|  |
| --- |
| **School of Information Technology**  **Diploma in Information Security**  **IT2539**  **Applications Security Project**  **AY2012/2013 S2**  **MWN**  **Project Report v1.0.0**  MODULE GROUP: **DIS1102**  PROJECT TEAM: **4**  **Alvin Lee Yong Teck (LEADER)**  **Lam Yong Quan, Terence**  **Lim Kian Hock Bryan**  **Wong Zi Feng, Benjamin** |

Contents

[Document Control 5](#_Toc347663216)

[1. Introduction 5](#_Toc347663217)

[2. Project Proposal 5](#_Toc347663218)

[2.1 Background 5](#_Toc347663219)

[2.2 Business Case 6](#_Toc347663220)

[2.3 Business Functions 6](#_Toc347663221)

[2.4.1 Security Functions 9](#_Toc347663222)

[2.4.2 Secure Coding 10](#_Toc347663223)

[2.5 Architecture Diagram 12](#_Toc347663224)

[2.6 Resources 12](#_Toc347663225)

[3. Project Plan 13](#_Toc347663226)

[3.1 Team Organization 13](#_Toc347663227)

[3.2 Task Allocation 13](#_Toc347663228)

[3.3 Schedule 14](#_Toc347663229)

[3.4 Gantt Chart 16](#_Toc347663230)

[4. Requirements Specification 19](#_Toc347663231)

[4.1 Use Case Diagrams 19](#_Toc347663232)

[4.1.1 Use Case Diagram (Overall Summary) 19](#_Toc347663233)

[4.1.1.1 Use Case Diagram (Overall Summary) – Login System 20](#_Toc347663234)

[4.1.1.2 Use Case Diagram (Overall Summary) – BMI Calculator 20](#_Toc347663235)

[4.1.1.3 Use Case Diagram (Overall Summary) – RSS Feeds 21](#_Toc347663236)

[4.1.1.4 Use Case Diagram (Overall Summary) – Sharing System 21](#_Toc347663237)

[4.1.1.5 Use Case Diagram (Overall Summary) – Commenting System 22](#_Toc347663238)

[4.1.1.6 Use Case Diagram (Overall Summary) – In-House Mail System 22](#_Toc347663239)

[4.1.1.7 Use Case Diagram (Overall Summary) – Advertising System 23](#_Toc347663240)

[4.1.2 Use Case Diagram (All Use Cases) 24](#_Toc347663241)

[4.2 Misuse Case Diagrams 25](#_Toc347663242)

[4.3 Use Case Descriptions 25](#_Toc347663243)

[4.3.1 Alvin’s Use Case Descriptions 25](#_Toc347663244)

[4.3.2 Bryan’s Use Case Descriptions 31](#_Toc347663245)

[4.3.3 Benjamin’s Use Case Descriptions 34](#_Toc347663246)

[4.3.4 Terence’s Use Case Descriptions 35](#_Toc347663247)

[4.4 Domain Class Diagrams 37](#_Toc347663248)

[4.5 Security Requirements 37](#_Toc347663249)

[5. Design Specification 38](#_Toc347663250)

[5.1 Sequence Diagrams 38](#_Toc347663251)

[5.1.1.1 Alvin’s Sequence Diagrams - Login 38](#_Toc347663252)

[5.1.1.2 Alvin’s Sequence Diagram – Register Account 39](#_Toc347663253)

[5.1.1.3 Alvin’s Sequence Diagram – Retrieve Forgotten Username 40](#_Toc347663254)

[5.1.1.4 Alvin’s Sequence Diagram – Retrieve Forgotten Password 41](#_Toc347663255)

[5.1.1.5 Alvin’s Sequence Diagram – Verify Account 42](#_Toc347663256)

[5.1.2.1 Bryan’s Sequence Diagram – Send Mail 43](#_Toc347663257)

[5.1.2.2 Bryan’s Sequence Diagram – Receive Mail 43](#_Toc347663258)

[5.1.2.3 Bryan’s Sequence Diagram – Delete Mail 43](#_Toc347663259)

[5.1.2.4 Bryan’s Sequence Diagram – Calculate BMI 44](#_Toc347663260)

[5.1.2.5 Bryan’s Sequence Diagram – Convert Height (Cm/Inch) 44](#_Toc347663261)

[5.1.2.6 Bryan’s Sequence Diagram – Convert Weight (Kg/Lbs) 44](#_Toc347663262)

[5.1.3.1 Benjamin’s Sequence Diagram - Advertisement 45](#_Toc347663263)

[5.1.3.2 Benjamin’s Sequence Diagram - Article 45](#_Toc347663264)

[5.1.4.1 Terence’s Sequence Diagram - Comment 46](#_Toc347663265)

[5.1.4.2 Terence’s Sequence Diagram – Sharing to Facebook 46](#_Toc347663266)

[5.1.4.3 Terence’s Sequence Diagram – Retweet to Twitter 47](#_Toc347663267)

[5.2 Design Class Diagrams 48](#_Toc347663268)

[5.3 UI Prototypes 49](#_Toc347663269)

[5.3.1.1 Alvin’s UI Prototypes – Login Prototype 49](#_Toc347663270)

[5.3.1.2 Alvin’s UI Prototypes – Verify Account Prototype 49](#_Toc347663271)

[5.3.1.3 Alvin’s UI Prototypes – Register Prototype 50](#_Toc347663272)

[5.3.1.4 Alvin’s UI Prototypes – Forgot Username Prototype 50](#_Toc347663273)

[5.3.1.5 Alvin’s UI Prototypes – Forgot Password Prototype 51](#_Toc347663274)

[5.3.2.1 Bryan’s UI Prototypes – BMI Calculator Prototype 51](#_Toc347663275)

[5.3.2.2 Bryan’s UI Prototypes – Compose Mail Prototype 52](#_Toc347663276)

[5.3.2.3 Bryan’s UI Prototypes – Mail Inbox Prototype 52](#_Toc347663277)

[5.3.2.4 Bryan’s UI Prototypes – Mail Sent Messages Prototype 53](#_Toc347663278)

[5.3.2.5 Bryan’s UI Prototypes – Mail Deleted Messages Prototype 53](#_Toc347663279)

[5.3.2.6 Bryan’s UI Prototypes – Mail Received Messages Prototype 54](#_Toc347663280)

[5.3.3.1 Benjamin’s UI Prototypes – RSS Feeds Prototype 54](#_Toc347663281)

[5.3.4.1 Terence’s UI Prototypes – Comment Page Prototype 55](#_Toc347663282)

[6. Test Plan 56](#_Toc347663283)

[6.1 Alvin’s Test Plan 56](#_Toc347663284)

[6.1.1 Alvin’s Test Setup 56](#_Toc347663285)

[6.1.2 Alvin’s Test Procedures 56](#_Toc347663286)

[6.1.3 Alvin’s Test Cases 56](#_Toc347663287)

[6.2 Bryan’s Test Plan 60](#_Toc347663288)

[6.2.1 Bryan’s Test Setup 60](#_Toc347663289)

[6.2.2 Bryan’s Test Procedures 60](#_Toc347663290)

[6.2.3 Bryan’s Test Cases 60](#_Toc347663291)

[6.3 Benjamin’s Test Plan 61](#_Toc347663292)

[6.3.1 Benjamin’s Test Setup 61](#_Toc347663293)

[6.3.2 Benjamin’s Test Procedures 61](#_Toc347663294)

[6.3.3 Benjamin’s Test Cases 61](#_Toc347663295)

[6.4 Terence’s Test Plan 62](#_Toc347663296)

[6.4.1 Terence’s Test Setup 62](#_Toc347663297)

[6.4.2 Terence’s Test Procedures 62](#_Toc347663298)

[6.4.3 Terence’s Test Cases 63](#_Toc347663299)

[7. Conclusion 64](#_Toc347663300)

[7.1 Summary of Completed Tasks 64](#_Toc347663301)

[7.2 Summary of Future Enhancements 65](#_Toc347663302)

[7.3 Lessons Learnt 65](#_Toc347663303)

[Appendix A - Threat Modeling report 66](#_Toc347663304)

[Appendix B - Fortify report (Wk 16) 77](#_Toc347663305)

[Appendix C - Fortify report (Wk 17) 77](#_Toc347663306)

[Appendix D - Fortify report summary 77](#_Toc347663307)

# Document Control

This project report template shall be adopted by the project teams working on IT2539 Applications Security Project.

The document shall adopt the following version numbering corresponding to the project deliverables:

|  |  |
| --- | --- |
| **Version Number (#.#)** | **Project Deliverables** |
| 0.1.# | Project Proposal  Project Plan |
| 0.2.# | Requirements Specification |
| 0.3.# | Design Specification  Demo Prototype |
| 0.4.# | Test Plan |
| 1.0.0 | Final Project Report |

# 1. Introduction

Our project is called **My Well-being Network (MWN).** Our project will be a **Website Application** that will be develops in **C#** and **.NET Framework**. It is a health-related website which consists of sports, diets, and basically information that is related to human physical well-being.

Our project aims to create a health-related website to provide physical well-being information and tools (e.g. BMI calculator, messaging system) related to sports and diet for people who are concerned of their own health and well-being. Also, our project aims to create awareness of information related to health and well-being for people who are unaware of it around the globe!

# 2. Project Proposal

## 2.1 Background

As more and more people, not only Singaporeans, are not aware of their own personal health and well-being, we need to develop a website application portal to create awareness of this issue. Also, people who are more concerned of their own health and well-being may also use our website application to get related information and gain knowledge related to personal health and well-being. Our website application will be user-friendly whereby the users are able to obtain information through articles and use tools provided by our website application to further aid them in their choice of a healthy lifestyle!

## 2.2 Business Case

As it is going to be a free-for-all website application, we are unable to earn any revenue to keep our website keep on-going. Hence, to prevent such incident from happening, our project will implement an Advertising System. Purpose of this advertising system is to allow us to earn additional money resources to keep our website on-going.

As our website consists of 2 types of users – Free Account User & Premium Account User, Free Account Users will be able to remove these advertisements by upgrading their accounts to become Premium Account Users. Other than that, they Premium Account Users also benefits more than the Free Account Users to be able to enjoy many more additional features which Free Account Users who does not have access to. Free Account Users may also donate additional money resources to our website to be eligible to upgrade their accounts to Premium Account Users. Not forgetting, Premium Account Users, as well as Guest who does not have an account for our application may also donate additional money resources to our website!

With the additional money resources we gained from the advertising system, upgrading to premium account users and donation, we are able to solve the issue of not having an on-going website, able to keep our website application having the availability of 100%! Also should there be any surplus in money resources, we may also use it to invest in our web application to enhance it, further develop our web application to make it better, so as to create awareness and attract more people to our website application! As more and more people are aware of our website application, which means that our website application will be able to earn more and more revenue! With greater amount of revenue, we will be able to invest that money to advertise our website around the globe, and further develop our website application to be a well-known website application!

## 2.3 Business Functions

Our project will provide business functions for all of our users. We provide features such as the Login System, Free/Premium Account, RSS Feeds, Advertising System, BMI Calculator, In-house Mail System, Comment system, for personal views of the users, to discuss on specific interests topics, as well as Facebook/Twitter Sharing System. Each individual function was elaborated more in details below:

**Login System:**

As our website application involves 3 types of users – Guest, Free Account User and Premium Account User, we required a Login System for our project to identify which type of user is using our website application. Basically, out of the 3 types of users, Guest is the only type of user who will interact with this system, since Free User & Premium User are already logged into our system, other than the logout button, they will not interact with the system.

Our Login System consists of 6 parts – Login, Registration, Captcha, Two-Factor Authentication, Multiple Invalid Login Timer and Logout. Guest will interact either interact with either Login or Registration depends on whether the Guest has an existing account or not. Under Registration, Guest will require to input their username, password, email etc. to have their account created as a Free Account User.

Login System will also involve in Two-Factor Authentication, Captcha and Multiple Invalid Login which will be elaborated more on Security Functions.

**Free/Premium Accounts:**

As mentioned in the Login System, other than Guest, there are two other types of users – Free Account User and Premium Account User.

Difference is that premium account users are able to enjoy special privileges that free account users don’t. Example, premium account users are able to turn advertising system on and off if they don’t think there is a need for advertising.

After the Guest has registered an account, it will be initially be a Free Account User. To upgrade their Free Account User into a Premium Account User, they can either donate to our company or buy the premium account. A verification code will be given to the user for authentication purposes.

**RSS Feed:**

The RSS feed will provide the latest health and sports news to our users. News articles will be taken from a variety of other websites and compiled into the RSS feed for simpler access. Articles will include both the latest sports news such as scores from recent sporting events to Interviews with famous sports stars, as well as the latest health news such as tip on how to maintain a healthy body to well-being general knowledge. With this function, users of our website will be able to not only learn health tips from us, but also keep up with the latest news of the Health & Fitness world.

The Android app will include a mobile RSS feed. The RSS function of the RSS feed will function similarly to the website application version. It displays the exact same articles as those available on the full website application version. With this app, users of our website can access the latest news via our website on the go.

**Advertising System:**

Advertisements will be displayed on our website. We will approach reputable companies who are related to the Health & Fitness industry (e.g. Nike, Milo, etc.) and offer to advertise their products for a small monthly fee. The money earned from the advertisements will be used for the upkeep of the website. Advertisements can be removed if users decide to purchase a premium account. These advertisements will also provide cheaper deals for products that said companies are offering.

**BMI Calculator:**

The BMI calculator will allow users to calculate their BMI and know their BMI rating before or after they surf our website and to look up articles on how to live healthier. The purpose of the BMI calculator is to allow the users to track their BMI and see how they have been progressing with their diet.

**In-house Mail System:**

The in-house mail system will allow users to mail each other privately to ask about things regarding the articles or to gather additional information on certain things they would like to enquire about. The purpose of the system is to allow the users to mail other users regarding any matter privately as compared to the comment system where everyone can see all the comments being posted.

**Comment System:**

This function allows the user to comment on an article and/or tools that can be found on our website.

By allowing users to comment on our articles can help us generate feedbacks on how to further improve an article or even our whole website. We are given the opportunity to interact with the users and the users are able to interact with each other as well.

Other than helping us understand the users, the users can also help one another by starting a discussion within the comments or answering questions that a user might have depending on the article.

**Facebook/Twitter Sharing System:**

This function will allow users to share the website and/or free tools through the use of social networking websites. As there are a lot of people using social networking websites nowadays, we feel that this function will help to advertise our web application. The user is also able to share articles and/or tools that are found within our website.

Through the use of Facebook and Twitter, the user can introduce the website application to his or her friends by using the Share function or the Like function in Facebook; and the Tweet or Retweet function in Twitter. This will help advertise our website application efficiently as social networking sites are very powerful and can reach out to a large amount of audience in a short amount of time.

## 2.4.1 Security Functions

In addition, our project will also implement security features for all of our users; make them feel secured using our application. We provide features such as On-Screen Keyboard, Two-Factor Authentication, Multiple Invalid Login Timer, Android App 2FA and Captcha System. Each individual security function was elaborated more in details below:

**On Screen Keyboard:**

On Screen Keyboard is a security feature which is basically a virtual keyboard on the computer screen. Users will be able to click on the displayed letters for the letters to be entered. Purpose of this feature is to prevent Keylogger Attacks.

Users who are attacked by the attacker using Keylogger, will get all their information sniffed from the typing of their keyboard. This is dangerous as the attacker will be able to get the username and password from the users. This feature will be able to use on the Login Page and Registration Page by the Guest.

**Two-Factor Authentication (2FA):**

Two-Factor Authentication (2FA) is a security feature to allow the registered user to authenticate himself/herself that he/she is the one who had registered in our system. During registration, they are required to enter their email address and phone number, which will be required for Two-Factor Authentication.

After registration, they will either receive a verification code from their stated email address or a SMS from their stated phone number from our system. They are still unable to login now as their account is not yet verified. After receiving the verification code from either SMS or email, they will then enter the given verification code into the verification field text. If successful, their account will be verified and they are able to login using their verified account. If it fails, they can choose to re-enter the verification code or ask for a new verification code should there be any discrepancy.

**Multiple Invalid Login Timer:**

Multiple Invalid Login Timer is a timer that will be activated at the login page if the Guest fails to enter the correct password at least 5 times. If login fails due to invalid password for 5 consecutive times, an Invalid Login Timer will be shown to countdown for the next available login.

It starts at 30secs, and slowly times 2 for each consecutive time for a maximum of 24hrs. (Example: 5 fails -> 30secs countdown, 6 fails -> 1min countdown, 7 fails -> 2mins countdown, 8 fails -> 4mins countdown, etc etc.) Purpose of this security feature is to prevent attackers from using brute-force attack to retrieve the password of a particular user.

**Android App 2FA:**

The Android app is not only an RSS feed, but doubles as a 2FA device as well. By using the IMEI number of the phone and running it through with an algorithm with a random string provided by the website, the app will then present a new string, and the user is required to enter said string into the website, in which a similar algorithm is used on the IMEI number provided by the user at startup, and a random string generated. If the 2 input strings match, access will be granted.

This will provide a safer way for users to log in or retrieve their password, without worrying if someone has both their username and password as the users phone is still needed for authentication.

**Captcha System:**

The Captcha function will be able to distinguish a human user and a bot user. This will make sure that the user logging in is indeed human and not being controlled by a bot. The purpose of Captcha is to distinguish between a legit user and a remote robot user.

## 2.4.2 Secure Coding

Our project also practices secure coding, which will prevent attackers from attacking into our system. We will be preventing 4 out of Top 10 Web Application Security Vulnerabiliies such as (A1) SQL Injection, (A3) Broken Authentication and Session Management, (A7) Insecure Cryptographic Storage, and (A10) Invalidated Redirects and Forwards.

To prevent this, we will implement HTTPS, Hashing, and Encryption techniques and many more; aim to allow our users to feel safe and secured from attacks when using our application. Each individual secure coding function was elaborated more in details below:

**(A1) SQL Injection Prevention – Parameterized Queries & Stored Procedure**

As known by all, database is very important that it must be secured as it stores sensitive information about our users. So, a secured database is a must to make sure our users’ sensitive information are stored securely without being deleted or read. When an attacker is able to insert a series of SQL statements into a ‘query’ by manipulating data input into our application, which means that our web application was suffered by SQL Injection.

To prevent SQL Injection issues, our project will use parameterized queries to bind variables rather than concatenating SQL statements together as strings. We may also use stored procedures to prevent users from directly interacting with our SQL codes. With either one of these methods, database is highly secured from SQL Injection attacks, making our website application more secured!

**(A3) Broken Authentication and Session Management – HTTPS:**

The HTTPS will allow users to surf our website securely and know that their connection to our website is indeed secure and safe. The purpose of HTTPS is to securely protect the connection between the user and the server

**(A7) Insecure Cryptographic Storage – Hashing:**

For this project, hashing will be used for the passwords. We will be using secure salted password hashing.

As the passwords of the users will be saved in a database, we will have to hash the passwords to ensure that even if there was a breach of files, the user’s passwords will remain unknown. But this may not be enough as a user may use a rainbow table or pre-compute a table of hashes of every 10 character passwords, or the hashes of every common word, and use that to figure out which password was used to generate the hash. The hacker will then be able to figure out the plain-text of the password.

To complicate matters for the hackers, or the instance that this scenario may happen, a random but known string known as Salt, is added to the password before hashing. This isn’t necessarily a secret but it forces the hacker to re-compute his table of possible password hashes.

**(A7) Insecure Cryptographic Storage – Encryption:**

For this project encryption will be used on the credit card number of our clients.

One of the actors for our project is the premium users. The premium users are grated access to all the tools that are available in our website.

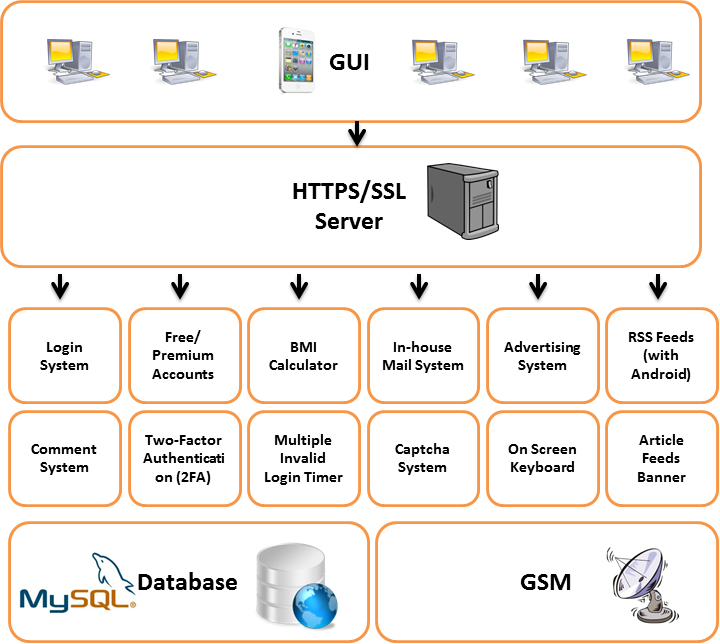
To become a premium user, the user will have to pay a sum of subscription fee or donate to us and this is to be done through online. This means that we will have to obtain the customers personal details and credit card number. Therefore we have the responsibility to keep it safe from any threat.

