**Foreign Currency Buying**

**Software Design Document**

**Version <1.0>**

**Team 8:**

**Aidan Dobkins**

**Juliana Tomaiko**

**Justin Hancock**

**Patrick Miller**

**Tyler Stephens**

**2021-10-29**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 2021-10-29 | <1.0> | Initial Version of Document | Team 8 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1 Introduction....................................................................................................................................... 4

1.1 Purpose............................................................................................................................ 4

1.2 Scope............................................................................................................................... 4

1.3 Definitions, Acronyms, and Abbreviations........................................................................ 5

1.4 References....................................................................................................................... 6

1.5 Overview.......................................................................................................................... 6

2 Glossary............................................................................................................................................ 7

3 Uses Cases....................................................................................................................................... 7

3.1 Actors................................................................................................................................ 7

3.2 List of Use Cases.............................................................................................................. 7

3.3 Use Case Diagrams.......................................................................................................... 8

3.4 Use Cases........................................................................................................................ 8

4 Design Overview............................................................................................................................... 17

4.1 Introduction..................................................................................................................................... 17

4.2 System Architecture........................................................................................................................ 17

4.3 System Interfaces........................................................................................................................... 17

4.4 Constraints and Assumptions......................................................................................................... 19

5 Object Descriptions............................................................................................................................ 19

5.1 Objects............................................................................................................................................ 19

6 Object Collaboration........................................................................................................................... 25

6.1 Object Collaboration Diagram......................................................................................................... 25

**Software Design Document**

**1 Introduction**

The Software Design Document is a document that serves to plan and aid software development that provides a clear, concise narrative that is easy to follow. The SDD also includes use case diagrams, use cases, object class descriptions, user interface mockups, an architectural diagram, and an object collaboration diagram.

**1.1 Purpose**

The purpose of the Software Design Document is to represent software design by including design descriptions and information in order to implement that software in future stages. The SDD includes diagrams and detailed descriptions to effectively communicate the design to customers and stakeholders, and to address any concerns for the Foreign Currency Buying System.

**1.2 Scope**

The Software Design Document is a part of the design process that serves as a building block to implement the software itself. This stage involves detailed description of the system, elements of architectural design, and UML diagrams and mockups to demonstrate how the system will behave and how potential users may interact with the system. This particular SDD will include these descriptions and diagrams for the Foreign Currency Buying system.

**1.3 Definitions, Acronyms, and Abbreviations**

* Object – Used in object-oriented programming, can be a variable, function, or data structure that is an instance of a class. <https://en.wikipedia.org/wiki/Object_(computer_science)>
* Object collaboration diagram – A UML graphical method that shows objects and actors and how they collaborate on a task. The purpose is to show relationships between objects.
* <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml-collaboration-diagram/>
* Object-oriented programing (OOP) – A way of programming that is based on objects that contain attributes and methods. <https://en.wikipedia.org/wiki/Object-oriented_programming>
* System Architecture – A system architecture is a form of conceptual modelling that outlines the possible structure of a software system and how it may behave once implemented.
* <https://en.wikipedia.org/wiki/Systems_architecture>
* Unified Modeling Language (UML) – UML is a modeling language that provides a standard of graphical representation for the purpose of designing software systems.
* <https://en.wikipedia.org/wiki/Unified_Modeling_Language>
* Use case diagram – A use case diagram is a graphical representation of user’s possible interactions with a system, showcasing different “actors” and how each interact with a specific system. <https://en.wikipedia.org/wiki/Use_case_diagram>

**1.4 References**

* IEEE Recommended Practice for Software Design Descriptions
  + IEEE 1016-2009

**1.5 Overview**

The Software Design Document is divided into 6 sections with their corresponding subsections. The sections of the Software Design Document are:

1 Introduction

2 Glossary

3 Use Cases

4 Design Overview

5 Object Descriptions

6 Object Collaborations

**2 Glossary**

2.1 Glossary is unused in current document due to Section 1.3 Definitions, Acronyms, and Abbreviations providing terms and definitions for internal use of the document.

**3 Use Cases**

Use-Case Model Survey

**3.1 Actors**

**3.1.1 General User**

3.1.1.1 Information: The general user of the site is a person who has/is going to create an account through the system. This person once verification is complete, can purchase currency through the system.

