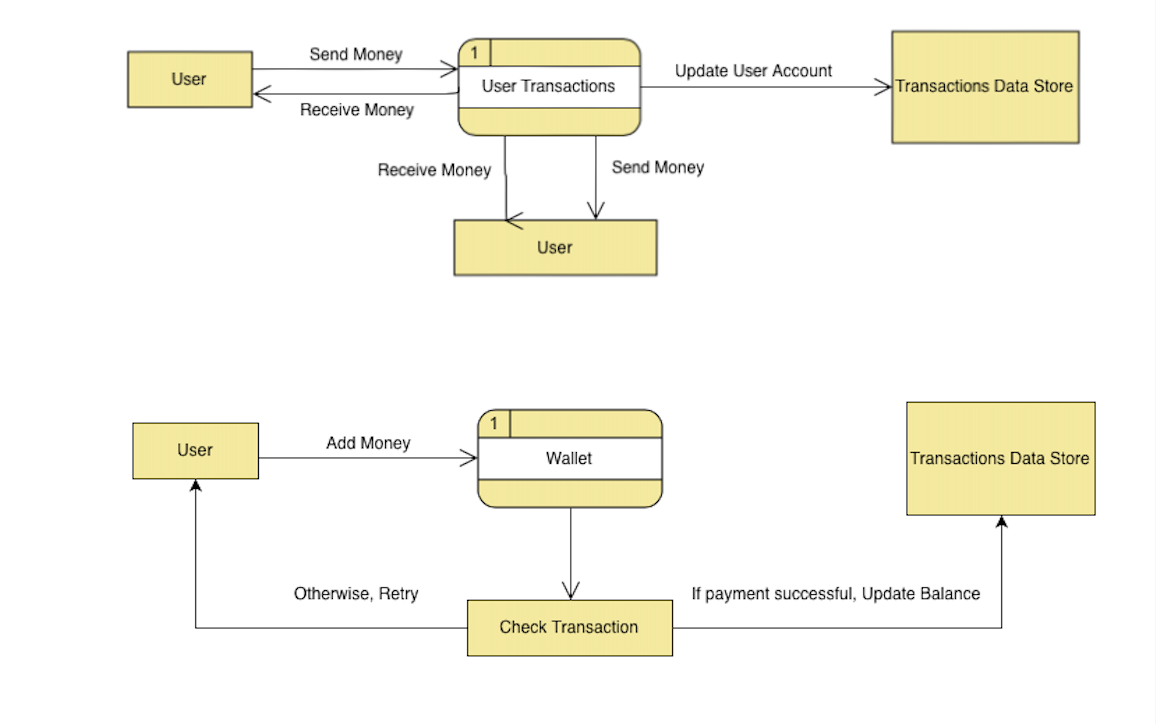
The security controls allocated to the system’s components is shown in the diagram below.



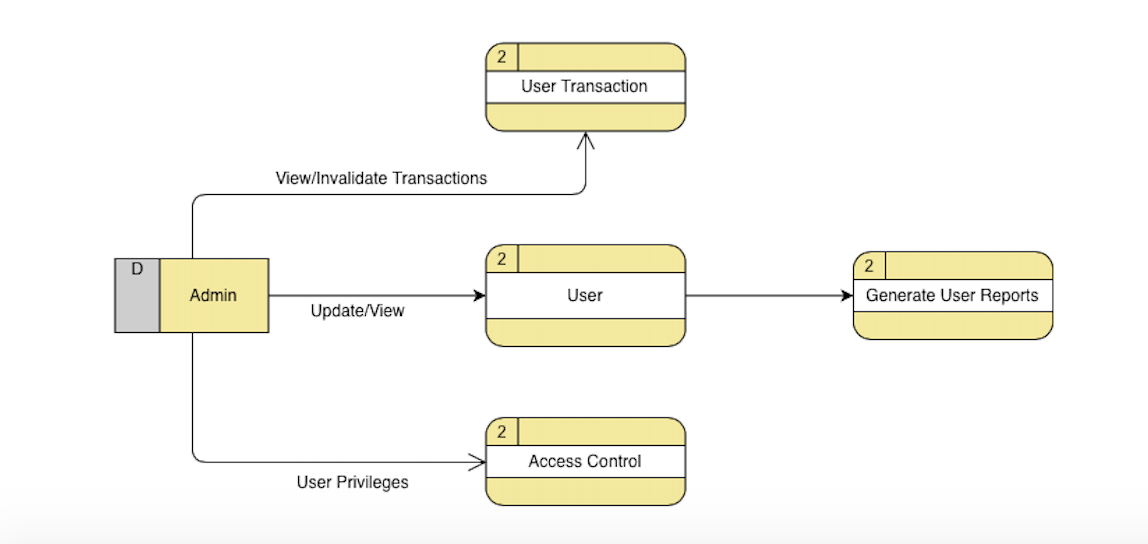


* 1. DATA FLOW DIAGRAM

The data flow diagram for the user is shown below.



The data flow diagram for the administrator is shown below.





1. THREAT MODELLING AND ATTACK SURFACE ANALYSIS
   1. ATTACK SURFACE ANALYSIS

The following attack surfaces have been identified in terms of the main use cases.

* USER REGISTRATION

**Exploiting Trust in Client:**

An attack of this type uses the vulnerabilities in a client/server channel authentication and data integrity. An attacker places themselves in the communication channel such that the server believes it is communicating with the only valid client.

* USER AND ADMIN LOGIN
* **Exploitation of Trusted Credentials:**

Attacks on session and resource identities can happen when user input is accepted without verifying its authenticity. The attackers can gain access to vital information like payment information of users.

* **Authentication Abuse:**

The attacker obtains unauthorized access to the application because of a faulty authentication.

* **Authentication Bypass:**

The attacker gains access to the application by circumventing the authentication mechanism.

* UPDATE PROFILE
* **Manipulating User State:**

The attacker modifies state information maintained by the software in user-accessible locations. The target software will use this modified information in an unintended manner.

* **Shared Data Manipulation:**

The attacker exploits the data structure that is shared among multiple threads in the application. Once the shared data is modified, other components will trust the validity of the shared data.

* ADD MONEY, SEND MONEY, TRANSFER TO BANK
* **Flooding:**

The attacker consumer the resources of the application thereby denying access of the application by authenticated users.

* **Input Data Manipulation:**

The attacker exploits a weakness in the input validation mechanism by controlling the format and structure of the input. This can also lead to bypassing protection mechanisms.

* **Content Spoofing:**

The attacker modifies content so that it contains something other than the intended material, but the apparent source of the content is not changed. This can lead to financial fraud and privacy violations.

* **Action Spoofing:**

The attacker can disguise one action for another and trick the user to do one action, but another action is initiated.

* **Sustained Client Engagement:**

The attacker denies access of the application to authenticated users by continuously engaging a specific resource to keep the resource occupied as long as possible. This causes processes to take longer than usual.

* **Functionality Bypass:**

The attacker attacks the system by passing some or all of the functionality intended to protect it.

* VIEW USER DETAILS, TRANSACTIONS, AND BLOCK USER
* **Exploitation of Trusted Credentials:**

Attacks on session and resource identities can happen when user input is accepted without verifying its authenticity.

* **Excessive Allocation:**

The attacker causes the application to allocate excessive resources according to their requests, therefore reducing access to legitimate users.

* **Resource Leak Exposure:**

The attacker exploits a resource leak present in the application to reduce the quantity of resources available to authenticated users if the application.

* **Privilege Abuse:**

Faulty access control mechanisms can give attackers access to confidential information that is intended to be accessed only be administrators.

* 1. MITIGATION

The various threats can be mitigated in a number of ways.

* During the design phase, client validation should not be relied upon for security purposes.
* Ensure that all client processes are authenticated and not accepted by the system if not properly validation.
* Input validation should be performed on all remote content.
* The use of cookies can protect the session ID if intercepted by attackers.
* A session timeout can be put for all sessions. If the user does not logout in 20 minutes, the server terminates the session after 20 minutes of inactivity.
* Confidential information like passwords and payment details should be avoided in user controllable locations.
* Proper behavior conditions should be specified when resource allocation reaches its limits.
* Suspicious sites and links should be avoided.
* Proper cybersecurity training should be given to the employees.
* Access to resources can be limited per IP address to avoid denial of service.
* Memory should always be allocated and freed using matching functions.

1. SECURITY REQUIREMENTS

The web service being built should not be vulnerable to following categories and threats.

A list of possible vulnerabilities and threats in the system are:

1. Input Validation

2. Session Management

3. Sensitive Data Exposure

4. Parameter Manipulation

5. Access Control

The possible attacks that could happen through exploitation of these vulnerabilities are:

1. **Input Validation:**

* Username Enumeration.
* SQL Injection.
* Cross-site Scripting.

1. **Session Management:**

* Session Hijack.
* Session Replaying.

1. **Sensitive Data Exposure:**

* Weak/No Encryption.
* SQL Attack.

1. **Parameter manipulation:**

* Query string manipulating.
* Cookies.
* HTTP Headers.

1. **Access Control:**

* Brute force attack.
* Confidential data breach.
* Data Tampering

The traceability for verifying the requirements are:

1. **Input Validation:**

* Display appropriate error message to avoid disclosure of usernames.
* Test cases including SQL queries as a user’s input in every web application functionality.
* Include various input cases in manual testing for checking input validation implementations.

1. **Session Management:**

* Session validation checks against user’s system specific attributes needs to be checked. Encryption of session information should be checked.
* Checking the session data signature such that the source IP or system is mapped with session.

1. **Data Exposure:**

* Encryption of all data in transit and at rest using secure protocols and algorithms.
* User awareness about possible site phishing and iframe injection.

1. **Parameter Manipulation:**

* Test cases exploiting the sessions for query strings changes.
* Checking the exposure of sensitive data in the cookies.
* Check on valid headers allowed.

1. **Access Control:**

* Script or manual login attempts tried with large attempts.
* Check every possible location of the Web application for exposure of system information.
* Manual web page testing for checking the user perimeter for changing the data.

1. CODING STANDARDS

The following coding standards need to be followed for:

* **Validation:**

All Web pages will be verified against the W3C validator to ensure that the markup is well formed. This helps to churn out problems that are able to be tested via automation. This is not an alternative for manual code review.

* **Self-closing Slash:**

All tags must be properly closed. For tags that are self-closing, the forward slash should have exactly one space preceding it:   
<br />   
rather than the compact but incorrect:   
<br/>   
The W3C specifies that a single space should precede the self-closing slash.

* **Attributes and tags:**

All tags and attributes must be written in lowercase. Additionally, attribute values should be lowercase when the purpose of the text therein is only to be interpreted by web browsers. For instances in which the data needs to be human readable, proper title capitalization should be followed.

For machines:   
<meta http-equiv="content-type" content="text/html; charset=utf-8" /> For humans:   
<a href="http://example.com/" title="Description Here">Example.com</a>

* **Quotes:**

According to the W3C specifications for XHTML, all attributes must have a value, and must use double- or single-quotes (source). The following are examples of proper and improper usage of quotes and attribute/value pairs.

Correct:   
<input type="text" name="email" disabled="disabled" />   
<input type='text' name='email' disabled='disabled' />   
Incorrect:   
<input type=text name=email disabled>

A comprehensive view on the proper use of HTML is as follows:

**1. Always Declare a Doctype**

The doctype declaration should be the first thing in your HTML documents. The doctype declaration tells the browser about the XHTML standards you will be using and helps it read and render your markup correctly.

<!DOCTYPE html PUBLIC "-//W3C//**DTD XHTML 1.0 Strict**//EN" “http:// www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**2. Use Meaningful Title Tags**

The <title> tag helps make a web page more meaningful and search-engine friendly. For example, the text inside the <title> tag appears in Google’s search engine results page, as well as in the user’s web browser bar and tabs.  
Take for instance, the following example:

<title>Six Revisions - Web Development and Design Information</title>

**3. Use Descriptive Meta Tags**

Meta tags make your web page more meaningful for user agents like search engine spiders.

**Description Meta Attribute**The description meta attribute describes the basic purpose of your web page (a summary of what the web page contains). For each web page, you should place a concise and relevant summary inside the meta description tag.

For example, this description:  
<meta name="description" content="Six Revisions is a blog that shares useful information about web development and design, dedicated to people who build websites." />

**Keywords Meta Attribute**

<meta name="keywords" content="web design, web development" />  
The keywords meta attribute contains a comma-separated list of key words and phrases that relate to your web page. These keywords make your web page even more meaningful.

**4. Use Divs to Divide Your Layout into Major Sections**

Consider dividing your web page into major sections as the first step in constructing a website design.

**5. Separate Content from Presentation**

Your HTML is your content. CSS provides your content’s visual presentation. Never mix both.  
Don’t use inline styles in your HTML. Always create a separate CSS file for your styles. This will help you and future developers that might work on your code make changes to the design quicker and make your content more easily digestible for user agents.