We will use encryption and have the credit card number saved in our database to access it whenever a purchase is made.

**(A10) Invalidated Redirects and Forwards – Website Redirection Prevention:**

Our website application will also feature website redirection prevention. As hackers would take advantage of the ads and links available on the website and use them to redirect our users to their own malicious sites. Thus, by setting up a website redirection system, we can ensure our users a safe browsing experience without having to worry about having their valuable information stolen or computers getting infected by viruses and malware.

## 2.5 Architecture Diagram

****

## 2.6 Resources

The following software is needed for the development of our project:

* Microsoft Visual Studio 2012 Ultimate
* Microsoft Visual Studio 2010 Ultimate (Fortify)
* Microsoft Office 2010

The following hardware is needed for the development of our project:

* Laptops
* Server
* Smart Phone

Other facilities that are needed for the development of our project:

* Computer Labs
* Meeting/Discussion Rooms
* Library

# 3. Project Plan

## 3.1 Team Organization

**Module Group:** IT2539- DIS1102

**Team Name:** MWN

**Team No:** 4

**Team Leader:** Alvin Lee

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S/N** | **Name** | **Admin No** | **Handphone No** | **Email Address** |
| 1\* | Alvin Lee Yong Teck | 113340E | 83664422 | [113340E@mymail.nyp.edu.sg](mailto:113340E@mymail.nyp.edu.sg) |
| 2 | Lam Yong Quan Terence | 111070Z | 91119235 | [111070Z @mymail.nyp.edu.sg](mailto:114905L@mymail.nyp.edu.sg) |
| 3 | Lim Kian Hock Bryan | 113871T | 91371197 | [113871T @mymail.nyp.edu.sg](mailto:114853Y@mymail.nyp.edu.sg) |
| 4 | Wong Zi Feng Benjamin | 111199P | 96354416 | [111199P @mymail.nyp.edu.sg](mailto:111413S@mymail.nyp.edu.sg) |

\*denotes the team leader

## 3.2 Task Allocation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/N** | **Task** | **Alvin Lee** | **Bryan Lim** | **Benjamin Wong** | **Terence Lam** |
| 1 | Login System | ✓ |  |  |  |
| 2 | Free/Premium Accounts | ✓ |  |  |  |
| 3 | On Screen Keyboard | ✓ |  |  |  |
| 4 | Two-Factor Authentication (2FA) | ✓ |  |  |  |
| 5 | Multiple Invalid Login Timer | ✓ |  |  |  |
| 6 | BMI Calculator |  | ✓ |  |  |
| 7 | In-house Mail System |  | ✓ |  |  |
| 8 | Captcha System |  | ✓ |  |  |
| 9 | Advertising System |  |  | ✓ |  |
| 10 | RSS Feeds |  |  | ✓ |  |
| 11 | Android RSS Feeds |  |  | ✓ |  |
| 12 | Article Feeds Banner |  |  | ✓ |  |
| 13 | Android App 2FA |  |  | ✓ |  |
| 14 | Comment System |  |  |  | ✓ |
| 15 | Sharing System (Facebook/Twitter) |  |  |  | ✓ |
| 16 | SQL Injection Prevention | ✓ |  |  |  |
| 17 | HTTPS |  | ✓ |  |  |
| 18 | Unvalidated Redirects (Phishing) |  |  | ✓ |  |
| 19 | Salt Hashing |  |  |  | ✓ |
| 20 | Encryption |  |  |  | ✓ |
| 21 | Integration | ✓ |  |  |  |
| 22 | SQL Database Design | ✓ |  |  |  |
| 23 | Cloud Hosting |  | ✓ |  |  |
| 24 | Overall Project Design |  |  | ✓ |  |
| 25 | Information Gathering |  |  |  | ✓ |

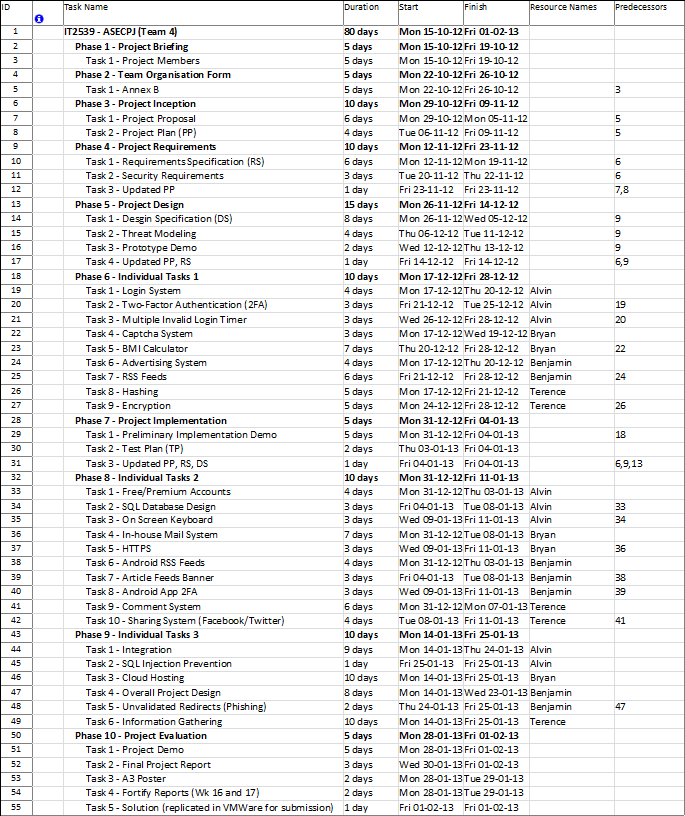
\*S/N 1 to 15 (Main Content & Security Features), 16 to 20 (Secure-coding), 21 to 25 (Backend)

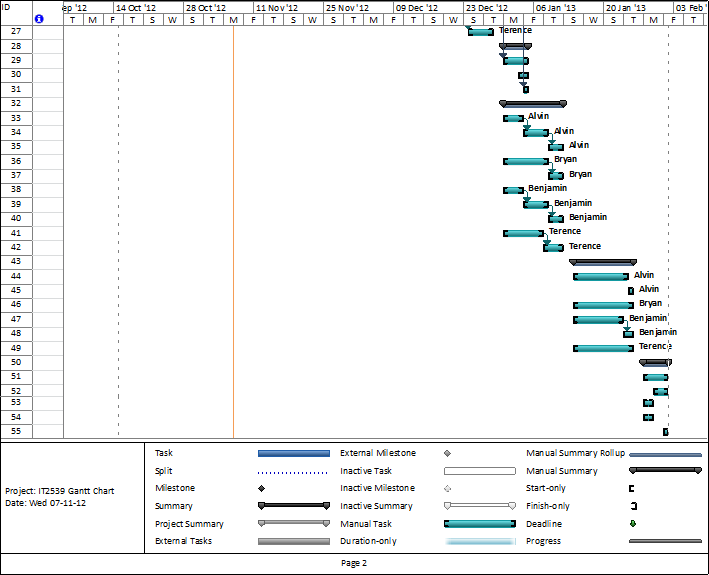
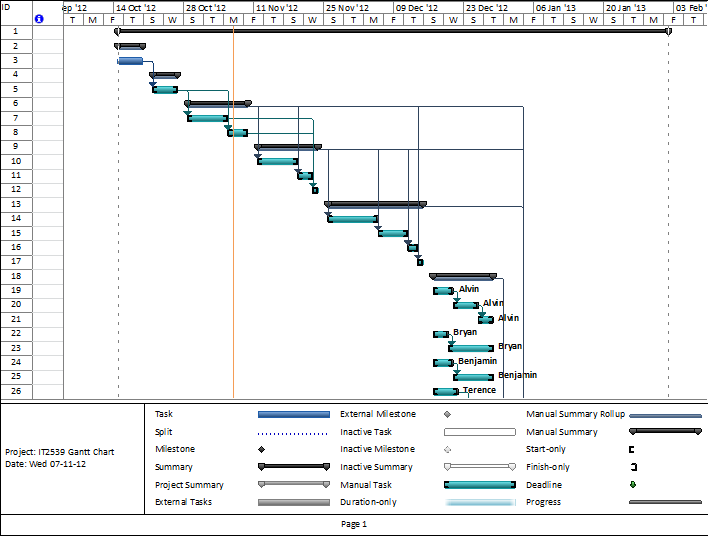
## 3.3 Schedule

Our project phases, deliverables and schedule are as follows:

|  |  |  |
| --- | --- | --- |
| **Phases/Deliverables** | **Week** | **Deadlines of Deliverables** |
| **Project Briefing**   * Project members | **1** | **19th Oct 2012** |
| **Team Organization Form**   * Annex B | **2** | **25th Oct 2012** |
| **Project Inception**   * Project Proposal * Project Plan (PP) | **4** | **9th Nov 2012** |
| **Project Requirements**   * Requirements Specification (RS) * Security Requirements * Updated PP | **6** | **23th Nov 2012** |
| **Project Design**   * Design Specification (DS) * Threat Modeling * Prototype Demo * Updated PP, RS | **9** | **14th Dec 2012** |
| **Individual Tasks 1**   * Login System **Alvin** * Two-Factor Authentication (2FA) **Alvin** * Multiple Invalid Login Timer **Alvin** * Captcha System **Bryan** * BMI Calculator **Bryan** * Advertising System **Benjamin** * RSS Feeds **Benjamin** * Encryption **Terence** * Salt Hashing **Terence** | **11** | **28th Dec 2012** |
| **Project Implementation**   * Preliminary Implementation Demo * Test Plan (TP) * Updated PP, RS, DS | **12** | **4th Jan 2013** |
| **Individual Tasks 2**   * Free/Premium Accounts **Alvin** * SQL Database Design **Alvin** * On Screen Keyboard **Alvin** * In-house Mail System **Bryan** * HTTPS **Bryan** * Android RSS Feeds **Benjamin** * Article Feeds Banner **Benjamin** * Android App 2FA **Benjamin** * Comment System **Terence** * Sharing System (Facebook/Twitter) **Terence** | **13** | **11th Jan 2013** |
| **Individual Tasks 3**   * Integration **Alvin** * SQL Injection Prevention **Alvin** * Cloud Hosting **Bryan** * Overall Project Design **Benjamin** * Unvalidated Redirects (Phishing) **Benjamin** * Information Gathering **Terence** | **15** | **25th Jan 2013** |
| **Project Evaluation**   * Project Demo * Final Project Report * A3 Poster * Fortify Reports (Wk 16 and Wk 17) * Solution (replicated in VMWare for submission) | **16** | **1st Feb 2013** |

## 3.4 Gantt Chart

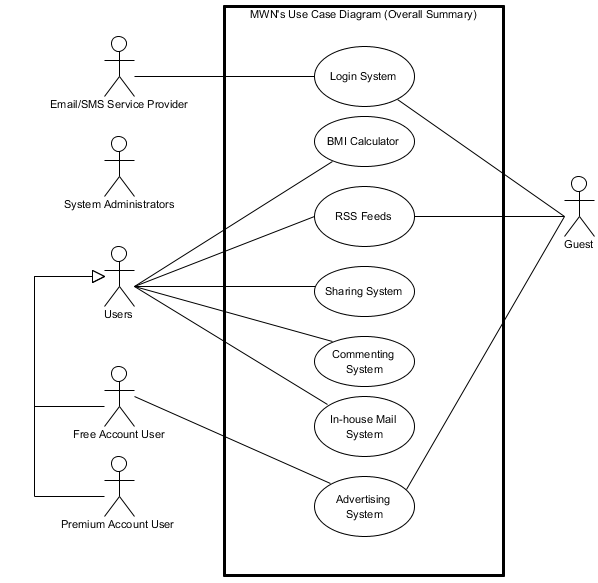
****



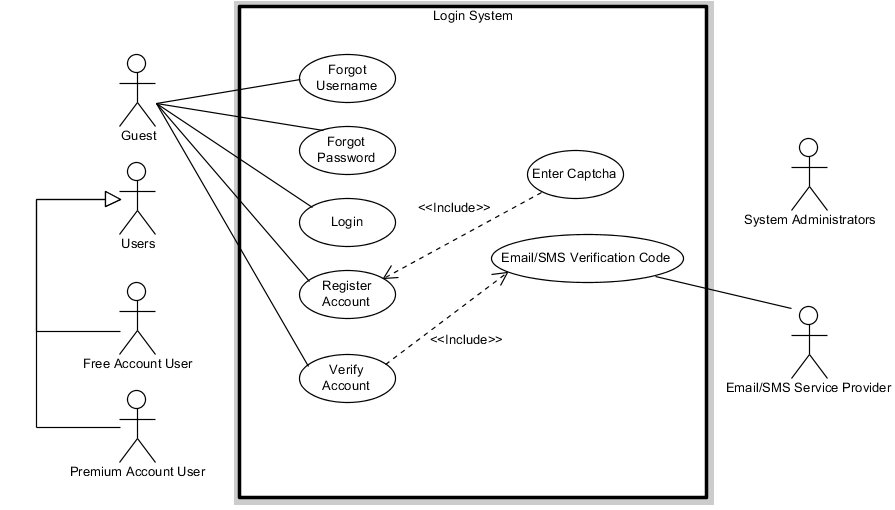
# 4. Requirements Specification

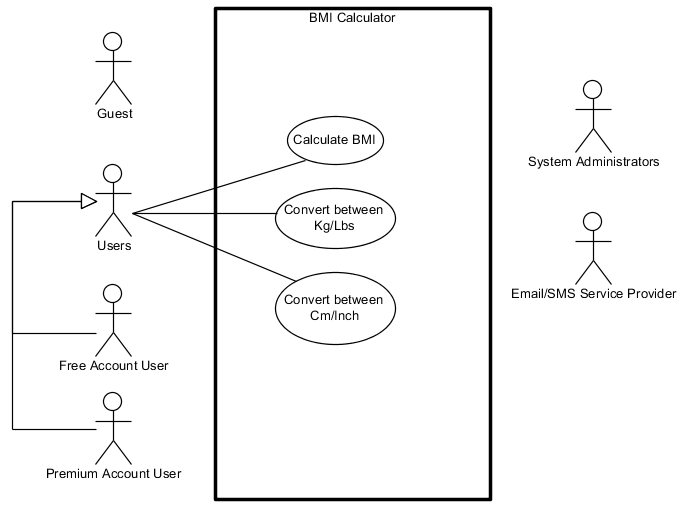
## 4.1 Use Case Diagrams

## 4.1.1 Use Case Diagram (Overall Summary)

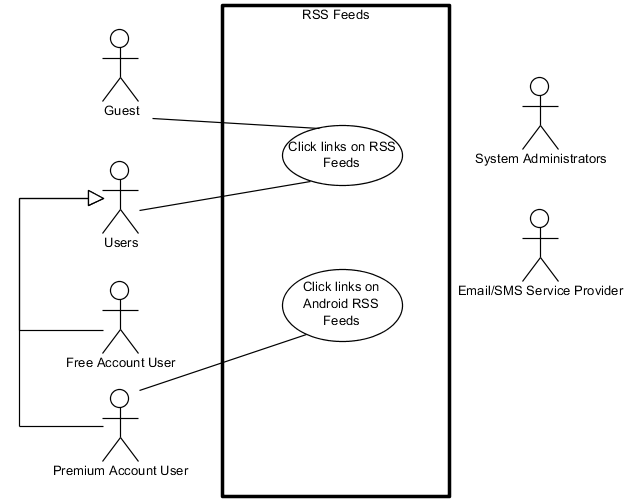


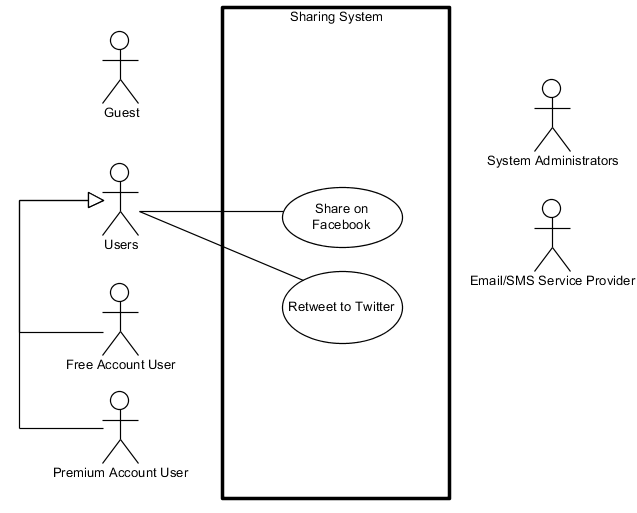
## 4.1.1.1 Use Case Diagram (Overall Summary) – Login System



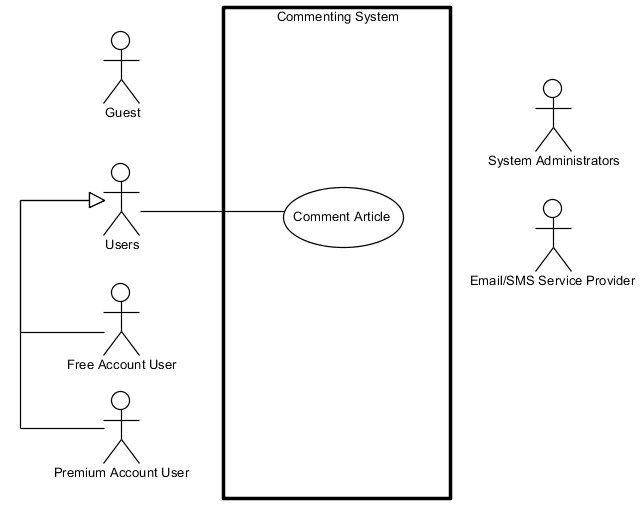
4.1.1.2 Use Case Diagram (Overall Summary) – BMI Calculator

## **4.1.1.3 Use Case Diagram (Overall Summary) – RSS Feeds**

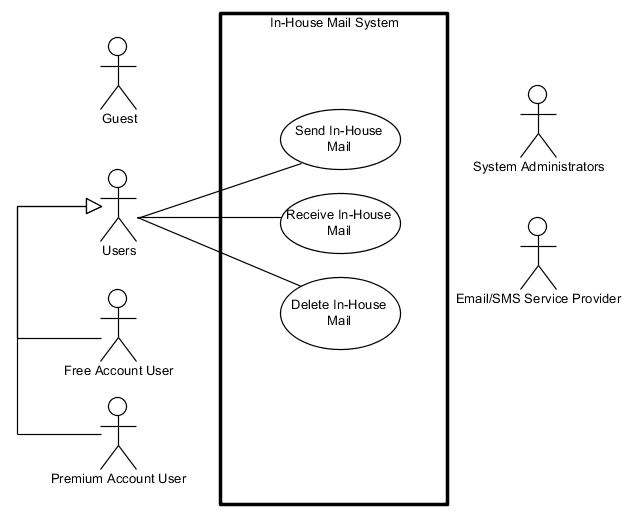
****

4.1.1.4 Use Case Diagram (Overall Summary) – Sharing System

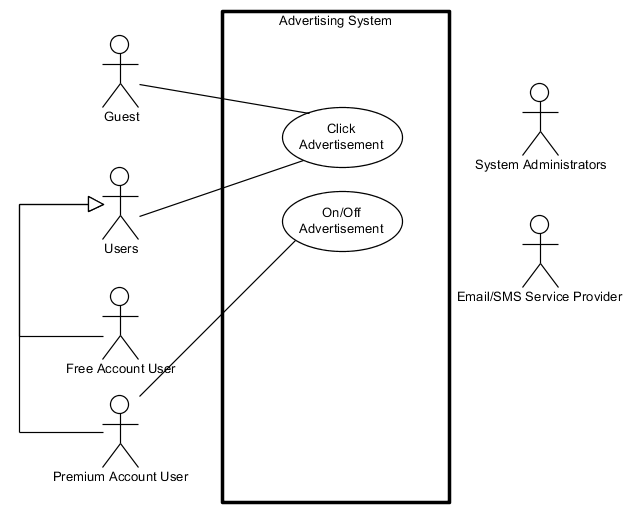
## **4.1.1.5 Use Case Diagram (Overall Summary) – Commenting System**