**3.1.2 Administrative User**

3.1.2.1 Information: The administrative user is a user who oversees the verification process of general user account creation (via checking photo ID) as well as the account verification of delivery agents. The administrative user is also responsible for managing issues reported by general users.

**3.1.3 Delivery Agent**

3.1.3.1 Information: The delivery agent is a user who upon account creation/verification by the administrator can complete orders, mark them as delivered, see current orders waiting to be delivered and certify a delivery for each transaction.

**3.1.4 System Under Design**

3.1.4.1 The system under design is the currency exchange system that is being created. The actor represents the system and the actions that it takes.

**3.2 List of Use Cases**

**3.2.1 General User Use Cases**

3.2.1.1 Purchase Currency

3.2.1.2 Submit Issue

3.2.1.3 Cancel Order

3.2.1.4 Create Account

**3.2.2 Administrative User Use Cases**

3.2.2.1 Create Account

3.2.2.2 Verify General User Photo ID

3.2.2.3 Verify Delivery Agent Account

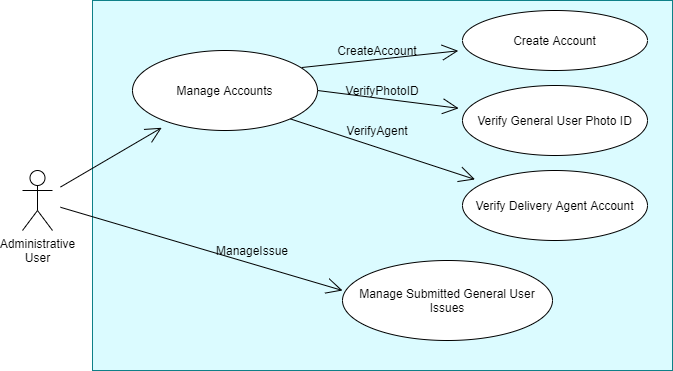
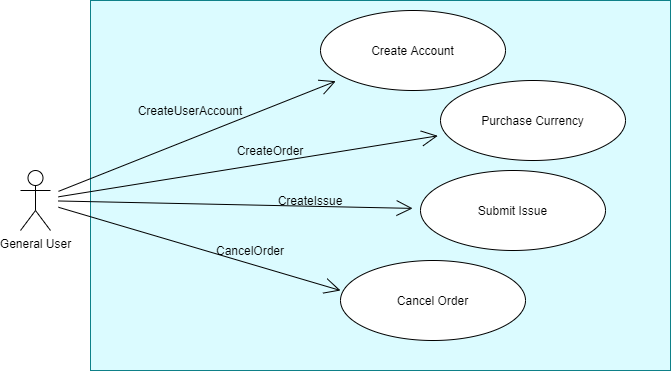
3.2.2.4 Manage Submitted General User Issues

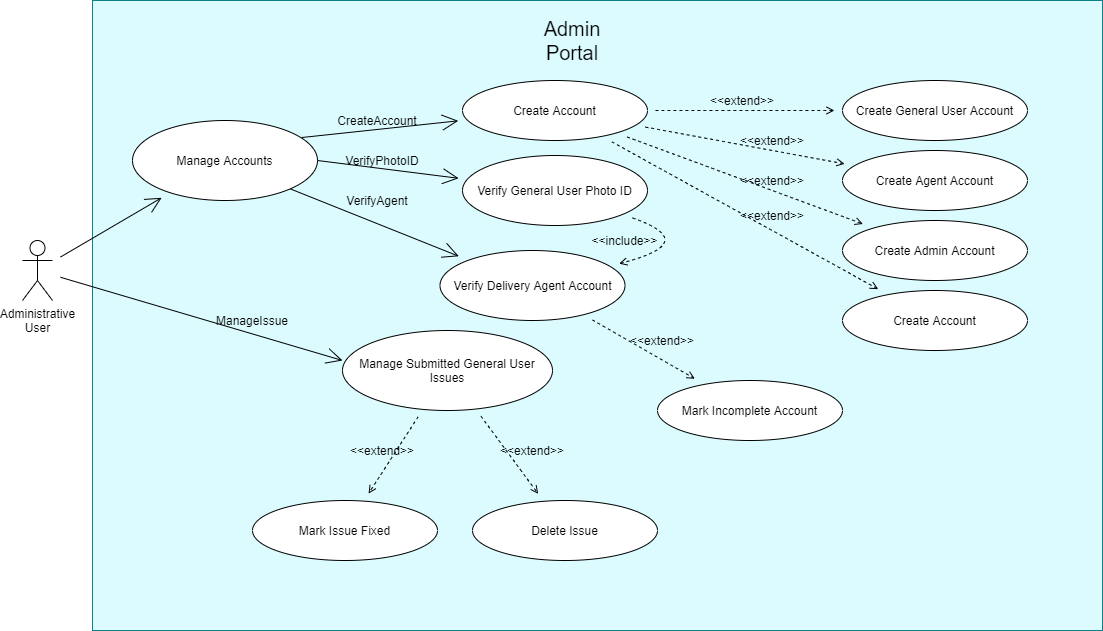
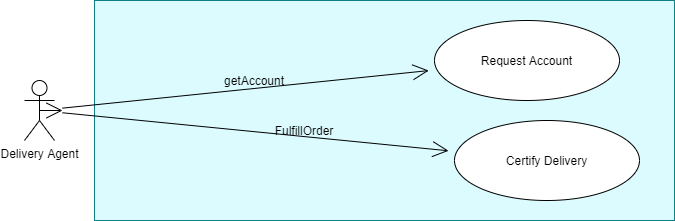
**3.2.3 Delivery Agent Use Cases**