**6. Use Heading Elements Wisely**

Learn to use <h1> to <h6> elements to denote your HTML’s content hierarchy. This helps make your content more meaningful for screen-reading software and search engines, as well as other user agents.

<h1 class="entry-title"><?php the\_title(); ?></h1>

**7.Use the Right HTML Element at the Right Place**

Learn about all the available HTML elements and use them correctly for a semantic and meaningful content structure.  
Use <em> for *em*phasis and <strong> for **heavy emphasis**, instead of <i> or <b> (which are deprecated).  
**Example**<em>emphasized text</em>  
<strong>strongly emphasized text</strong>

**8. Don’t Use Divs for Everything**

Sometimes developers end up wrapping <div> tags around multiple <div> tags that contain more <div> tags, creating a mountain of divs.  
Under the latest draft of the W3C HTML specification, a <div> is a meaningless element that should be used "as an element of last resort, for when no other element is suitable."

But many use it even for menial things like displaying inline elements as block elements (instead of the display:block; CSS property).

**9. Use an Unordered List (<ul>) for Navigation**

Navigation is a very important aspect of a website design and the <ul> element combined with CSS makes your navigation menus semantic (because, after all, a navigation menu is a list of links) as well as beautiful.  
Also, by convention, an unordered list for your navigation menu has been the accepted markup.

**An Example of an Unordered List**

<ul id="main\_nav">  
<li><a href="#" class="active">Home</a></li> <li><a href="#">About</a></li>  
<li><a href="#">Portfolio</a></li>  
<li><a href="#">Services</a></li>  
<li><a href="#">Blog</a></li>  
<li><a href="#">Contact Us</a></li>  
</ul>

**10. Use Alt Attributes with Images**

Using a meaningful alt attribute with <img> elements is a must for writing valid and semantic code.  
**Bad Practice**<img id="logo" src="images/bgr\_logo.png"/>

**<!-- has an alt attribute, which will validate, but alt value is meaningless -->** <img id="logo" src="images/bgr\_logo.png" **alt="brg\_logo.png"** />  
**Good Practice**<img id="logo" src="images/bgr\_logo.png" **alt="Six Revisions Logo"** />

**11. Use Title Attributes with Links (When Needed)**

Using a title attribute in your anchor elements will improve accessibility when used the right way.  
It is important to understand that the title attribute should be used to increase the meaning of the anchor tag.

**Bad Practice**

**<!-- Redundant title attribute -->**<a href="http://blog.com/all-articles" **title="Click Here"**>Click here.</a>

When a screen reader reads the anchor tag, the listener has to listen to the same text twice. What’s worse is that it doesn’t explain what the page being linked to is.  
If you are just repeating the anchor’s text or aren’t intending to describe the page being linked, it’s better not to use a title at all.

**12. Write Consistently Formatted Code**

A cleanly written and well-indented code base shows your professionalism, as well as your consideration for the other people that might need to work on your code. This is another advice that is more emphasized upon to make code more readable.

1. SUPPLY CHAIN RISK MANAGEMENT

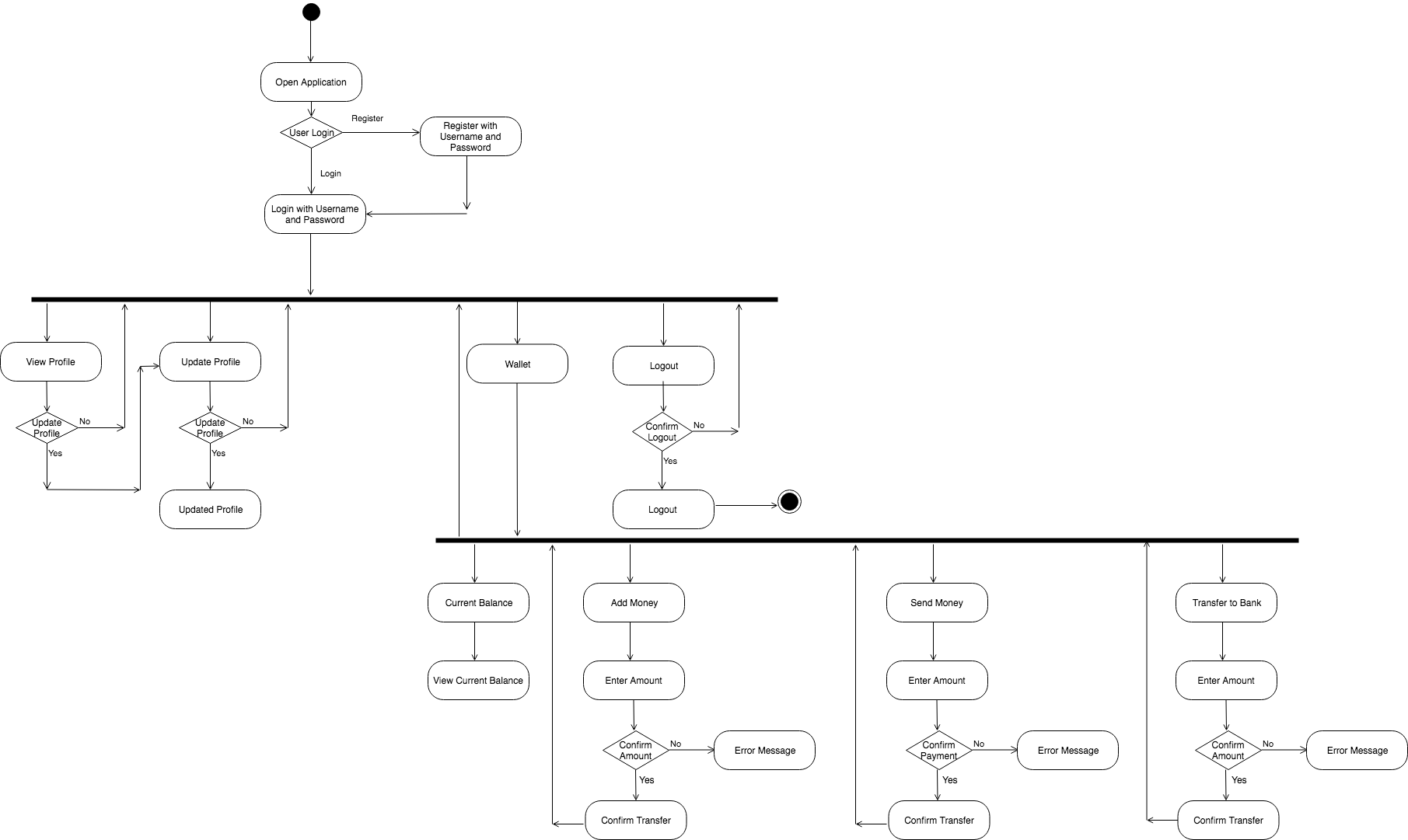
Supply-chain risk management (SCRM) is "the implementation of strategies to manage both every day and exceptional risks along the supply chain based on continuous risk assessment with the objective of reducing vulnerability and ensuring continuity.”

There are certain Risks associated with the SCRM and methods to mitigate them in order to ensure that the application can handle against the security threats.

|  |  |
| --- | --- |
| **Risk** | **Mitigate** |
| Inadequate Security Knowledge to engineering team. | a) Training the Engineering team can be given from security experts.  b) Security certifications can be provided to the team members. |
| Trading confidential transactional data. | Limited access rights to database and system should be provided. |
| Application user access to data. | Enforce minimum privileges to user to access the internal data only according to the requirements. |
| SQL injection in Python. | Use of parameterized query, also use regular expression for input validation. |
| Risk in session management. | Using specific module for session management and avoid any malicious activity. |
| Possible DOS attacks. | Provide a mechanism to detect and prevent DOS attacks. |
| 3rd party data interaction risk.  E.g.: payment gateways. | Make sure we cross check the security guidelines and that the 3rd party has SSL to ensure any risk of data. |
| Card holder’s data security | Enforce strong multi-factor authentication for access to critical systems with credit/debit card holder’s data. |

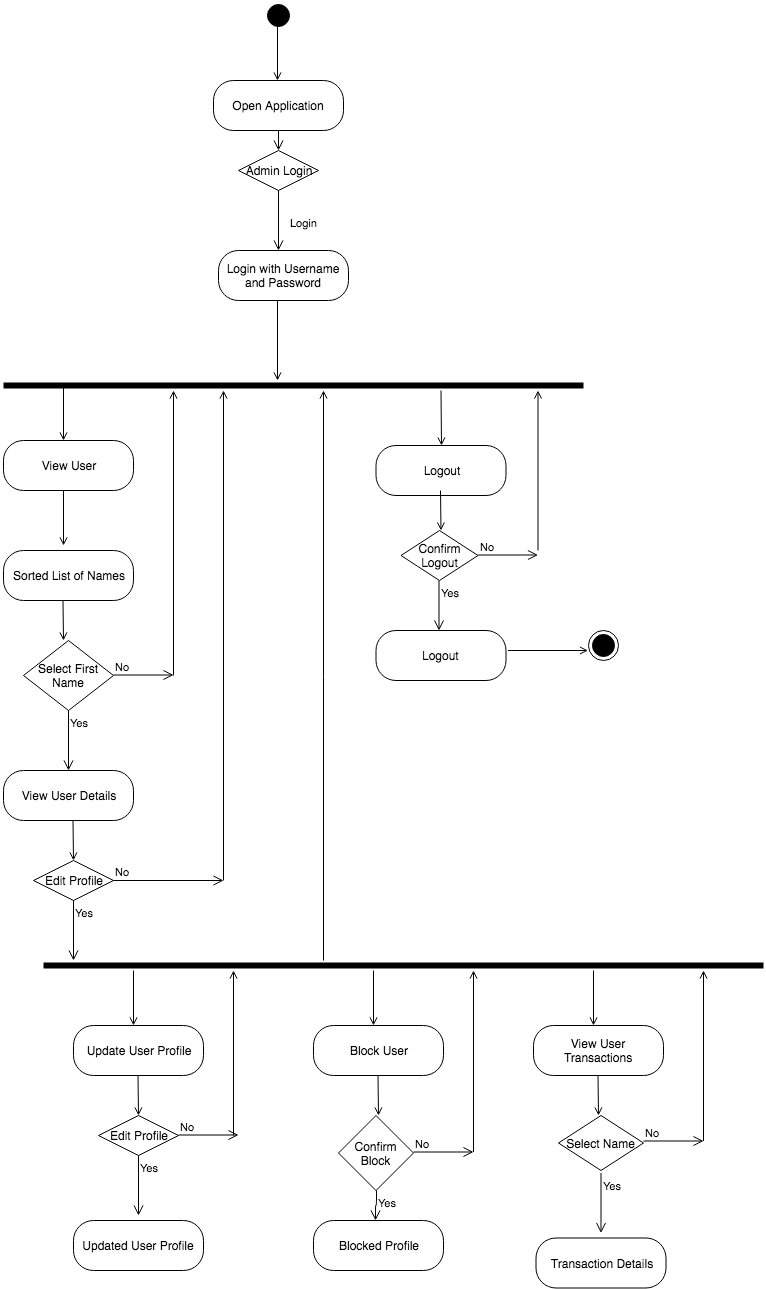
1. INFORMATION, FUNCTIONAL, BEHAVIORAL MODELING

The Activity Diagram of the user is:



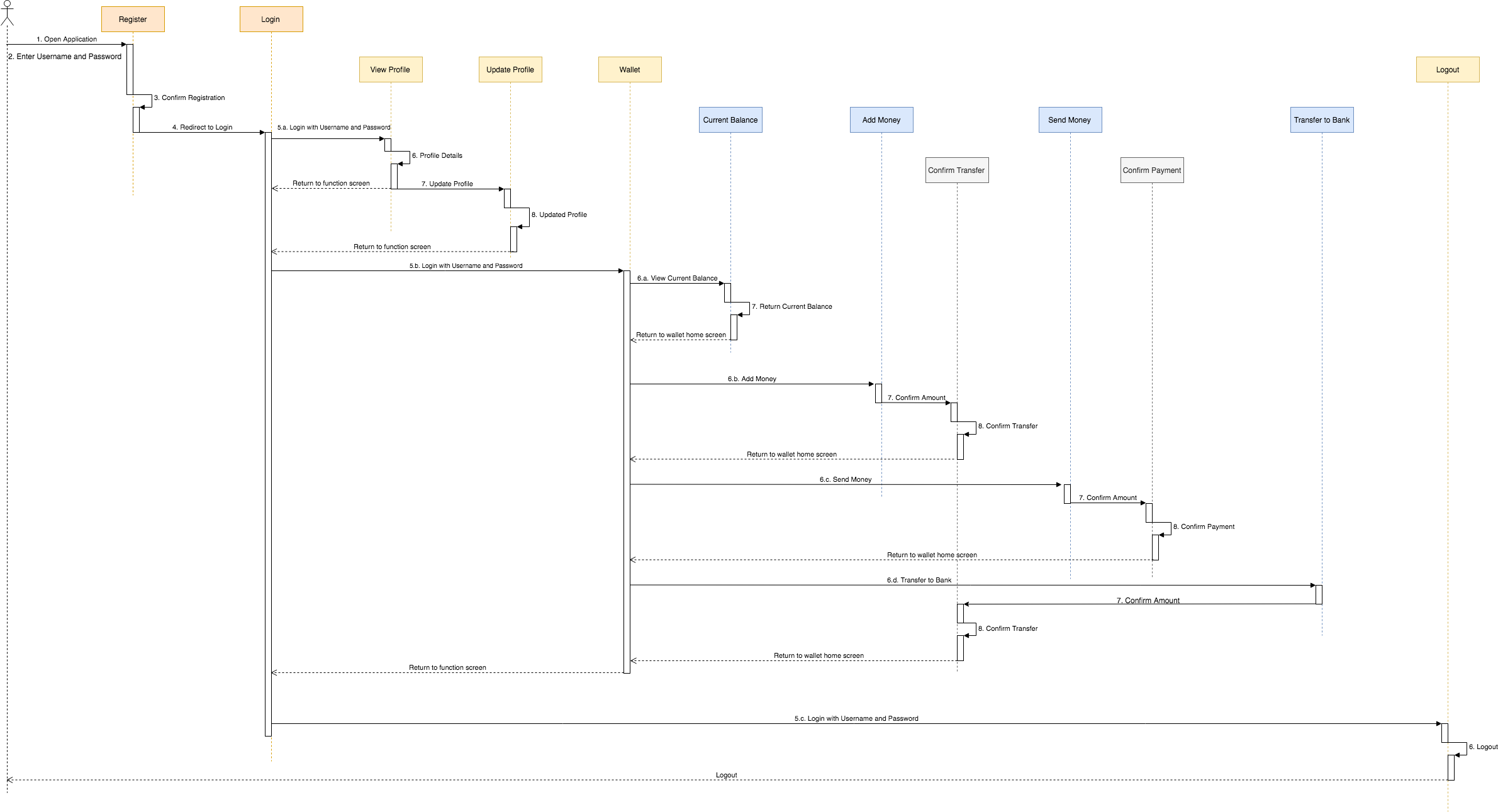


The Activity Diagram for the administrator is:



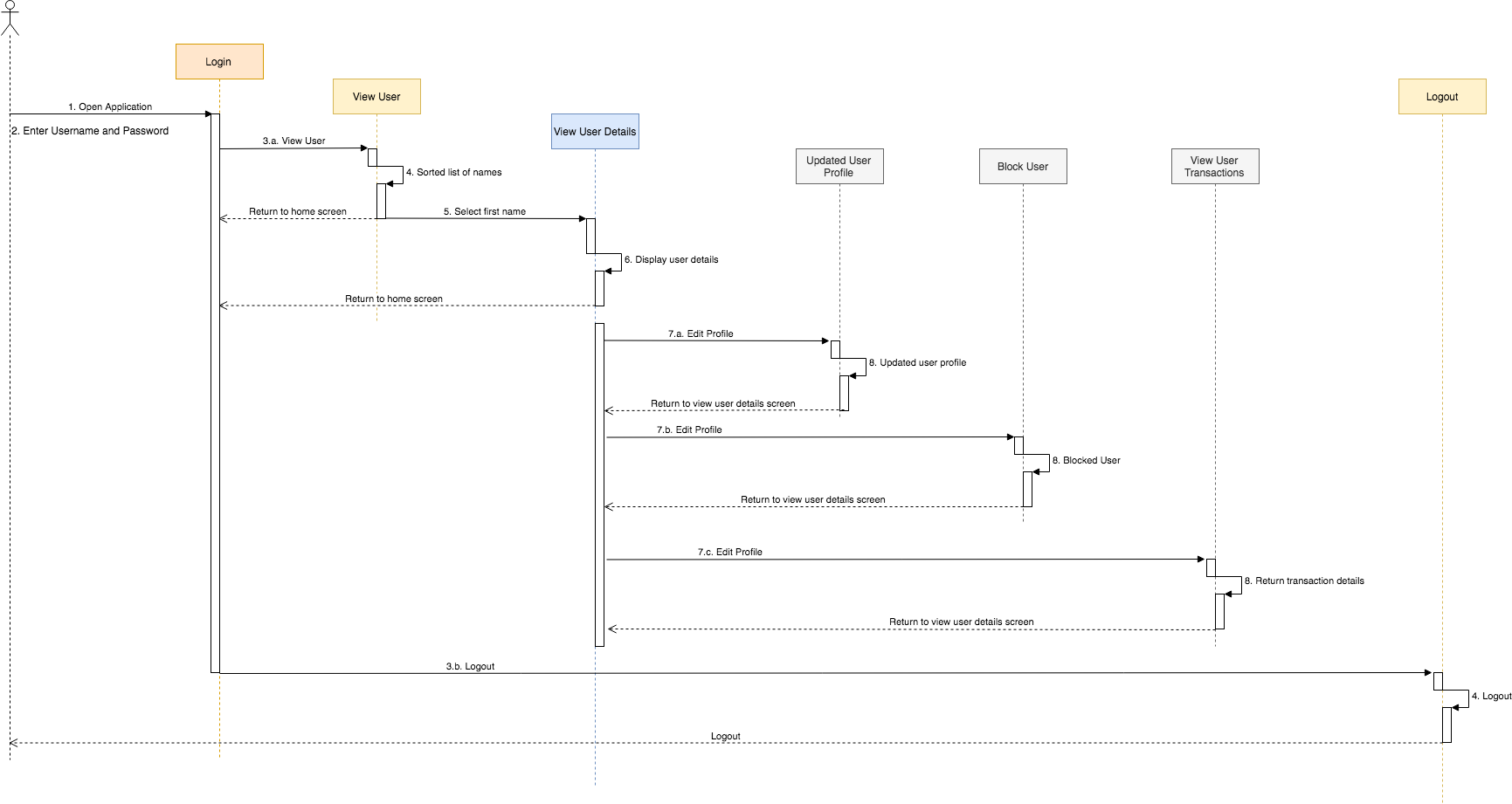


The Sequence Diagram for the user is:

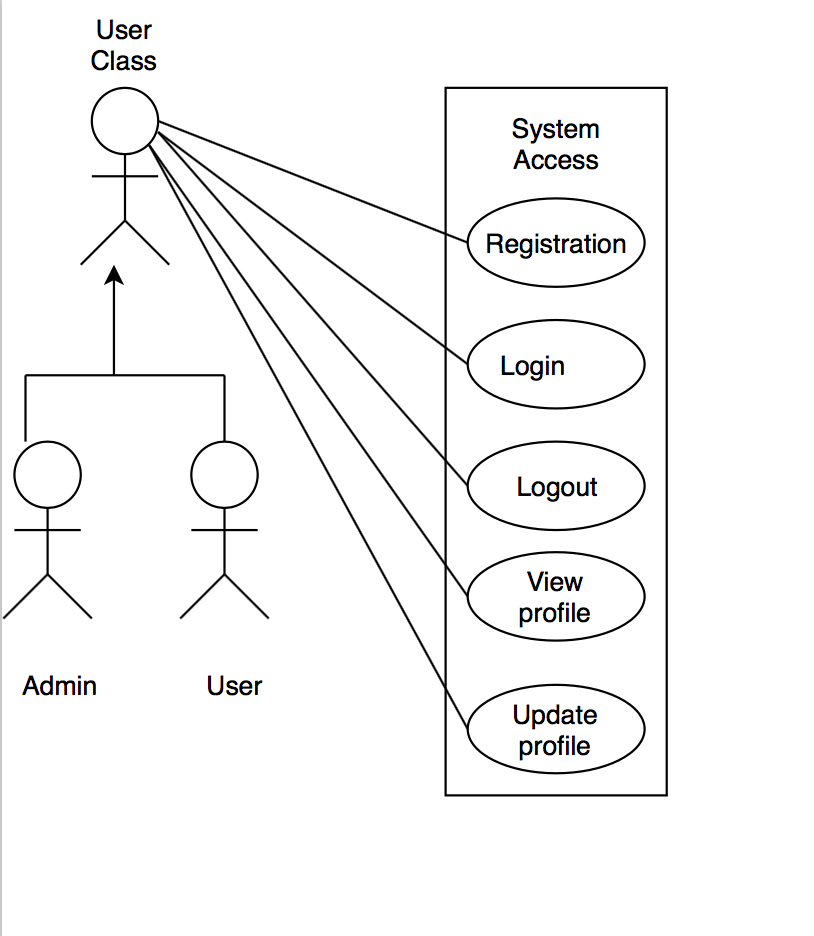




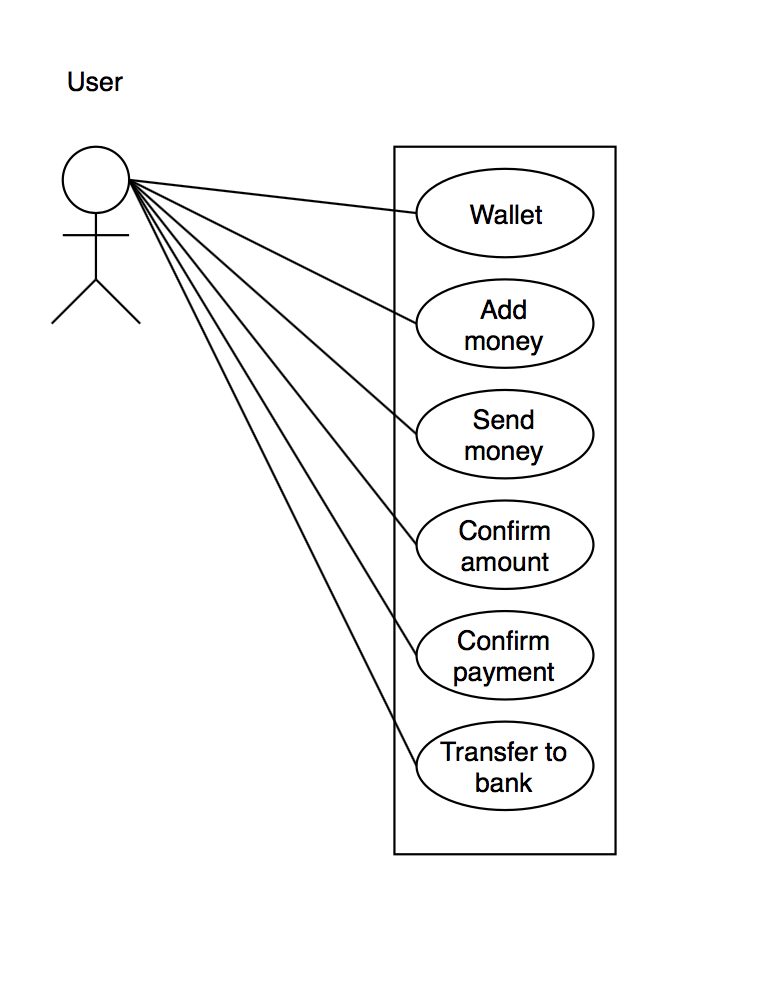
The Sequence Diagram for the administrator is:



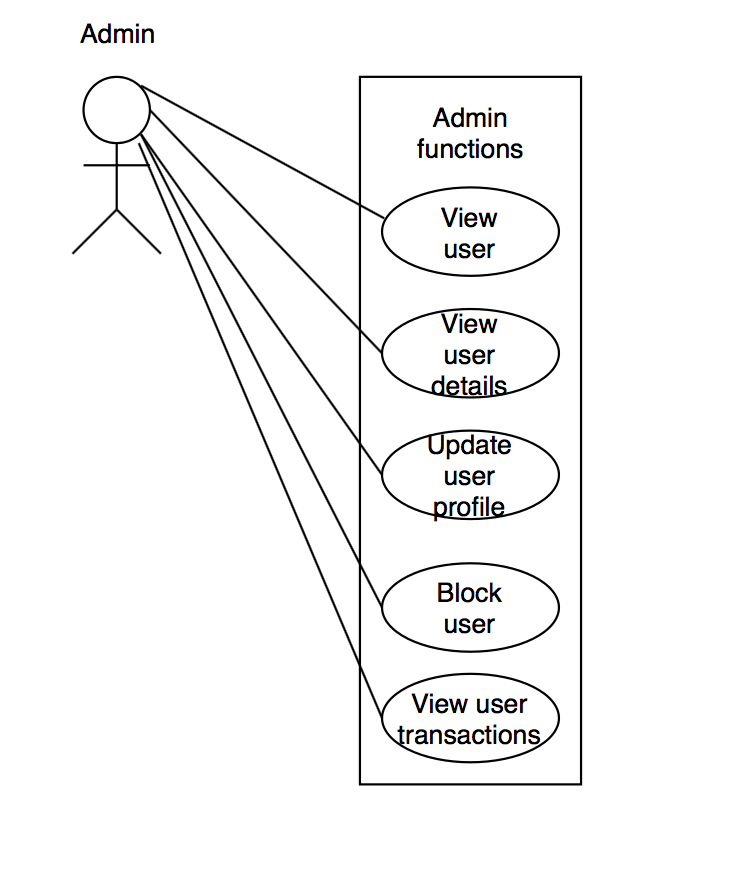
The Use Cases are as follows:













1. TESTING STRATEGY

The testing strategy will be implemented in the following ways.

