Bug Tracker

Version 1.5

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 20/09/2016 | 1.0 | Start iteration plan | Tessa |
| 24/10/2016 | 1.1 | Add Overall Plan, iteration table, use cases, resources, demos of add bug and view bug | Tessa |
| 24/10/2016 | 1.2 | Add demos of login, register, edit profile and view profile | Xinwei Jiang |
| 25/10/2016 | 1.5 | Finish the iteration plan for edit profile and view profile | Xinwei Jiang |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Plan 4

3. Resources: 12

4. Use Cases 12

5. Evaluation Criteria 12

# Introduction

This iteration plan focuses on documenting iterations involved with the build of bug tracking software following the RUP software development process. The allocation of tasks for each weekly iteration is recorded along with a demonstration of each user interface as it’s built.

## Purpose

The purpose is to document the plan, iterations and build of Bug Tracking software using RUP. Iterations cover the construction stage through to Alpha testing.

## Scope

The project associated with this iteration plan is a bug tracking software application currently under development known as Bug Tracker. Bug Tracker includes functionality to report and comment on bugs in specific software applications and record the bugs life cycle, from new through to fixed. It also calculates reputation points for both software developers and reporters along with statistical reports on bugs. This iteration plan documents construction phase through to the start of Alpha testing.

## Definitions, Acronyms and Abbreviations

See Glossary.docx

## References

SRS.docx Version 1.3

TestPlan.docx Version 1.2

## Overview

The document contains:

* An overall plan of each iteration for the entire construction stage. Time frame for an iteration is one week.
* An iteration table outlining tasks allocated to each team member and when the tasks are due.
* Demos of interfaces and elements being added as they are built.

# Plan

Overall iteration plan for the entire construction phase through to Alpha Testing:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Milestone** | **Artifacts** | **Planned** | **Revised** | **Actual** |
| Develop an executable prototype of the basic componentry incorporating an SQL database.  **Includes:**  sequence diagrams and architectural design for each use case.  Database design diagram.  State diagram for 2 entity classes, Bug and Users | Build user interfaces, (include diagrams, see user interface prototype in SRS V1.2) and functionality to allow a user to:   * Register * Log-in | 19/09/2016 |  | 19/09/2016 |
| * View main window of the system * Report a bug * View/edit profile * Create database design and build | 27/09/2016 |  | 4th October |
| View/Edit bug  Search for a bug  Search other users  Comment on a bug  Subscribe to a bug  Display and export a statistical report  Assign a bug to a developer | 4th October 2016 | 10th October 2016 | View/Edit bug complete  Search bug complete  Comment on bug complete  Subscribe to bug complete  12th October 2016-10-23  Search Users  Assign bug  Display & export a report  Complete by 23rd October 2016 |
| Complete integration and Alpha testing to refine functionality.  Present the new system to the class (users). | * Complete overall architectural design document * Complete data persistence design * Integrate each members code into one application. * Create a test plan * Create unit tests | 11/10/2016 | 24/10/2016 | * Code integration completed by 17th October 2016-10-23 * Architectural Design diagram complete by 24/10/2016 * Test Plan complete 27/10/2016 * Unit tests complete 27/10/2016 * Data persistence design complete 27/10/2016 |
|  | * Carry out testing * Begin work on presentation | 18/10/2016 | 25/10/2016 | Presentation complete 25/10/2016  Specific use cases tested by  27/10/2016 |
|  | * Complete presentation | 26/10/2016 |  | Complete 26/10/2016 |