3.2.3.1 Request Account

3.2.3.2 Certify Delivery

**3.3 Use Case Diagrams**





**3.4 Use Cases**

3.4.1.1 General User Use Cases – Purchase Currency

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:**  Purchase Currency | | **ID:**  PC | | **Priority:**  High | |
|  | **Primary actor:**  General User | **Source:**  Currency Buyers, | **Use case type:**  Business | **Level:**  Detail |  |
| **Interested Stakeholders:**  Foreigners, travelers, anyone looking to exchange currency | | | | | |
| **Brief Description:**  This use case describes the process of purchasing available currency on the site. It is a very general use case that encompasses the entire currency purchasing process. | | | | | |
| **Goal:**   * To successfully purchase selected currency through purchasing system. | | | | | |
| **Success Measurement:**   * The user purchases desired currency | | | | | |
| **Precondition:**   * General user has successfully created an account and has had that account verified through the admin photo ID verification * The user has provided all necessary information in account sign up process   **Trigger:**   * The general user has reached a point where they are ready to purchase currency after all necessary information has been provided. | | | | | |
| **Relationships:**  Include:  Extend:  Depends On:   * General User Create Account | | | | | |
| **Typical flow of events:**  1. The general user uses the search function to find desired currency based on name/country  2. The user is then presented with a list view of currencies according to search  3. The user is able to select a currency and its denomination  4. The system under design populates the relative currency exchange information for the selected  currency  5. The user is able to select shipping option  6. The user is able to submit the order | | | | | |
| **Assumptions**  1. The system automatically updates the exchange rate at the given point intime and the exact  Exchange rate is made visible to the general user. The exchange rate is also applied and used  In transaction.  2. The general user has a verified account | | | | | |
| **Implementation Constraints and Specifications:** | | | | | |

3.4.1.2 General User Use Cases – Submit Issue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:**  Submit Issue | | **ID:**  SI | | **Priority:**  High | |
|  | **Primary actor:**  General User | **Source:**  Currency Buyers | **Use case type:**  Business | **Level:**  Detail |  |
| **Interested Stakeholders:**  Foreigners, travelers, anyone looking to exchange currency, System administrators | | | | | |
| **Brief Description:**  If an issue occurs at any point for the the general user, they are able to submit an issue within the transaction process. | | | | | |
| **Goal:**   * The general user has the ability to have their issues resolved by system administrators | | | | | |
| **Success Measurement:**   * General user is successfully able to submit issues. These issues are then viewed/resolved by system administrators | | | | | |
| **Precondition:** The general user has created an account that has been verified. They have made a transaction with the system  **Trigger:** | | | | | |
| **Relationships:**  Include:  Extend:  Depends On:   * General User Create Account * Purchase Currency | | | | | |
| **Typical flow of events:**  1. The general user once logged in can go to the Submit Issues page.  2. The general user has to provide the transaction number given to them when order was placed  3. The general user can describe the particular issues they have had with the transaction  4. Finally, the general user can submit the issue with the transaction into the system. | | | | | |
| **Assumptions**  1. The user is logged into their account  2. The user has submitted an order within the system | | | | | |
| **Implementation Constraints and Specifications:** | | | | | |

