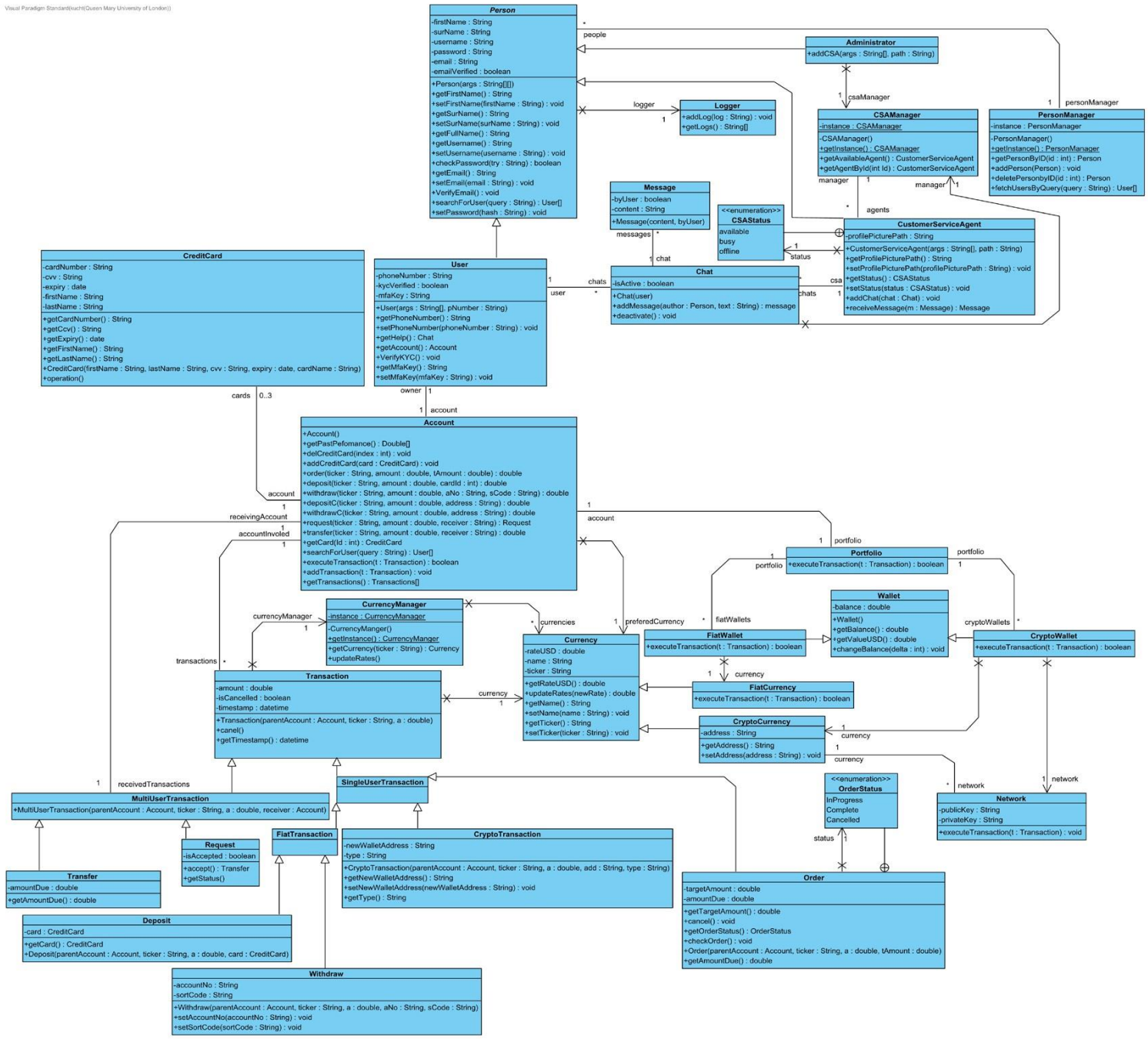
CryptoGram –

Faris Javaid, Group 15

Design Report

Contents

1. Class Diagram



2. Traceability Matrix

Class	Requirement ID	Explanation
Wallet	1	The balance for each crypto asset is held in the Wallet class. Each account is made up of multiple wallets.
Person	3	Each user is a person; Person stores the username of a user
CustomerServiceAgent	6	CustomerServiceAgent stores the file path of a picture
CreditCard	16	Each account can have up to 3 credit cards
Transaction	18	Each transaction contains a timestamp
User	19	User contains a multi-factor authentication key, used by google authentication
CustomerServiceAgent	25	CustomerServiceAgent stores the status as either ‘available’, ‘busy’, or ‘offline’
Logger	28	Logger contains and creates logs
Person	29	Person stores the first name (firstName) and surname (surName) of every user

User	31	User stores the phone number of a user
Person	32	Person stores a username for every user

3. Design Discussion

Changes of entities and relations

- The Wallet class now holds a single reference to a Currency class rather than many like in the Domain Model.
- The administrator entity was added to the system and is now represented as a class.
- Each message is now an instance of the Message class.
- CreditCard is now also stored as an object.

Design Patterns

