Mobile Trading System

A Case Study in Software Design

By:

Group - 13

Samay Sharma, Rashika Kheria,

Raghav Katyal, Nakka Mithun, Naveen Panwar

MOBILE TRADING SYSTEM

- Develop an integrated solution designed specifically for the Stock Exchange Board.
- Address an important and compelling need of investors on move.
- Making timely investment and trading decisions using their handheld devices such as PDAs, and smart phones.
- The application had to provide comprehensive market information, analytical tools and recommendations.

High Level Decisions

- Usage of **Waterfall model** as the SDLC model :-

Terrain was pretty much known and requirements were pretty clear.

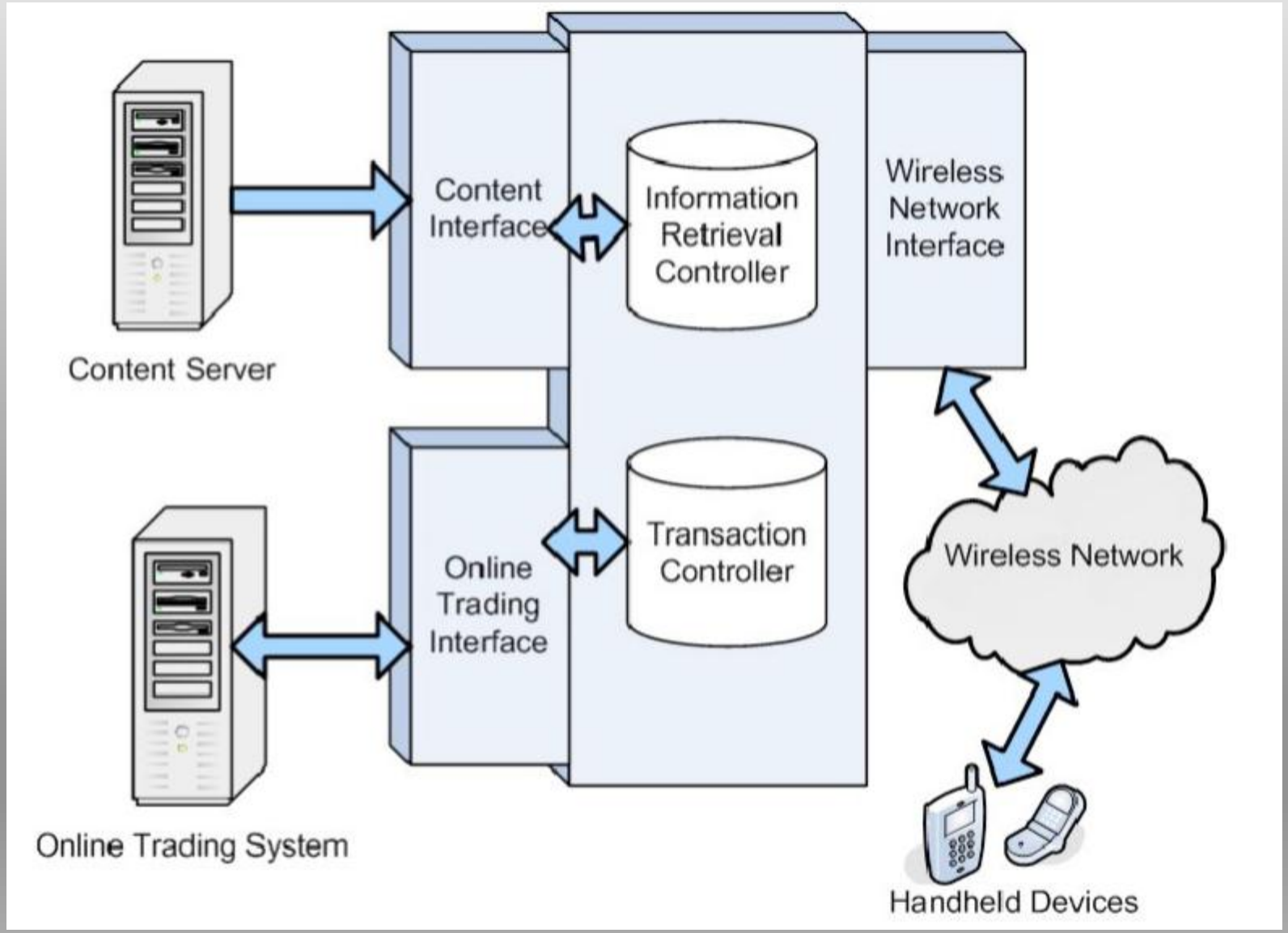
- **Use Case model** for requirements engineering:-

  Obvious choice as it is the most widely used method for requirement engineering.

- **Design Methodology** :-

  Object Oriented analysis and design methodology for a strong and extensible design..

BLOCK DIAGRAM OF MOBILE TRADING SYSTEM



Design Goals

- ***Security***
  Need to instill confidence in all stakeholders that confidentiality of private information is adequately protected, transactions are legally binding and cannot be altered.
- ***Extensibility and Scalability***
  New WAP enabled and wireless gadgets keep flooding the market periodically.
- ***Algorithmic Efficiency***
  The data should be processed with little processing power and memory.
- ***Retrieval Efficiency***
  The data should be processed in less time and help the user in quick decision making.

Architectural Decisions - GCM

- Usage of GCM server.
- It allows 3rd-party application servers to send messages to their Android applications.
- lightweight message telling the Android application that there is new data to be fetched from the server
- Handles all aspects of queuing of messages and delivery to the target application.
- Android application on an Android device doesn't need to be running to receive messages. System will wake it up and hence makes it power efficient.

Architectural Decisions - Security

➢ ***Using Encryption mechanism*** – WPA2 for encryption of data and communication between the user device and the server.

Advantages of WPA2 :-

➢Current standard for Wifi-security

➢WPA2 uses AES (Advanced Encryption Standard) to provide strong encryption.

➢WPA2 uses a Temporary Key Integrity Protocol (TKIP), which dynamically changes the key as data packets are sent across the network.

➢Since the key is constantly changing, it makes cracking the key more difficult.

Architectural Decisions - Extensibility and Scalability

- Lot of types of mobile devices available in the market.
- Important to have a cross platform solution.
- Use open source cross platform development solution like RhoMobile.

Advantages of using RhoMobile

- Opensource
- Development languages are HTML/CSS and ruby which are very commonly used.
- Has support for all major mobile OS.
- Apps take full advantage of the Hardware

Architectural Decisions – Retrieval Efficiency

- Usage of cache for retrieval efficiency
- Using custom made cache replacement policy to suit our needs.
- Cache policy includes two factors user preferences and recently used stocks.
- ½ of the cache is specifically filled with stocks which are in user preferences and remaining on the basis of time.
- Usage of such a cache policy drastically reduces the number of times the server has to fetch data from the content server (which is a very slow operation as content server is to be accessed by the network)

Data Analysis Algorithm

- Using artificial neural networks.
- Use a feed forward network utilizing the backward propagation of errors algorithm to update the network weights.
- Large dataset available for balancing and updating the network weights.
- May also use genetic algorithms for estimation but quick algorithms and less power consumption is required.