3.4.1.3 General User Use Cases – Cancel Order

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:**  Cancel Order | | **ID:**  CO | | **Priority:**  High | |
|  | **Primary actor:**  General User | **Source:**  Currency buyers | **Use case type:**  Business | **Level:**  Detail |  |
| **Interested Stakeholders:**  Foreigners, travelers, anyone looking to exchange currency, System administrators, Delivery agents | | | | | |
| **Brief Description:** Once the user has placed an order, the user has a time window to cancel that order. | | | | | |
| **Goal:**   * The user has the option to cancel the order at any point within cancellation window | | | | | |
| **Success Measurement:**   * The order is successfully cancelled by choice of the user | | | | | |
| **Precondition:**   * The general user is logged into their account * The general user has already placed an order   **Trigger:** | | | | | |
| **Relationships:**  Include:  Extend:  Depends On:   * General User Create Account * Purchase Currency | | | | | |
| **Typical flow of events:**  1. The user accesses the order cancellation window  2. The user can either enter the transaction number or select one of the transactions  from the drop down list  3. The general user can see if the selected transaction is within the cancellation window  4. The general user can then submit the cancellation | | | | | |
| **Assumptions**  1. The general user has access to transaction information | | | | | |
| **Implementation Constraints and Specifications:** | | | | | |

3.4.1.4 General User Use Cases – Create Account

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:**  General User Create Account | | **ID:**  GUCA | | **Priority:**  High | |
|  | **Primary actor:**  General User | **Source:**  Currency Buyers | **Use case type:**  Business | **Level:**  Detail |  |
| **Interested Stakeholders:**  Foreigners, travelers, anyone looking to exchange currency, System administrators | | | | | |
| **Brief Description:**  The general user via the login/signup page can create an account. | | | | | |
| **Goal:**   * The user successfully signs up for an account | | | | | |
| **Success Measurement:**   * Creation of user account | | | | | |
| **Precondition:**  The user has access to the system  **Trigger:**   * The general user once all information is entered signs up | | | | | |
| **Relationships:**  Include:  Extend:  Depends On: | | | | | |
| **Typical flow of events:**  1. The general user has the option to login or create an account via the sign up page.  2. The general user provides name, mailing address, email, phone number into given text fields. The  general user also selects the payment method.  3. The general user also uploads their photo identifcation | | | | | |
| **Assumptions**  1. The user is able to access the system | | | | | |
| **Implementation Constraints and Specifications:** | | | | | |

3.4.2.1 Administrative User Use Cases – Create Account

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:**  Administrator Create Account | | **ID:**  ACA | | **Priority:**  High | |
|  | **Primary actor:**  Administrative User | **Source:**  System administrators | **Use case type:**  Business | **Level:**  Detail |  |
| **Interested Stakeholders:**  System Administrators | | | | | |
| **Brief Description:**  The system administrators if they don’t have an account shall be able to create an account. | | | | | |
| **Goal:**   * The admin is successfully able to create an account | | | | | |
| **Success Measurement:**   * The system administrator is able to login to account after account creation | | | | | |
| **Precondition:**   * The administrator is an employee of the company * They have a password that can be used for account creation   **Trigger:**   * Once all sign-up steps are followed the administrative user sign up for account | | | | | |
| **Relationships:**  Include:  Extend:  Depends On: | | | | | |
| **Typical flow of events:**  1. The administrative user goes to the account creation page  2. The administrative user provides name, email, phone number, address, and password  3. The administrative user will then receive a confirmation email for the account creation  4. Once confirmation email is received, the account can be logged in | | | | | |
| **Assumptions**  1. The administrative user is given access to the system with administrative view/privileges | | | | | |
| **Implementation Constraints and Specifications:** | | | | | |