****

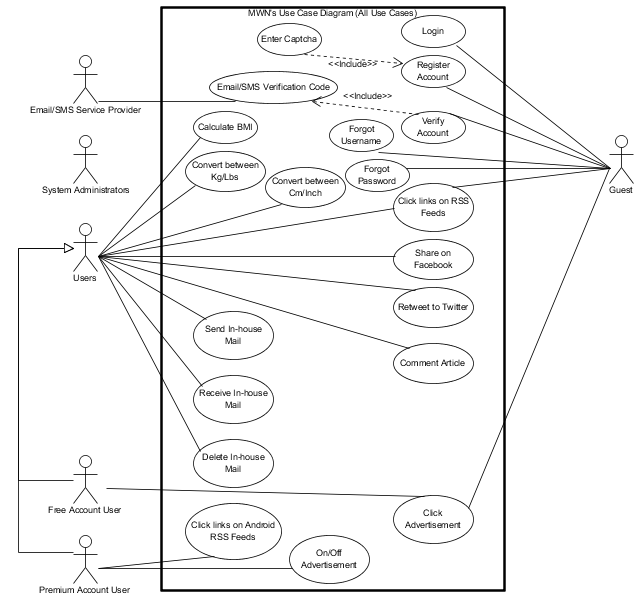
## **4.1.1.6 Use Case Diagram (Overall Summary) – In-House Mail System**

****

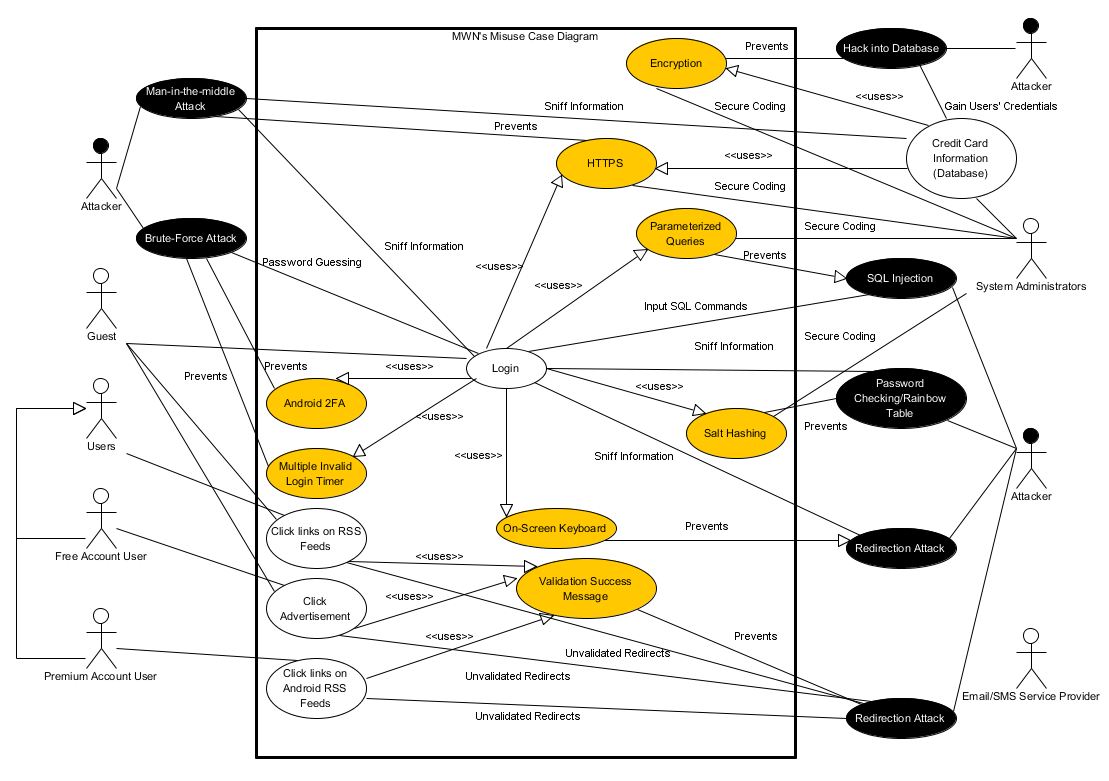
## 4.1.1.7 Use Case Diagram (Overall Summary) – Advertising System



## 4.1.2 Use Case Diagram (All Use Cases)



## 4.2 Misuse Case Diagrams



## 4.3 Use Case Descriptions

## 4.3.1 Alvin’s Use Case Descriptions

|  |  |
| --- | --- |
| **Use Case Name:** | Register Account |
| **Brief Description:** | This use case allows the Guest to create an account by registering their information onto the required fields. Successful registration will be given a Free Account which is able to gain access rights to certain features which Guest don’t benefits. |
| **Actors:** | Guest |
| **Pre-Conditions:** | * Guest must not be logged in. * Guest does not have an existing account. |
| **Post-Conditions:** | * Account successfully created, but is yet to be verified. * System will prompt the Guest to verify his/her account. * Email/SMS Service Provider will send a verification code via Email/SMS to the registered Guest. |
| **Basic Flow:** | 1. At the Registration Page, Guest will require to enter their required information such as Username, Password, Confirm Password, First Name, Last Name, E-mail Address, Confirm E-mail Address, Phone Number, Security Question, Security Answer, and Captcha (refer to Enter Captcha’s Use Case). 2. System will check whether the input Username is existing or not in the database. |
| **Alternate Flow:** | 1. Guest enters invalid information or required field is not yet included. Guest will then require to re-enter valid information and empty required fields. 2. If the system found an existing Username in the database, the system will validate and prompt an error message (e.g. “Existing Username. Please re-enter.”). Guest will then require to re-enter a non-existing Username. |

|  |  |
| --- | --- |
| **Use Case Name:** | Verify Account & Email/SMS Verification Code |
| **Brief Description:** | This use case allows the Guest to verify his/her account after he/she has registered their account in the Register Account use case. Successful verification will allow the user to login successfully using their verified account. |
| **Actors:** | Guest |
| **Pre-Conditions:** | * Guest must not be logged in. * Account must not be verified yet |
| **Post-Conditions:** | * Account successfully verified. |
| **Basic Flow:** | 1. From the login page, Guest will click ‘Verify Account’ to proceed to the Verification Page. 2. At the Verification Page, Guest will require to enter their Username, followed by a verification code provided by the Email/SMS Service Provider via Email/SMS. 3. System will check whether the verification code provided by the Guest matches the verification code given to the Guest via Email/SMS. |
| **Alternate Flow:** | 1. If verification code is not sent to Guest via Email/SMS, Guest may click on the “Re-send verification code” link and enter their registered Username to have their verification code re-send to their Email/SMS. 2. If the system found the input verification code does not match with the verification code given via Email/SMS, the system will validate and prompt an error message (e.g. “Incorrect Verification Code. Please re-enter.”). Guest will then re-enter a valid verification code. |

|  |  |
| --- | --- |
| **Use Case Name:** | Login |
| **Brief Description:** | This use case allows the Guest to login using his/her registered account as Free Account User or Premium Account User, to access respective accessible features and benefits. |
| **Actors:** | Guest |
| **Pre-Conditions:** | * Guest must not be logged in. * Guest must have an existing account to login. * Account must be verified. |
| **Post-Conditions:** | Guest successfully logged in as Free Account User or Premium Account User. |
| **Basic Flow:** | 1. From the login page, Guest will enter his/her Username and Password in the given text field. 2. System will first check for the existing of the account by searching Username in the database. 3. System will then check for the Username and Password against the database whether does it match. 4. System will check whether the account is verified or not. 5. System will finally check whether the account is Free Account User or Premium Account User, then allow the Guest to login with respective access rights. |
| **Alternate Flow:** | 1. Guest enters invalid information (e.g. Username and Password mismatch), re-enter valid Username and Password. 2. If the system cannot find the input Username in the database, the system will validate and prompt an error message (e.g. “Incorrect Password. Please re-enter.” Due to security purposes, it will not prompt an error message that says the Username was not found). Guest will then re-enter valid Username and Password. 3. If the system found the input Username in the database, however the Username and Password does not match, the system will validate and prompt an error message (e.g. “Incorrect Password. Please re-enter.”). Guest will then re-enter valid Username and Password. 4. If the system found out that the input Username and the password in the database matches, however found out that the account is not yet verified, the system will validate and prompt an error message (e.g. “Account not yet verified. Please verify.”). Guest will then know that the account is not yet verified and needs to verify the account. |

|  |  |
| --- | --- |
| **Use Case Name:** | Retrieve Forgotten Password |
| **Brief Description:** | This use case allows the Guest who forgotten his/her registered account’s password to enter their username and security answer, and he/she will be given a random generated password via E-mail/SMS |
| **Actors:** | Guest |
| **Pre-Conditions:** | * Guest must not be logged in. * Guest must have an existing account. |
| **Post-Conditions:** | Guest’s password successfully changed to the random generated password given via E-mail/SMS. |
| **Basic Flow:** | 1. From the forgot password page, Guest will enter his/her Username and Security Answer in the given text field. 2. System will first check for the existing of the account by searching Username in the database. 3. System will then check for the Username and Security Answer against the database whether does it match. 4. System then sends a request for SMS/Email Service Provider to notify the Guest’s new random generated password via SMS/Email 5. Guest will then require to login using the given password and upon login successful, he/she can choose to change into their desire password. |
| **Alternate Flow:** | 1. Guest enters invalid information (e.g. Username and Security Answer mismatch), re-enter valid Username and Security Answer. 2. If the system cannot find the input Username in the database, the system will validate and prompt an error message (e.g. “Incorrect Security Answer. Please re-enter.” Due to security purposes, it will not prompt an error message that says the Username was not found). Guest will then re-enter valid Username and Security Answer. 3. If the system found the input Username in the database, however the Username and Security Answer does not match, the system will validate and prompt an error message (e.g. “Incorrect Security Answer. Please re-enter.”). Guest will then re-enter valid Security Answer. 4. If the Guest was not notified by the SMS/Email Service Provider, he/she will have to repeat the steps from the beginning to let the SMS/Email Service Provider to resend the new random generated password, due to security purposes. Guest will require to double check his/her input particulars upon submission. |

|  |  |
| --- | --- |
| **Use Case Name:** | Retrieve Forgotten Username |
| **Brief Description:** | This use case allows the Guest who forgotten his/her registered account’s username to enter their registered E-mail Address and security answer, and his/her username will be send to them via E-mail |
| **Actors:** | Guest |
| **Pre-Conditions:** | * Guest must not be logged in. * Guest must have an existing account. |
| **Post-Conditions:** | Guest’s username successfully notified by the SMS/Email Service Provider via E-mail. |
| **Basic Flow:** | 1. From the forgot username page, Guest will enter his/her E-mail Address and Security Answer in the given text field. 2. System will first check for the existing of the account by searching the E-mail Address in the database. 3. System will then check for the E-mail Address and Security Answer against the database whether does it match. 4. System then sends a request for SMS/Email Service Provider to notify the Guest’s username via SMS/Email |
| **Alternate Flow:** | 1. Guest enters invalid information (e.g. E-mail Address and Security Answer mismatch), re-enter valid E-mail Address and Security Answer. 2. If the system cannot find the input E-mail Address in the database, the system will validate and prompt an error message (e.g. “This e-mail address does not have a registered account. Please re-enter.”). Guest will then re-enter valid E-mail Address and Security Answer. 3. If the system found the input E-mail Address in the database, however the E-mail Address and Security Answer does not match, the system will validate and prompt an error message (e.g. “Incorrect Security Answer. Please re-enter.”). Guest will then re-enter valid Security Answer. 4. If the Guest was not notified by the SMS/Email Service Provider, he/she will have to repeat the steps from the beginning to let the SMS/Email Service Provider to resend the Guest’s username, due to security purposes. Guest will require to double check his/her input particulars upon submission. |

|  |  |
| --- | --- |
| **Use Case Name:** | Multiple Invalid Login Timer |
| **Brief Description:** | This use case is a security feature to prevent the attacker to spam login and sniff account information using Brute-Force Attack for password guessing. This feature will require Guest to login after a certain period of time if there are more than 5 consecutive failed logins occurs. |
| **Actors:** | Guest |
| **Pre-Conditions:** | * Guest must not be logged in. |
| **Post-Conditions:** | Successfully preventing the attacker from using Brute-Force Attack for login spams. |
| **Basic Flow:** | 1. System check the login input against the database to determine whether it is a successful login or not. 2. System will trigger a counter to count for the number of consecutive failed logins if unsuccessful login encounters. 3. System will trigger the Multiple Invalid Login Timer if counter reaches 5 consecutive failed logins, to prevent the user from logging in for the next 30 seconds. (Time will increase if the times for consecutive failed logins increase). |
| **Alternate Flow:** | - |

|  |  |
| --- | --- |
| **Use Case Name:** | On-Screen Keyboard |
| **Brief Description:** | This use case is a security feature to prevent the attacker to sniff account information using Keylogger Attack to record information entered by the victim. This feature will allow the Guest to have an option to enter their sensitive information (e.g. Username, Password) using On-Screen Keyboard to prevent Keylogger Attack. When using On-Screen Keyboard, Guest must take note of shoulder surfing as well. |
| **Actors:** | Guest |
| **Pre-Conditions:** | * Guest must not be logged in. |
| **Post-Conditions:** | Successfully preventing the attacker from using Keylogger Attack to record sensitive information entered by the victim. |
| **Basic Flow:** | 1. From the Login Page or Registration Page, Guest will be given an option to use On-Screen Keyboard to enter their sensitive information such as Username and Password. |
| **Alternate Flow:** | - |

## 4.3.2 Bryan’s Use Case Descriptions

|  |  |
| --- | --- |
| **Use Case Name:** | Enter Captcha |
| **Brief Description:** | To differentiate if the user registering is a bot or a human |
| **Actors:** | Guests |
| **Pre-Conditions:** | none |
| **Post-Conditions:** | Users are verified as humans |
| **Basic Flow:** | 1. System display captcha 2. User enters captcha and system validates captcha 3. Successfully registered |
| **Alternate Flow:** | 1. Captcha is entered wrongly – Prompts user to re-enter |

|  |  |
| --- | --- |
| **Use Case Name:** | Calculate BMI |
| **Brief Description:** | To allow users to calculate their BMI |
| **Actors:** | Free Account Users, Premium Account Users |
| **Pre-Conditions:** | Users have to be logged in |
| **Post-Conditions:** | BMI rating is displayed |
| **Basic Flow:** | 1. User enters height & weight 2. System validates height & weight 3. System display BMI |
| **Alternate Flow:** | 1. Either height or weight is entered wrongly – Prompts user to re-enter |

|  |  |
| --- | --- |
| **Use Case Name:** | Convert between Kg/Lbs |
| **Brief Description:** | To allow users to convert their weight from Kg to Lbs or vice versa |
| **Actors:** | Free Account Users, Premium Account Users |
| **Pre-Conditions:** | Users have to be logged in |
| **Post-Conditions:** | Weight is converted from Kg to Lbs or vice versa |
| **Basic Flow:** | 1. User enters weight 2. System validates weight 3. System convert between Kg and Lbs or vice versa |
| **Alternate Flow:** | 1. Weight is entered wrongly – Prompts user to re-enter |

|  |  |
| --- | --- |
| **Use Case Name:** | Convert between Cm/Inch |
| **Brief Description:** | To allow users to convert their height from Cm to Inch or vice versa |
| **Actors:** | Free Account Users, Premium Account Users |
| **Pre-Conditions:** | Users have to be logged in |
| **Post-Conditions:** | Height is converted from Cm to Inch or vice versa |
| **Basic Flow:** | 1. User enters height 2. System validates height 3. System convert between Cm and Inch or vice versa |
| **Alternate Flow:** | 1. Height is entered wrongly – Prompts user to re-enter |

|  |  |
| --- | --- |
| **Use Case Name:** | Send In-house Mail |
| **Brief Description:** | To allow users to mail each other |
| **Actors:** | Free Account Users, Premium Account Users |
| **Pre-Conditions:** | Users have to be logged in |
| **Post-Conditions:** | Mail will be sent out to the recipient |
| **Basic Flow:** | 1. Users enters in all the field 2. System validates all fields 3. Mail is sent out |
| **Alternate Flow:** | 1. One field is not entered – Prompts user to re-enter |

|  |  |
| --- | --- |
| **Use Case Name:** | Receive In-house Mail |
| **Brief Description:** | To allow users to receive their mails |
| **Actors:** | Free Account Users, Premium Account Users |
| **Pre-Conditions:** | Users have to be logged in and mail is received |
| **Post-Conditions:** | Mail will be received by the recipient |
| **Basic Flow:** | 1. Mail is sent out by the sender 2. Mail is received by the recipient |
| **Alternate Flow:** |  |

|  |  |
| --- | --- |
| **Use Case Name:** | Delete In-house Mail |
| **Brief Description:** | To allow users to delete their mails |
| **Actors:** | Free Account Users, Premium Account Users |
| **Pre-Conditions:** | Users have to be logged in |
| **Post-Conditions:** | Mails are deleted |
| **Basic Flow:** | 1. Mail is either sent out or received 2. User selects the mails to be deleted 3. Mail is deleted |
| **Alternate Flow:** | 1. No mails are selected to be deleted - – Prompts user to re-select mails to be deleted |

## 4.3.3 Benjamin’s Use Case Descriptions

|  |  |
| --- | --- |
| **Use Case Name:** | Click Advertisement |
| **Brief Description:** | To generate revenue for our website |
| **Actors:** | Guests, Free Account Users |
| **Pre-Conditions:** | None |
| **Post-Conditions:** | Users get redirected to main site of displayed advertisement. |
| **Basic Flow:** | 1. System display advertisement 2. User selects shown advertisement 3. User gets redirected to main site of advertisement |
| **Alternate Flow:** | 1. Premium user can choose to disable advertisements all together. |

|  |  |
| --- | --- |
| **Use Case Name:** | Click links on RSS Feeds |
| **Brief Description:** | To allow users to view collection of articles regarding a certain topic from various websites. |
| **Actors:** | Free Account Users, Premium Account Users |
| **Pre-Conditions:** | User must be logged in |
| **Post-Conditions:** | User views article he has selected |
| **Basic Flow:** | 1. System displayed all compiled article headers 2. User selects article 3. System displays article in web app itself |
| **Alternate Flow:** | 1. User chooses to view article in original website. |

|  |  |
| --- | --- |
| **Use Case Name:** | Click links on Android RSS Feeds |
| **Brief Description:** | Allow users to validate their login. |
| **Actors:** | Premium Account Users |
| **Pre-Conditions:** | Users must own an Android phone |
| **Post-Conditions:** | User validates him/herself successfully and logs in |
| **Basic Flow:** | 1. System displays short random string to be inserted into app 2. User inputs said string into app 3. App returns another string 4. User inserts string into System 5. System validates and grants access |
| **Alternate Flow:** | 1. String does not match and access is denied. |

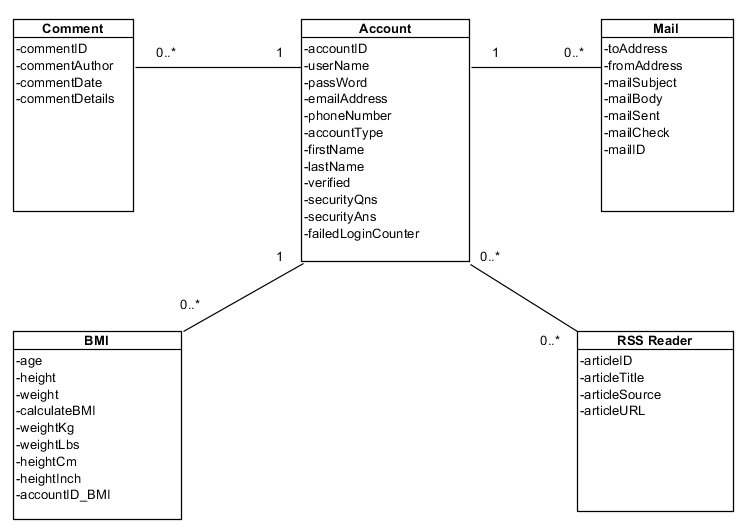
## 4.3.4 Terence’s Use Case Descriptions

|  |  |
| --- | --- |
| **Use Case Name:** | Share on Facebook |
| **Brief Description:** | Grants user ability to share an article or tool on Facebook |
| **Actors:** | Free Account Users, Premium Account Users |
| **Pre-Conditions:** | Login |
| **Post-Conditions:** | Article or tool is shared on the user’s Facebook account |
| **Basic Flow:** | 1. User logs in 2. User clicks on “Share” button 3. User verifies Facebook account 4. User is asked to confirm sharing of article/tool 5. Article/tool is shared |
| **Alternate Flow:** | 1. User fails to log in 2. User Facebook account authentication fails 3. User declines sharing of article/tool 4. Article/tool is not shared |

|  |  |
| --- | --- |
| **Use Case Name:** | Retweet to Twitter |
| **Brief Description:** | Grants user ability to retweet an article or tool on Twitter |
| **Actors:** | Free Account Users, Premium Account Users |
| **Pre-Conditions:** | Login |
| **Post-Conditions:** | Article or tool is shared on the user’s Twitter account |
| **Basic Flow:** | 1. User logs in 2. User clicks on “Retweet” button 3. User verifies Twitter account 4. User is asked to confirm sharing of article/tool 5. Article/tool is retweeted |
| **Alternate Flow:** | 1. User fails to log in 2. User Twitter account authentication fails 3. User declines sharing of article/tool 4. Article/tool is not shared |

|  |  |
| --- | --- |
| **Use Case Name:** | Comment Article |
| **Brief Description:** | Grants user ability to comment on an article or tool on Twitter |
| **Actors:** | Free Account Users, Premium Account Users |
| **Pre-Conditions:** | Login |
| **Post-Conditions:** | User’s comment is seen on article/tool |
| **Basic Flow:** | 1. User logs in 2. User clicks on clicks on article/tool 3. User types in comment in the comment box 4. User is asked to confirm comment on article/tool 5. Comment on article/tool is updated |
| **Alternate Flow:** | 1. User fails to log in 2. User declines commenting of article/tool 3. Article/tool is not updated |

## 4.4 Domain Class Diagrams



## 4.5 Security Requirements

As we all know, there are more and more people in this world was a victim of many different types of attacks across the network. It will be very serious if the victims had their sensitive and confidential information (e.g. Credit Card Number) being sniffed out by attackers or hackers. This will eventually led to the financial loss of victims. Reason is mainly because of the numerous amounts of security vulnerabilities in the website application the victims used.