Design Principles

- ***Single responsibility design principle*** :-

A class should have only a single responsibility. If we have to make a change to the class, it should not change the interfaces which it provides to the other classes. If a functionality is to be added, it should change only one class.

- ***Modularity and maintainability*** :-

The code should be modular so that it can easily be modified in the future. Also, it increases the maintainability of the code in the future. Only particular job is to be done by a module and system components can be easily interchanged.

Use Case Description

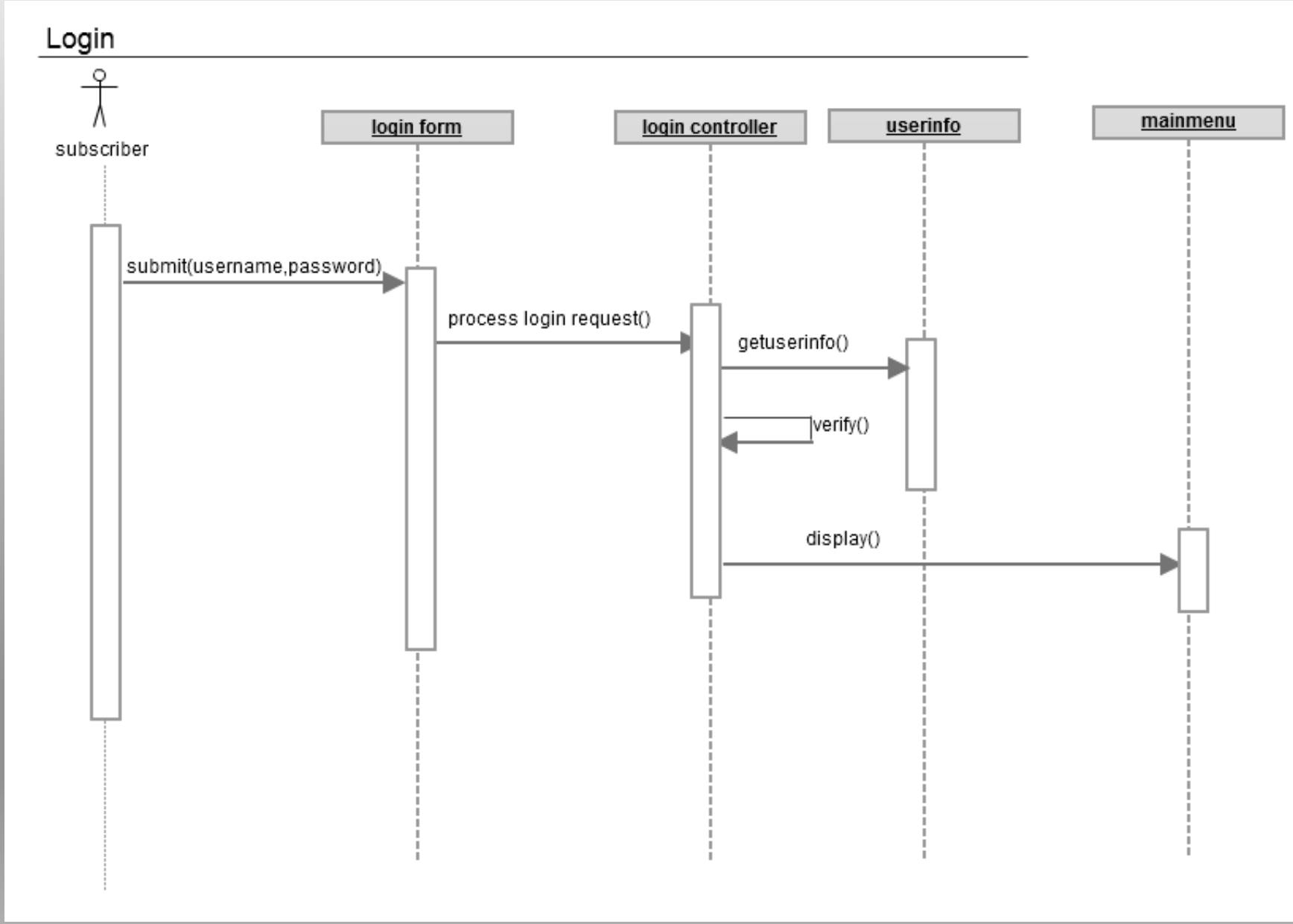
Use Case ID: UC1	Use Case Name: Login
Description:	This use case allows the subscriber to log into the system and use the system after verification. The subscriber cannot initiate any use case till he has not successfully completed this use case.
Actor:	*Subscriber*
Preconditions:	NA
Postconditions:	NA
Frequency of Use:	Whenever Subscriber starts to use the system.

Basic Flow:	When the subscriber initiates the client software installed on his hand-held device, the use case starts. The normal course of events are following: 1. System requests login credentials. 2. The subscriber enters the username and password and clicks submit. 3. System verifies the subscriber details. 4. The main menu is displayed at the subscriber's device.
Alternate Flow:	On step 4, if the authentication server finds the information sent by the subscriber is not a valid one, it performs the following: 1. The system sends a message that the subscriber cannot be logged into the system. 2. The message is displayed on the subscriber's hand-held device along with the login screen.