3.4.2.2 Administrative User Use Cases – Verify General User Photo ID

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:**  Verify General User Photo ID | | **ID:**  VGUPI | | **Priority:**  High | |
|  | **Primary actor:**  Administrative User | **Source:**  System Administrators | **Use case type:**  Business | **Level:**  Detail |  |
| **Interested Stakeholders:**  General Users who want to purchase currency, System Administrators | | | | | |
| **Brief Description:**  For every general user that creates an account, they have to submit a photo ID. The photo ID has to be verified by a system administrator in order for the general user account creation process to be finalized. | | | | | |
| **Goal:**   * An administrative user is successfully able to verify a general user photo ID | | | | | |
| **Success Measurement:**   * General user photo ID is either verified or rejected | | | | | |
| **Precondition:**   * The administrative is logged into their account * The administrative user is able to view photo ID of general users   **Trigger:**   * The administrative user is ready to approve general user account | | | | | |
| **Relationships:**  Include:  Extend:  Depends On:   * General User Create Account | | | | | |
| **Typical flow of events:**  1. The general user has submitted a photo ID in the general user account sign up.  2. The administrative user is able to view the photo ID of the general user and scan it  3. The administrative user then verifies the photo ID  4. The administrative user can approve or reject the general user account | | | | | |
| **Assumptions**  1. The system is populated with general users who are awaiting photo ID verification | | | | | |
| **Implementation Constraints and Specifications:** | | | | | |

3.4.2.3 Administrative User Use Cases – Verify Delivery Agent Account

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:**  Verify Delivery Agent Account | | **ID:**  VDAA | | **Priority:**  High | |
|  | **Primary actor:**  Administrative User | **Source:**  Delivery Agent | **Use case type:**  Business | **Level:**  Detail |  |
| **Interested Stakeholders:**  System administrators, delivery employees | | | | | |
| **Brief Description:** Once a delivery agent has requested account creation, the administrative user can then approve or reject the account request. | | | | | |
| **Goal:**   * To successfully verify/reject a delivery agent’s account | | | | | |
| **Success Measurement:**   * Delivery agent’s account is approved | | | | | |
| **Precondition:**   * Administrative user is logged into account and has access to the system   **Trigger:**   * Administrative user has reached a point where delivery agent account can be approved | | | | | |
| **Relationships:**  Include:  Extend:  Depends On:   * Delivery Agent Request Account | | | | | |
| **Typical flow of events:**  1. The administrative user can view the delivery agent’s account request  2. The administrative user is then able to approve the account based on proper guidelines/information.  3. The delivery agent is able to login to the account once approval is completed. | | | | | |
| **Assumptions**  1. The administrative user is able to view delivery agent account requests that are populated through the system. | | | | | |
| **Implementation Constraints and Specifications:** | | | | | |

3.4.2.4 Administrative User Use Cases – Manage Submitted General User Issues

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:**  Manage Submitted General User Issues | | **ID:**  MSGUI | | **Priority:**  High | |
|  | **Primary actor:**  Administrative User | **Source:**  General Users | **Use case type:**  Business | **Level:**  Detail |  |
| **Interested Stakeholders:** | | | | | |
| **Brief Description:** The administrative user can see the issues submitted by the general users and respond to the issue via selected response. | | | | | |
| **Goal:**   * The administrative user is successfully able to respond to the general user submitted issues | | | | | |
| **Success Measurement:**   * The general user receives a response from administrative user | | | | | |
| **Precondition:**   * The general user has submitted an issues via the submit issue page   **Trigger:**   * Once administrative user has selected a response path (email, phone, letter) and formulated a response to the given issue. The response is carried out and sent to user. | | | | | |
| **Relationships:**  Include:  Extend:  Depends On:   * Submit Issue | | | | | |
| **Typical flow of events:**  1. The general user submits an issue through the submit issue page.  2. The administrative user can view this submission.  3. The administrative user can then formulate a response to the submitted issue  4. The administrative user selects form of response: letter, email, or phone call 5. | | | | | |
| **Assumptions**  1. The administrative user is provided with user submissions via the system | | | | | |
| **Implementation Constraints and Specifications:** | | | | | |

