Group 6: WeWantA Inc.

Coding Turk System
Software Requirements Specification
For Web Application

Version <1.0>

Coding Turk System	Version <1.0>
Software Requirements Specification	Date: <13/10/2017>
<WWA13102017-1.0>	

Revision History

Date	Version	Description	Author
<13/10/2017>	<1.0>	First Version of Coding Turk System	Thierno Diallo Chris Panican Anthony Tsui Jin Zhang

Table of Contents

Software Requirements Specification

1. Introduction

1.1 Purpose

This document's purpose is to explain and describe the Coding Turk System. This includes the purpose, usage, and features of the system, along with the technical details such as languages and system interactions. The document will also provide external resource about the technologies being used for this project.

1.2 Scope

This Coding Turk System will be utilized by clients who post system requests to be developed, and by Developers who will implement and fulfill the Client's requests. It is an online platform for people who would like to hire developers to do certain system requests.

Developers can bid on a post specifying the time and money required to fulfil the request, and clients can choose which developer to hire. A rating system will be available to both clients and developers, and they will be able to rate each other. Higher rating developers will have more positive benefits such as the ability to accept more requests. However, poor ratings will cause a limit of less than 2 requests and will be marked as poor performing. Suspected irresponsible evaluations will also result in warnings, and multiple warnings will warrant removal from the system.

In addition to those roles, there will also be a super-user which has the power to handle accounts, money issues, and manage user activities. The super-user can also manage warnings and can be appealed to for evaluation of warnings/removal.

Visitors will be able to view information made publicly from clients and developers, including ratings and past project details. However, to unlock all of the features such as being a client or being a developer, then the visitor must make an account.

1.3 Definitions, Acronyms, and abbreviations

Terms	**Definition**
Super-User (SU)	The administrator of the system. They have the rights to check every registered user, block users with violations, unban users who were reported by mistake, and etc.
Client	The users who post system requests. They must deposit money to be paid to developers that they choose to hire. Their rating is determined by developers.
Developer	The users who implements the system(s) a client posted. They can be rated by the clients on their performance.
Visitor	Temporary user who is not registered in the system. They can only view publicly made posts, and public information for clients and developers, including ratings and past work.
HTML + CSS	Hypertext Markup Language and Cascading Style Sheet. HTML provides a website structure and CSS can style it.
Python	A powerful programming language that we will be using in back-end to process various tasks.
Flask	A web framework. Provides backend tools to build complex, database-driven websites.
MySQL	A relational database management based on SQL. It has a wide range of purposes and mainly used in web development.
Twitter Bootstrap	A front-end web framework used to design websites. It makes styling easier and features mobile-responsiveness.

1.4 Reference

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirement Specifications.

IEEE Computer Society, 1998.

Flask Documentation: http://flask.pocoo.org/docs/0.10/

1.5 Overview

The next section, the Overall Description, gives an overview of the functionality of the system. It details the information requirements and establishes context for the technical requirements specification.

The third section, Specific Requirements, will go deeper into the functionality of the product by using developer/technical terms. With this, the developers will have a better understanding on the product's inner workings.

2. Overall Description

2.1 Use-Case Model Survey

The use-case model survey is a visual guide on how the functionalities of the system are implemented. The diagram below shows the basic features of the application and user interactions. In this section, we briefly explain this diagram and will go into detail in Section 3.1: Use-Case Reports.