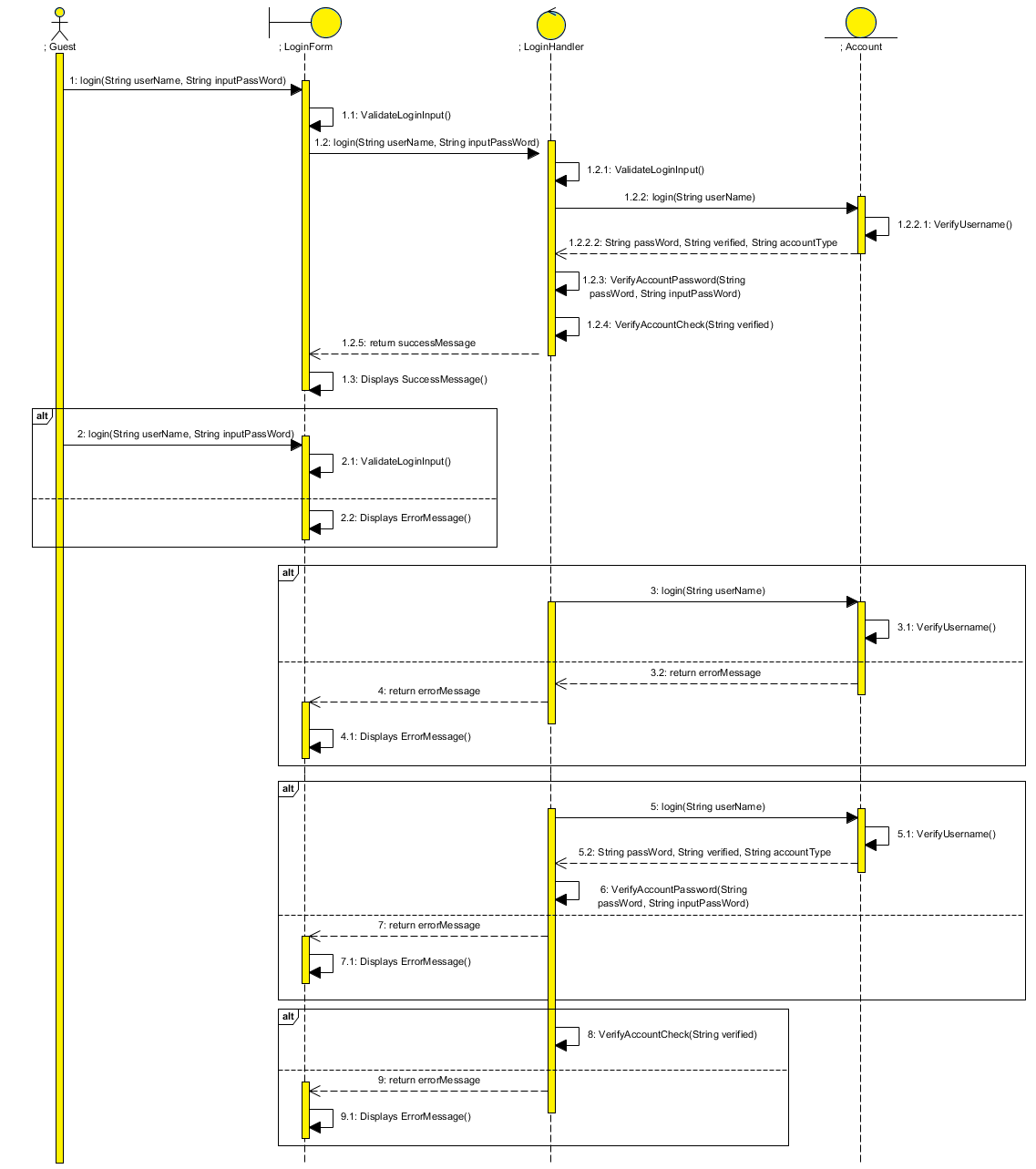
To prevent any users in our system to be victims of those attackers or hackers, our system must be as secured as possible, with little or no security vulnerabilities to be invoked by attackers or hackers. So it is important that our project will implement security features for all of our users; make them feel secured using our application!

Additionally, our project must also practice secure coding so that we can prevent any attackers or hackers across the network from attacking into our system, and then to our users. Our project will be able to prevent 4 out of Top 10 Web Application Security Vulnerabiliies, such as (A1) SQL Injection, (A3) Broken Authentication and Session Management, (A7) Insecure Cryptographic Storage, and (A10) Invalidated Redirects and Forwards. For future enhancement, our project will be able to prevent the other 6 out of Top 10 Web Application Security Vulnerabilities, such as (A2) Cross Site Scripting, (A4) Insecure Direct Object Reference, (A5) Cross Site Request Forgery, (A6) Security Misconfiguration, (A8) Failure to Restrict URL Access, and (A9) Insufficient Transport Layer Protection.

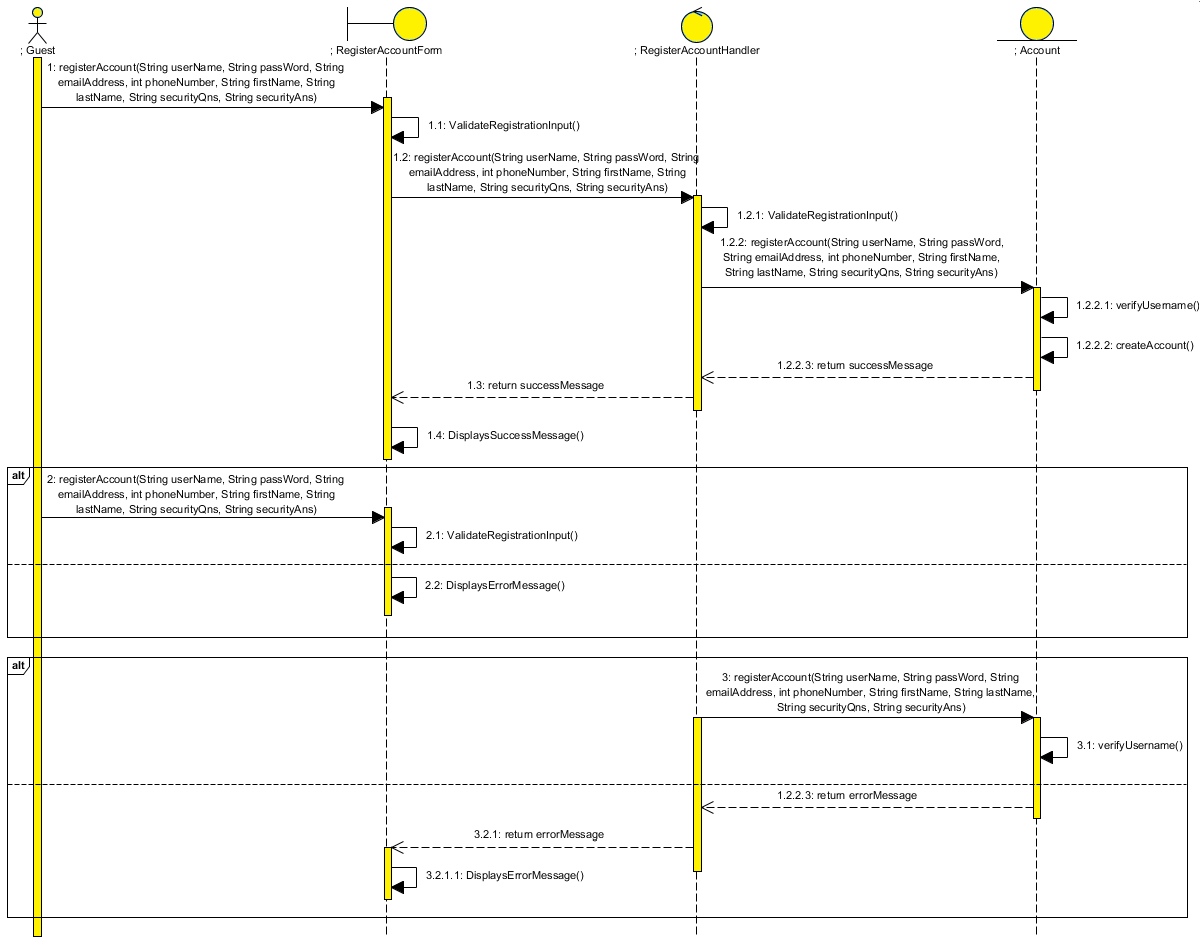
# 5. Design Specification

## 5.1 Sequence Diagrams

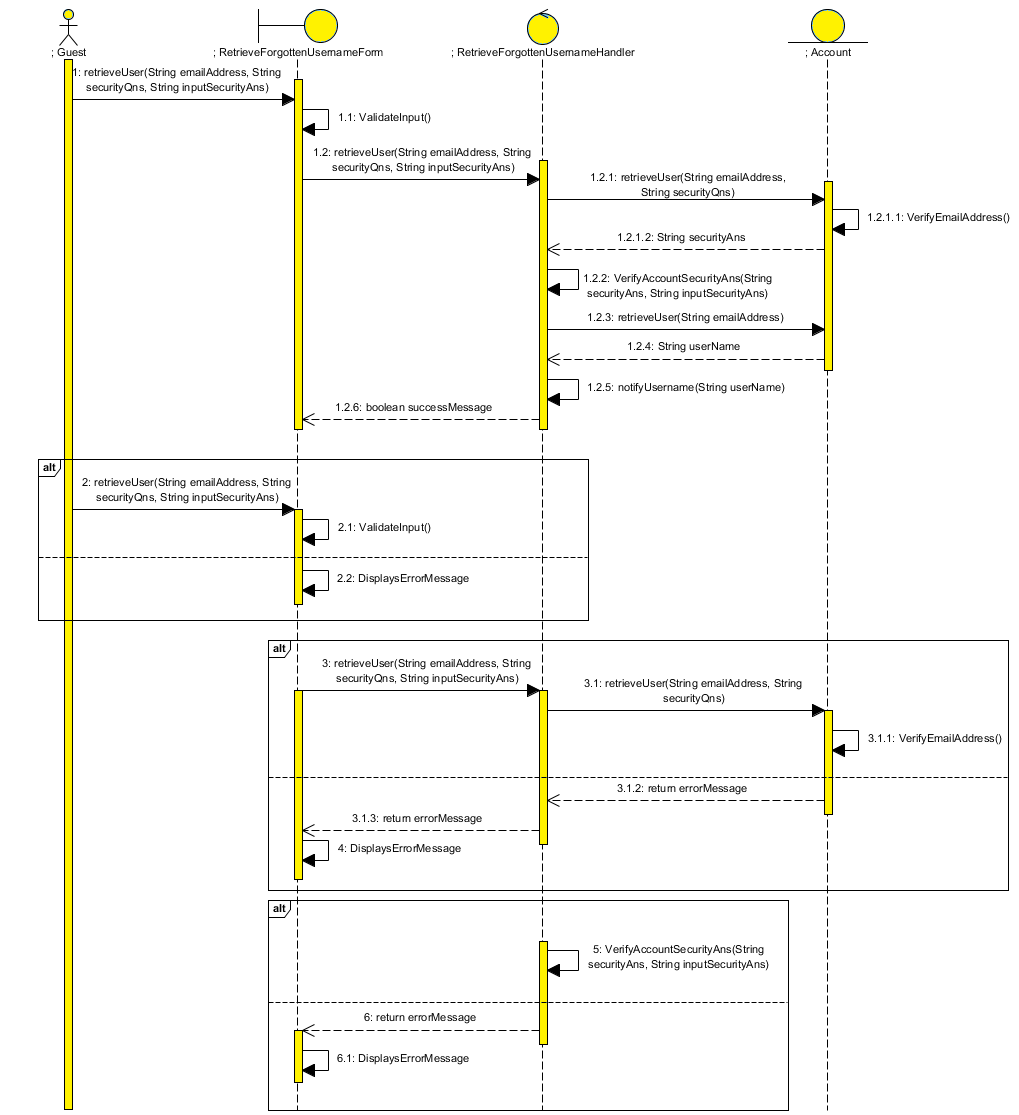
## 5.1.1.1 Alvin’s Sequence Diagrams - Login



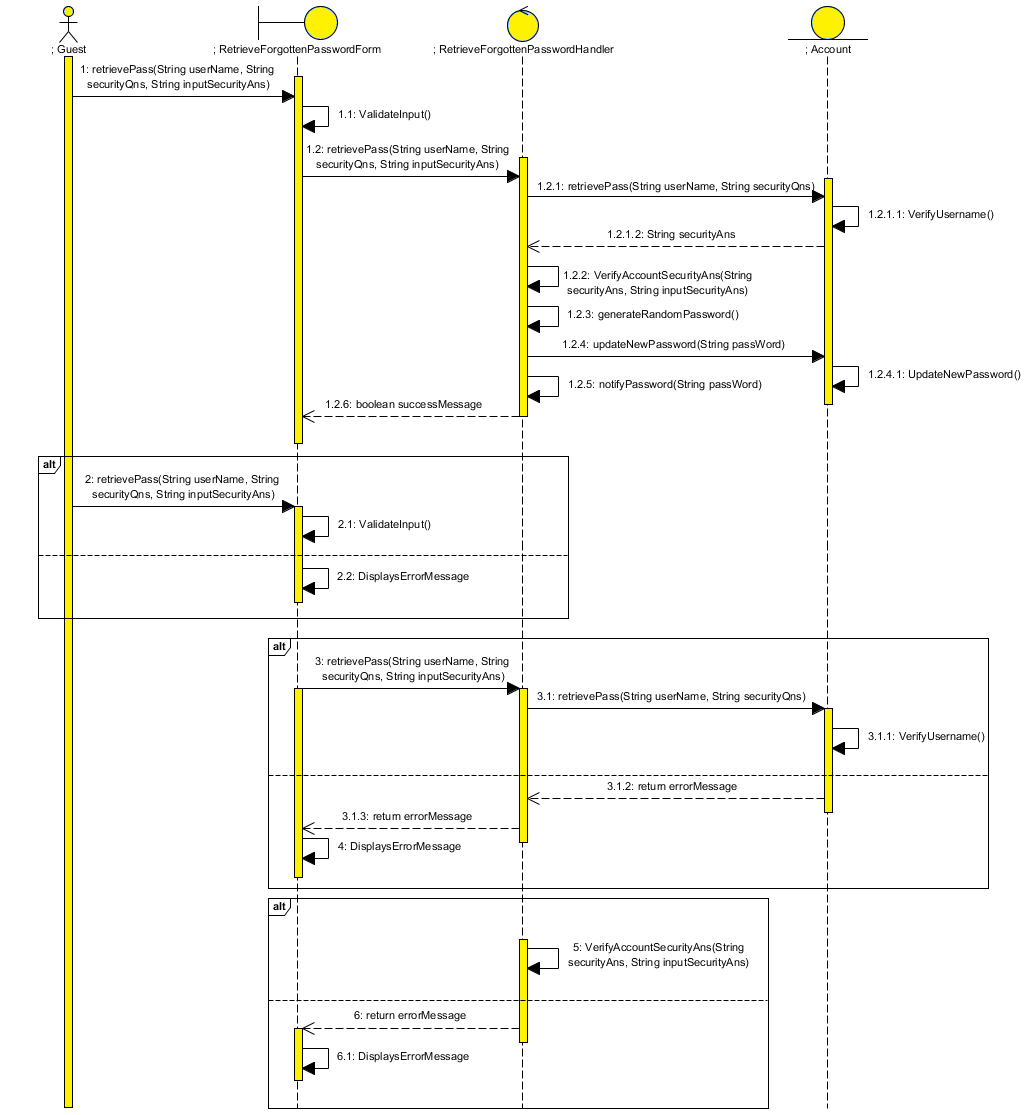
## 5.1.1.2 Alvin’s Sequence Diagram – Register Account



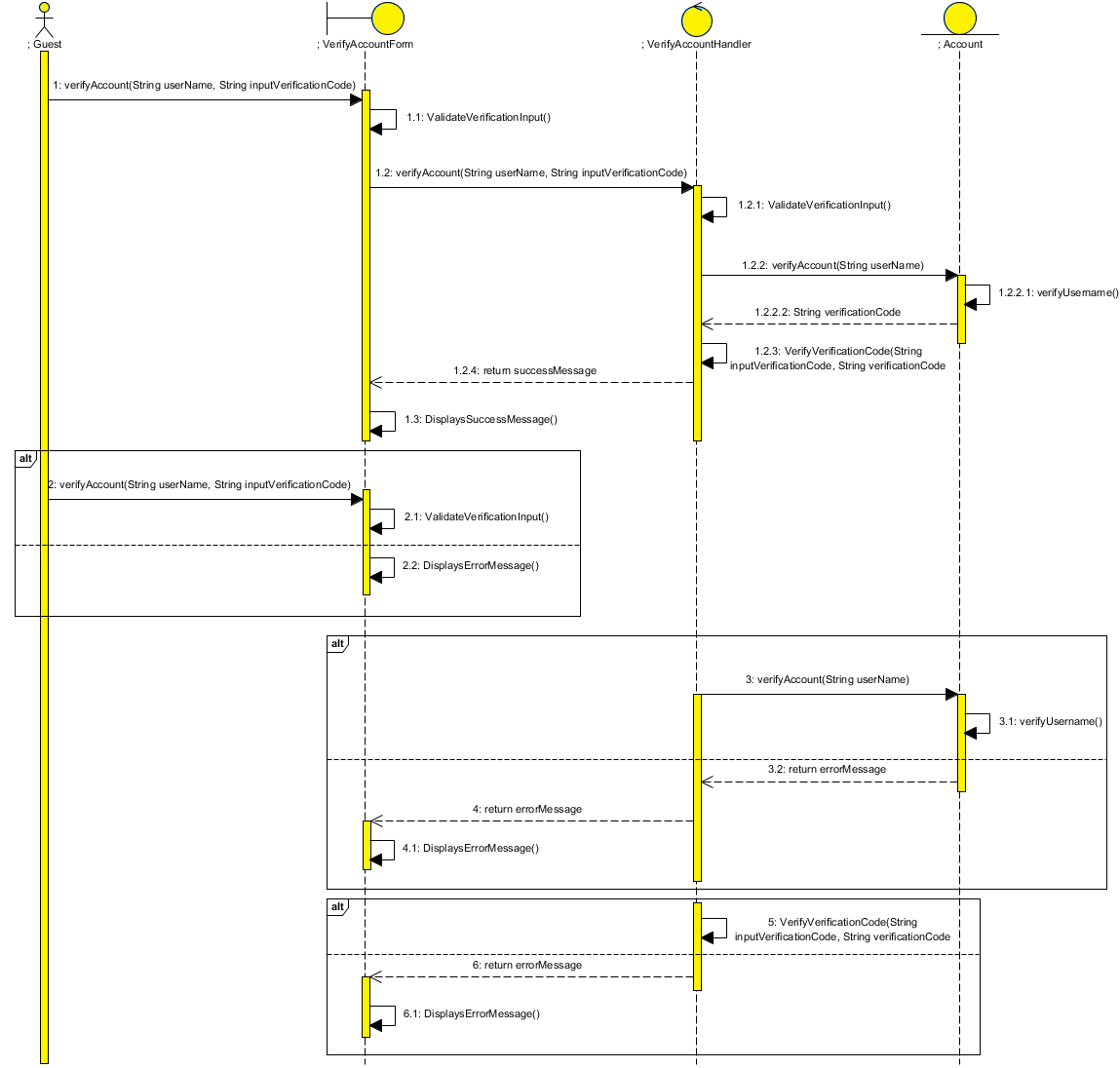
## 5.1.1.3 Alvin’s Sequence Diagram – Retrieve Forgotten Username