* **Unit Testing:**

Unit tests are small program fragments written and maintained by developers to check the source code is working properly. This can either be automated or done using Junit.

* **API and Service Testing:**

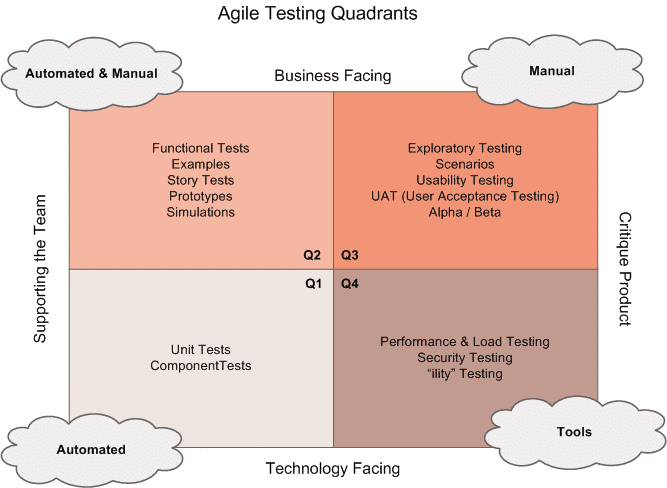
This is done by developers to ensure proper communication between all the components of the system. SoapUI can be used for this method.

* **Acceptance Testing:**

This is done by developers and quality analysts to check if the client’s requirements are met. The testing is done with respect to user needs, requirements, and business processes to determine if the system satisfies the acceptance criteria and to enable the customers to decide whether or not they accept the system.

* **System Testing:**

This testing is done to ensure that the whole system works when it is integrated and meets the necessary requirements. This is done by quality analysts, business analysts, and the product owner.





* **Product Backlog:**

Software development failure usually occurs due to unspecific requirements and different interpretations of the requirements by different members of the team. The requirements should be clear, concise, and unambiguous. They should have an acceptance criterion and have the following properties:

1. Independent
2. Negotiable
3. Valuable
4. Estimable
5. Small
6. Testable

* **Sprint Planning:**

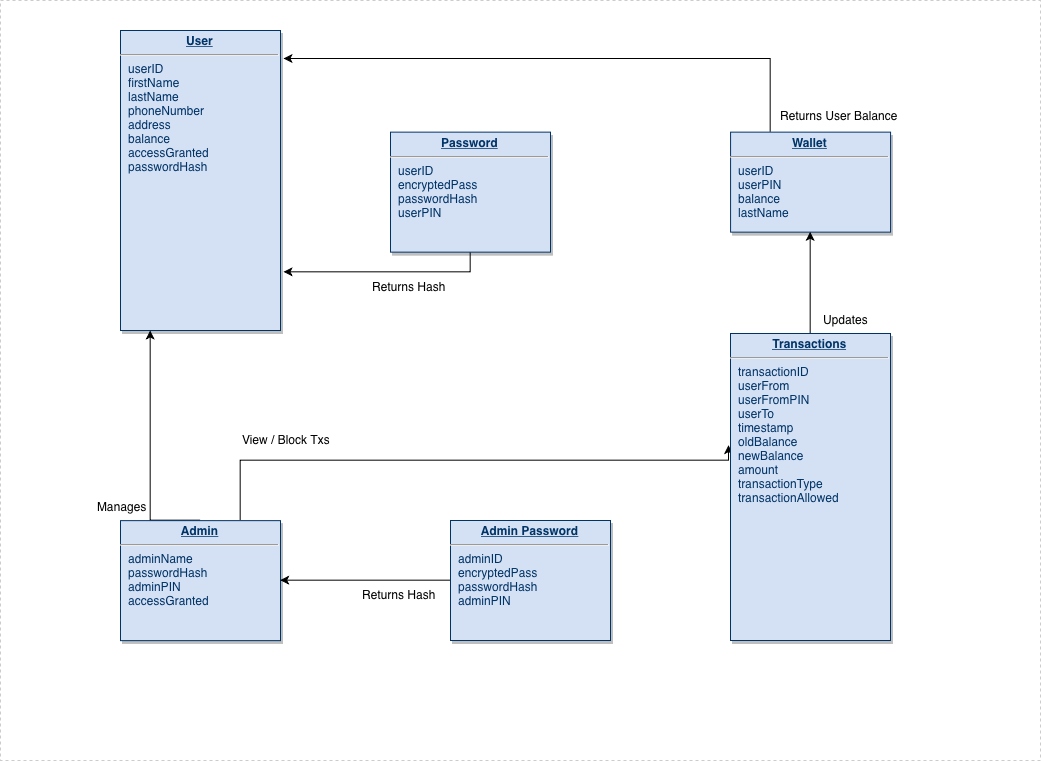
Developers should have a good understanding of the technical details that are necessary to meet the user requirements. To prevent defects, business analysts, quality analysts, and developers should be involved during the requirements stage. This will reduce the chances of incorrect implementation and ensure that the requirements are interpreted by everybody in the same way.

* **Dynamic Analysis Security Testing:**

This will help find certain vulnerabilities in the application when they are running in production. This can help find input and output validation problems, SQL injection, and configuration mistakes. Veracode will be used to perform DAST. It will simulate the actions of an attacker to discover vulnerabilities. A detailed report on critical vulnerabilities and information that lets them recreate the flaws is given to the developers and quality analysts. It also helps develop long-term strategies that helps application security.

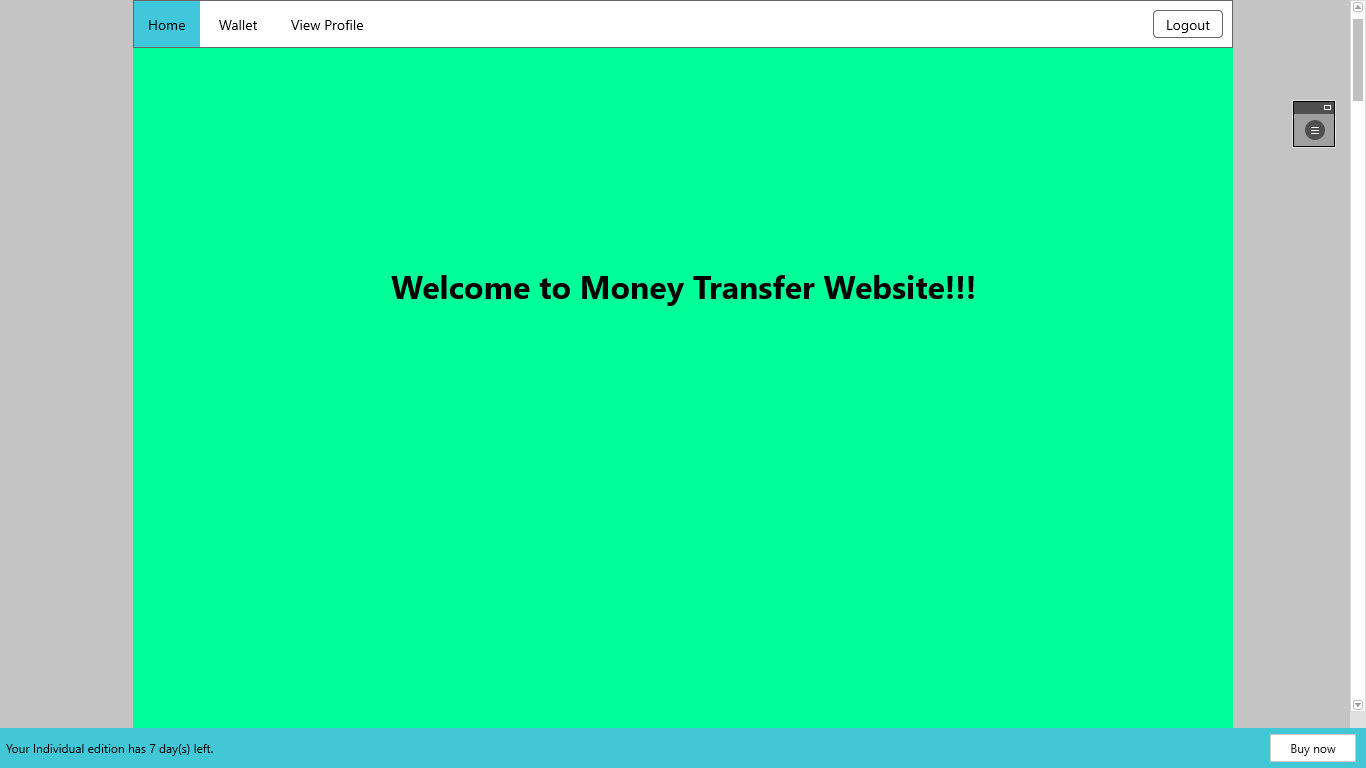
1. DATABASE TABLE DESIGN

The Entity Relationship diagram for the database is:

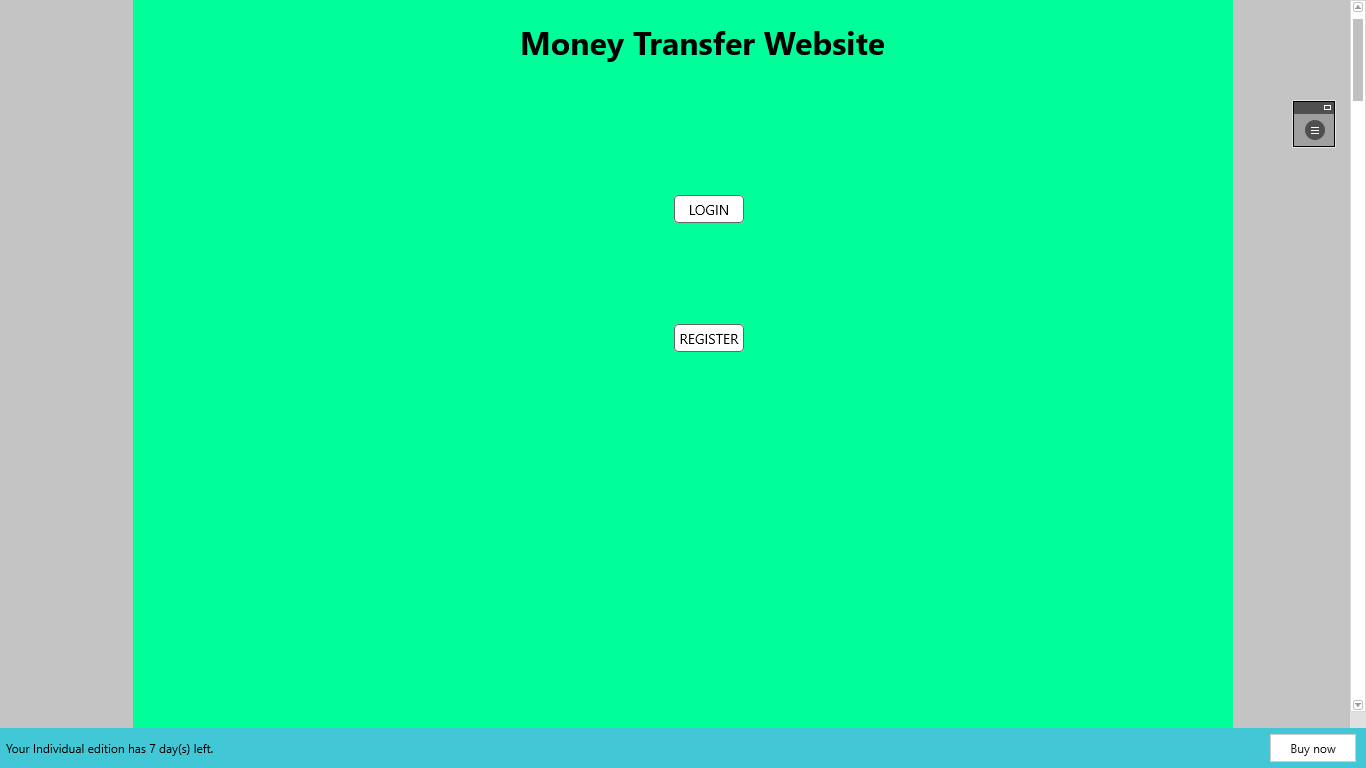




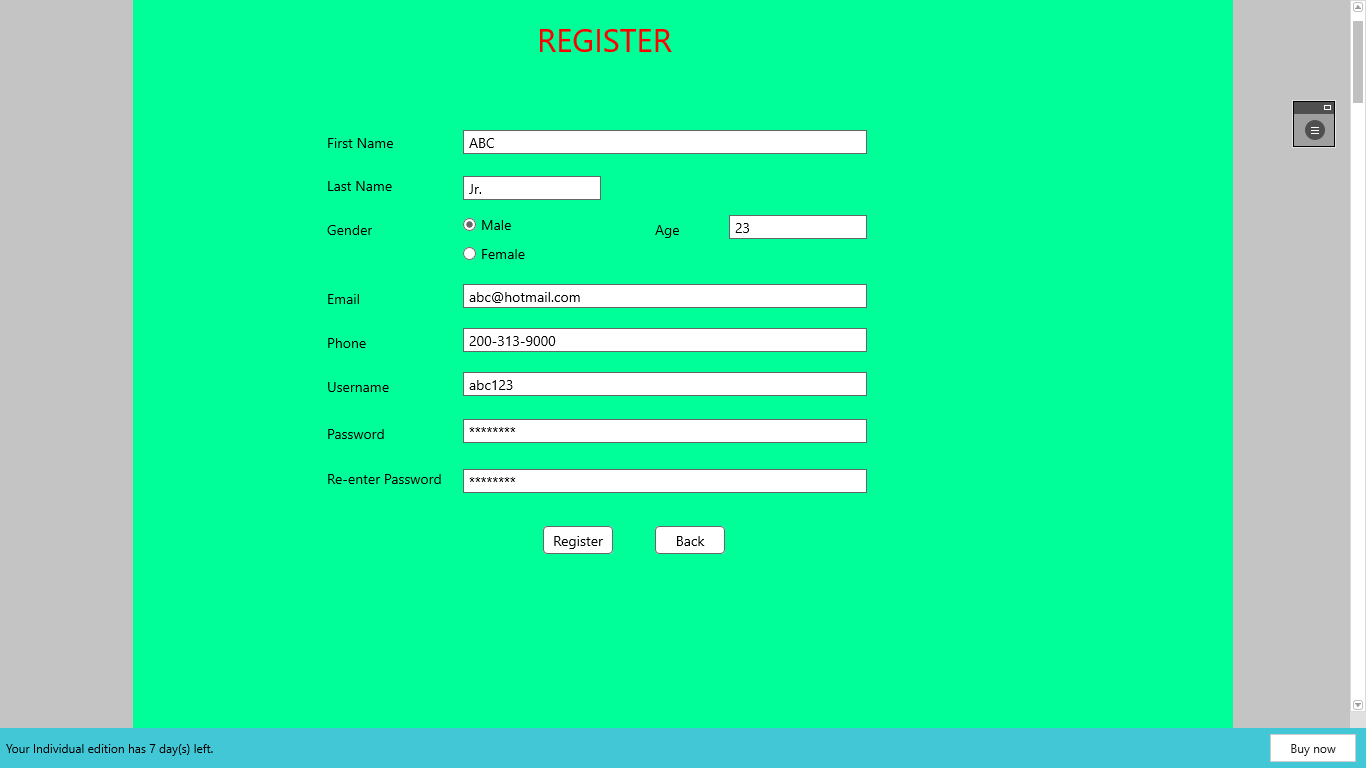
1. UI MOCKUPS

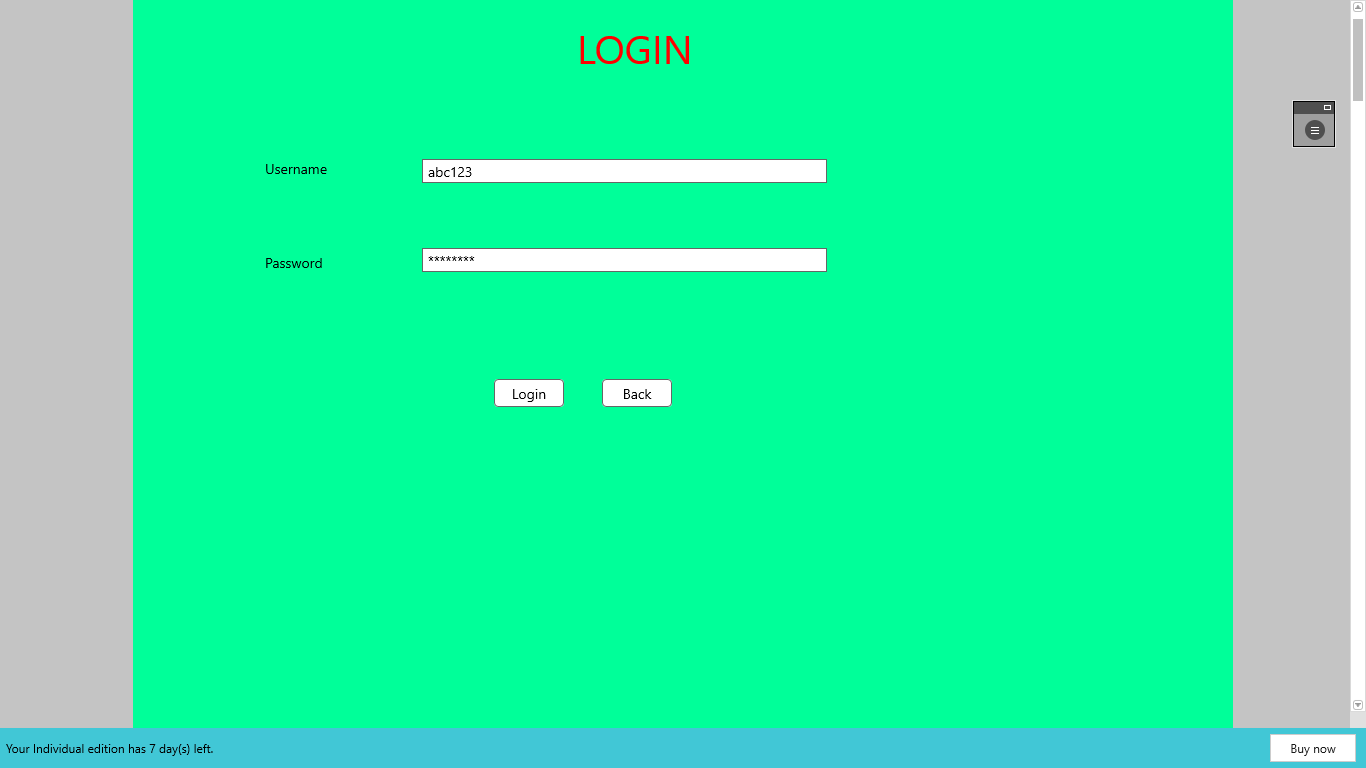