**Iteration Table Showing Intermediate Milestones and Task Allocation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Iteration** | **Due** | **Responsible** | **Functionality Added** |
| Week 8 | 19/09/2016 | Every group member – to become familiar with QT and complete some tutorials. | Begin construction of the application by building guis for log-in and registration.  Covers Uses cases: 1 and 2. A user can register and log-in to the system.  Functionality added:   * New user password verification. * Store user details to a database by creating an SQL connection. * Verify user name and password on log in. * Provide entry to main window.   Create a sequence diagram  Document the new controller classes to be added to the overall architectural diagram |
| Week 9 | 27/09/2016 | Nathaniel | * Create an SQL database on a remote server * Create entity class tables – Users, Bug |
| Week 9 | 27/09/2016 | Tessa | * Build Report a Bug user interface. Covers Use Case 12 – A user wants to report a bug. * Start Iteration Plan |
| Week 9 | 27/09/2016 | Selene | * Build Edit Profile user interface. Covers Use Case 5 - A user wants to change personal information. |
| Week 9 | 27/09/2016 | Toby | * Build search profile. Covers Use case 6 |
| Week 10 | 11/10/2016 | Tessa | * Build View Bug user interface. Covers use Case 7. The system should provide a User with a GUI to view Bugs. |
| Week 10 | 11/10/2016 | Nathaniel | * Build the main user interface. Use Case 4 |
| Week 10 | 11/10/2016 | Selene | * Build view user profile – Use Case 7 |
| Week 10 | 11/10/2016 | Toby | * Build Search User |
| Week 11 | 18/10/2016 | Nathaniel | * Integrate User interfaces into one application. Tidy up each class so the program can work as one. |
| Week 11 | 18/10/2016 | Tessa | * Start Test Plan Document |
| Week 11 | 18/10/2016 | Selene | * Complete functionality for the Register class |
| Week 12 | 25/10/2016 | Tessa | * Complete Architectural Design diagrams and description. Create black box test cases for View Bug and Add Bug. Complete project management documents. |
| Week 12 | 25/10/2016 | Selene | * State diagram for user and bug classes * Black Box Test cases for View and Add Profile. |
| Week 12 | 25/10/2016 | Toby | * Carry out white box testing using QT tests on controller class functions. |
| Week 12 | 25/10/2016 | Everyone | * Complete individual interface class diagrams and upload to github |
| Week 12 | 25/10/2016 | Nathaniel | * Complete Developer & user Reputation point functionality. * Create user interface diagram * Create ER diagram * Complete data dictionary for database |
| Week 13 | 27/10/2016 | Everyone | * Practice presentation of program * Complete the final class diagram |
| Week 13 | 29/10/2016 | Everyone | * Print and collate a hard copy of the complete assignment. |

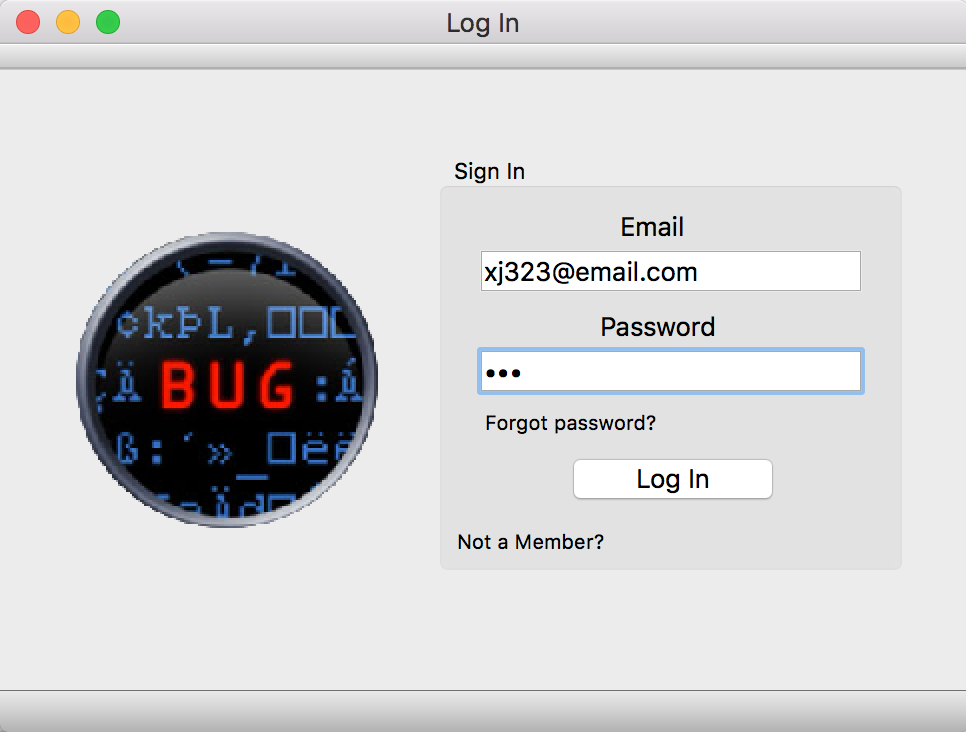
**Demonstrations of some of the User Interface Builds:**