3.4.3.1 Delivery Agent User Use Cases – Request Account

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:**  Request Account | | **ID:**  RA | | **Priority:**  Medium | |
|  | **Primary actor:**  Delivery Agent | **Source:**  Delivery Employees | **Use case type:**  Business | **Level:**  Detail |  |
| **Interested Stakeholders:**  Delivery employees working for the company, system administrators | | | | | |
| **Brief Description:**  The delivery agent is able to request an account and apply for verification once all neccessary information is provided | | | | | |
| **Goal:**   * To successfully apply for account within the system | | | | | |
| **Success Measurement:**   * Delivery agent user has account verified by admin | | | | | |
| **Precondition:**   * Delivery agent user has access to the system * Delivery agent user is a registered employee of the company   **Trigger:**   * Once registration is successful the delivery agent has applied for verification | | | | | |
| **Relationships:**  Include:  Extend:  Depends On: | | | | | |
| **Typical flow of events:**  1. Delivery agent registers on the website by providing an email address, home address, name, phone number, and password.  2. Delivery agent has applied for verification.  3. An administrative user will approve the account (verification).  4. The delivery agent will receive a confirmation email from the administrative user.  5. The delivery agent is now able to login to the system. | | | | | |
| **Implementation Constraints and Specifications:** | | | | | |

3.4.3.2 Delivery Agent User Use Cases – Certify Delivery

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name:**  Certify Delivery | | **ID:**  CD | | **Priority:**  High | |
|  | **Primary actor:**  Delivery Agent | **Source:** | **Use case type:**  Business | **Level:**  Detail |  |
| **Interested Stakeholders:**  General Users who have placed an order for currency, delivery agents accessing the system | | | | | |
| **Brief Description:** Once an order has been placed by the general user, the delivery agent can verify the delivery. | | | | | |
| **Goal:**   * To successfully verify the delivery for the general user | | | | | |
| **Success Measurement:**   * The delivery is marked as active or verified | | | | | |
| **Precondition:**   * Delivery agent is logged in * User provided signature input   **Trigger:**   * Given the delivery meets guidelines and signature is present, the delivery agent is ready to verify the delivery. | | | | | |
| **Relationships:**  Include:  Extend:  Depends On:   * Purchase currency | | | | | |
| **Typical flow of events:**  1. General User has already placed an order for currency  2. General User has provided signature input  3. Delivery agent user enters the transaction I.D.  4. Delivery agent user verifies the delivery or marks delivery active if no general user signature is present | | | | | |
| **Assumptions**  1. System has successfully updated delivery status/purchases that are active | | | | | |
| **Implementation Constraints and Specifications:** | | | | | |

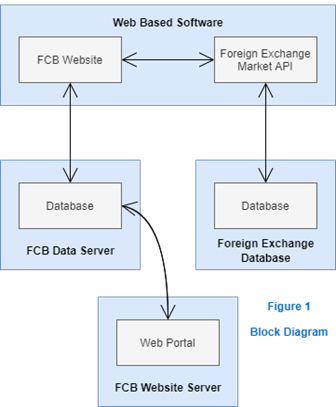
**4 Design overview**

**4.1 Introduction**

The design overview section gives a concise summary of the design of the software. The System Architecture section gives a visual view of the system components and the context that it shares with external systems. Before going into the specific details of the design of the software, the reader should familiarize themselves with the overview.

**4.2 System Architecture**

4.2.1 Overall Structure for the Foreign Currency Buying Software

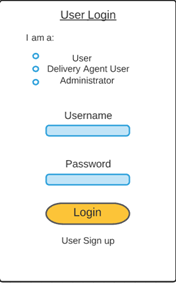
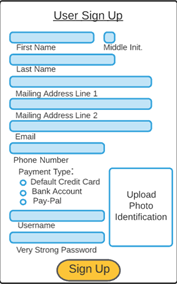


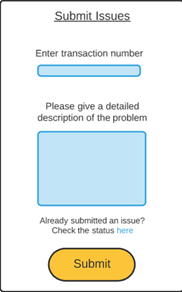
**4.3 System Interfaces**

**4.3.1 External User Interface Requirements**

4.3.1.1 User Interfaces

The user interface will allow a first-time user to easily sign up for a user account. Then the user can login and enter the system. The user will be presented with all the main functions on the first user interface. The user shall be able to navigate through different pages through clicking or touch and can enter information via keyboard or touch. A delivery agent or admin shall also be able to easily sign up for an account and be presented with a login screen. The delivery agent and admin will then be able to navigate through the software in the same form as the user. Centralized hosting will be utilized via a web interface and the software may be used by many operating systems. The diagrams below will present a visual representing the interface during various stages of navigating the software.