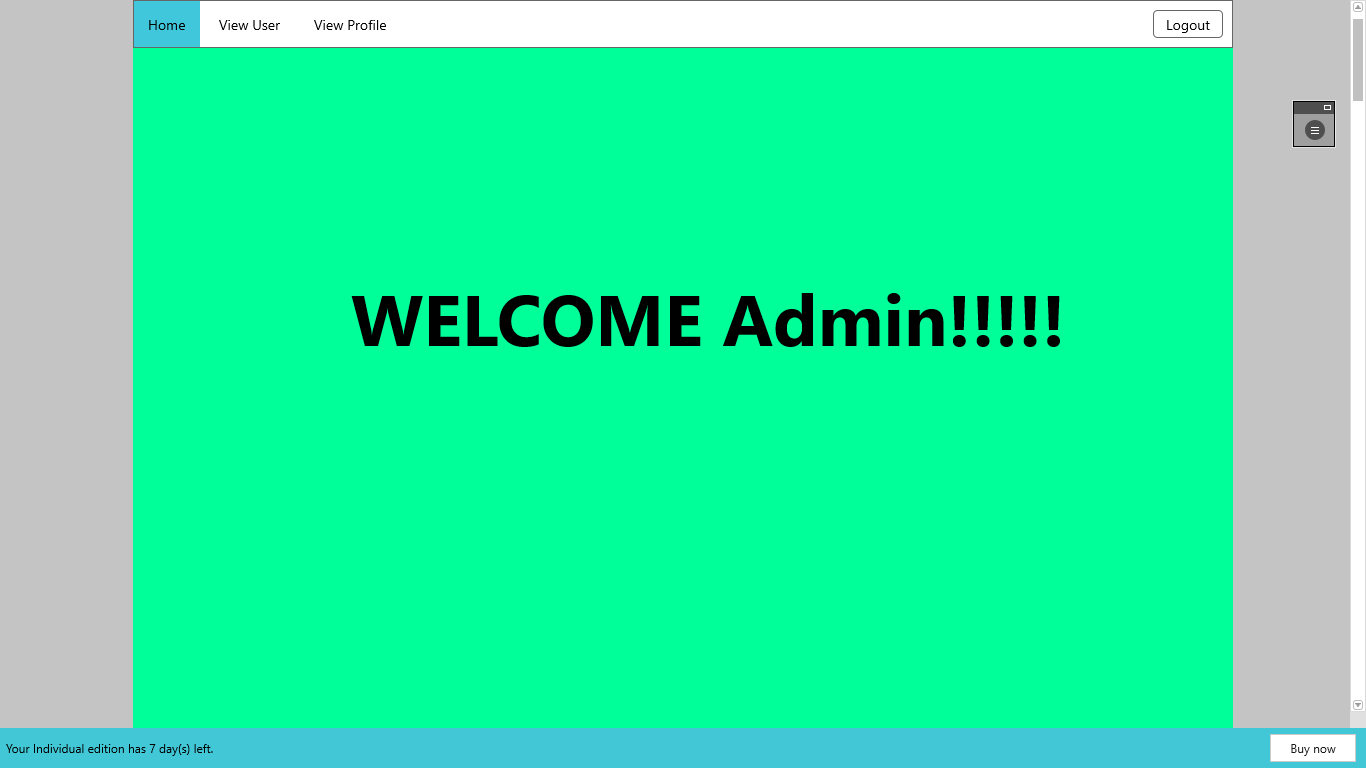




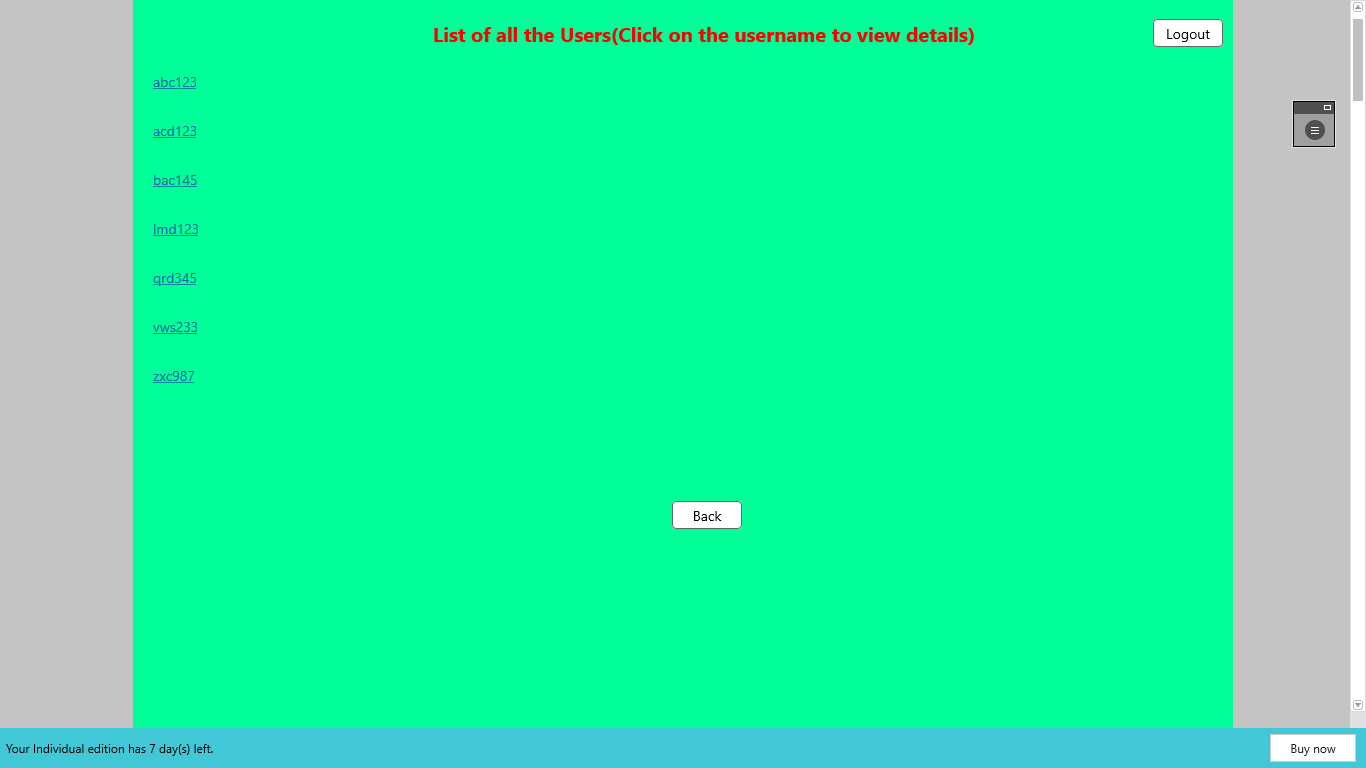


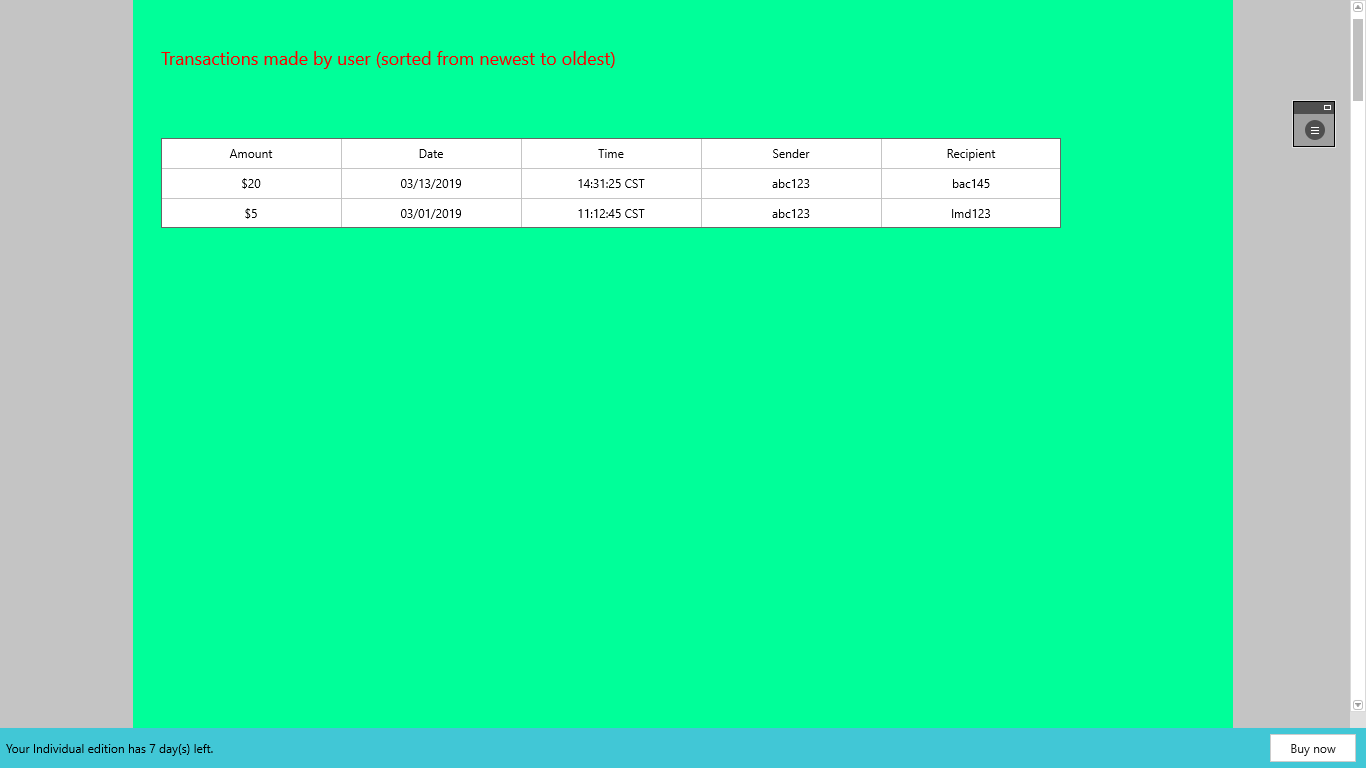
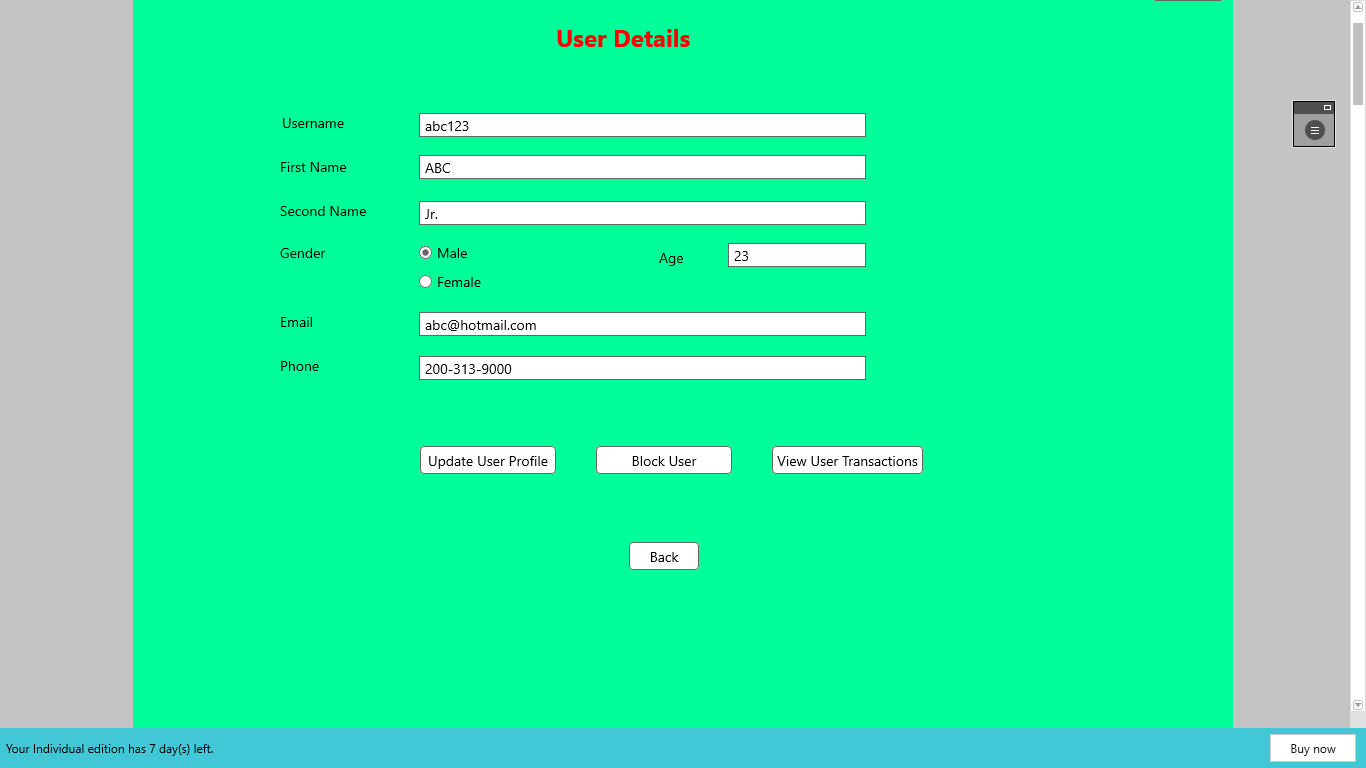