**Iteration 1:**

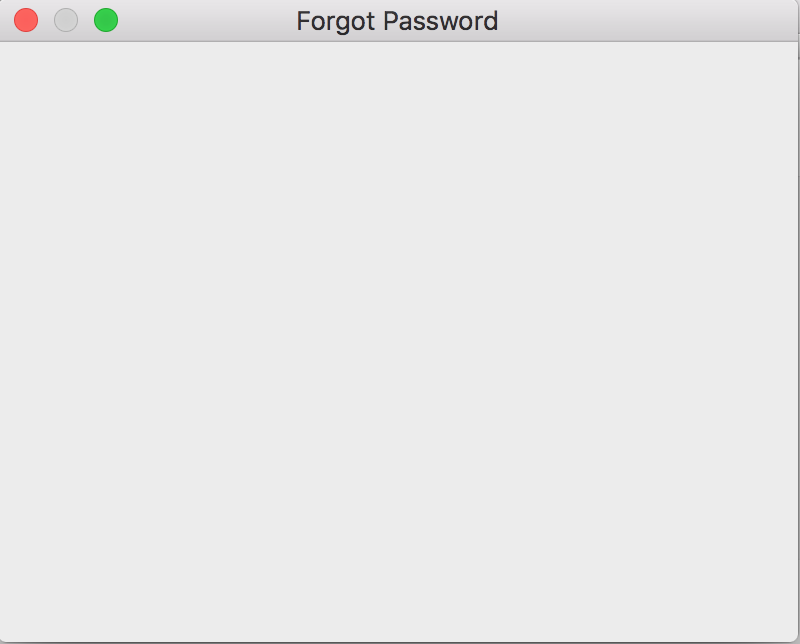
**Creation of Login interface:**

**Initial interface:**

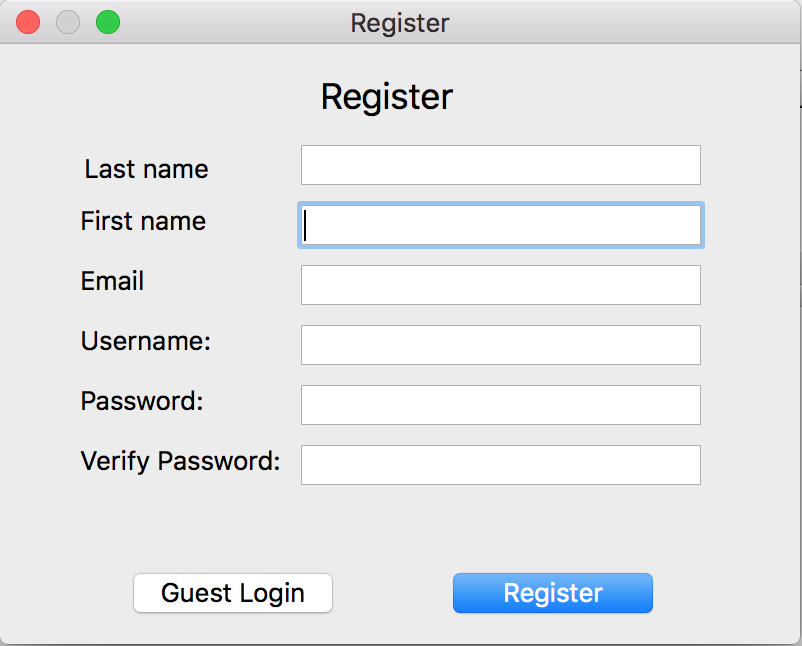
Users can type their email and password to sign in if they have the account.



The user can click “Forgot password?” to get the password if they forget the password.



Click “Not a member” to create an account.



**Still To Be Completed:**

* Set up the data base, so the system can check the user.
* Finish the “Forgot Password” dialog.