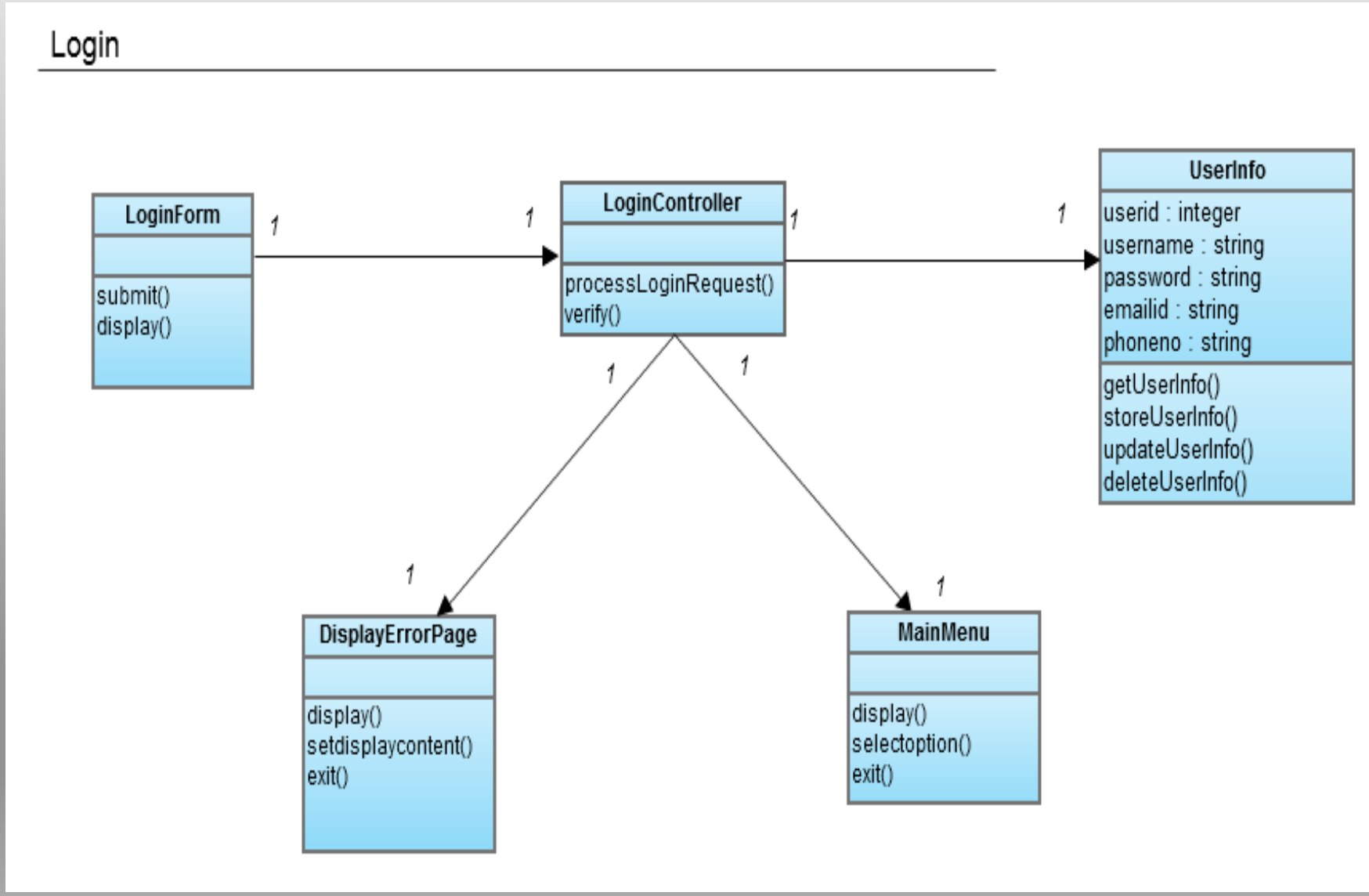
Supplement Use Case

Use Case ID: UC1	Use Case Name: Login
Basic Flow:	1) User opens the application on the PDA and the system displays the login screen to the user. 2) User enters the username and password on the login screen and clicks on the submit button. 3) These details are passed to the login controller which checks if the username belongs to a valid subscribed user. 4) If it does, then the login controller sends a request to the database to retrieve the password for the corresponding user name. 5) The database processes the request and returns the password for the mentioned username to the login controller. 6) Login Controller checks if the passwords match. If they do, then the user is shown the main screen.
Alternate Flows:	**AF 1**: In 3), is username is not valid, Login Controller displays an error page stating that the username is invalid. **AF 2**: In 6), if the passwords do not match, an error page is displayed stating that the username and passwords do not match.

Sequence Diagram



Class Diagram



Use Case Description

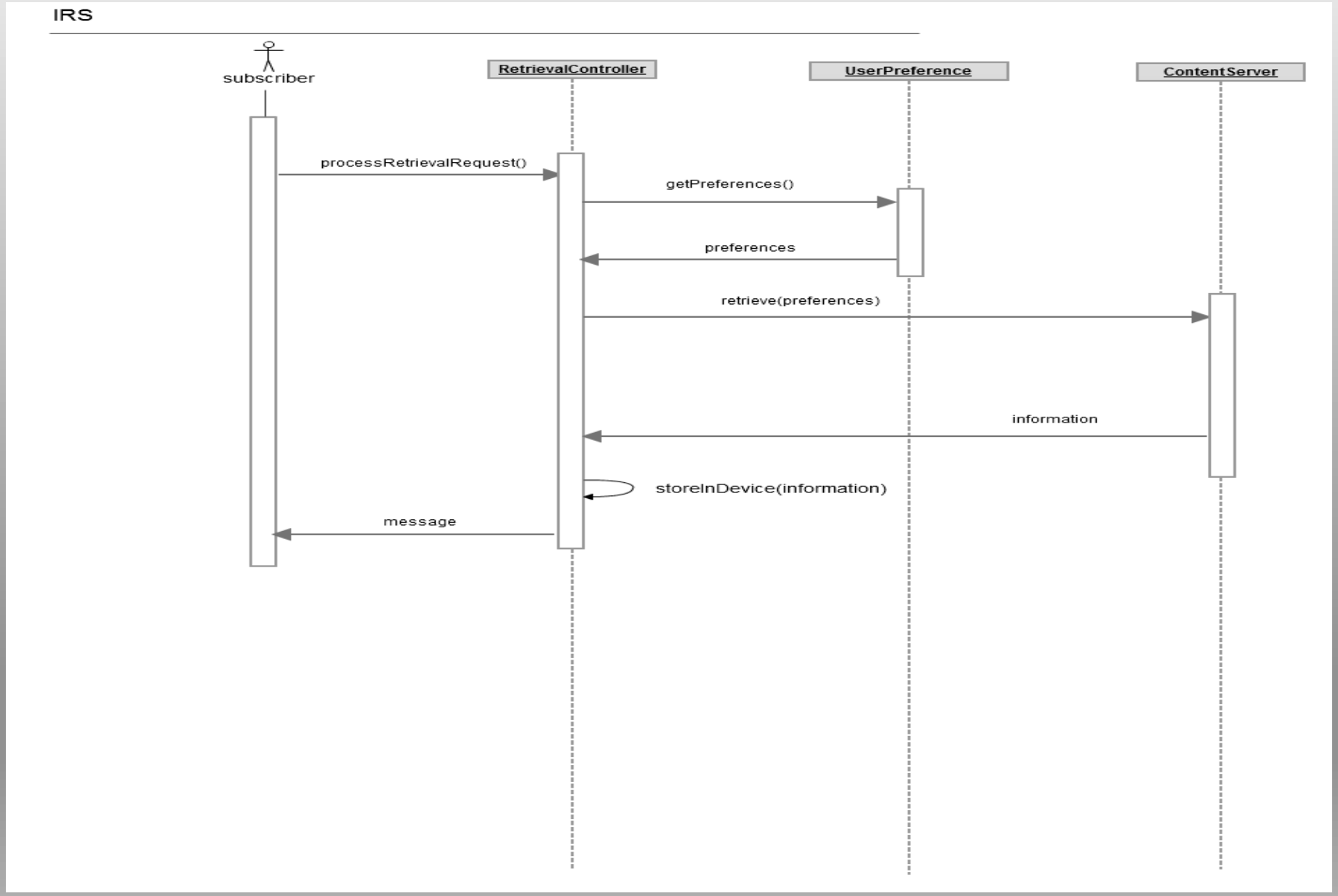
Use Case ID: UC2	Use Case Name: Retrieve Information and Store
Description:	This use case retrieves the financial locally on the PDA. This use case is a continuous one and starts at the time of the startup of the system and continues till the system is terminated or an interrupt is used to terminate the process.
Actor:	*Subscriber*
Preconditions:	User must be logged in; sufficient memory available in the device to store contents.
Postconditions:	NA
Frequency of Use:	High

Basic Flow:	The system initiates the use case. The normal course of events for this use case are following: 1. The system initiates the retrieval process when the user logs on. 2. System checks user preferences. 3. System receives data from content server in specified format according to user preferences/portfolio. 4. IRC stores the data in hand-held device in suitable format
Alternate Flow:	On step 1, user can initiate the process by choosing to refresh the content.
Exceptions:	If the system fails to execute the use case for any reason, it should pop-up a message stating the same. In that case, the use case should be initiated again.