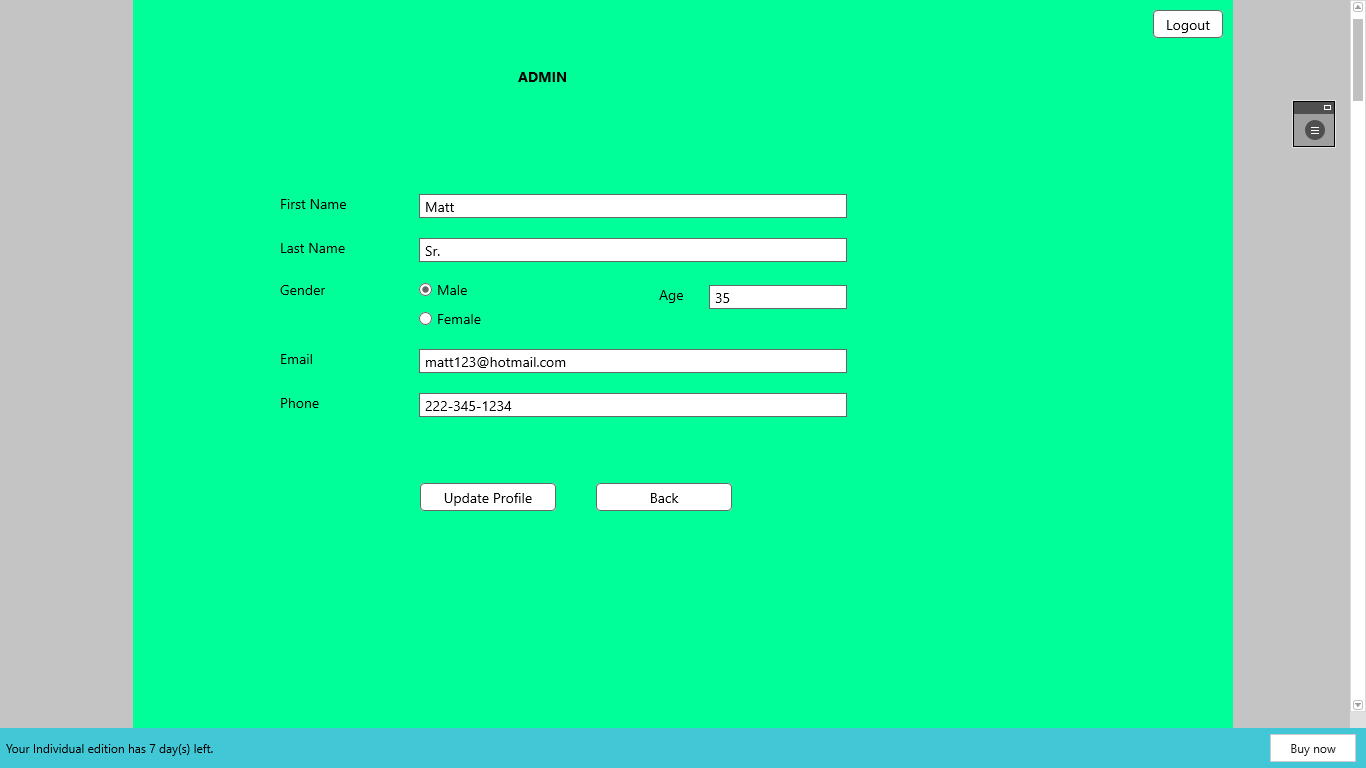


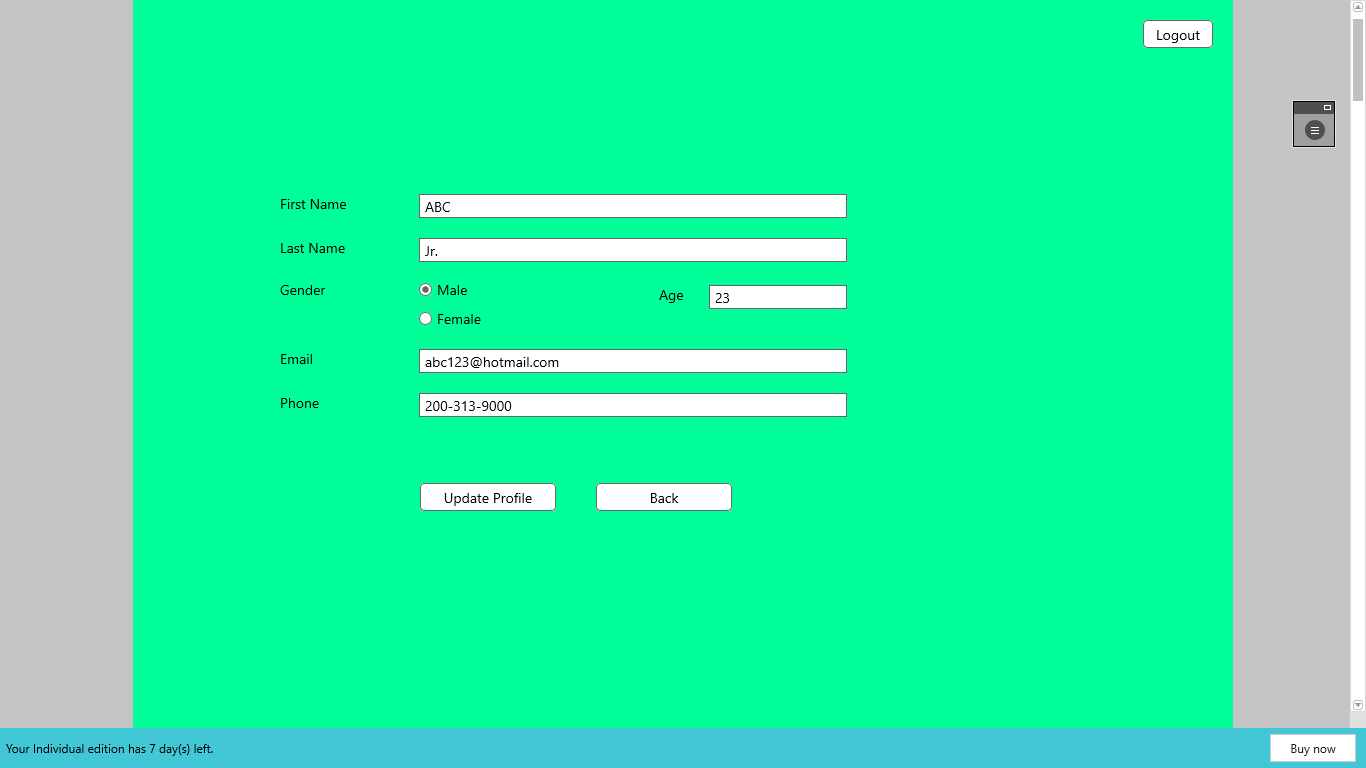


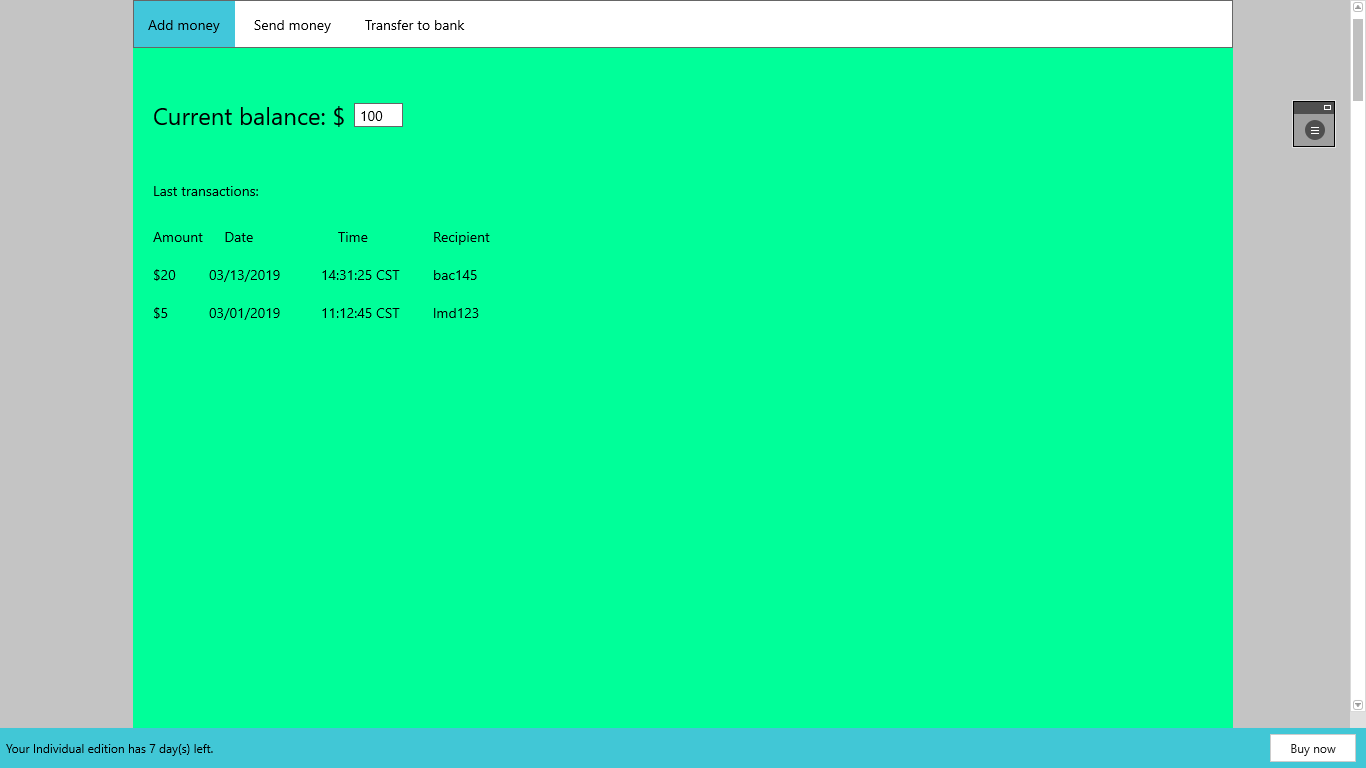


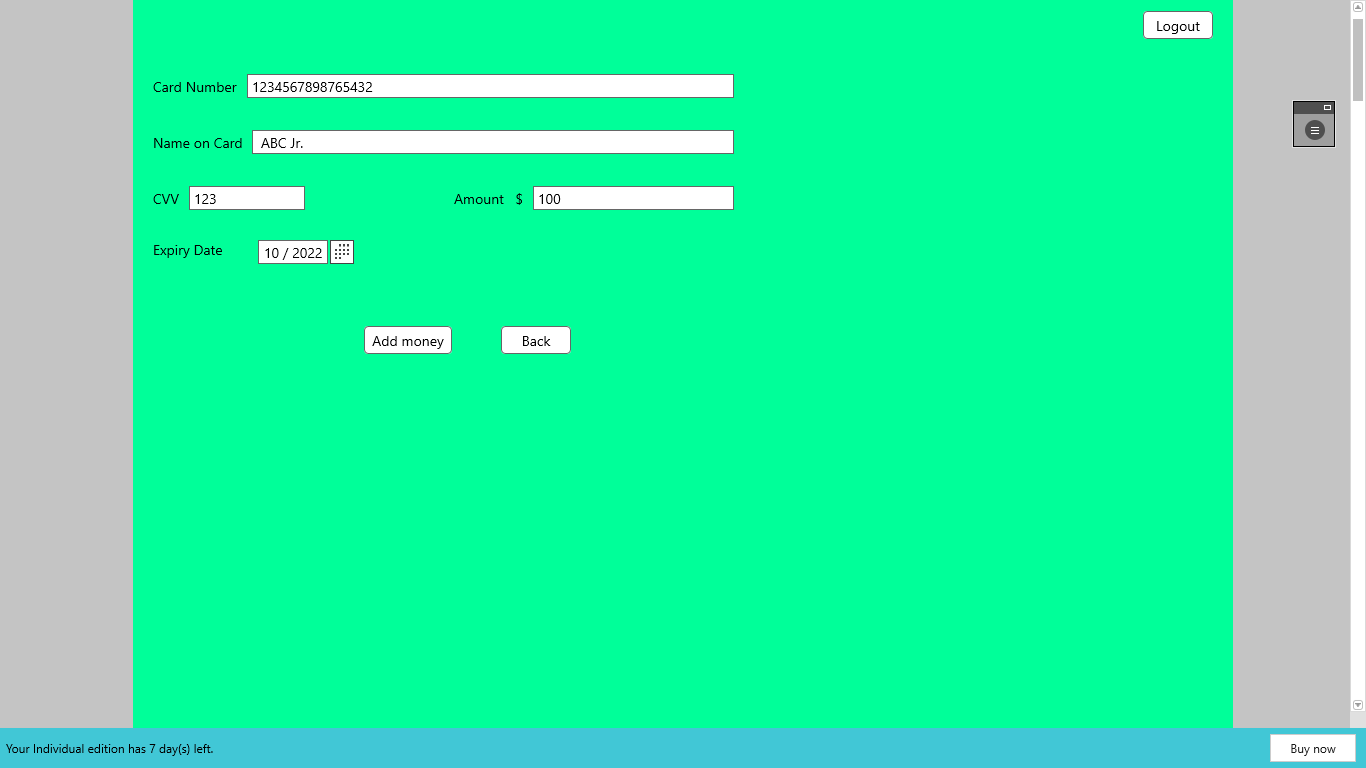




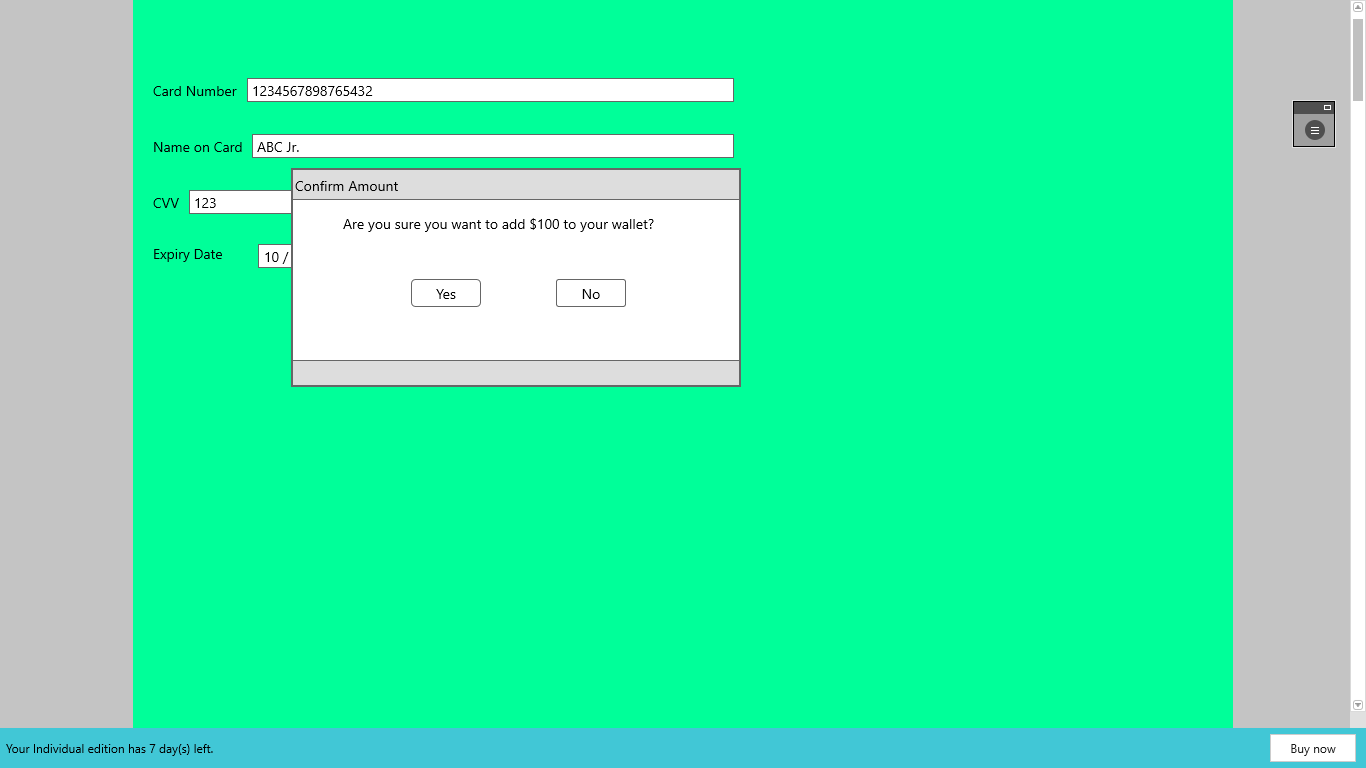






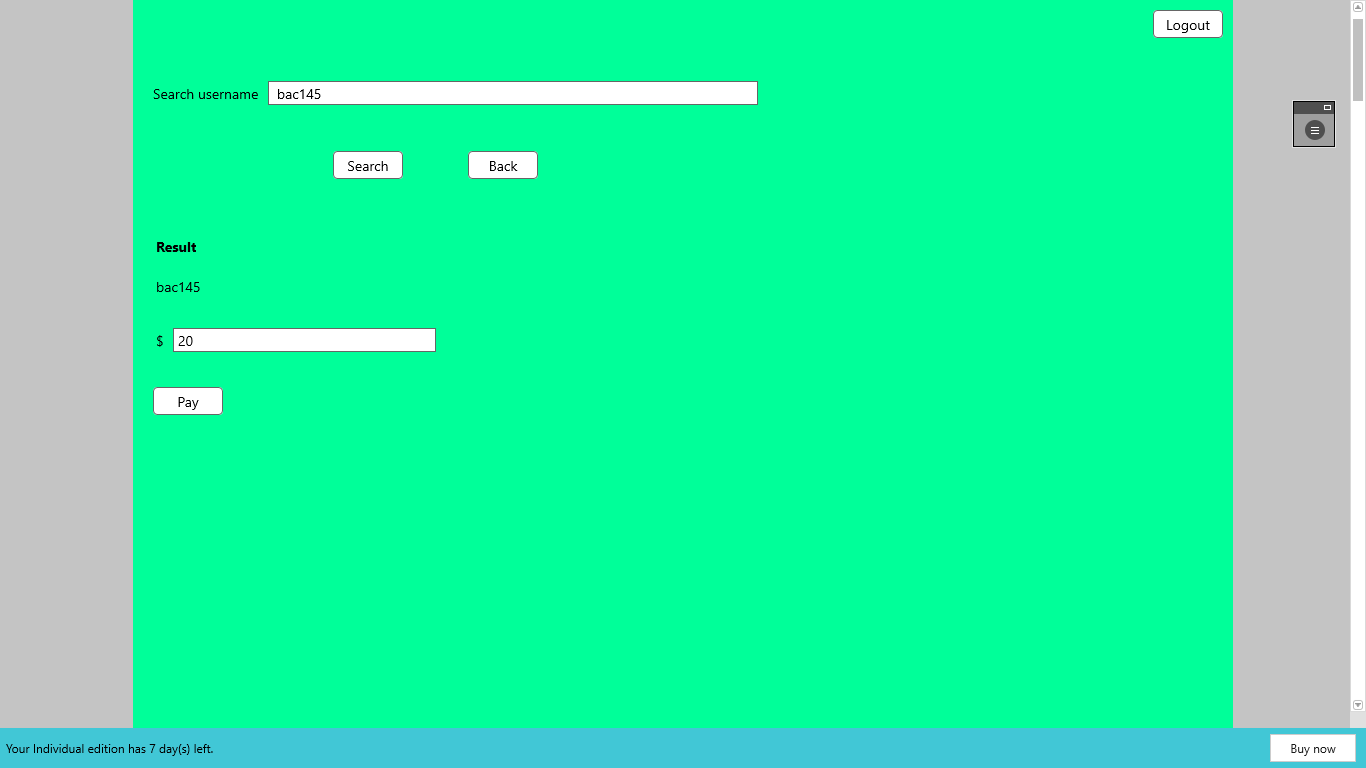


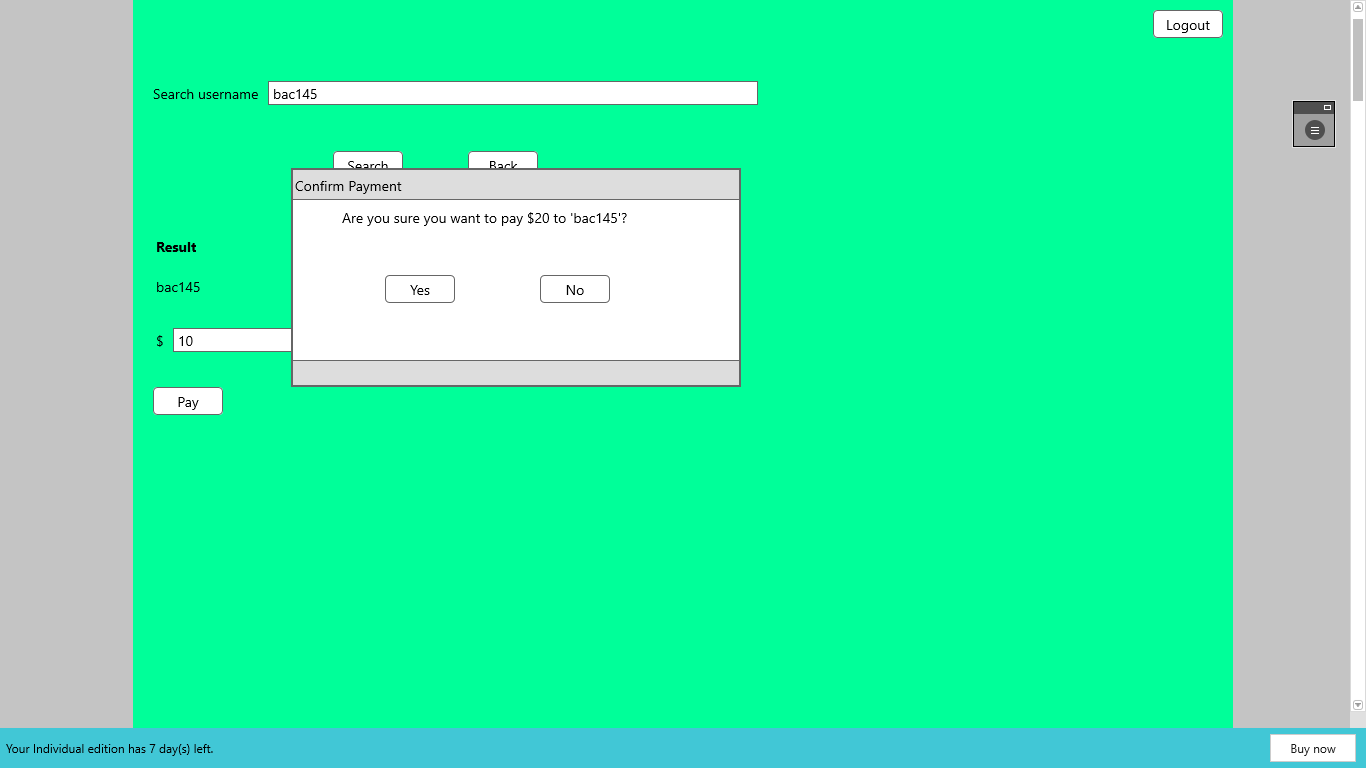




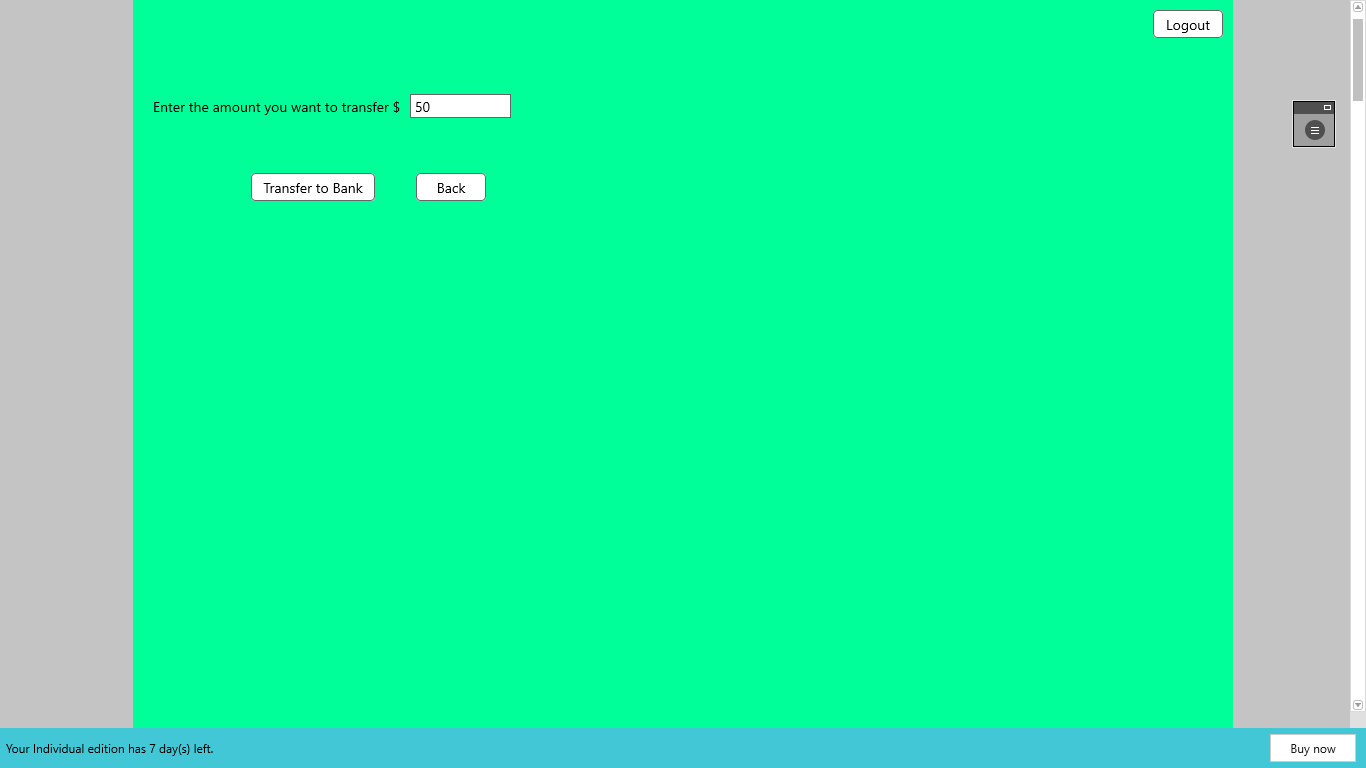