4.3.1.2 Software Interfaces

* + - Due to the software needing to communicate with various foreign markets it will interface with a case management system to pull and push data updates. Utilizing an API such as JDBC or ODBC there will be a traditional database connection.

4.3.1.3 Communication Interfaces

* There will be communication between the application and various foreign databases. While the web portal consists of operations concerning both reading and modifying the data, the communication between database and the mobile application will consists of only reading operations. Again, the connection will consist of utilizing an API such as JDBC or ODBC using a traditional database connection.

**4.4 Constraints and Assumptions**

4.4.1 List of Assumptions

4.1.1.1 It can be assumed that the website API whether JDBC or ODBC shall uphold a sufficient speed and stability to support the structure of the software. The transactions and request through users, delivery agents, and admins shall not put the API at risk of going down resulting in the entire system crashing. Constant modifications to the software shall not be required if the API maintains stability.

4.4.2 List of Dependencies

4.4.2.1 The software system will be dependent on one or multiple case management systems to push and pull data. Such as, if the transaction data is not appropriate for the case management system, or the transaction is declined due to insufficient funds. Another option is for a separate service to be created that allows for the pulling of transaction data.

**5 Object Descriptions**

## 5.1 Objects

## 5.1.1

|  |  |
| --- | --- |
| **Class name: Create Account** | |
| **Brief description: Class in charge of assigning and setting up all the details a user account will need to operate in our service. Includes setting up commonalities such as a username, password, assigns a email address. Should also be able to check a database for too similar usernames and password and duplicate emails.** | |
| **Attributes (fields)** | **Attribute Description** |
| setEmail() | Declaration where the new created account sets this as its users email address. |
| **Program Description Language** |
| public void setEmail(string newEmail) {  string setEmail();  } |
| setUsername() | **Attribute Description** |
| Declaration where the new created account sets this as its users Username. |
| **Program Description Language** |
| public void setUsername(string newUsername) {  string setUsername();  } |
| setPassword() | **Attribute Description** |
| Declaration where the new created account sets this as its users password. |
| **Program Description Language** |
| public void setPassword(string newPassword) {  string setPassword();  } |
| getUsername() | **Attribute Description** |
| Declaration where the function gets the username inputted for creation to be used in the cross validation of the users account information. |
| **Program Description Language** |
| public void getUsername(string getUsername) {  string getUsername();  } |
| crossCheckUser() | **Attribute Description** |
| Declaration where the accounts inputed information is cross checked with already created accounts to see if any information clashes. |
| **Program Description Language** |
| public void crossCheckUser( getUsername()){  NewUser = getUsernaem()  if NewUser == UserNameInDatabase  print(“Username is taken”)  } |

5.2.1

|  |  |
| --- | --- |
| **Class name: Security Check** | |
| When a user logs into the app or from the website for the first time on a new device the user will get a email with a security code that is required to log into the application. This class will be responsible for emailing the security code and checking the entered code with the one that the class emailed. | |
| **Attributes (fields)** | **Attribute Description** |
| getEmail() | The method is responsible for checking the database of current users and getting that users email. |
| **Program Description Language** |
| private void getEmail(database userEmails) {  string getEmail();  } |
| sendSecCode() | **Attribute Description** |
| The method is responsible for using the email gotten from the method above and using it so send the email a security validation code. |
| **Program Description Language** |
| private void sendSecCode(database userEmails) {  string SecurtiyCode;  sendToUserEmail();  } |
| validateSecCode() | **Attribute Description** |
| This method is used to check the security code sent against the one the user entered. If the code is wrong, it lets the user know and sends out a new code for the user to try again. Allows only for a maximum of 3 incorrect inputs before users account is locked for a short duration. |
| **Program Description Language** |
| private void sendSecCode(database userEmails) {  string enteredCode;  if enteredCode == SecurityCode  allowLogIn()  Else  Print(“Incorrect security code please try again”)  resendSecurtiyCode()  restartIfStatement  } |
| allowLogIn() | **Attribute Description** |
| This method once it receive conformation from the security codes validation method proceeds to log the user completely into the app. |
| **Program Description Language** |
| private void allowLogIn(){  createUserInterface()  InputUserAccountInfo()  } |