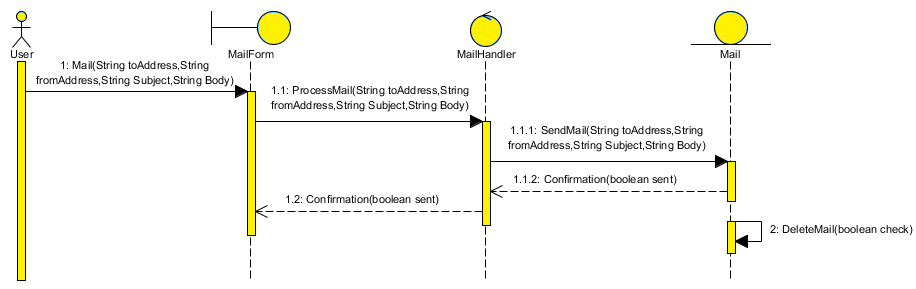
## 5.1.1.4 Alvin’s Sequence Diagram – Retrieve Forgotten Password



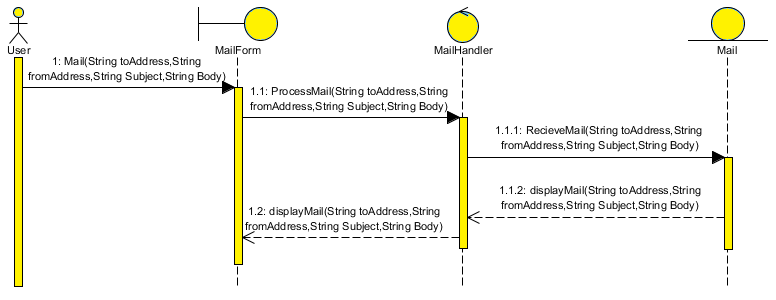
## 5.1.1.5 Alvin’s Sequence Diagram – Verify Account



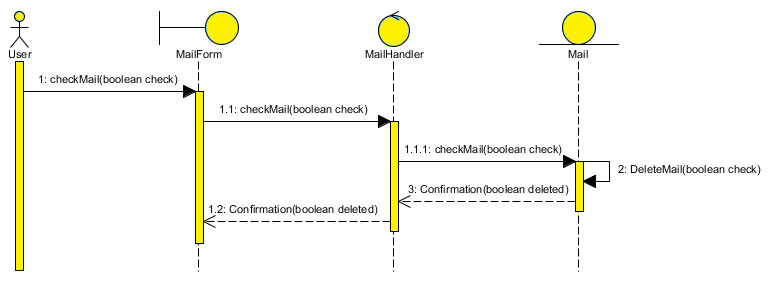
## 5.1.2.1 Bryan’s Sequence Diagram – Send Mail



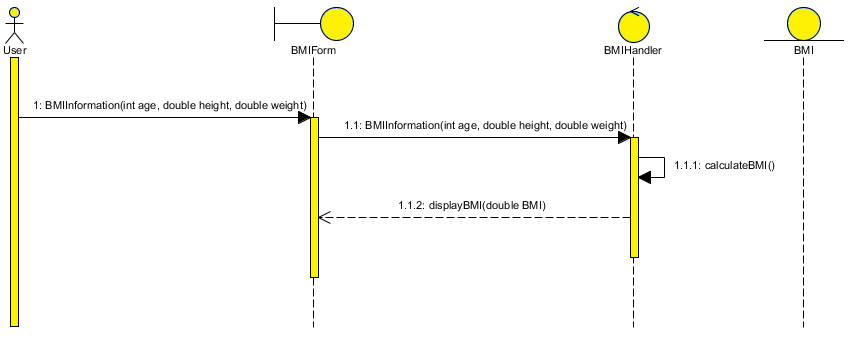
## 5.1.2.2 Bryan’s Sequence Diagram – Receive Mail



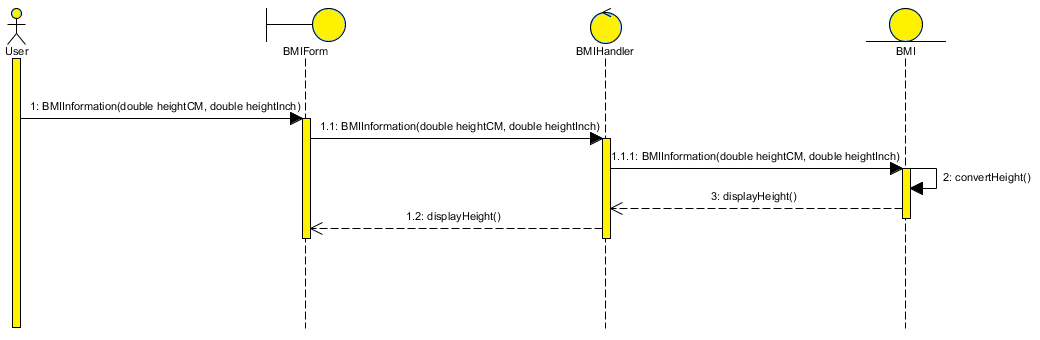
## 5.1.2.3 Bryan’s Sequence Diagram – Delete Mail



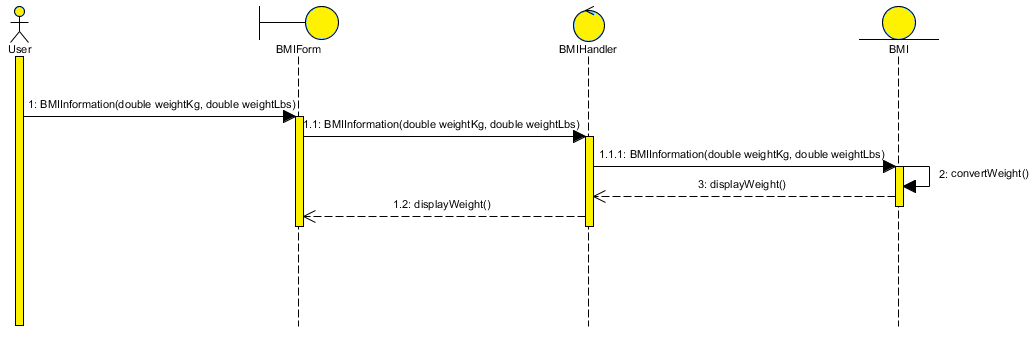
## 5.1.2.4 Bryan’s Sequence Diagram – Calculate BMI



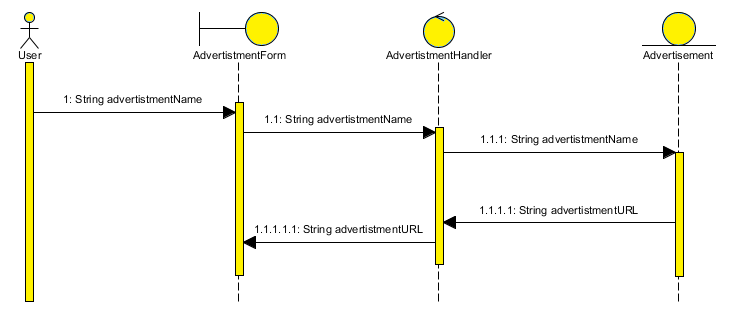
## 5.1.2.5 Bryan’s Sequence Diagram – Convert Height (Cm/Inch)



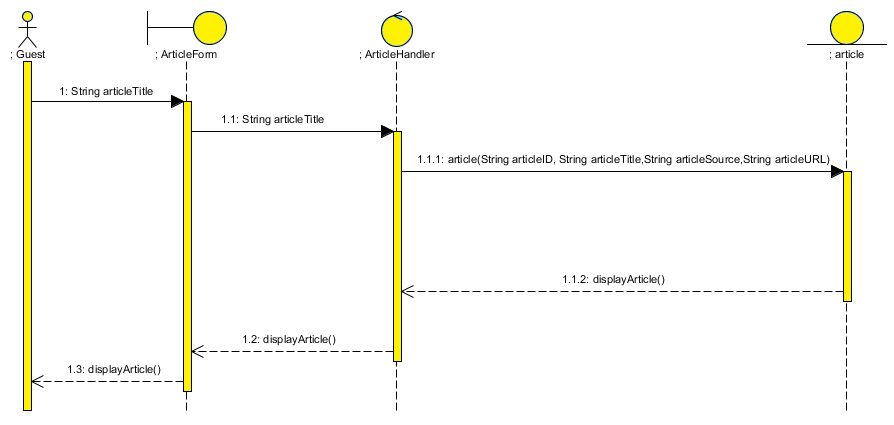
## 5.1.2.6 Bryan’s Sequence Diagram – Convert Weight (Kg/Lbs)



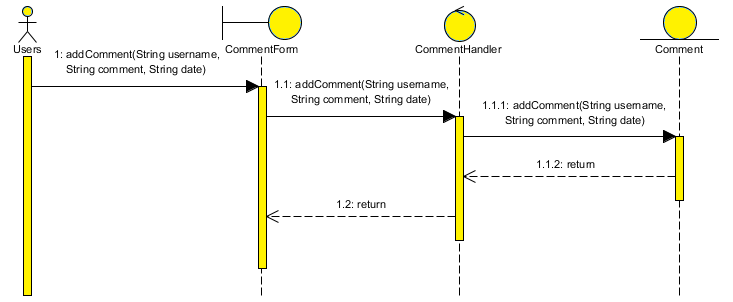
## 5.1.3.1 Benjamin’s Sequence Diagram - Advertisement



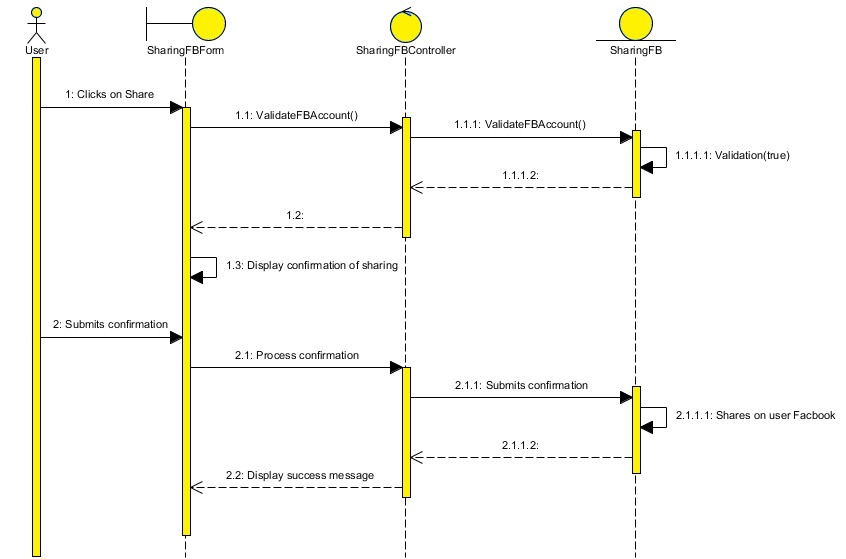
## 5.1.3.2 Benjamin’s Sequence Diagram - Article



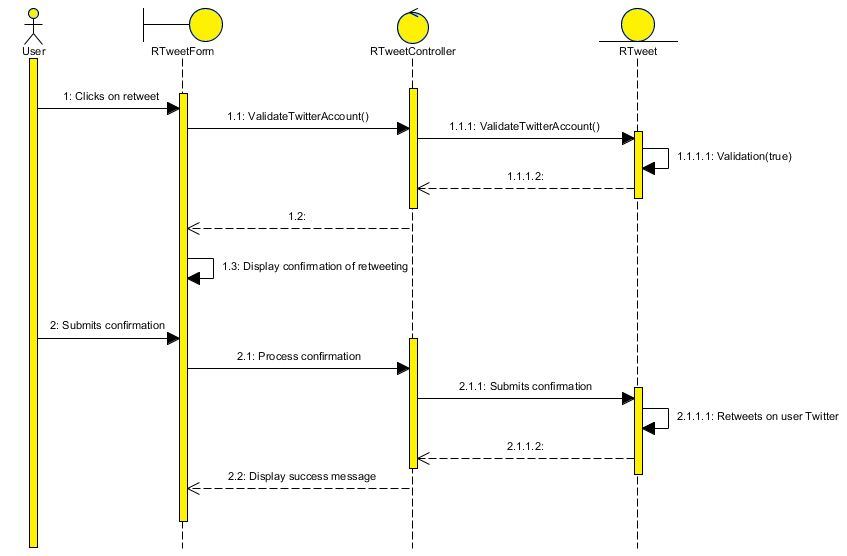
## 5.1.4.1 Terence’s Sequence Diagram - Comment



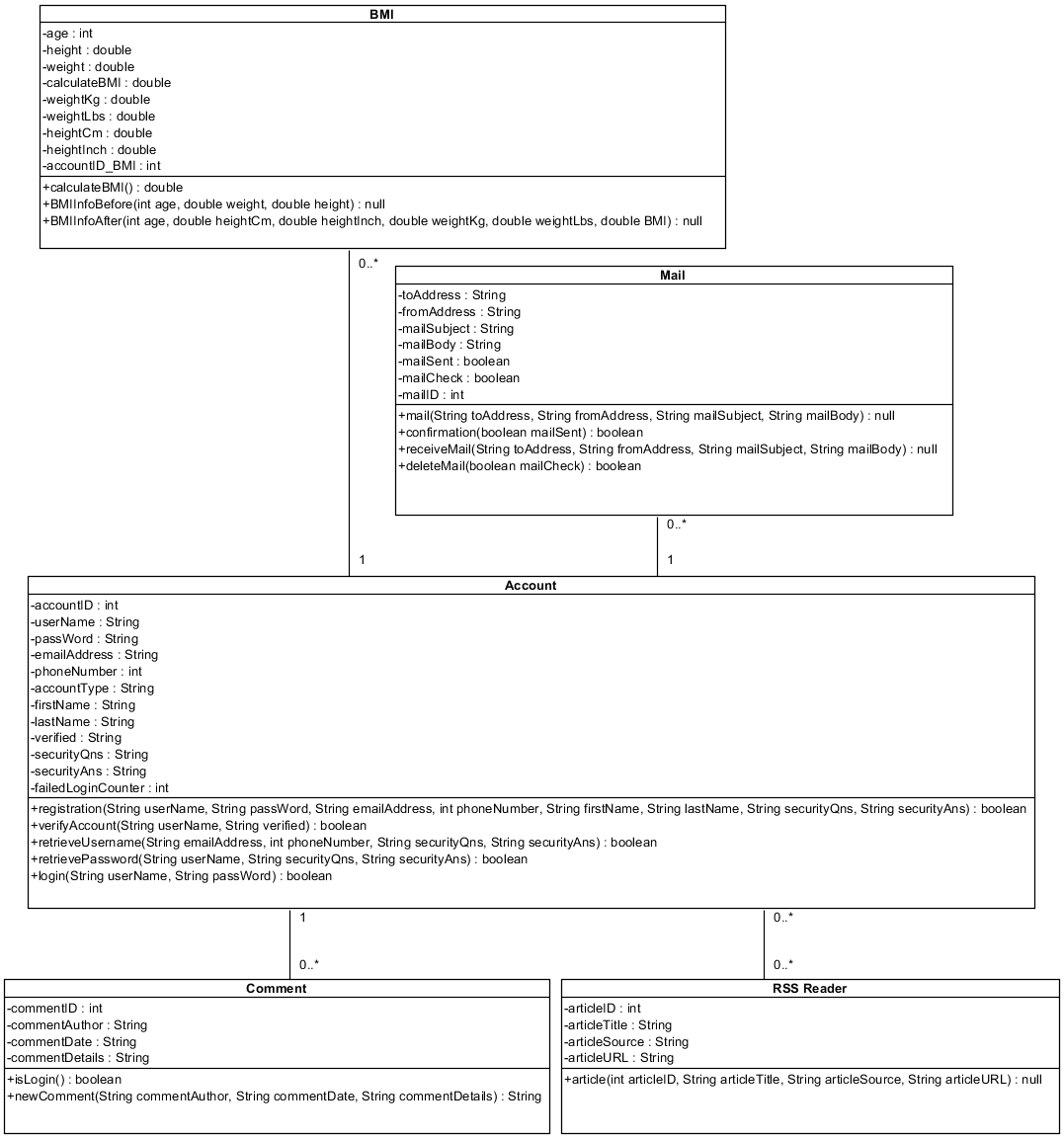
## 5.1.4.2 Terence’s Sequence Diagram – Sharing to Facebook