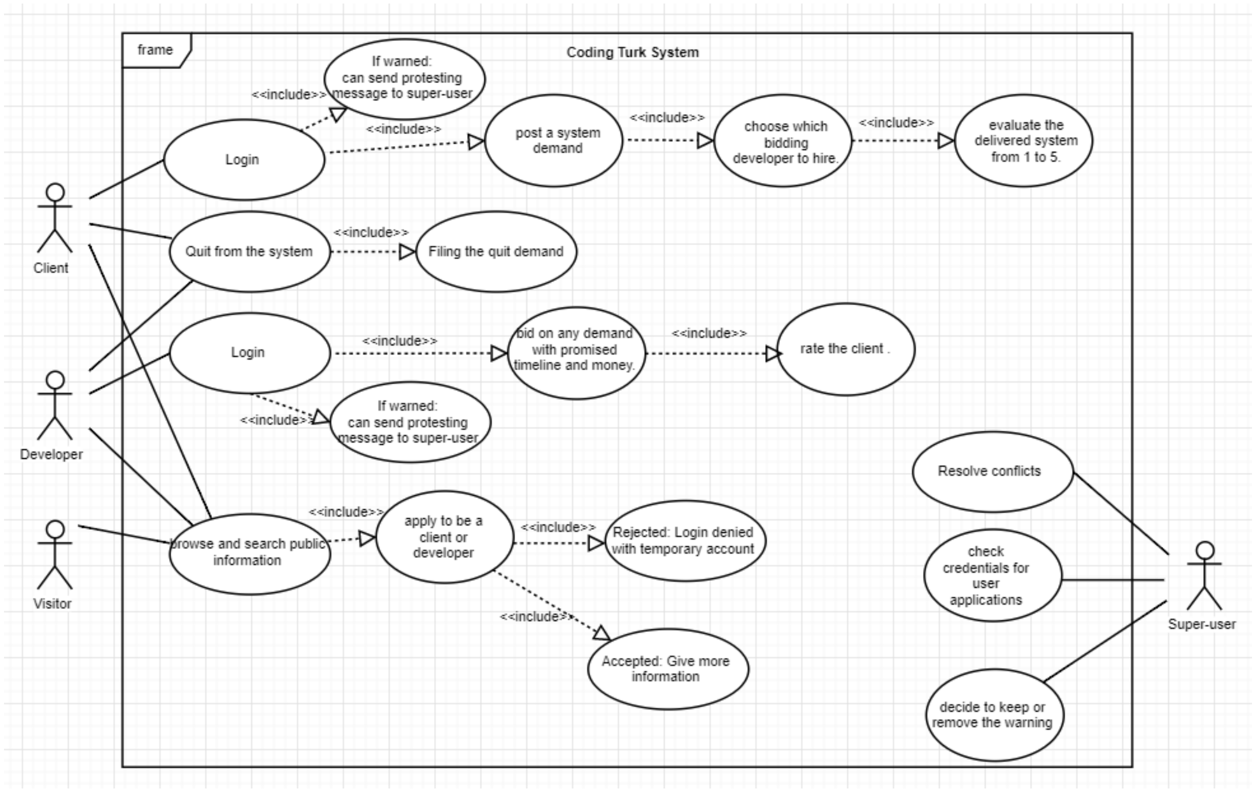

frame
Coding Turk System
If warned: can send protesting message to super-user
<<include>>
Login
post a system demand
choose which bidding developer to hire.
evaluate the delivered system from 1 to 5.
Client
Quit from the system
Filing the quit demand
Login
bid on any demand with promised timeline and money.
rate the client .
If warned: can send protesting message to super-user
Developer
Resolve conflicts
browse and search public information
apply to be a client or developer
Rejected: Login denied with temporary account
check credentials for user applications
Visitor
Super-user
Accepted: Give more information
decide to keep or remove the warning

Figure 1: Use-Case Diagram

There are four user types in this system: visitor, developer, client, and super-user. Each user have a unique role in the system and contribute in different ways.

- Visitor: Users that are not registered in the system are classified as visitors. They can view public posts and ratings on developers and clients. They have the option to register as a Client or Developer.
- Developer: Registered users who classify themselves as developers. They can look up projects and bid their price. If accepted, they can start working on the project within designated time. They can also quit partnership with customer.
- Client: Registered users who post system requests. They can view the list of available developers and choose whoever they want. After receiving the finished project, they will rate the developer's performance.
- Super-user: Administrators and moderators who manage users in the system. They can manage existing accounts, the blacklist, and projects. Super-users resolve various conflicts that may occur between the Developer and Client.