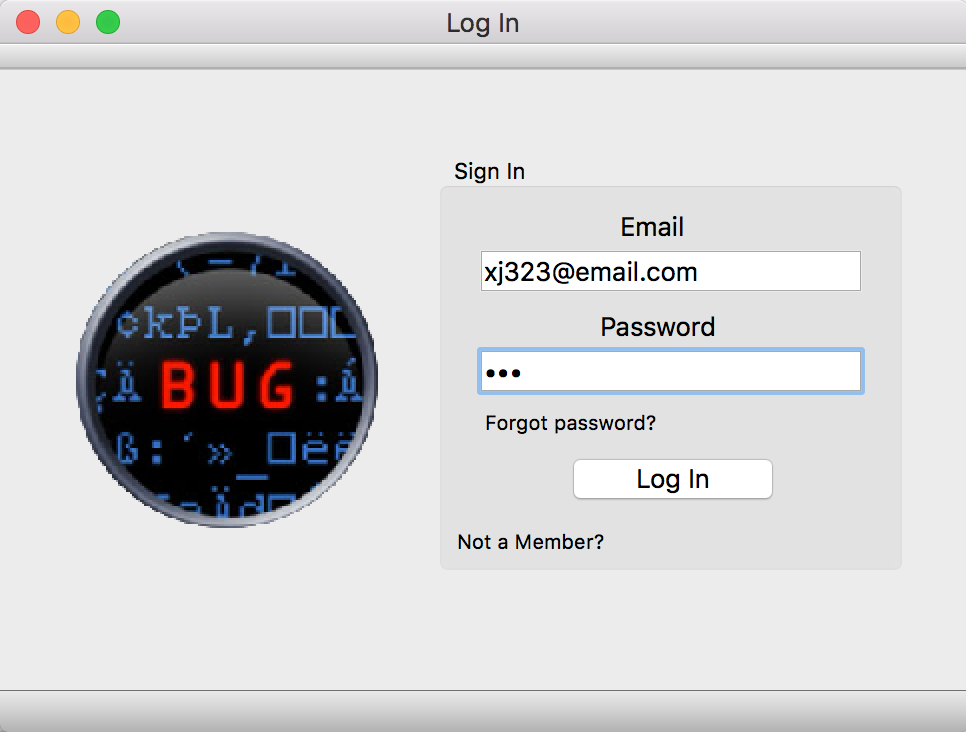
**Iteration 2:**

Set up database Login User Interface

* When the user want to login, the system can check the password and email.

**Test Performed**

User enter the email and password:



When the user clicks Log In button, the system will check the data. If the data is right, user can enter the main screen.

**Functionality Still To Complete**:

* Finish the “Forgot Password” dialog.