- In order to keep track of all instances of Person class, PersonManager will hold an array list of all Person objects. To ensure that there is a single PersonManager object with the array list, the **Singelton design pattern** is used. The Singelton design pattern will guarantee that creation of multiple objects of PersonManager class is not possible.
- In order to keep track of all instances of CustomerServiceAgent class, CSAManager will hold an array list of all CustomerServiceAgent objects. To ensure that there is a single CSAManager object with the array list, the **Singelton design pattern** is used. The Singelton design pattern will guarantee that creation of multiple objects of CSAManager class is not possible.
- In order to keep track of all instances of Currency class, CurrencyManager will hold an array list of all Currency objects. To ensure that there is a single CurrencyManager object with the array list, the **Singelton design pattern** is used. The Singelton design pattern will guarantee that creation of multiple objects of CurrencyManager class is not possible.
- In order to minimise mistakes in code and encapsulate updates of balances of fiat and crypto wallets during the execution of transactions, Wallets class was created (**Facade design pattern**). When a transaction has to be executed it will be passed to executeTransaction(t : Transaction) in Wallets that will do all verifications and execute the transaction in the corresponding FiatWallet and CryptoWallet objects in the right order. Additionally, decoupling is achieved as crypto and non-crypto transactions are split and the correct wallet is called accordingly.