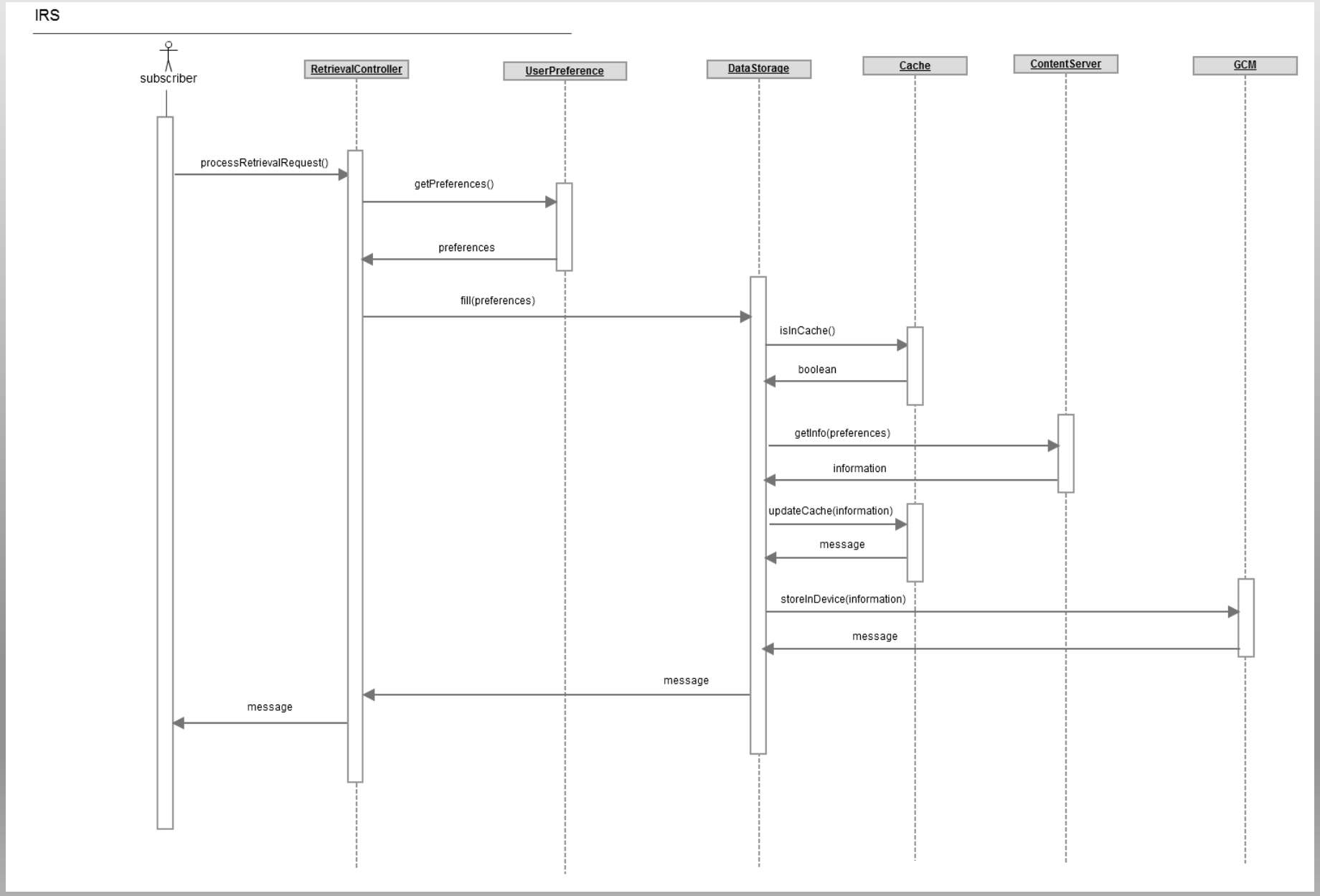
Supplement Use Case

Use Case ID: UC2	Use Case Name: Retrieve Information and Store
Basic Flow:	1) Retrieval process is initiated when the user logs in using the PDA. 2) IR Controller gets the user preferences from the database. 3) IR Controller requests the Data-storage for the Data on the basis of given preferences. 4) Data-storage checks whether the desired information is in the cache. 5) If it is not there, the Data-Storage requests the content server for the desired information. 6) The data is then updated in the Data-Storage. 7) Then, a GCM message is sent and the corresponding information is stored on the mobile.
Alternate Flows:	**AF1:** In 1), if the user is unable to login then an error page is displayed. **AF2:** In 3), if the connection is not established, an error page is displayed stating "Unable to connect to Data-Storage". **AF3:** In 7), if GSM server is unable to send the message, an error is issued.