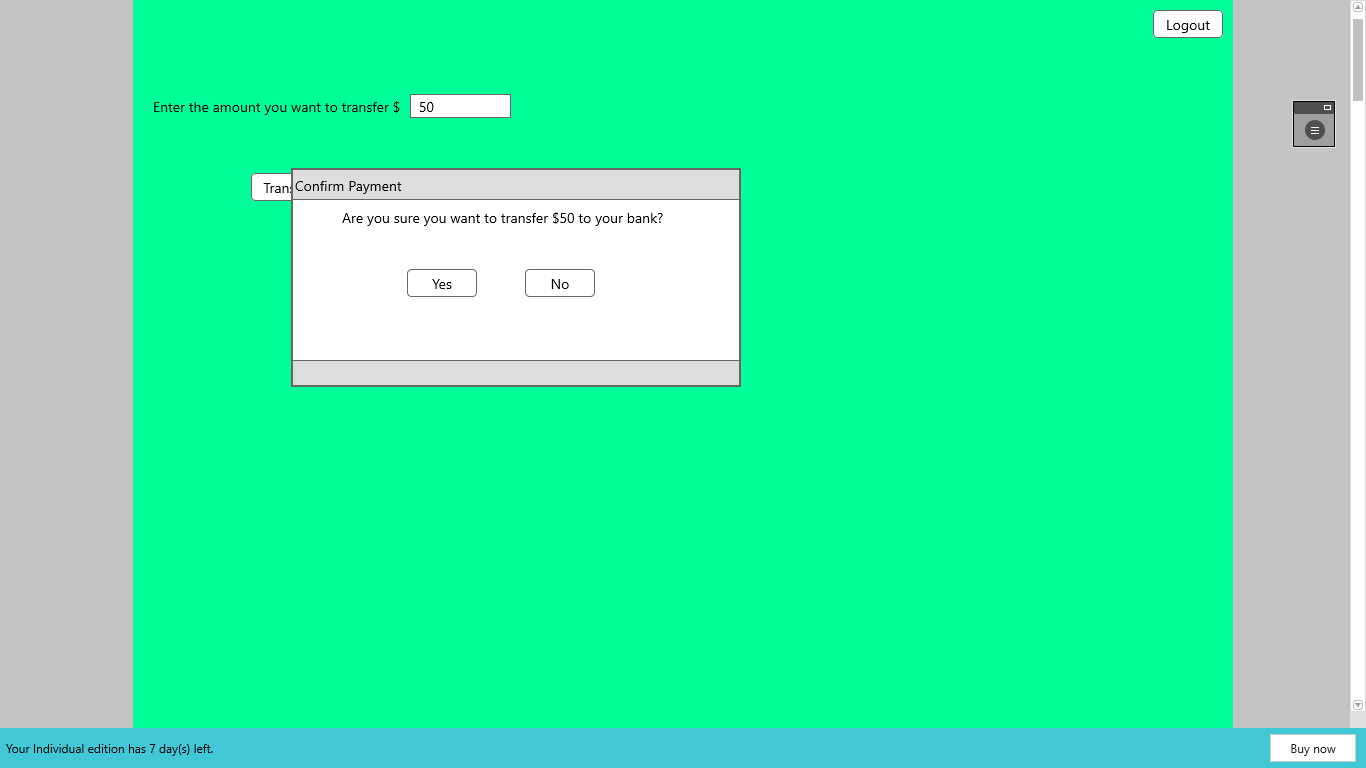














REFERENCES

* <https://www.testingexcellence.com/agile-test-strategy-example-template/>
* <https://www.agilealliance.org/glossary/unit-test>
* <http://softwaretestingfundamentals.com/acceptance-testing/>
* <https://capec.mitre.org/>
* <https://www.veracode.com/security/dast-test>