Other Design Decisions

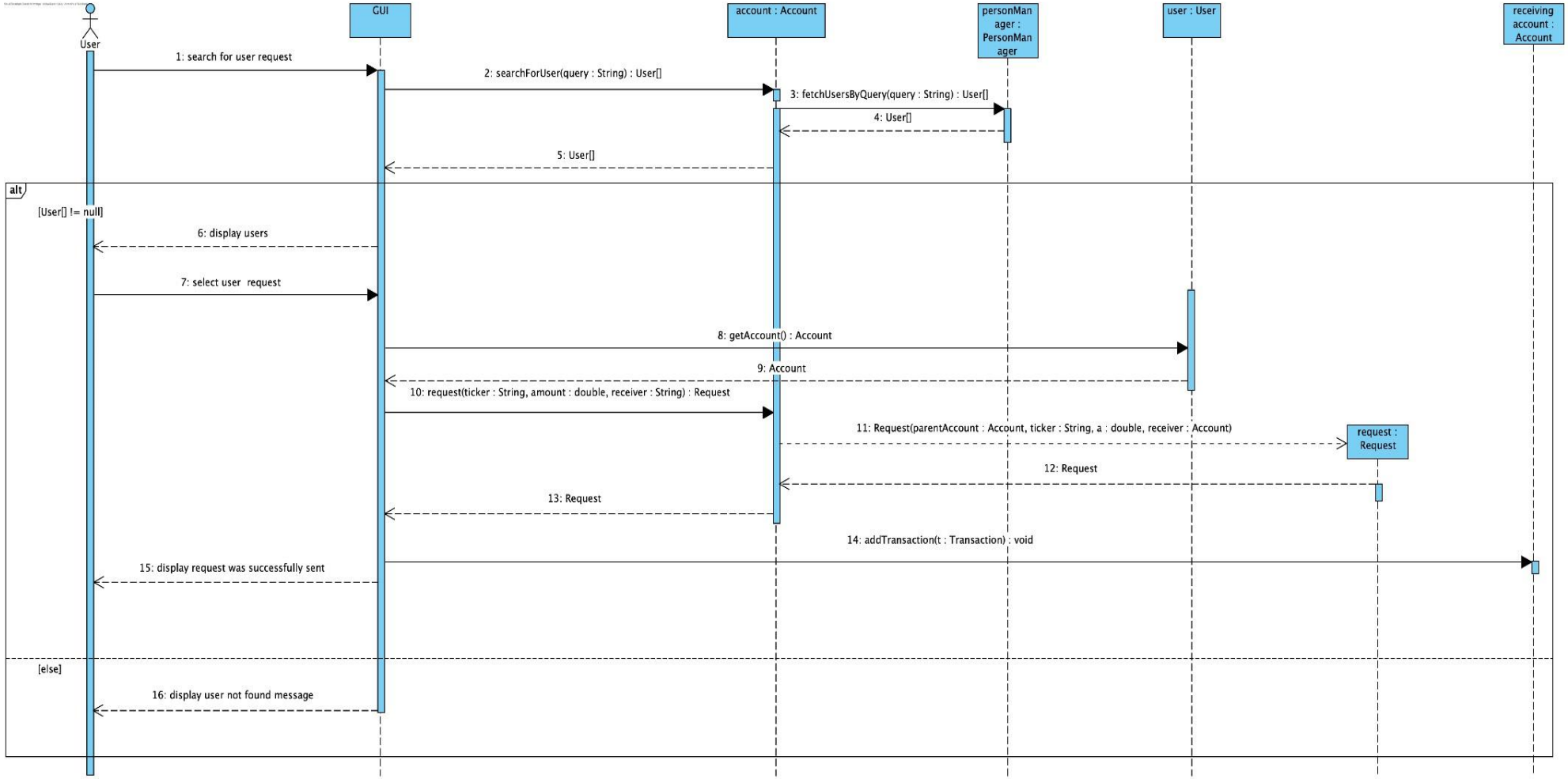
- Since the platform will be used in multiple countries with different fiat currencies the system will fetch the price of each crypto asset in every one of those fiat currencies. In order to use memory efficiently, all Currency objects

have an instance variable rateUSD that can be used to calculate the price of any asset in any fiat currency. (e.g [Price Of BTC in £] = [Price of BTC in $] ÷ [Price of £ in $]) Note: since FiatCurrency is a subclass of Currency it inherits rateUSD needed for the calculation above.