## 5.1.4.3 Terence’s Sequence Diagram – Retweet to Twitter

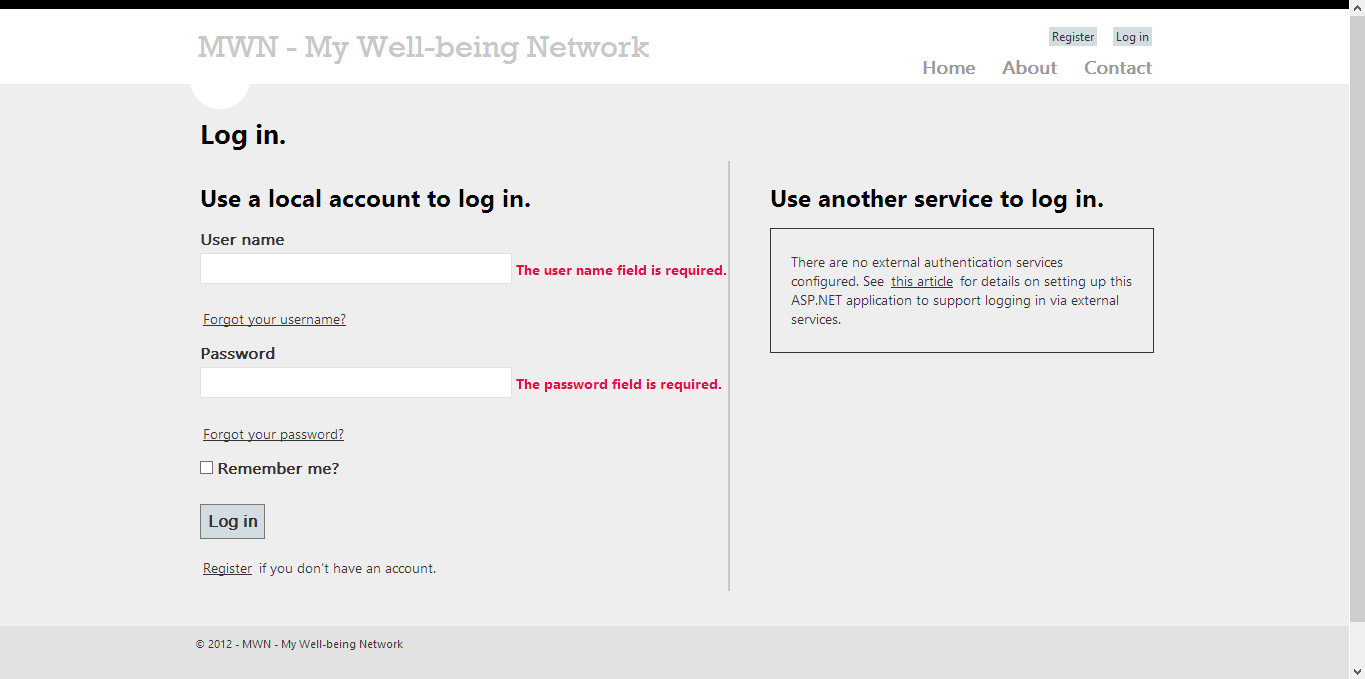


## 5.2 Design Class Diagrams

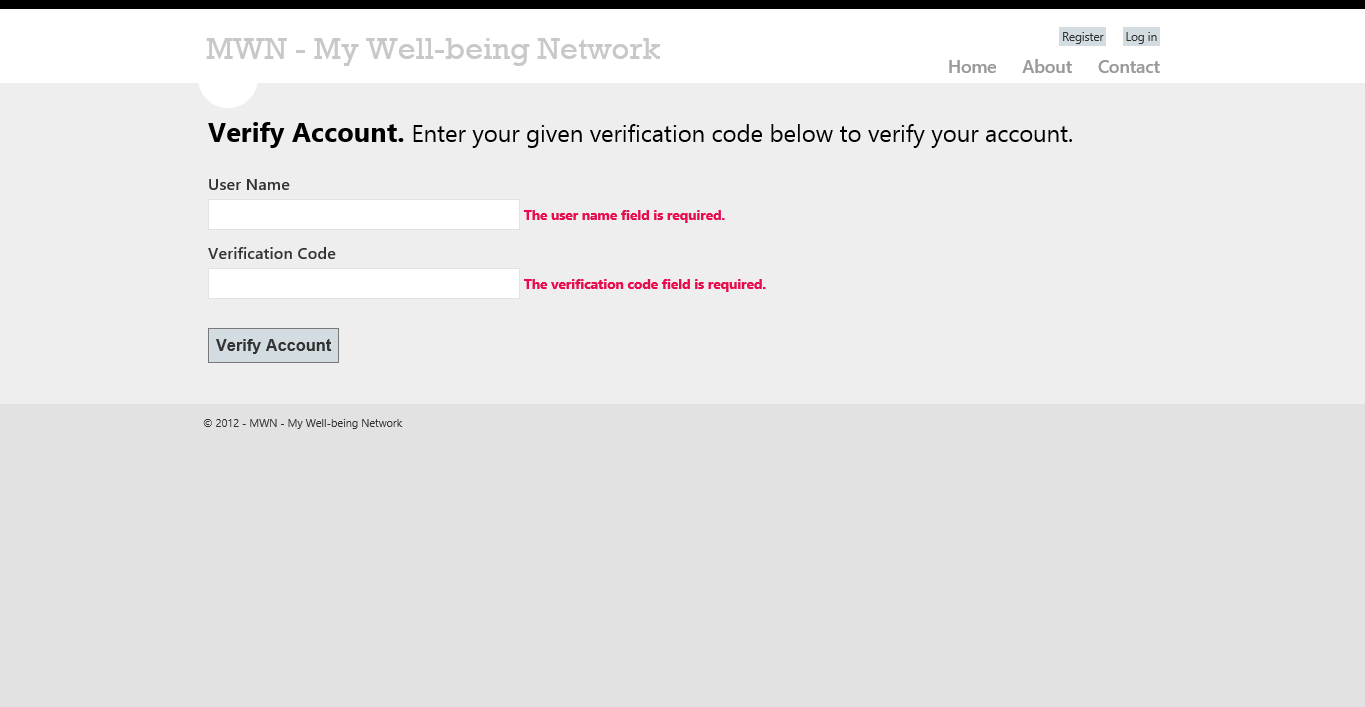


## 5.3 UI Prototypes

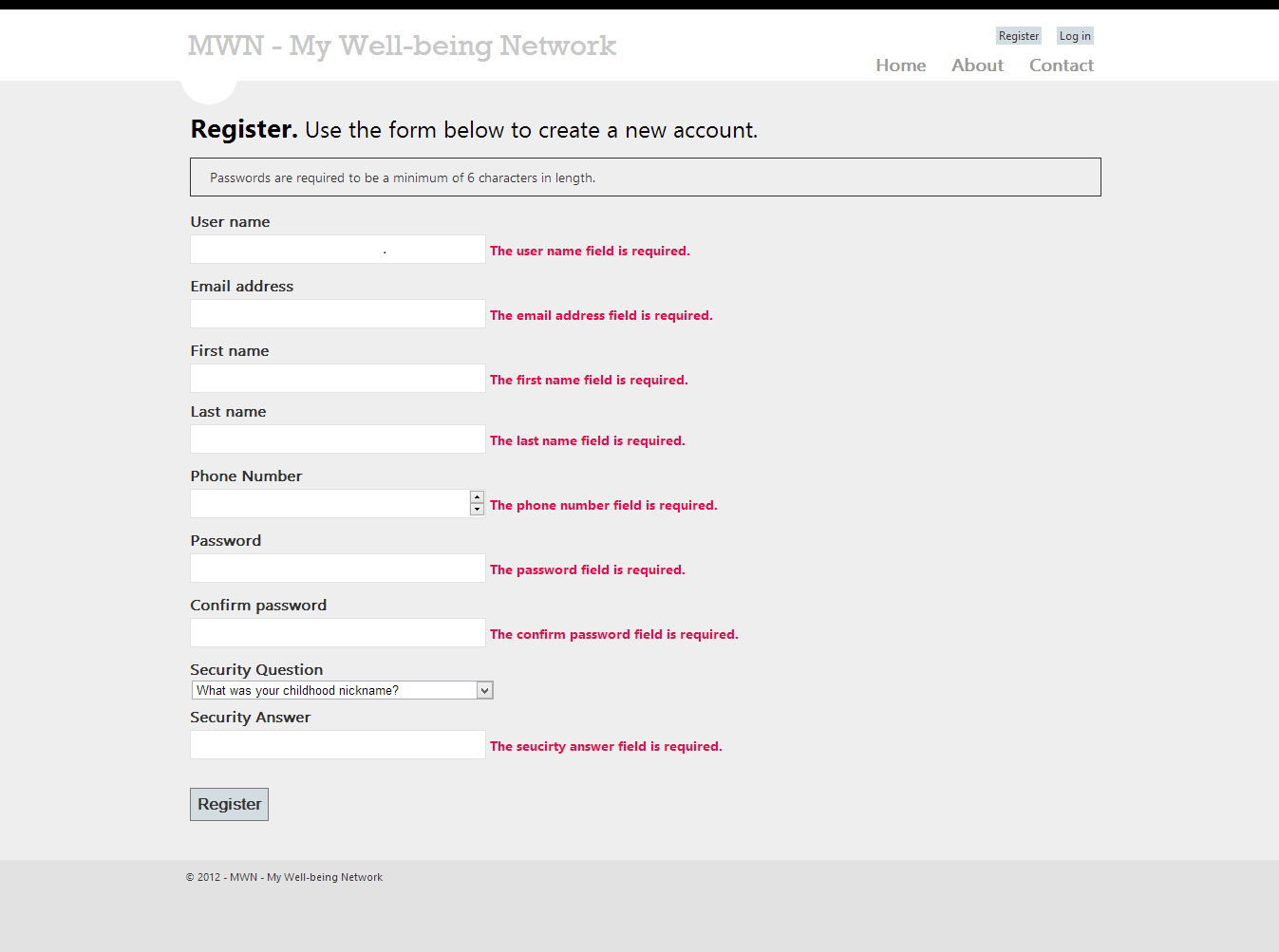
## 5.3.1.1 Alvin’s UI Prototypes – Login Prototype



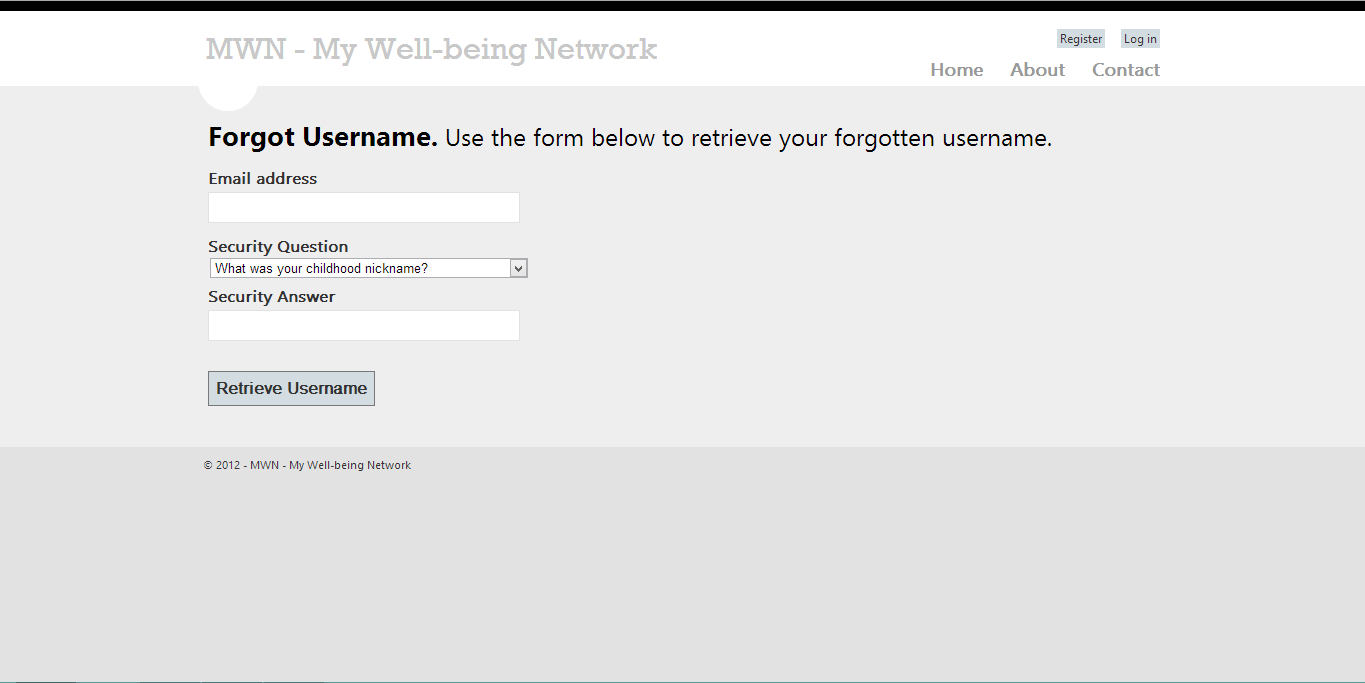
## 5.3.1.2 Alvin’s UI Prototypes – Verify Account Prototype



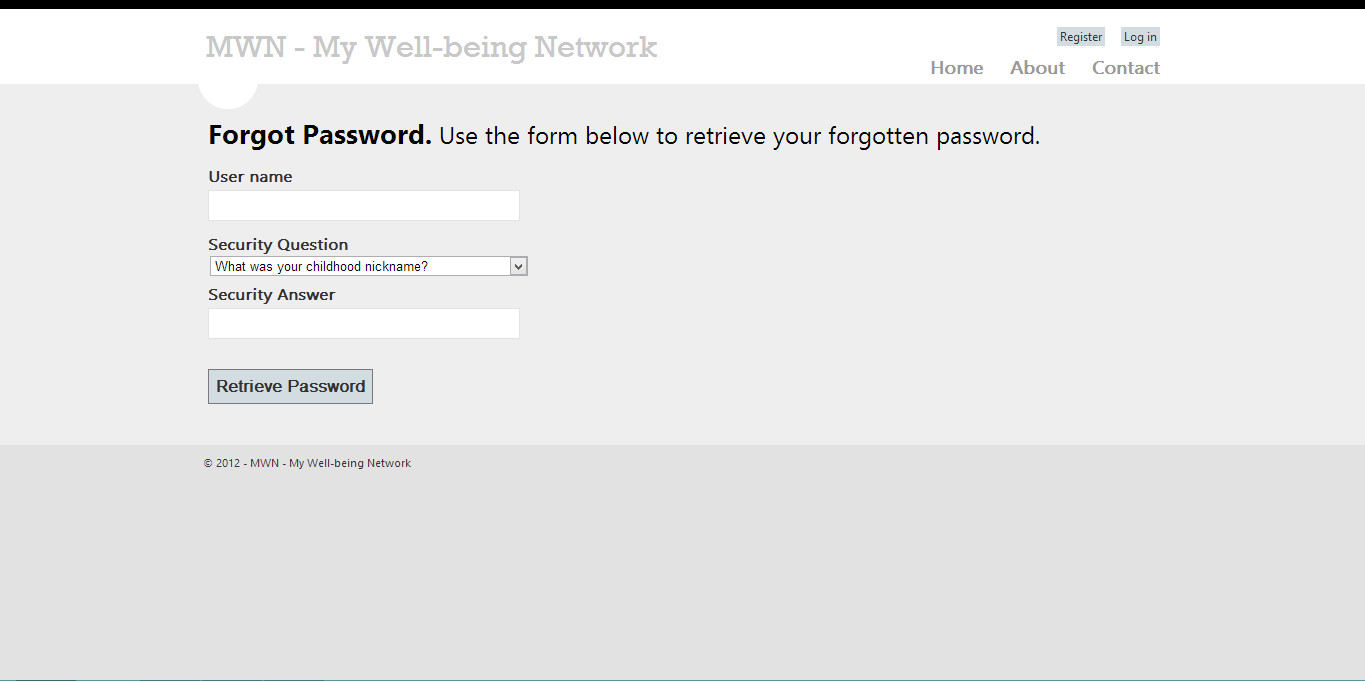
## 5.3.1.3 Alvin’s UI Prototypes – Register Prototype



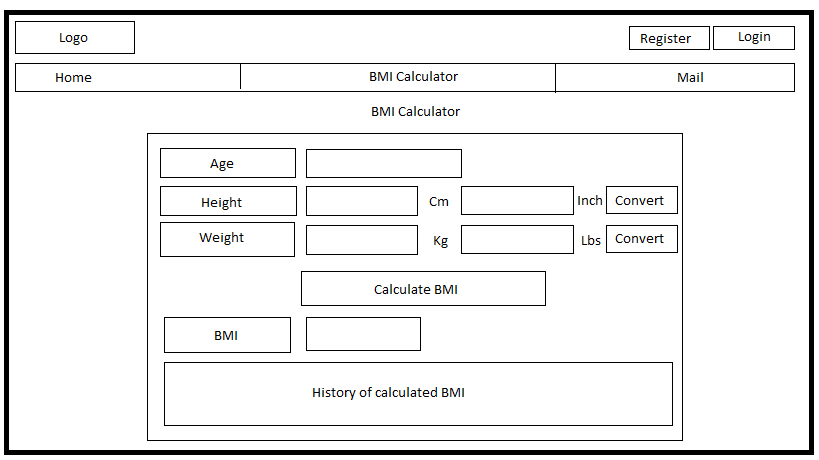
## 5.3.1.4 Alvin’s UI Prototypes – Forgot Username Prototype



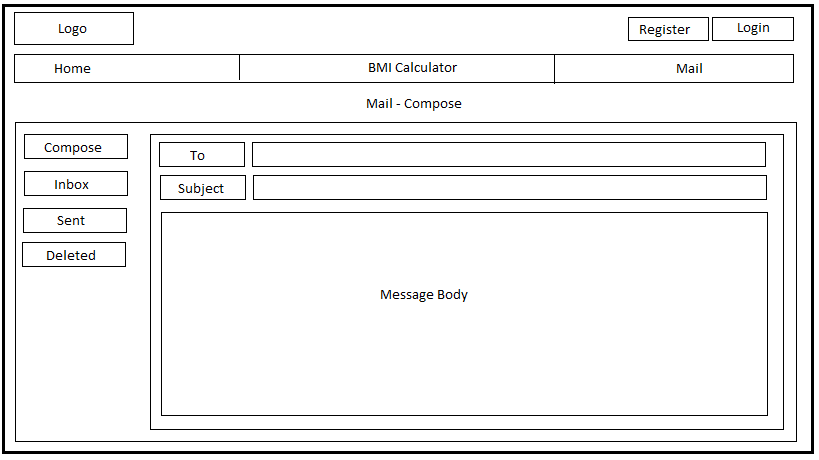
## 5.3.1.5 Alvin’s UI Prototypes – Forgot Password Prototype



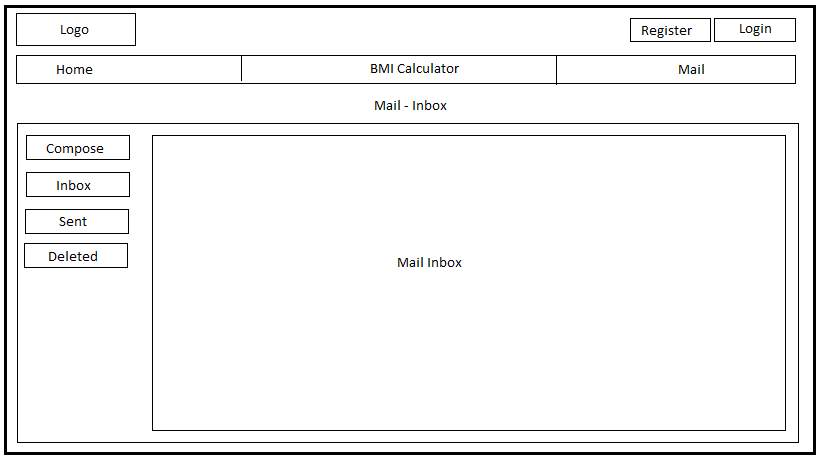
## 5.3.2.1 Bryan’s UI Prototypes – BMI Calculator Prototype



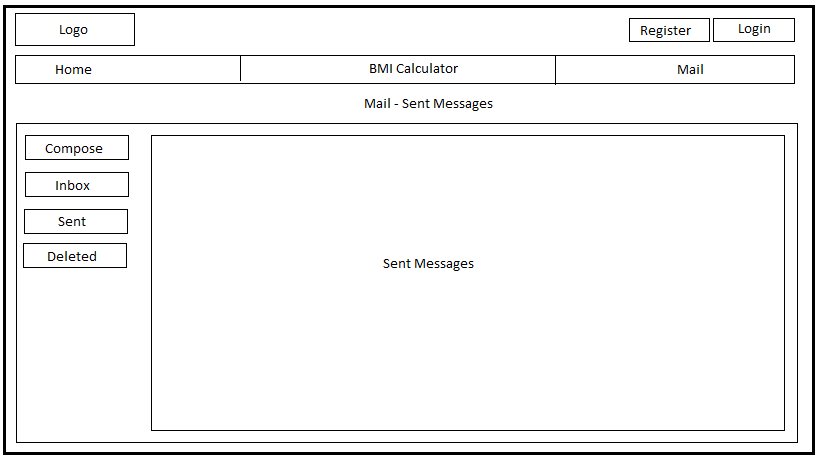
## 5.3.2.2 Bryan’s UI Prototypes – Compose Mail Prototype



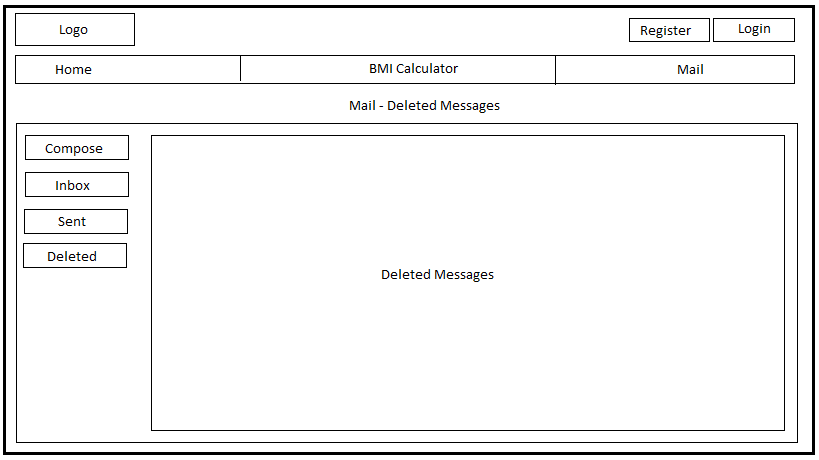
## 5.3.2.3 Bryan’s UI Prototypes – Mail Inbox Prototype