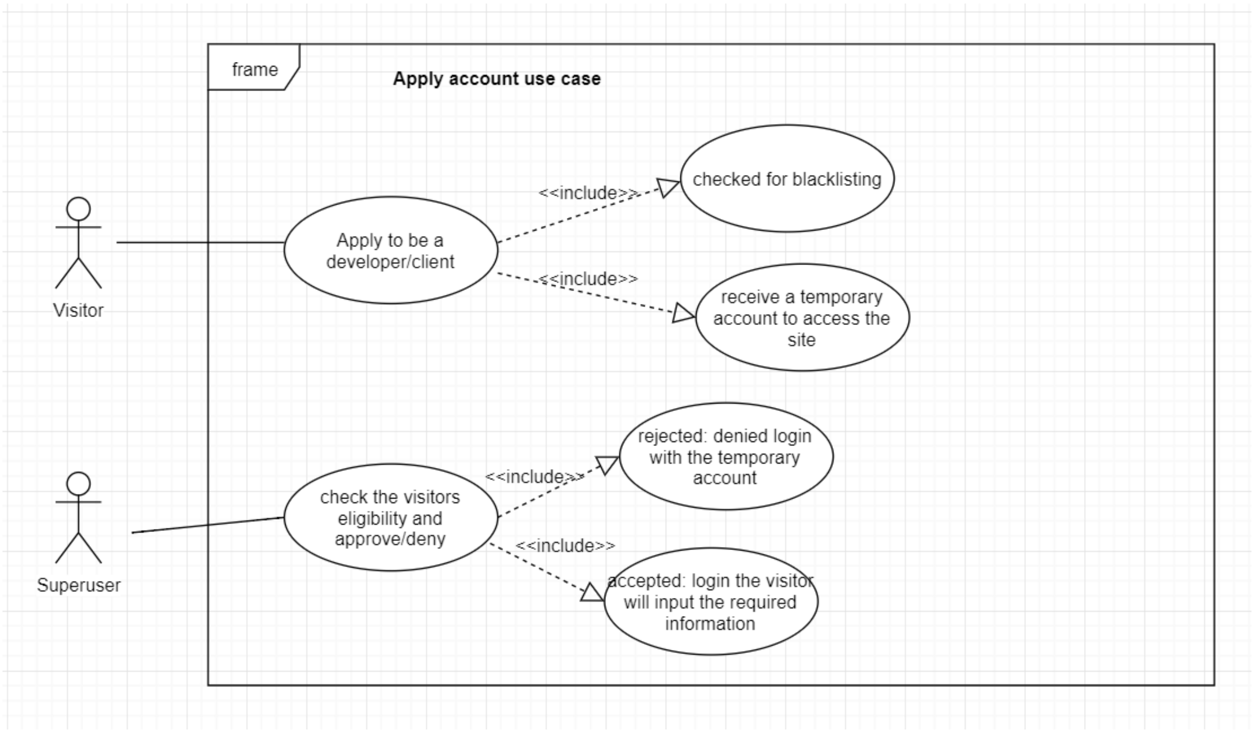
2.2 Assumptions and Dependencies

The application is expected to work until the host server is on disposal. The frontend and backend can be improved over time, and new features may be added as well. In the case where no one uses the site, it will be deemed unprofitable and be shut down.

The web application will also be dependent on the quality of hosting services and the amount of data we can put on the database server. Free tiers usually have a limit, while paid tiers have higher limits for a fee. Processing data on the server should be inexpensive since there are no complicated actions in the system.

3. Specific Requirements

3.1 Use-Case Reports


frame
Apply account use case
Visitor
Superuser
Apply to be a developer/client
<<include>>
checked for blacklisting
<<include>>
receive a temporary account to access the site
check the visitors eligibility and approve/deny
<<include>>
rejected: denied login with the temporary account
<<include>>
accepted: login the visitor will input the required information