5.3.1

|  |  |
| --- | --- |
| **Class name: Total Balance/Current Value of Investments** | |
| Brief description: This class will solely be responsible for keeping the information of the user’s balance. Will be able to quickly determine the current value of the separate amounts of currencies the user owns through the app and quickly calculate the total worth of them all added together based on current market values. | |
| **Attributes (fields)** | **Attribute Description** |
| getForiegnCurrency() | Method will be responsible for getting information from the Foreign and local currency class to use in calculations. |
| **Program Description Language** |
| Public getForiegnCurrency( foriegnCurrencyInfo){  getForiegnCurrency()  getLocalCurrency()  } |
| valueBasedOnLocalCurrency() | **Attribute Description** |
| Method will calculate the values of the users currently owned currencies based on the specified local currency of the user. Example $1 US = $2 Canada |
| **Program Description Language** |
| Public getForiegnCurrency( ){  X = getForiegnCurrency()  Y = getLocalCurrency()  Value = X/Y  } |
| calculateTotalValue() | **Attribute Description** |
| Method will calculate the total values of all the currencies based on local values, the total number of currencies owned, value of those currencies. All this information will be added together to find the total value of the users current investments. |
| **Program Description Language** |
| Private calculateTotalValue(getTotalNumberofCurrencies, valueBasedOnLoaclCurrency){  TotalValue = (TotalNumber \* valueBasedOnLocal)  } |
| getTotalNumberOfCurrencies() | **Attribute Description** |
| Method will have subclasses for all the various currencies and find out the total number of currencies the user has. Pass that information along to other methods which need it. |
| **Program Description Language** |
| Public getTotalNumberOfCurrencies( userCurrencyInfo){  getTotalNumberOfCurrencies()  } |

5.4.1

|  |  |
| --- | --- |
| **Class name: Create User Interface** | |
| **Brief description:** Brief description: This class is responsible for creating the design and setting up the interface the user will interact with when the user interacts with. Should include be designed to be streamlined easy to maintain and user friendly | |
| **Attributes (fields)** | **Attribute Description** |
| userInterface | This is a generic for attributes used to represent user interface components to be displayed and used in the user interface. |
| dataForInterface | This is a generic for the data holding attributes that will need to be created based on the set of elements to be edited. |
| **Methods (operations)** | **Method Description** |
| getUIInterfaceComponent() | This is a generic for the getters used for user interface. |
| setUIInterfaceComponent() | This is a generic for the setters used for user interface. |
| getDataComponent() | This is a generic for the getters used for data used with the interface for documents. |
| setDataComponent() | This is a generic for the setters used for data used with the interface for documents. |
| validate(data, validationType) | An overloaded method set which is used to validate modifications to a data set based on the type of element being edited. |

5.5.1

|  |  |
| --- | --- |
| **Class name: Calculate Profit Margins** | |
| Brief description: This class will be able to gather the information from the current investment total class and get information of what the information was in the past, for example on the first day the user purchased said currency and be able to calculate the profit/loss margin of the users investment. | |
| **Attributes (fields)** | **Attribute Description** |
| getTotalBalance() | This method gets the total balance from the Total Balance class. |
| **Program Description Language** |
| private void getTotalBalance(totalBalance) {  float getTotalBalance();  } |
| getFirstDayBalance() | **Attribute Description** |
| This method will be used every time a purchase is made by the user it will store the currencies value on that day and how much the user bought. |
| **Program Description Language** |
| private void getFirstDayBlance(currencyInformation) {  float getTotalBalance();  } |
| getYesterdaysBalance() | **Attribute Description** |
| This method will store Today’s currency values and balance and yesterday’s values for the same. Once the day ends Todays will become yesterdays and a new today’s values will be calculated. |
| **Program Description Language** |
| private void getFirstDayBlance(currencyInformation) {  float getTodaysBalance();  float getYesterdaysBalance();  } |
| calculateForProfit() | **Attribute Description** |
| This method calculates profits based on yesterday, first day, and todays changes in the currencies value. |
| **Program Description Language** |
| private void calcualteForProfit(currencyInformation, totalBalance) {  ProfitTodayVsYesterday()  ProfitTodayVsFirstDay()  ProfitTodayVsBasedOnTodaysChanges()  } |

5.6.1

**6 Object Collaboration**

**6.1 Object Collaboration Diagram**