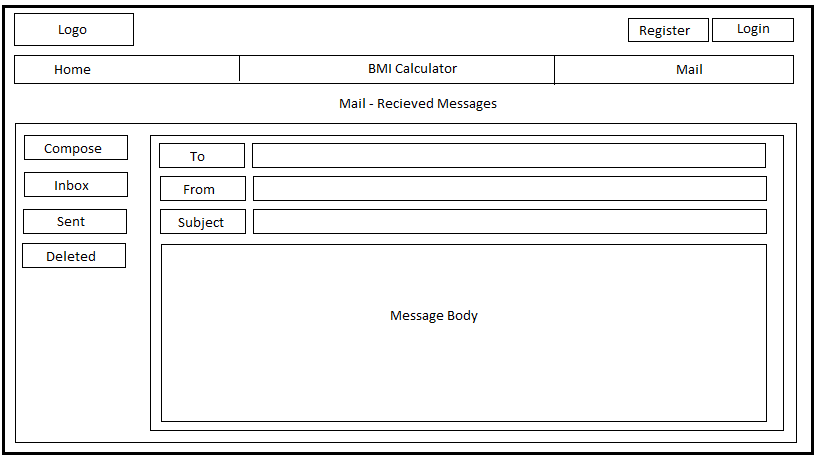
## 5.3.2.4 Bryan’s UI Prototypes – Mail Sent Messages Prototype



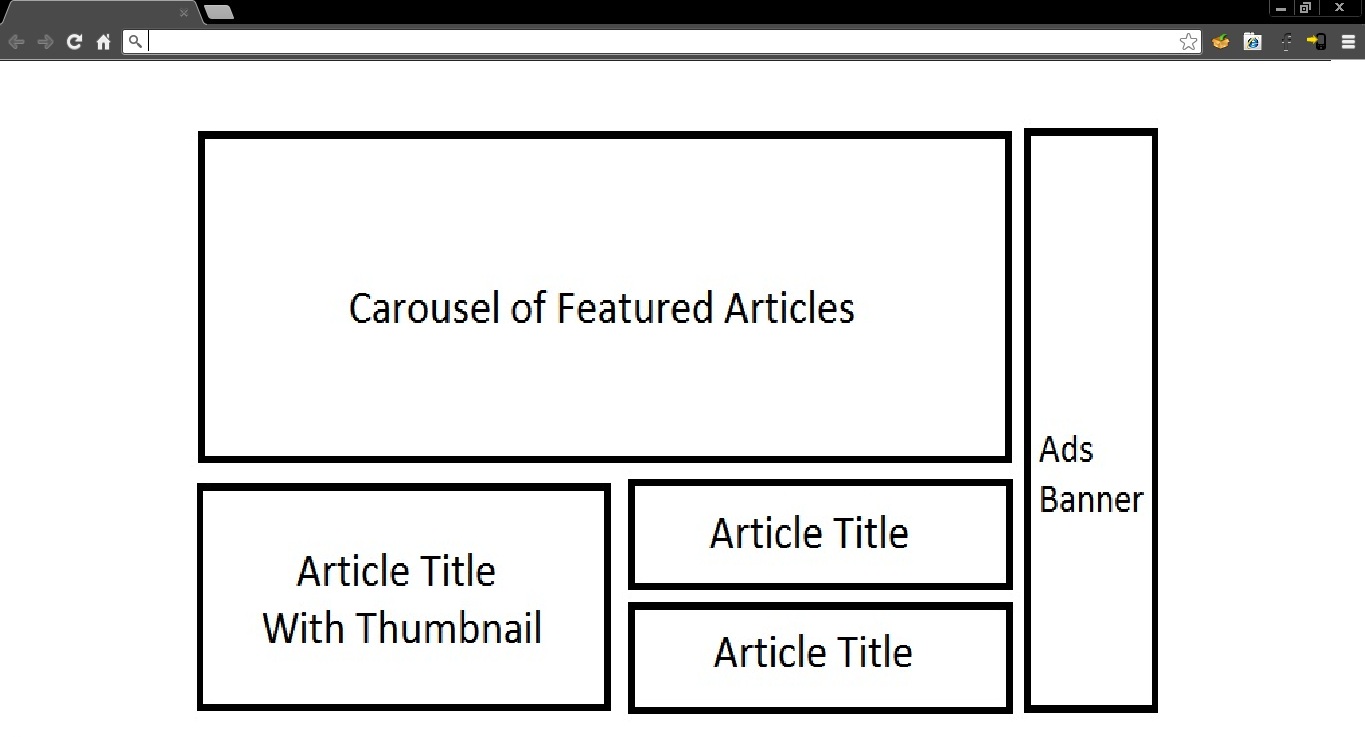
## 5.3.2.5 Bryan’s UI Prototypes – Mail Deleted Messages Prototype



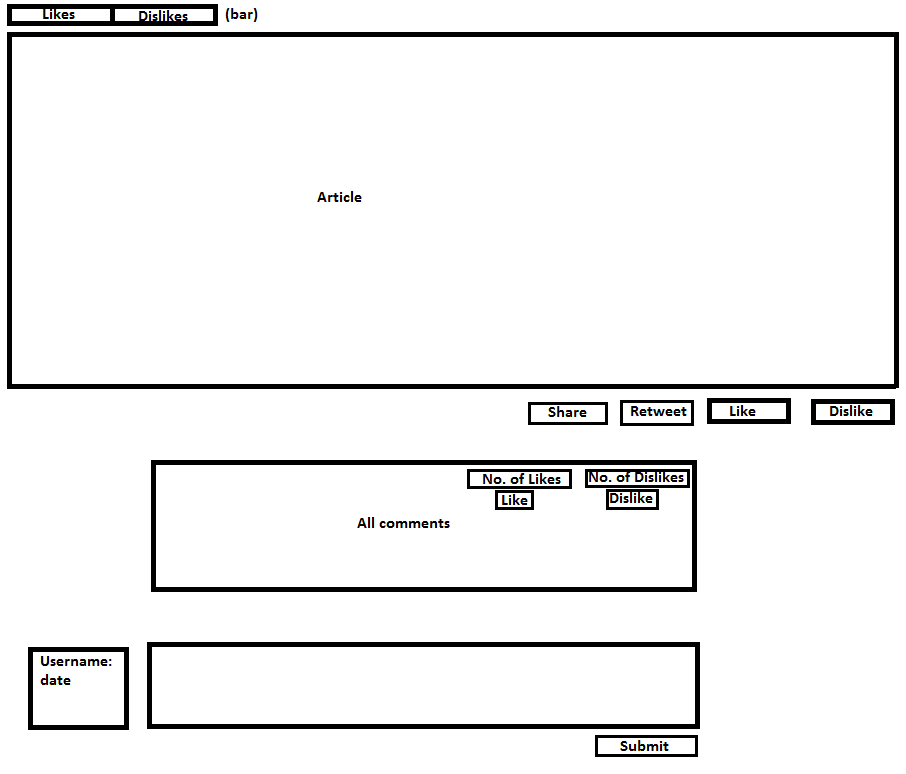
## 5.3.2.6 Bryan’s UI Prototypes – Mail Received Messages Prototype



## 5.3.3.1 Benjamin’s UI Prototypes – RSS Feeds Prototype



## 5.3.4.1 Terence’s UI Prototypes – Comment Page Prototype



# 6. Test Plan

## 6.1 Alvin’s Test Plan

## 6.1.1 Alvin’s Test Setup

1. Startup the web server of MWN (**SQL Workbench**)
2. Startup **Microsoft Visual Studio 2012** or **Microsoft Visual Studio 2010**
3. Run the MWN’s Project Solution file ***MWN.sln***
4. In the **Solution Explorer**, double click on the ***Homepage.aspx*** to open it up
5. Go to **DEBUG -> Start Debugging** or can also click on the green **play** button (Google Chrome/Internet Explorer)
6. ***Homepage.aspx*** will then start up and may proceed to do further testing

## 6.1.2 Alvin’s Test Procedures

1. **(Test Case 1 - 2)** Homepage -> **Login**
2. **(Test Case 3 - 7)** Homepage -> **Register**
3. **(Test Case 8 - 10)** Homepage -> Login -> **Forgot Username**
4. **(Test Case 11 - 13)** Homepage -> Login-> **Forgot Password**
5. **(Test Case 14 - 15)** Homepage -> Login-> Forgot Password -> **Change Password**
6. **(Test Case 16 - 18)** Homepage -> Login-> **Verify Account**
7. **(Test Case 19 - 21)** Homepage -> Login-> Verify Account -> **Resend Verify**

## 6.1.3 Alvin’s Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/N** | **Test Case** | **Input** | **Expected Output** | **Actual Output** | **Pass / Fail** |
| 1. | **Login:** Entered an invalid password (Username & Password do not match) | **Attributes:** Username, Password | **Error Message:** "Invalid Password, or account is not yet verified!” | **Error Message:** "Invalid Password, or account is not yet verified!” | **PASS** |
| 2. | **Login:** Entered an invalid password more than 5 times (Username & Password do not match) | **Attributes:** Username, Password | **Error Message:** "Invalid Password, or account is not yet verified!”  **Error Message:** "Unable to login. Please wait... \*timer\*” | **Error Message:** "Invalid Password, or account is not yet verified!”  **Error Message:** "Unable to login. Please wait... \*timer\*” | **PASS** |
| 3. | **Register:** Password and Confirm Password entered do not match | **Attributes:** Username, Password, ConfirmPassword, Email, Phone, FirstName, LastName, SecurityQns, SecurityAns, Captcha | **Error Message:** "Password mismatch. Please re-enter.” | **Error Message:** "Password mismatch. Please re-enter.” | **PASS** |
| 4. | **Register:** Username is not available in the database (Duplication of username in the database) | **Attributes:** Username, Password, ConfirmPassword, Email, Phone, FirstName, LastName, SecurityQns, SecurityAns, Captcha | **Error Message:** "Username is not available.” | **Error Message:** "Username is not available.” | **PASS** |
| 5. | **Register:** Email Address is not available in the database (Duplication of email address in the database) | **Attributes:** Username, Password, ConfirmPassword, Email, Phone, FirstName, LastName, SecurityQns, SecurityAns, Captcha | **Error Message:** "Email Address is not available.” | **Error Message:** "Email Address is not available.” | **PASS** |
| 6. | **Register:** Entered an invalid captcha | **Attributes:** Username, Password, ConfirmPassword, Email, Phone, FirstName, LastName, SecurityQns, SecurityAns, Captcha | **Error Message:** "Invalid Captcha. Please re-enter captcha.” | **Error Message:** "Invalid Captcha. Please re-enter captcha.” | **PASS** |
| 7. | **Register:** Successfully registered an account from the registration page | **Attributes:** Username, Password, ConfirmPassword, Email, Phone, FirstName, LastName, SecurityQns, SecurityAns, Captcha | **Successful Message:** "Successfully Registered Your Account! A Verification Code has been sent to your email."  **Database:**  Successfully created a new registered account.  **Successful Redirect:** VerifyAccount.aspx | **Successful Message:** "Successfully Registered Your Account! A Verification Code has been sent to your email."  **Database:**  Successfully created a new registered account.  **Successful Redirect:** VerifyAccount.aspx | **PASS** |
| 8. | **Forgot Username:** Entered an email address that is not found in the database | **Attributes:** Email, SecurityAns | **Error Message:** "Email Address is not available.” | **Error Message:** "Email Address is not available.” | **PASS** |
| 9. | **Forgot Username:** Entered an invalid security answer against the security question | **Attributes:** Email, SecurityAns | **Error Message:** "Invalid Security Answer. Please re-enter.” | **Error Message:** "Invalid Security Answer. Please re-enter.” | **PASS** |
| 10. | **Forgot Username:** Successfully retrieve forgotten username from the forgot username page | **Attributes:** Email, SecurityAns | **Successful Message:** "Your Username has been successfully sent to your email address!”  **Email Server:**  Successfully sent a retrieve username to their email address respectively.  **Successful Redirect:** Login.aspx | **Successful Message:** "Your Username has been successfully sent to your email address!”  **Email Server:**  Successfully sent a retrieve username to their email address respectively.  **Successful Redirect:** Login.aspx | **PASS** |
| 11. | **Forgot Password:** Entered an username that is not found in the database | **Attributes:** Username, SecurityAns | **Error Message:** "Username is not available.” | **Error Message:** "Username is not available.” | **PASS** |
| 12. | **Forgot Password:** Entered an invalid security answer against the security question | **Attributes:** Username, SecurityAns | **Error Message:** "Invalid Security Answer. Please re-enter.” | **Error Message:** "Invalid Security Answer. Please re-enter.” | **PASS** |
| 13. | **Forgot Password:** Successfully redirect to the change password page | **Attributes:** Email, SecurityAns | **Successful Redirect:** ChangePassword.aspx | **Successful Redirect:** ChangePassword.aspx | **PASS** |
| 14. | **Change Password:** Password and Confirm Password entered do not match | **Attributes:** Password, ConfirmPassword | **Error Message:** "Password mismatch. Please re-enter.” | **Error Message:** "Password mismatch. Please re-enter.” | **PASS** |
| 15. | **Change Password:** Password successfully changed | **Attributes:** Password, ConfirmPassword | **Successful Message:** "Successfully Changed Your Password!”  **Successful Redirect:** Login.aspx | **Successful Message:** "Successfully Changed Your Password!”  **Successful Redirect:** Login.aspx | **PASS** |
| 16. | **Verify Account:** Entered an username that is not found in the database | **Attributes:** Username, VerificationCode | **Error Message:** "Invalid Verification Code. Please re-enter.” | **Error Message:** "Invalid Verification Code. Please re-enter.” | **PASS** |
| 17. | **Verify Account:** Entered an invalid verification code that does not match with the username entered | **Attributes:** Username, VerificationCode | **Error Message:** "Invalid Verification Code. Please re-enter.” | **Error Message:** "Invalid Verification Code. Please re-enter.” | **PASS** |
| 18. | **Verify Account:** Correct verification code entered, account successfully verified | **Attributes:** Username, VerificationCode | **Successful Message:** "Successfully Verified Your Account!”  **Successful Redirect:** Login.aspx | **Successful Message:** "Successfully Verified Your Account!”  **Successful Redirect:** Login.aspx | **PASS** |
| 19. | **Resend Verify:** Entered an username that is not found in the database | **Attributes:** Username, SecurityAns | **Error Message:** "Username is not available.” | **Error Message:** "Username is not available.” | **PASS** |
| 20. | **Resend Verify:** Entered an invalid security answer against the security question | **Attributes:** Username, SecurityAns | **Error Message:** "Invalid Security Answer. Please re-enter.” | **Error Message:** "Invalid Security Answer. Please re-enter.” | **PASS** |
| 21. | **Resend Verify:** New generated verification code successfully send to user email address | **Attributes:** Username, SecurityAns | **Successful Message:** "A New Generated Verification Code has been sent to your email."  **Email Server:**  Successfully sent a new generated verification code to their email address respectively.  **Successful Redirect:** VerifyAccount.aspx | **Successful Message:** "A New Generated Verification Code has been sent to your email."  **Email Server:**  Successfully sent a new generated verification code to their email address respectively.  **Successful Redirect:** VerifyAccount.aspx | **PASS** |

## 6.2 Bryan’s Test Plan

## 6.2.1 Bryan’s Test Setup

1. Startup the web server of MWN (**SQL Workbench**)
2. Startup **Microsoft Visual Studio 2012** or **Microsoft Visual Studio 2010**
3. Run the MWN’s Project Solution file ***MWN.sln***
4. In the **Solution Explorer**, double click on the ***Homepage.aspx*** to open it up
5. Go to **DEBUG -> Start Debugging** or can also click on the green **play** button (Google Chrome/Internet Explorer)
6. ***Homepage.aspx*** will then start up and may proceed to do further testing

## 6.2.2 Bryan’s Test Procedures

1. **(Test Case 1 - 2)** Homepage -> Tools -> BMI Calculator
2. **(Test Case 3 - 5)** Homepage -> Tools -> Compose

## 6.2.3 Bryan’s Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/N** | **Test Case** | **Input** | **Expected Output** | **Actual Output** | **Pass / Fail** |
| 1. | **BMI Calculator:** No amount for weight entered | **Attributes:** Weight | **Error Message:** "Please enter a weight!" | **No Error Message** | **PASS** |
| 2. | **BMI Calculator:** No amount for height entered | **Attributes:** Height | **Error Message:** "Please enter a height!" | **No Error Message** | **PASS** |
| 3. | **Private messaging system (Compose):** No receiver name entered | **Attributes:** Receiver | **Error Message:** "Please enter the receiver name!" | **No Error Message Redirect:** Compose page | **PASS** |
| 4. | **Private messaging system (Compose):** No subject entered | **Attributes:** Subject | **Error Message:** "Please enter the subject of the mail!" | **No Error Message**  **Redirect:** Compose page | **PASS** |
| 5. | **Private mail messaging system (Compose):** No message body entered | **Attributes:** Body | **Error Message:** "Please enter the message body of the mail!" | **No Error Message Redirect:** Compose page | **PASS** |

## 6.3 Benjamin’s Test Plan

## 6.3.1 Benjamin’s Test Setup

1. Startup the web server of MWN (**SQL Workbench**)
2. Startup **Microsoft Visual Studio 2012** or **Microsoft Visual Studio 2010**
3. Run the MWN’s Project Solution file ***MWN.sln***
4. In the **Solution Explorer**, double click on the ***Homepage.aspx*** to open it up
5. Go to **DEBUG -> Start Debugging** or can also click on the green **play** button (Google Chrome/Internet Explorer)
6. ***Homepage.aspx*** will then start up and may proceed to do further testing

## 6.3.2 Benjamin’s Test Procedures

1. **(Test Case 1)** When user clicks on an article title on the carousel, he/she will be redirected to that specific article.
2. **(Test Case 2)** When user clicks on an article title in the RSS Reader, he/she will be redirected to that specific article's website.
3. **(Test Case 3)** When user clicks on an ad, a popup notice will inform the user that he/she is being redirected.
4. **(Test Case 4)** When user acknowledges that he/she is leaving the website, he/she will be redirected to the webpage of the specific ad he/she clicked on.
5. **(Test Case 5)** When user drags down "Go Premium" ribbon, it displays an overscreen shade stating features of being a premium user.
6. **(Test Case 6)** When clicks on "Become Premium User" button, it redirects him to premium user application form.

## 6.3.3 Benjamin’s Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/N** | **Test Case** | **Input** | **Expected Output** | **Actual Output** | **Pass / Fail** |
| 1. | **News Carousel:** When user clicks on article | **Attributes:** (Link to article) | **Redirect:** Article page. | **Redirect:** Article page. | **PASS** |
| 2. | **RSS Reader:** When user clicks on article | **Attributes:** (Link to external URL) | **Redirect:** External article page. | **Redirect:** External article page. | **PASS** |
| 3. | **Ad banner:** When user clicks on ad banner | **Attributes:** (Link to external URL) | **Popup Notice:** "You are leaving MWN." | **Popup :** "You are leaving MWN." | **PASS** |
| 4. | **Ad banner:** When user clicks "Okay" in the popup notice | **Attributes:** -NIL- | **Redirect:** External ad website. | **Redirect:** External ad website. | **PASS** |
| 5. | **Premium User Ribbon:** When user drags down premium user ribbon | **Attributes:** -NIL- | **Overscreen shade:** Displaying features of being a premium user. | **Overscreen shade:** Displaying features of being a premium user. | **PASS** |
| 6. | **Premium User Ribbon:**  When user clicks on "Become premium user" button | **Attributes:** -NIL- | **Redirect:** Premium user application page | **Redirect:** Premium user application page | **PASS** |

## 6.4 Terence’s Test Plan

## 6.4.1 Terence’s Test Setup

1. Startup the web server of MWN (**SQL Workbench**)
2. Startup **Microsoft Visual Studio 2012** or **Microsoft Visual Studio 2010**
3. Run the MWN’s Project Solution file ***MWN.sln***
4. In the **Solution Explorer**, double click on the ***Homepage.aspx*** to open it up
5. Go to **DEBUG -> Start Debugging** or can also click on the green **play** button (Google Chrome/Internet Explorer)
6. ***Homepage.aspx*** will then start up and may proceed to do further testing

## 6.4.2 Terence’s Test Procedures

1. **(Test Case 1)** When user registers, password is hashed with a randomly generated key and both the hashed password and key is stored in database.
2. **(Test Case 2)** When user logs in, user-entered-password is hashed with key retrieved from database, which will then be compared with hashed password from database. User is able to log in.
3. **(Test Case 3)** When user types into Facebook API to comment, the comment is displayed.