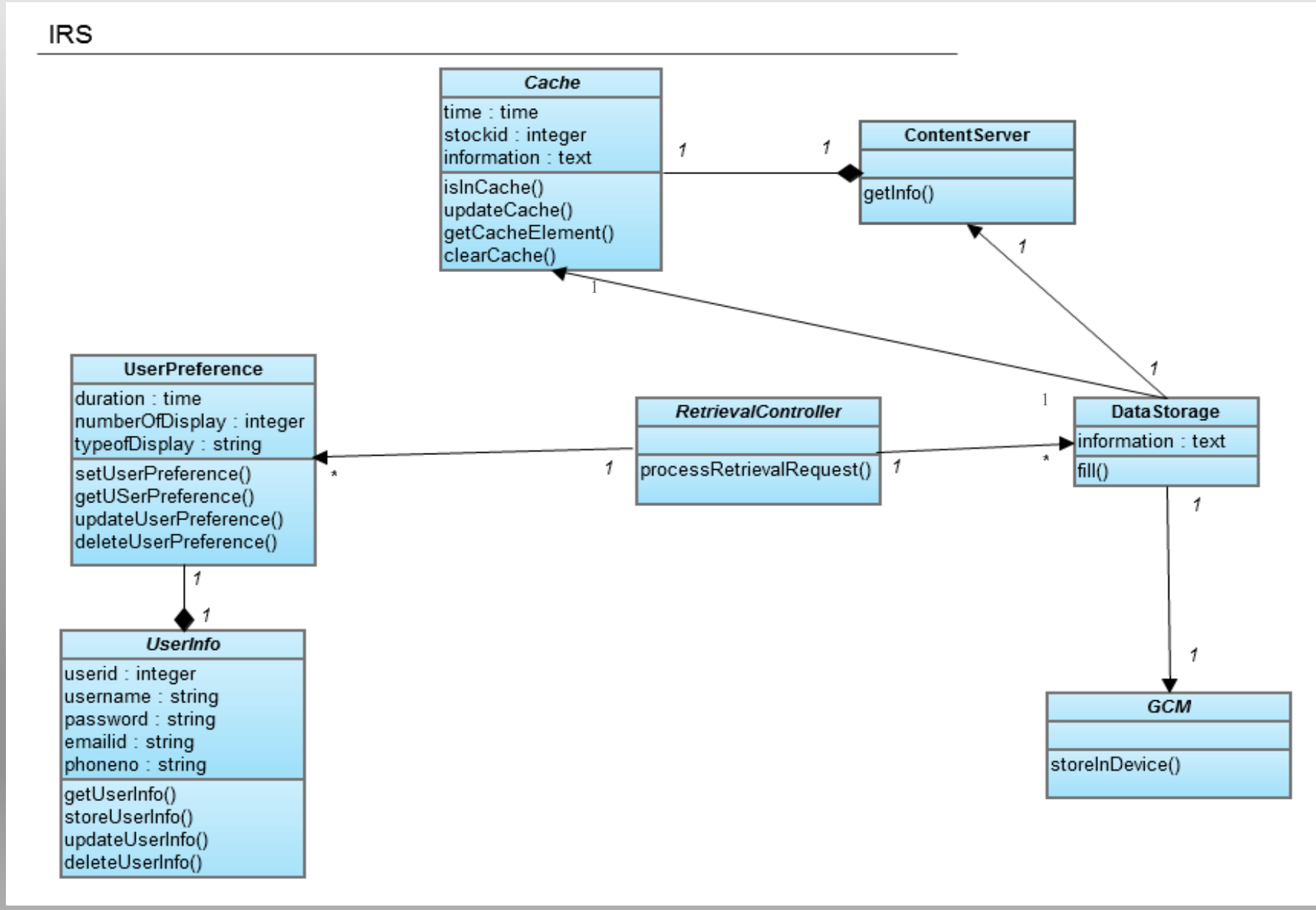
Sequence Diagram



Design Level Sequence Diagram



Class Diagram



Use Case Description

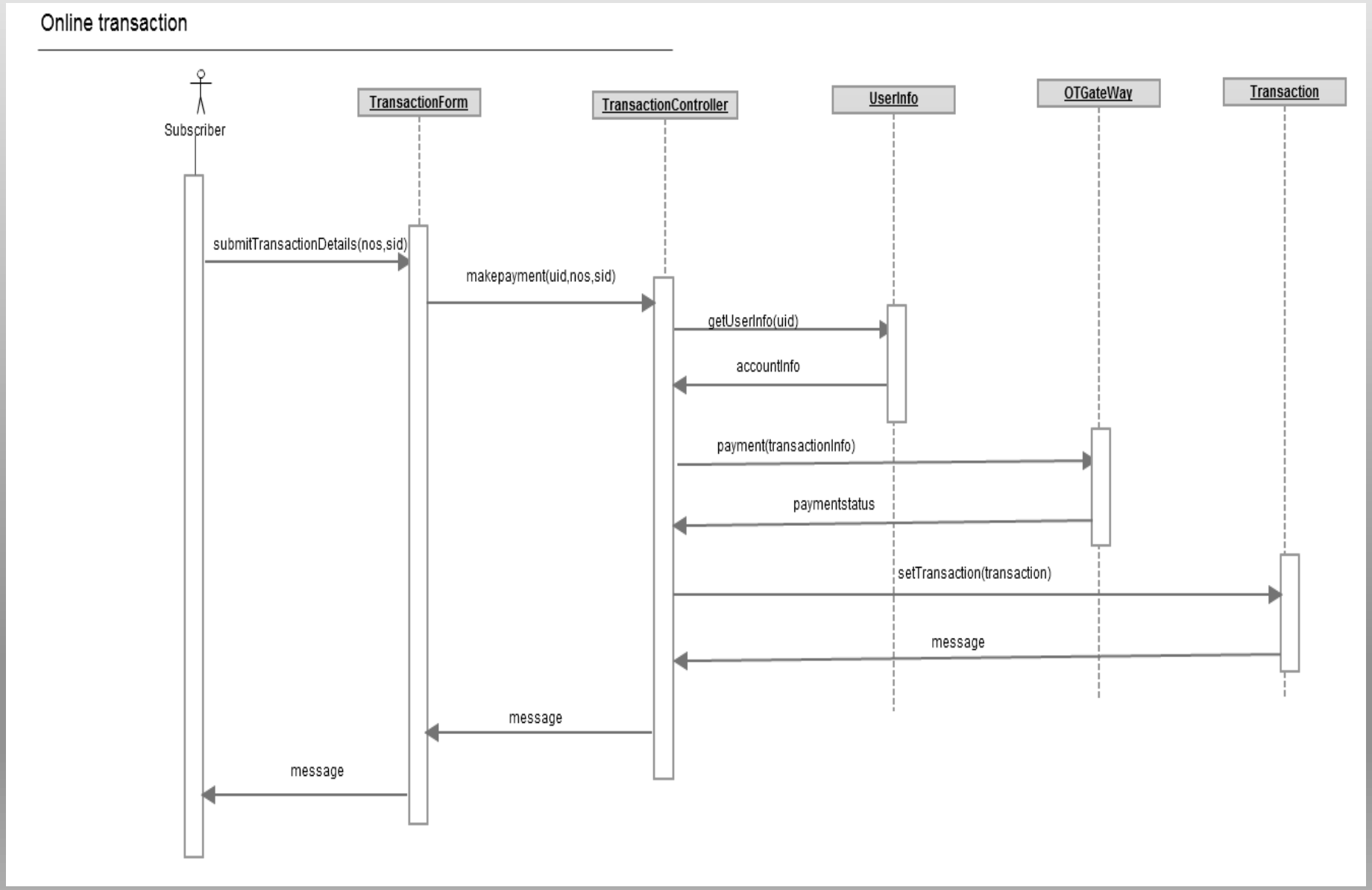
Use Case ID: UC3	Use Case Name: Online Transaction System
Description:	This use case describes the method of performing a transaction of stocks. It describes the purchase or sell of the stocks from the PDA device. The user cannot perform transaction till he initiates this use case.
Actor:	*Subscriber*
Preconditions:	User must be logged in
Postconditions:	NA
Frequency of Use:	Medium

Basic Flow:	The actor initiates the use case. The normal course of events for this use case are following: 1. The subscriber clicks on the buy/sell stock button after logging in. 2. Subscriber selects number of stocks and the type of transaction(buy/sell) and clicks the submit button. 3. Subscriber is redirected to OTS where he fills in the transaction details and clicks the submit button. 4. The subscriber is then displayed a message informing about the status of the transaction.
Alternate Flow:	In step 4, if the user receives a failed status message then he has to return back to step 1 in order to perform the transaction again.
Exceptions:	If the system fails to execute the use case for any reason, it should pop-up a message stating the same. In that case, the use case should be initiated again.