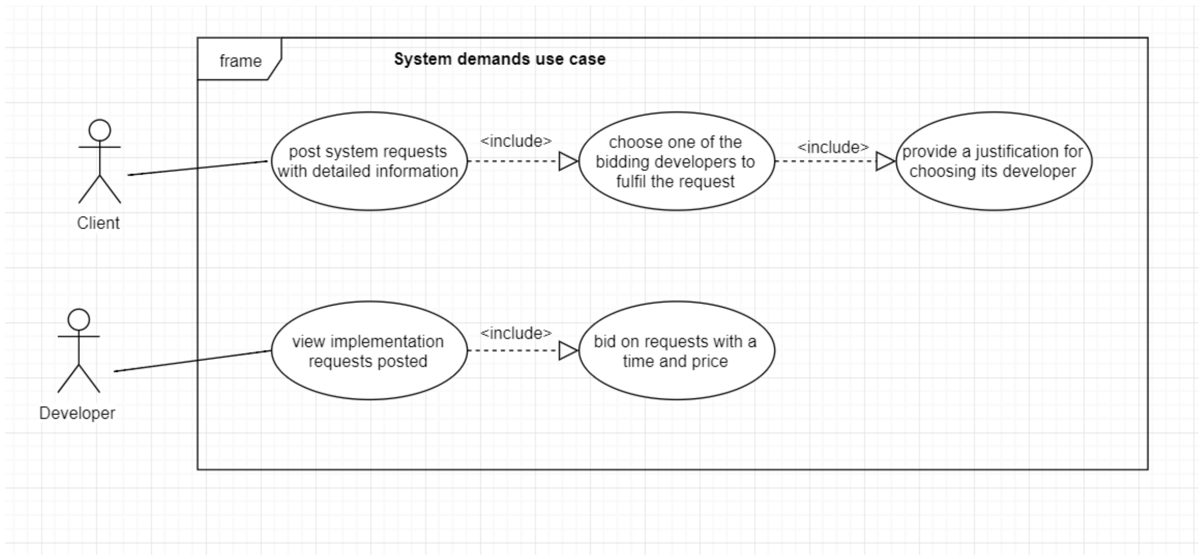
Use-case: Applying for an account

Figure 2: Use-case diagram for applying an account

Users: Visitors

An unregistered user can apply to be a developer/client, and will be checked on blacklist

1. A visitor accessing the site can apply to be a developer/client
2. Applicants will receive a temporary account to access the site
3. SU will check the visitors' eligibility and approve/deny
4. A. If rejected the visitor will be denied login with the temporary account
   B. If accepted, upon login the visitor will input registration information



frame
System demands use case
Client
post system requests with detailed information
<include>
choose one of the bidding developers to fulfil the request
<include>
provide a justification for choosing its developer
Developer
view implementation requests posted
<include>
bid on requests with a time and price

Use-case: System demands

Figure 3: Use-case diagram for system demands

Users: Clients/Developers

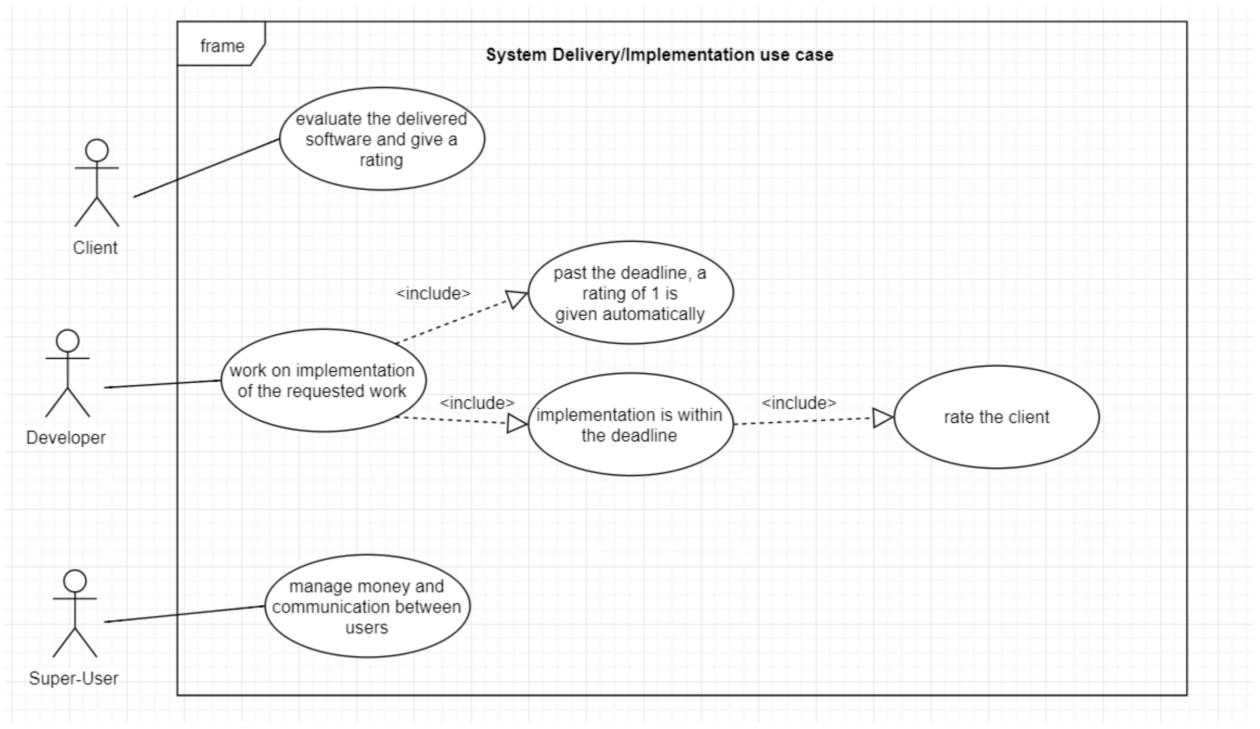
For Posting System Requests/Demands:

Clients can post system demands/implementations for developers to view/bid

1. Clients post detailed system requests on the website
2. Developers can bid on posted requests
3. Clients will choose one of the bidding developers to fulfil the request
4. Clients will provide a justification for choosing their developer
5. Half of the requested money will be transferred to the developer

Developers can bid on posted requests:

1. Developers can view posted system demands by Clients
2. Developers can then bid on requests with a proposed time and price
3. Half of the proposed payment is sent to the developer from the Client’s account


frame
System Delivery/Implementation use case
Client
evaluate the delivered software and give a rating
Developer
work on implementation of the requested work
<include>
past the deadline, a rating of 1 is given automatically
<include>
implementation is within the deadline
<include>
rate the client
Super-User
manage money and communication between users

Use-case: System Delivery/Implementation

Figure 4: Use-case diagram for System Delivery

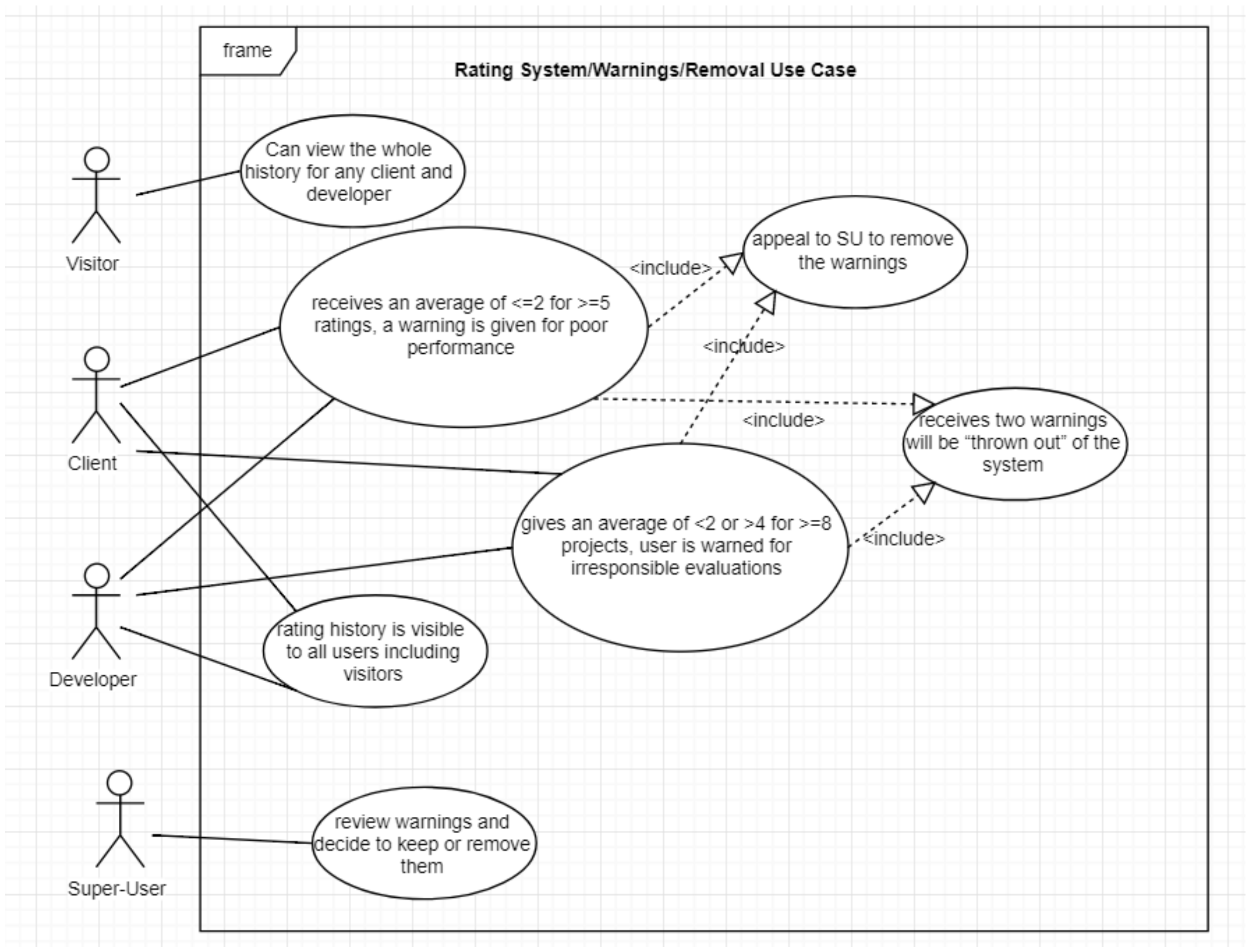
Users: Clients/Developers/Super-Users

After a Client has bid on a developer for their system request, the developer is expected to implement it within the predetermined deadline. A rating system will be used afterwards for both client and developer.

1. Developer will work on implementation of the requested work
2. a. If the implementation is past the deadline, the initial payment is transferred back to the Client, with an extra penalty fee from the Developer. A rating of 1 is also given to the Developer.
   b. If the implementation is within the deadline, the remaining payment is transferred to the SU

The following assumes the implementation was on time:

3. The Client will evaluate the delivered software and give a rating
4. a. If the rating is >= 3 the payment is transferred from the SU to the Developer
   b. If the rating is < 3 the client must give a reason for the rating, with the SU evaluating the low rating
5. 5% of the payment will be charged for using the system, and transferred from the Client and Developer to the SU
6. The developer will rate the Client, providing a reason for <=2 rating


frame
Rating System/Warnings/Removal Use Case
Visitor
Client
Developer
Super-User
Can view the whole history for any client and developer
receives an average of <=2 for >=5 ratings, a warning is given for poor performance
appeal to SU to remove the warnings
<include>
<include>
<include>
<include>
receives two warnings will be "thrown out" of the system