## 6.4.3 Terence’s Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/N** | **Test Case** | **Input** | **Expected Output** | **Actual Output** | **Pass / Fail** |
| 1. | When user registers for a new account | User inputs desired password | Key generated and password is hashed. Both key and hashed password is stored in database. | Key generated and password is hashed. Both key and hashed password is stored in database. **Characters are recognizable.** | **PASS** |
| 2. | When user registers for a new account | User inputs desired password | Key generated and password is hashed. Both key and hashed password is stored in database. | Key generated and password is hashed. Both key and hashed password is stored in database **but characters are unrecognized due to character conversions.** | **FAIL** |
| 3. | When user tries to log in to account | User inputs password | User-entered-password is hashed with key retrieved from database, which will then be compared with hashed password from database. User is able to log in. | User-entered-password is hashed with key retrieved from database, which will then be compared with hashed password from database. **User is unable to log in due to unknown characters retrieved from database.** | **FAIL** |
| 4. | When user tries to log in to account | User inputs password | User-entered-password is hashed with key retrieved from database, which will then be compared with hashed password from database. User is able to log in. | User-entered-password is hashed with key retrieved from database, which will then be compared with hashed password from database. **User is able to log in due to edited hashing algorithm.** | **PASS** |
| 5. | User comments on article through Facebook API | User logs into Facebook account and comments on article. | User able to comment after logging in to Facebook. | User able to comment after logging in to Facebook. | **PASS** |

# 7. Conclusion

## 7.1 Summary of Completed Tasks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/N** | **Task** | **Alvin Lee** | **Bryan Lim** | **Benjamin Wong** | **Terence Lam** |
| 1 | Login System | ✓ |  |  |  |
| 2 | On Screen Keyboard | ✓ |  |  |  |
| 3 | Two-Factor Authentication (2FA) | ✓ |  |  |  |
| 4 | Multiple Invalid Login Timer | ✓ |  |  |  |
| 5 | BMI Calculator |  | ✓ |  |  |
| 6 | Private Messaging System |  | ✓ |  |  |
| 7 | Captcha |  | ✓ |  |  |
| 8 | Advertisement System |  |  | ✓ |  |
| 9 | RSS Feeds |  |  | ✓ |  |
| 10 | XML Encryption |  |  | ✓ |  |
| 11 | Article Feeds Banner |  |  | ✓ |  |
| 12 | Facebook Comment System |  |  |  | ✓ |
| 13 | Sharing to Facebook |  |  |  | ✓ |
| 14 | Like to Facebook |  |  |  | ✓ |
| 15 | Retweet to Twitter |  |  |  | ✓ |
| 16 | Feedback Survey to be Premium |  |  |  | ✓ |
| 17 | SQL Parameterized Queries | ✓ |  |  |  |
| 18 | HTTPS |  | ✓ |  |  |
| 19 | Unvalidated Redirects (Phishing) |  |  | ✓ |  |
| 20 | Salt Hashing |  |  |  | ✓ |
| 21 | Integration | ✓ |  |  |  |
| 22 | SQL Database Design |  | ✓ |  |  |
| 23 | Overall Project Design |  |  | ✓ |  |
| 24 | Information Gathering |  |  |  | ✓ |

\*S/N 1 to 16 (Main Content & Security Features), 17 to 20 (Secure-coding), 21 to 24 (Backend)

## 7.2 Summary of Future Enhancements

|  |  |
| --- | --- |
| **S/N** | **Future Enhancements** |
| 1 | Enhance overall website UI design |
| 2 | Bug fixes (If any) |
| 3 | More health related information in our website |
| 4 | More diet related information in our website |
| 5 | More advertisements |
| 6 | More security vulnerabilities are fixed in the fortify reports |
| 7 | More tools and business features to enhance our website |
| 8 | Azure Cloud Hosting |
| 9 | More security features to let our users feel safe |
| 10 | Administrative panel to manage other users instead of using database |
| 11 | Users stand to win prizes for every clicks in the advertisement banners |
| 12 | Games related to health and diet to let user learn and gain knowledge from it |

## 7.3 Lessons Learnt

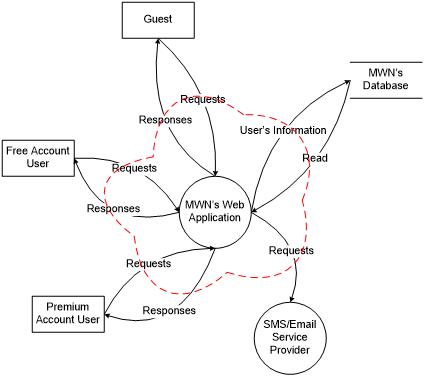
Through the project we have expanded our knowledge about web technology as well as how to protect our web applications from attacks. We have a good understanding of basic security measures as well as knowledge of why do we need to provide these services when working online with our customer and sending sensitive information and money.

# Appendix A - Threat Modeling report

|  |
| --- |
|  |

**Data Flow Diagrams**

**Context**



**Threats and Mitigations**

**Elements**

| **Element Type** | **Description** |
| --- | --- |
| Data Flow | Read |
| Data Flow | Requests |
| Data Flow | Requests |
| Data Flow | Requests |
| Data Flow | Requests |
| Data Flow | Responses |
| Data Flow | Responses |
| Data Flow | Responses |
| Data Flow | User’s Information |
| Data Store | MWN’s Database |
| Interactor | Free Account User |
| Interactor | Guest |
| Interactor | Premium Account User |
| Process | MWN’s Web Application |
| Process | SMS/Email Service Provider |
| TrustBoundary |  |
| TrustBoundary |  |
| TrustBoundary |  |
| TrustBoundary |  |
|  |  |

**External Interactors**

**Threats against Free Account User**

**Spoofing (Threat #33)**

**Threat:** The user would be able to gain priviledges it was otherwise denied.

**Mitigation:** Using strict authentication for example form validations, preventing SQL injection; and access control.

**Repudiation (Threat #34)**

**Threat:** If the user is able to retract what he/she says or sent, then the data has no integrity and it would be harder to track the culprit.

**Mitigation:** By using digital signatures, the parties will be able to ensure that the information is non-repudiated.

**Threats against Guest**

**Spoofing (Threat #1)**

**Threat:** The user would be able to gain priviledges it was otherwise denied.

**Mitigation:** Using strict authentication for example form validations, preventing SQL injection; and access control.

**Repudiation (Threat #2)**

**Threat:** If the user is able to retract what he/she says or sent, then the data has no integrity and it would be harder to track the culprit.

**Mitigation:** By using digital signatures, the parties will be able to ensure that the information is non-repudiated.

**Threats against Premium Account User**

**Spoofing (Threat #35)**

**Threat:** The user would be able to gain priviledges it was otherwise denied.

**Mitigation:** Using strict authentication for example form validations, preventing SQL injection; and access control.

**Repudiation (Threat #36)**

**Threat:** If the user is able to retract what he/she says or sent, then the data has no integrity and it would be harder to track the culprit.

**Mitigation:** By using digital signatures, the parties will be able to ensure that the information is non-repudiated.

**Processes**

**Threats against MWN’s Web Application**

**Spoofing (Threat #15)**

**Threat:** Unauthorised access to data in the system

**Mitigation:** Ensure that the user is acutally who he claims to be

**Tampering (Threat #16)**

**Threat:** Attackers can tamper the database data by using brute force attacks

**Mitigation:** Validate all input from the user and use minimal error information messages to limit the error encountered

**Repudiation (Threat #17)**

**Threat:** Data provided to the users is incorrect

**Mitigation:** Giving users as little privilege as possible to do their things and logging user information

**Information Disclosure (Threat #18)**

**Threat:** Information disclosed to other parties by an unauthorised party

**Mitigation:** Encrypting the data and validate all input from the user

**Denial of Service (Threat #19)**

**Threat:** System not being available to the users

**Mitigation:** Multiple invalid login timer

**Elevation of Privilege (Threat #20)**

**Threat:** Unauthorised access and changes to the system

**Mitigation:** Validate all input given by the user using a whitelist

**Threats against SMS/Email Service Provider**

**Spoofing (Threat #25)**

**Threat:** Unauthorised access to data in the system

**Mitigation:** Ensure that the user is acutally who he claims to be

**Tampering (Threat #26)**

**Threat:** Attackers can tamper the database data by using brute force attacks

**Mitigation:** Validate all input from the user and use minimal error information messages to limit the error encountered

**Repudiation (Threat #27)**

**Threat:** Data provided to the users is incorrect

**Mitigation:** Giving users as little privilege as possible to do their things and logging user information

**Information Disclosure (Threat #28)**

**Threat:** Information disclosed to other parties by an unauthorised party

**Mitigation:** Encrypting the data and validate all input from the user

**Denial of Service (Threat #29)**

**Threat:** System not being available to the users

**Mitigation:** Limit the rate of people accessing the system

**Elevation of Privilege (Threat #30)**

**Threat:** Unauthorised access and changes to the system

**Mitigation:** Validate all input given by the user using a whitelist

**Data Flows**

**Threats against Read**

**Tampering (Threat #12)**

**Threat:** Tampering will impact this Web Application as information and data being transmitted can be modified by a man in the middle and wrong information will be displayed.

**Mitigation:** Ensure that the data is untampered by encrypting the data. We can also keep backups of the data to ensure quick roll back in the case of tampering.

**Information Disclosure (Threat #13)**

**Threat:** The impact of Information Disclosure is that sensitive information that belongs to our users will be accessible by attackers. This will make our web application lose it's integrity as well as the trust of our users.

**Mitigation:** A solution to the information disclosure is to encrypt the data, that way, even if attackers get a hold of this sensitive data, they would only get the encrypted text instead of the plain text.

**Denial of Service (Threat #14)**

**Threat:** Denial of Service can affect accesibility as the web application will not be able to display the data that our users want to view.

**Mitigation:** Have a validation system to prevent buffer overflow or spamming as well as deny any unknown services.

**Threats against Requests**

**Tampering (Threat #3)**

**Threat:** While sending data from Guest's access point of view to MWN's Web Application, attackers can tamper the data being transmitted and thus, can be modified by a man-in-themiddle attack, causing incorrect information received.

**Mitigation:** To prevent such information to be a victim of man-in-the-middle attack, we have to improve the integrity of the messages between the Guest and to MWN's Web Application. To do so, we will require to compute the hash values of the data that we would like to preserve the integrity for, so that by comparing the hash values, we are able to check whether the messages are still intact and not modified in any point of transmission.

**Information Disclosure (Threat #4)**

**Threat:** When all the information are leaked, confidential information such as the personal particulars of our users will be accessible by unauthorised party. Reputation of MWN's Web Application will be greatly affected as well, thus losing the trust of our users.

**Mitigation:** Main solution to solve such issues is to encrypt the confidential data of our users, or any other data that are meant to be confidential. Hashing may also solve such issue to be available as plain text to attackers when our information were disclosed.

**Denial of Service (Threat #5)**

**Threat:** MWN's Web Application services or webpages would not be available and accessible by our Guest when they visits our Web Application. This would cause our Guest to mistaken that our Web Application was not available for the entire period, and not temporary.

**Mitigation:** Having schedule maintenance such as every week of the same day, same hour, MWN's Web Application will be down for 2 hours for maintenance. We can also validate any input to prevent any buffer overflow attacks that would cause denial of service.

**Threats against Requests**

**Tampering (Threat #52)**

**Threat:** While sending data from Premium Account User's access point of view to MWN's Web Application, attackers can tamper the data being transmitted and thus, can be modified by a man-in-themiddle attack, causing incorrect information received.

**Mitigation:** To prevent such information to be a victim of man-in-the-middle attack, we have to improve the integrity of the messages between the Premium Account User and to MWN's Web Application. To do so, we will require to compute the hash values of the data that we would like to preserve the integrity for, so that by comparing the hash values, we are able to check whether the messages are still intact and not modified in any point of transmission.

**Information Disclosure (Threat #53)**

**Threat:** When all the information are leaked, confidential information such as the personal particulars of our users will be accessible by unauthorised party. Reputation of MWN's Web Application will be greatly affected as well, thus losing the trust of our users.

**Mitigation:** Main solution to solve such issues is to encrypt the confidential data of our users, or any other data that are meant to be confidential. Hashing may also solve such issue to be available as plain text to attackers when our information were disclosed.

**Denial of Service (Threat #54)**

**Threat:** MWN's Web Application services or webpages would not be available and accessible by our Premium Account User when they visits our Web Application. This would cause our Premium Account User to mistaken that our Web Application was not available for the entire period, and not temporary.

**Mitigation:** Having schedule maintenance such as every week of the same day, same hour, MWN's Web Application will be down for 2 hours for maintenance. We can also validate any input to prevent any buffer overflow attacks that would cause denial of service.

**Threats against Requests**

**Tampering (Threat #43)**

**Threat:** While sending data from Free Account User's access point of view to MWN's Web Application, attackers can tamper the data being transmitted and thus, can be modified by a man-in-themiddle attack, causing incorrect information received.

**Mitigation:** To prevent such information to be a victim of man-in-the-middle attack, we have to improve the integrity of the messages between the Premium Account User and to MWN's Web Application. To do so, we will require to compute the hash values of the data that we would like to preserve the integrity for, so that by comparing the hash values, we are able to check whether the messages are still intact and not modified in any point of transmission.

**Information Disclosure (Threat #44)**

**Threat:** When all the information are leaked, confidential information such as the personal particulars of our users will be accessible by unauthorised party. Reputation of MWN's Web Application will be greatly affected as well, thus losing the trust of our users.

**Mitigation:** Main solution to solve such issues is to encrypt the confidential data of our users, or any other data that are meant to be confidential. Hashing may also solve such issue to be available as plain text to attackers when our information were disclosed.

**Denial of Service (Threat #45)**

**Threat:** MWN's Web Application services or webpages would not be available and accessible by our Free Account User when they visits our Web Application. This would cause our Free Account User to mistaken that our Web Application was not available for the entire period, and not temporary.

**Mitigation:** Having schedule maintenance such as every week of the same day, same hour, MWN's Web Application will be down for 2 hours for maintenance. We can also validate any input to prevent any buffer overflow attacks that would cause denial of service.

**Threats against Requests**

**Tampering (Threat #58)**

**Threat:** Attackers can tamper the data being transmitted from MWN's Web Application to SMS/Email Service Provider, causing the data being transmitted to be modified by a man-in-themiddle attack, thus causing incorrect information received.

**Mitigation:** To prevent such information to be a victim of man-in-the-middle attack, we have to improve the integrity of the messages between the MWN's Web Application and SMS/Email Service Provider. To do so, we will require to compute the hash values of the data that we would like to preserve the integrity for, so that by comparing the hash values, we are able to check whether the messages are still intact and not modified in any point of transmission.

**Information Disclosure (Threat #59)**

**Threat:** When all the information are leaked, confidential information such as the verification code, random generated password of the users will be accessible by unauthorised party. Reputation of MWN's Web Application will be greatly affected as well, thus losing the trust of our users.

**Mitigation:** Main solution to solve such issues is to encrypt the confidential data of our users, or any other data that are meant to be confidential. Hashing may also solve such issue to be available as plain text to attackers when our information were disclosed.

**Denial of Service (Threat #60)**

**Threat:** When the SMS/Email Service Provider is not available and accessible when our Web Application requests it, it will cause our features, that uses SMS/Email Service, unable to work perfectly as it should be.

**Mitigation:** Having schedule maintenance such as every week of the same day, same hour, MWN's Web Application will be down for 2 hours for maintenance. Under this period of maintenance, we would try to figure out the issue causing the denial of service of the SMS/Email Service Provider, and thus solving those availability and accessibility problems.

**Threats against Responses**

**Tampering (Threat #6)**

**Threat:** Tampering will impact this Web Application as information and data being transmitted can be modified by a man in the middle and wrong information will be provided to the user.

**Mitigation:** Ensure that the data is untampered by encrypting the data. We can also keep backups of the data to ensure quick roll back in the case of tampering.

**Information Disclosure (Threat #7)**

**Threat:** The impact of Information Disclosure is that sensitive information that belongs to our users will be accessible by attackers. This will make our web application lose it's integrity as well as the trust of our users.

**Mitigation:** A solution to the information disclosure is to encrypt the data, that way, even if attackers get a hold of this sensitive data, they would only get the encrypted text instead of the plain text.

**Denial of Service (Threat #8)**

**Threat:** Denial of Service can affect accesibility as the web application will not be able to display the data that our users want to view.

**Mitigation:** Have a validation system to prevent buffer overflow or spamming as well as deny any unknown services.

**Threats against Responses**

**Tampering (Threat #46)**

**Threat:** Tampering will impact this Web Application as information and data being transmitted can be modified by a man in the middle and wrong information will be provided to the user.

**Mitigation:** Ensure that the data is untampered by encrypting the data. We can also keep backups of the data to ensure quick roll back in the case of tampering.

**Information Disclosure (Threat #47)**

**Threat:** The impact of Information Disclosure is that sensitive information that belongs to our users will be accessible by attackers. This will make our web application lose it's integrity as well as the trust of our users.

**Mitigation:** A solution to the information disclosure is to encrypt the data, that way, even if attackers get a hold of this sensitive data, they would only get the encrypted text instead of the plain text.

**Denial of Service (Threat #48)**

**Threat:** Denial of Service can affect accesibility as the web application will not be able to display the data that our users want to view.

**Mitigation:** Have a validation system to prevent buffer overflow or spamming as well as deny any unknown services.

**Threats against Responses**

**Tampering (Threat #55)**

**Threat:** Tampering will impact this Web Application as information and data being transmitted can be modified by a man in the middle and wrong information will be provided to the user.

**Mitigation:** Ensure that the data is untampered by encrypting the data. We can also keep backups of the data to ensure quick roll back in the case of tampering.

**Information Disclosure (Threat #56)**

**Threat:** The impact of Information Disclosure is that sensitive information that belongs to our users will be accessible by attackers. This will make our web application lose it's integrity as well as the trust of our users.

**Mitigation:** A solution to the information disclosure is to encrypt the data, that way, even if attackers get a hold of this sensitive data, they would only get the encrypted text instead of the plain text.

**Denial of Service (Threat #57)**

**Threat:** Denial of Service can affect accesibility as the web application will not be able to display the data that our users want to view.

**Mitigation:** Have a validation system to prevent buffer overflow or spamming as well as deny any unknown services.

**Threats against User’s Information**

**Tampering (Threat #9)**

**Threat:** While sending data from MWN's Web Application to MWN's Database, attackers can tamper the database data using Brute Force Attacks

**Mitigation:** Validate all the available inputs from the user. Use of Multiple Invalid Login Timer

**Information Disclosure (Threat #10)**

**Threat:** Unauthorised Data Access by attackers using SQL Injection Attack

**Mitigation:** Secure Coding: SQL Parameterised Queries

**Denial of Service (Threat #11)**

**Threat:** Control messages are not properly integrity protected

**Mitigation:** Comparing hash values before and after sending messages to improve integrity of the messages

**Data Stores**

**Threats against MWN’s Database**

**Tampering (Threat #21)**

**Threat:** Hacker has access to data which could lead to modification or destruction of data.

**Mitigation:** As data tampering affects the integrity of the message, by using an additional shadow table, that will include the hash values of the data that we would like to preserve the integrity for, we are able to counter-check if the data has been tampered with.

**Repudiation (Threat #22)**

**Threat:** If the user is able to retract what he/she says or sent, then the data has no integrity and it would be harder to track the culprit.

**Mitigation:** By using digital signatures, the parties will be able to ensure that the information is non-repudiated.

**Information Disclosure (Threat #23)**

**Threat:** If confidential information can be accessed by unauthorised party, the information could be leaked, misused or modified.

**Mitigation:** Encrypting the information.

**Denial of Service (Threat #24)**

**Threat:** Services would not be provided when the admin needs it. This would cause the admin not being able to retrieve information in case of emergencies.

**Mitigation:** Maintenance and back up of the system.

**Certifications**

No certifications have been made for this threat model.

**External Dependencies**

There are no external dependencies.

**Implementation Assumptions**

There are no implementation assumptions.

**External Security Notes**

There are no external security notes.

# Appendix B - Fortify report (Wk 16)

Due to the numerous amount of pages in this project report as well as the fortify report, the fortify report (Week 17) will be kept separated from this project report.

Please refer to the file: **MWN’s Fortify Report (Week 16)**

# Appendix C - Fortify report (Wk 17)

Due to the numerous amount of pages in this project report as well as the fortify report, the fortify report (Week 17) will be kept separated from this project report.

Please refer to the file: **MWN’s Fortify Report (Week 17)**

# Appendix D - Fortify report summary

Due to the numerous amount of pages in this project report as well as the fortify report, the fortify report summary will be kept separated from this project report.

Please refer to both the file:

* **MWN’s Fortify Report (Week 16)**; and
* **MWN’s Fortify Report (Week 17)**