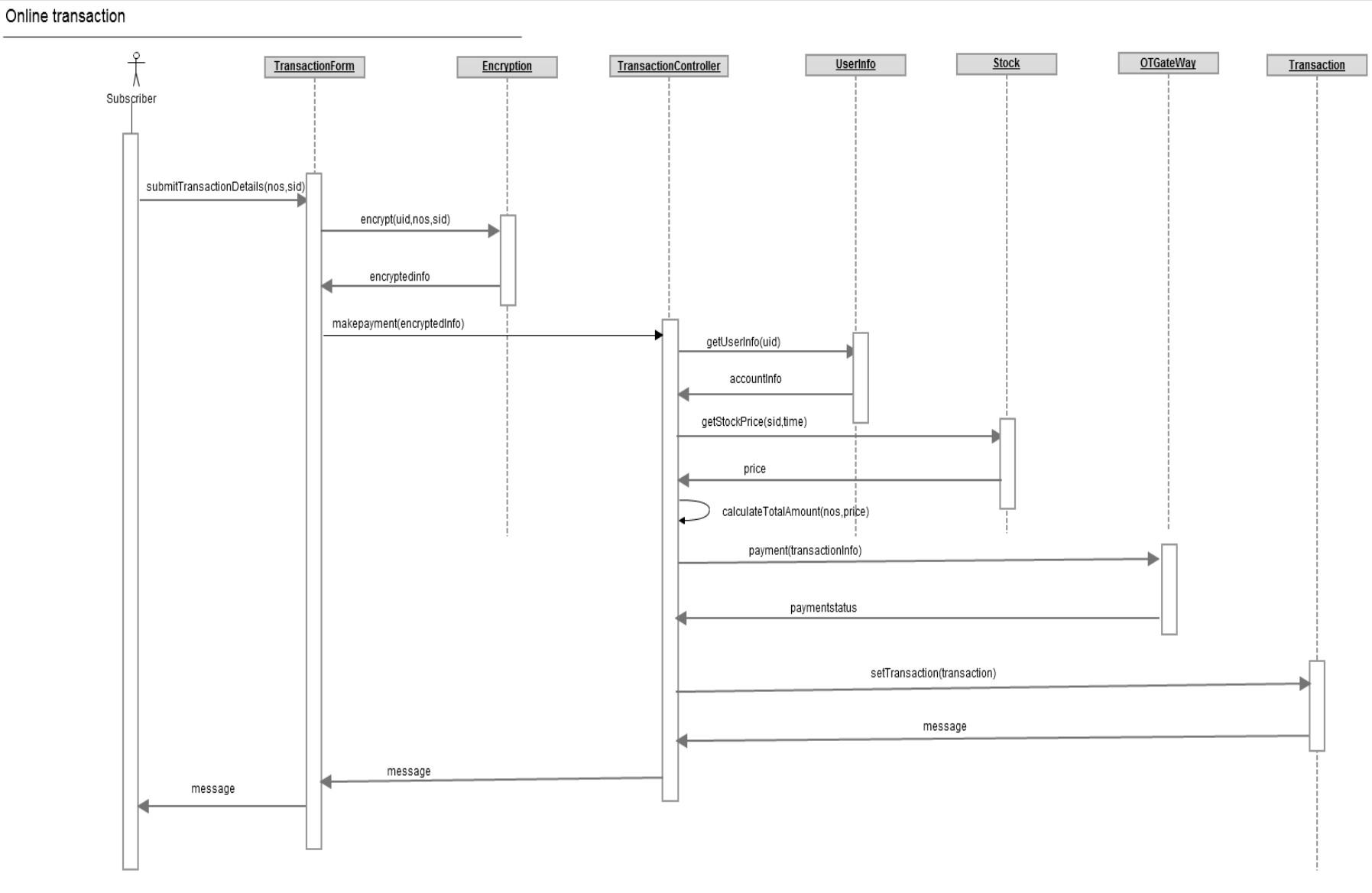
Supplement Use Case

Use Case ID: UC3	Use Case Name: Online Transaction System
Basic Flow:	1) The subscriber clicks on the buy/sell stock button. 2) Subscriber selects stock, number of stocks and the type of transaction(buy/sell) and clicks the submit button. 3) The transaction details are encrypted and sent to the transaction controller. 4) The transaction controller receives the data and calculates the current transaction cost by retrieving the current stock price. 5) Transaction controller then displays the final details of the transaction to the user. 6) The user clicks the confirm button. 7) The Transaction Controller then connects to the OTS. 8) OTS preforms the corresponding transaction and returns a response status message to the Transaction Controller. 9) The system then updates the database with the corresponding details and displays a screen to the user.
Alternate Flows:	**AF1:** In step 3, the transaction controller checks if that number of stocks are available, otherwise he is redirected to step 2. **AF2:** In step 5, if the user feels that he wants to change the details of the transaction, he is redirected to step 2 **AF3:** In step 7, if the system is unable to connect to the OTS, it displays error message.