gives an average of <2 or >4 for >=8 projects, user is warned for irresponsible evaluations
rating history is visible to all users including visitors
review warnings and decide to keep or remove them

Use-case: Rating System/Warnings/Removal

Figure 5: Use-case diagram for Rating/Warnings/Removal system

Users: Visitors/Clients/Developers/Super-Users
A rating system is used after a job between Clients and Developers, with an SU adjudicating.

1. Clients and Developers will have their rating history visible to all users
2. If a user receives an average of <=2 for >=5 ratings, a warning is given for poor performance
3. If a user gives an average of <2 or >4 for >=8 projects, user is warned for irresponsible evaluations
4. Warned users can appeal to SU to remove the warnings
5. SU will review warnings and decide to keep or remove them
6. Any user that receives two warnings will be "thrown out" of the system
    a. Removed users will be able to login once for system closing matters
    b. Removed users can still protest to super-users to reverse this action
    c. Removed users are put on a "black-list", and cannot register for one year

3.2 Supplementary Requirements

There are several requirements for the system to function properly. The following requirements ensure that the system runs correctly and efficiently.

1. Ease of transactions and efficiency
   We want to make sure that developers do not get involved in payment transactions after a job is taken. For every job start with half of the due payment given to the developer. The transactions are restricted from clients to the super-user and from the super-user to developers. We want to make sure that this process happens very efficiently to decrease future complications.

2. Scalability and system stability
   The coding Turk system is in the starting process. We intend to build a strong MySQL database that we will keep monitoring. As user population increases, we will need to upscale the servers and hardware of our system that need more storage capacity for our repositories.

3. Security and protection of user's privacy
   User's account and privacy are one of the priorities that we must consider. All accounts' passwords are encrypted by a Python hashing package. By doing this, if there is an event of data intrusion, the hackers will face difficulty attempting to access an account's private information. Additionally, users can also set specific personal information private and can also delete their account from the system.

4. Supporting Information

This Software Requirements Specification includes:

- Table of Contents
- Index