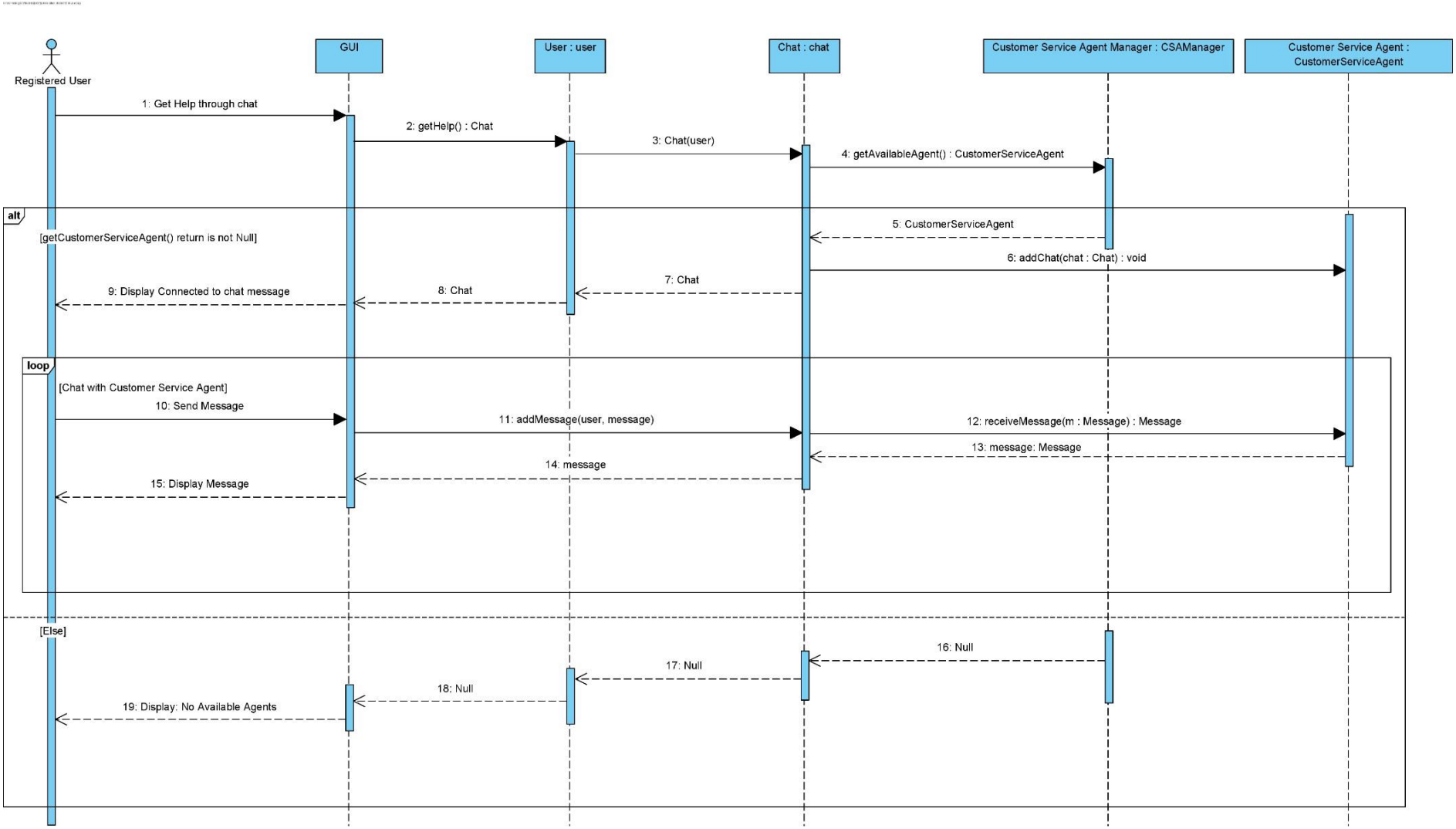
- Order class has an instance variable status that is of type OrderStatus enumeration. Since the order status can only be one of InProgress, Complete and Cancelled; boolean cannot be used here. Alternatively, a string could be used instead but incorrect capitalisation or spelling may lead to unexpected behaviour of the software.
- CustomerServiceAgent class has an instance variable status that is of type CSAStatus enumeration. Since the status can only be one of available, busy and offline; boolean cannot be used here. Alternatively, a string could be used instead but incorrect capitalisation or spelling may lead to unexpected behaviour of the software.
- The person class has an instance variable ‘password’ that represents the hash of the password of the user. Additionally, for extra security, the instance variable ‘password’ has no getters. Note that these security measures don’t affect the functionality as checkPassword can still assess the value of “password” for verification.
- All transactions are objects that inherit from one of the two classes: SingleUserTransaction and MultiUserTransaction, which are child classes of Transaction. This eliminates the need of copying code and decreases the number of possible bugs. The use of many small classes that inherit from each other rather than one big class also provides the ability for the Wallet class to match different behaviour of executeTransaction based on the transaction type passed to it.

4. Sequence Diagrams

a) Make a request



b) Chat with Customer Service Agent



c) *Purchase CryptoCurrency*