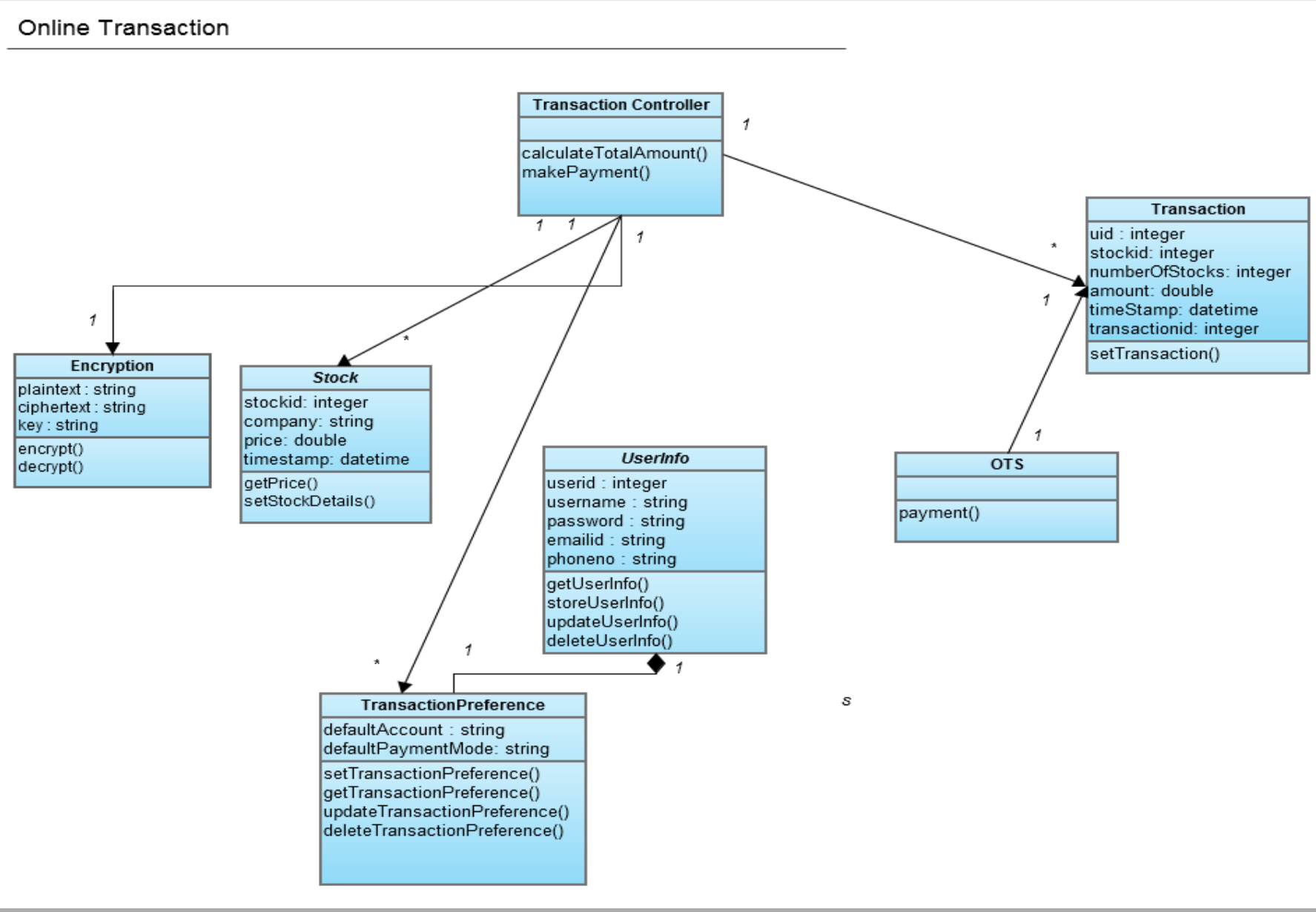
Sequence Diagram



Design Level Sequence Diagram

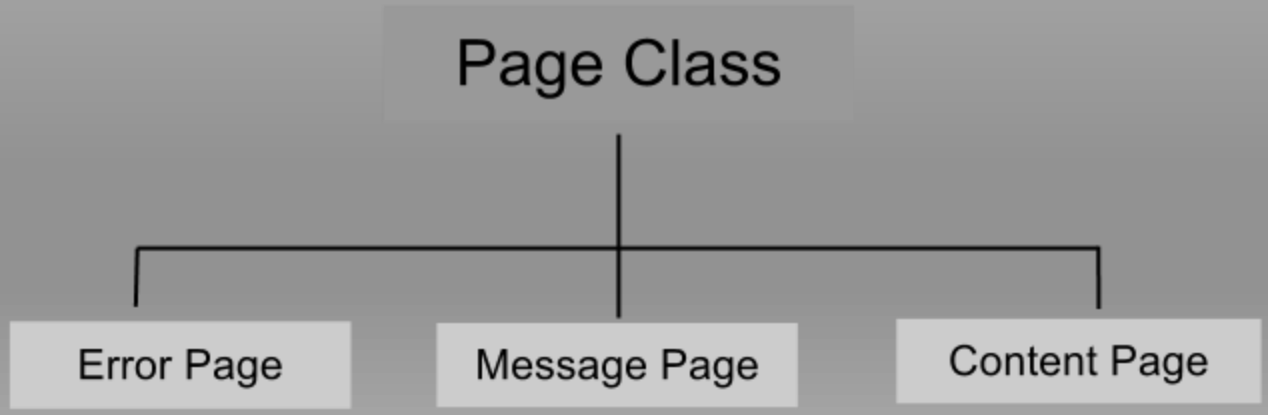


Class Diagram



Design Patterns – Abstract Factory

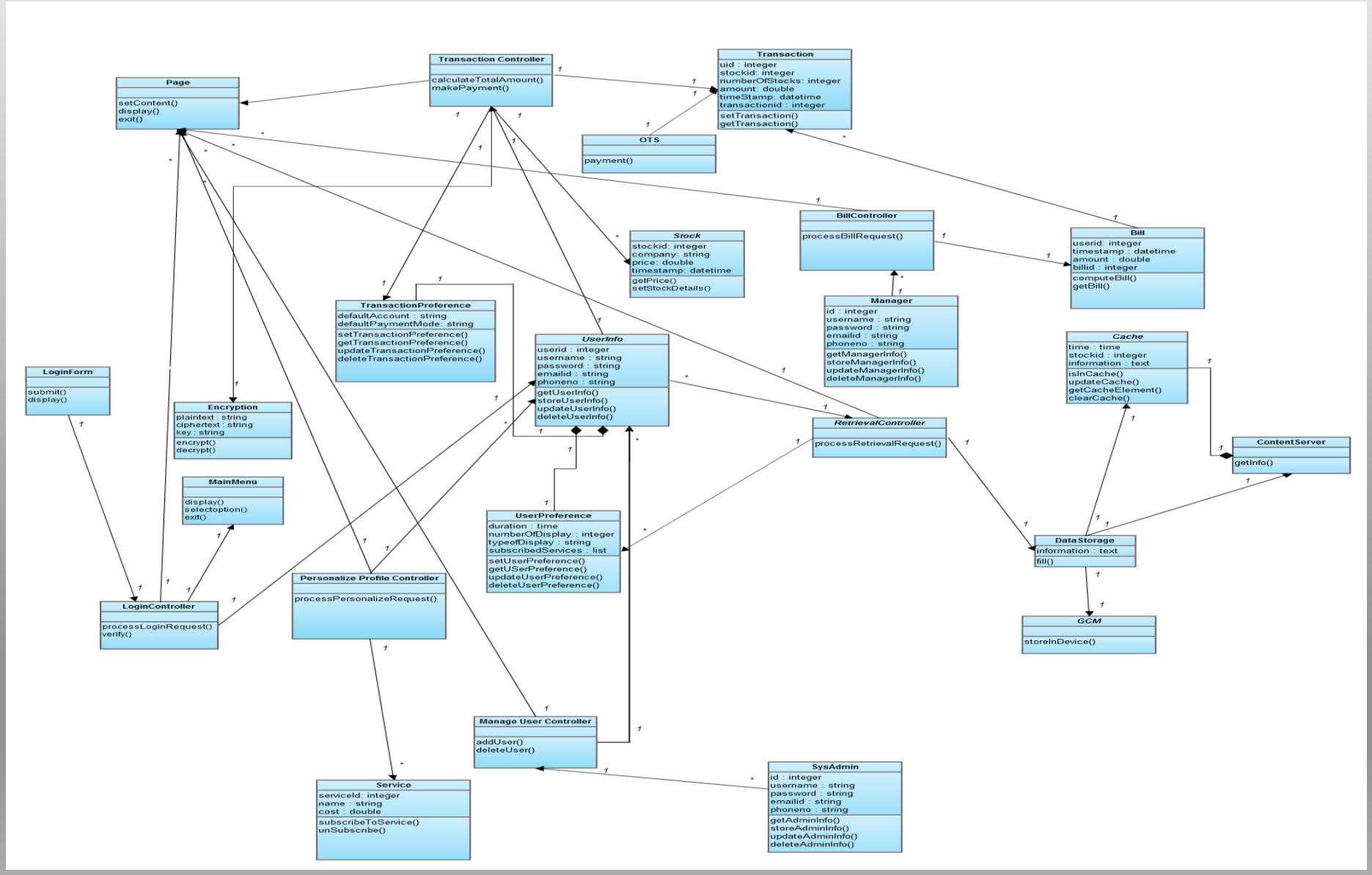
- Centralize decision of what factory to instantiate.
- One Factory which has multiple implementations on the basis of the current requirement.
- System decides which class type to instantiate by using a SetMethod() which indicates what type of Page is to be displayed.



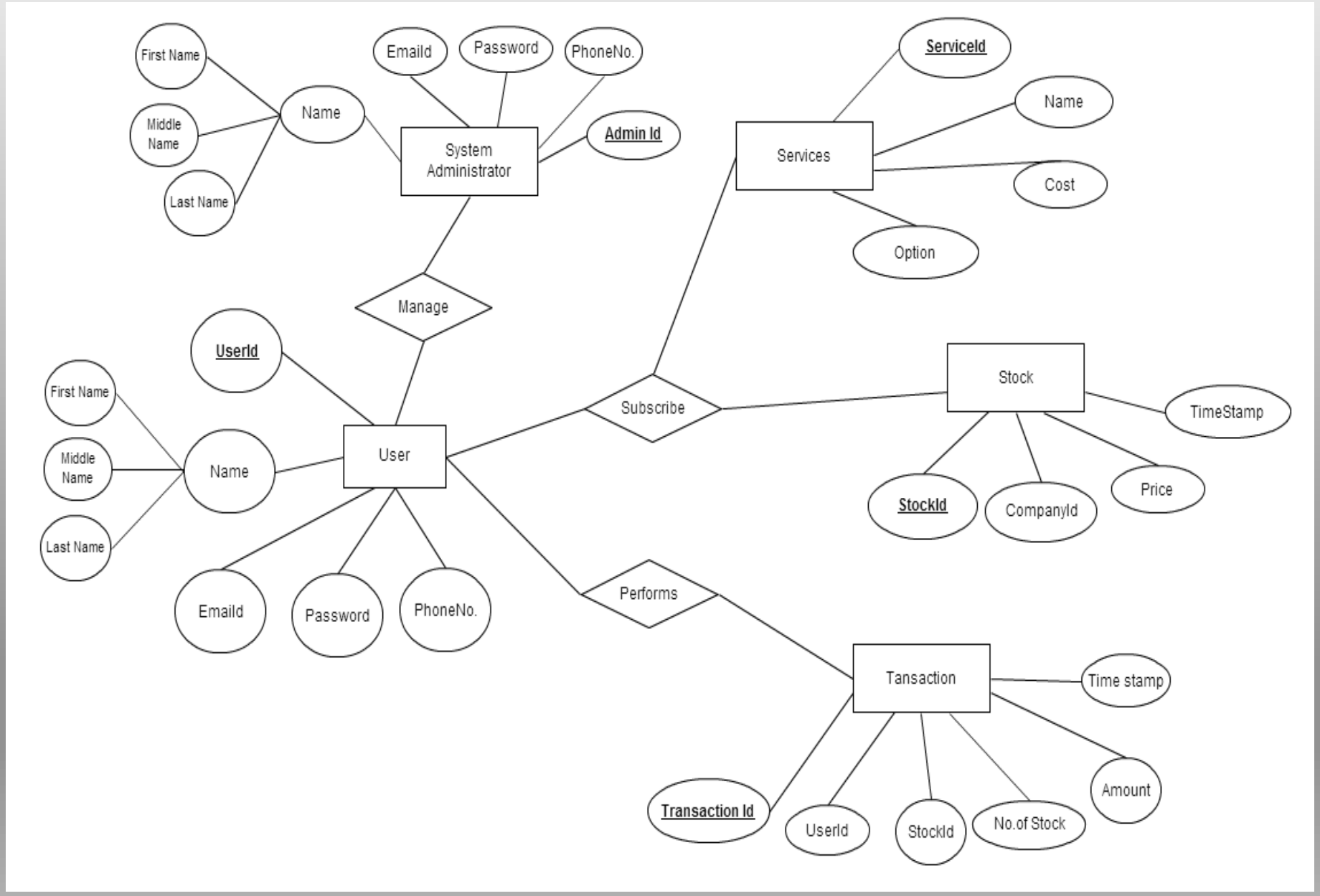
Design Patterns – Singleton

- Limit number of instances of a class to one.
- Provides a single global access point for all the classes.
- In our system, we have a single instance of the Cache for the whole system.

Final Class Diagram

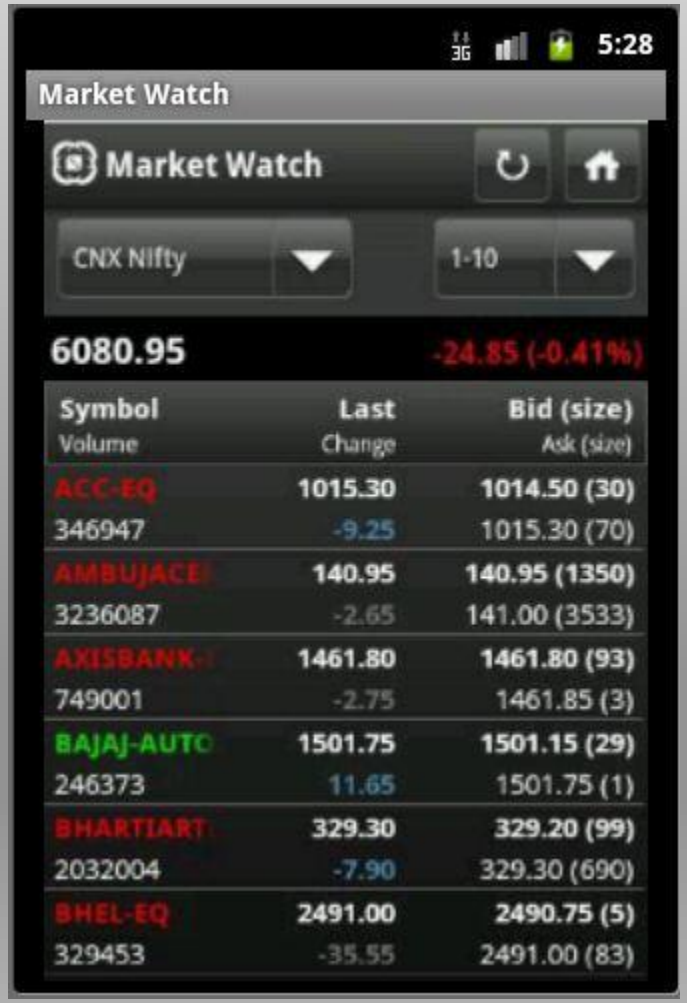


Logical Data Model



UI Design

Will be presented after review from Mr Ripul.



THANK YOU ☺

Questions ?