**Iteration 3:**

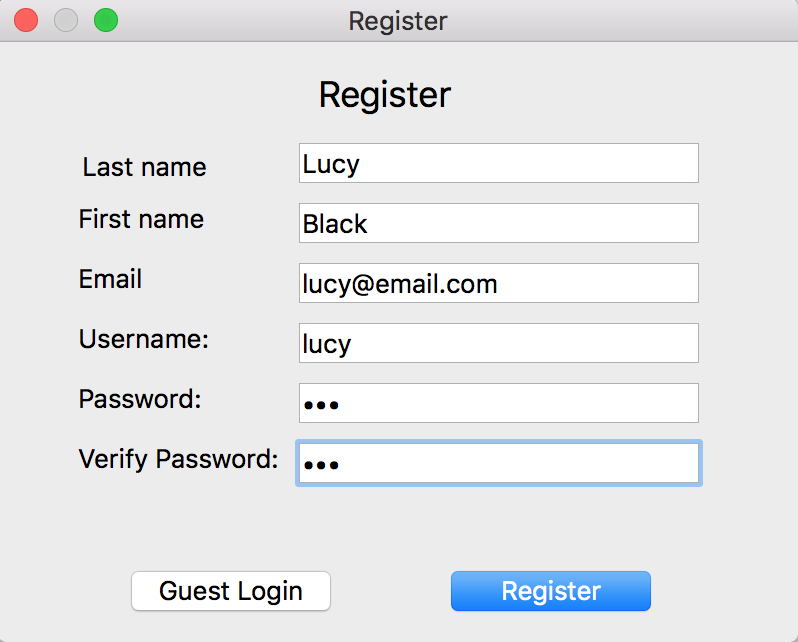
**Creation of Register interface:**

**Initial interface:**

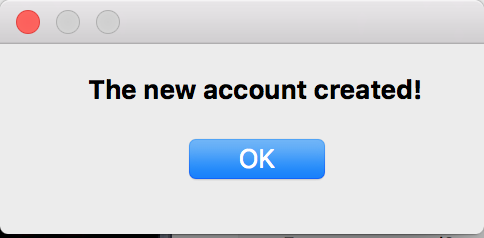
Complete the Register interface

**Test Performed**

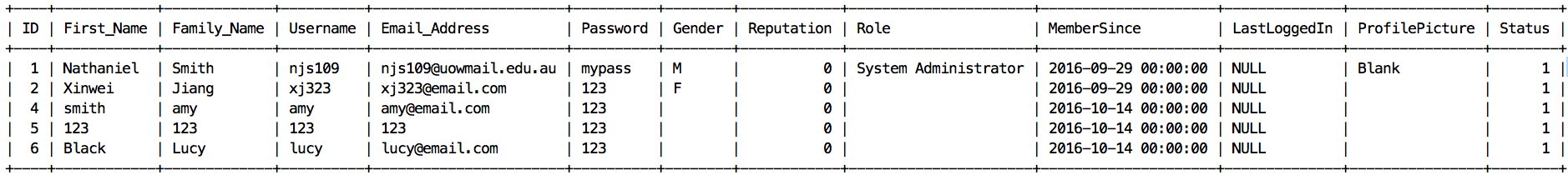
The user can enter the information to create an account, and the data can be inserted in to the database. And if the user doesn’t want to create the account, the user can log in as guest.



If the user creates the account successfully, a message will be shown.



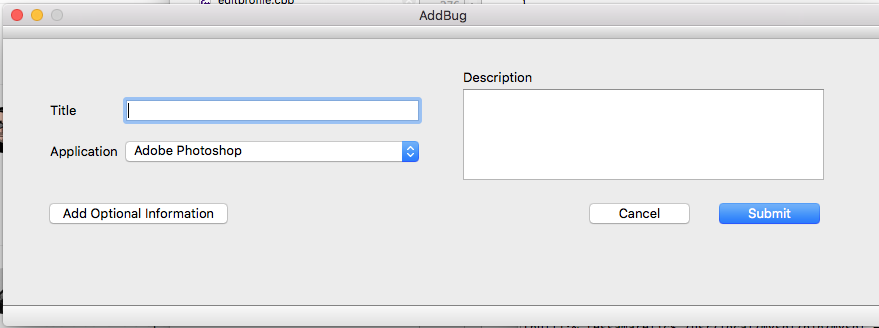
The data is inserted into the database.



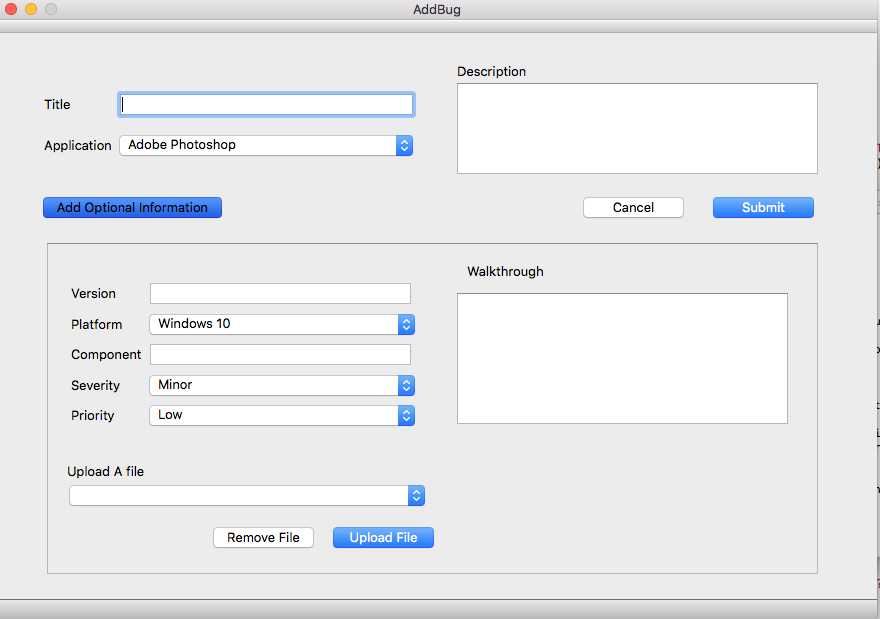
**Iteration 4:**

**Add Bug Initial User Interface:**

Users can choose to enter a minimum amount of information or click on Add Optional Information to enter extra data and upload a file if they wish.



Click Add Optional Information to show entire gui:



**Still To Be Completed:**

Functionality for the Add Optional Information toggle button so it toggles between open and closed.

Functionality to clear the interface once submit has been clicked.

**Outcome of Test:**



**Further Testing Needed:**

Test again once the correct driver has been loaded.

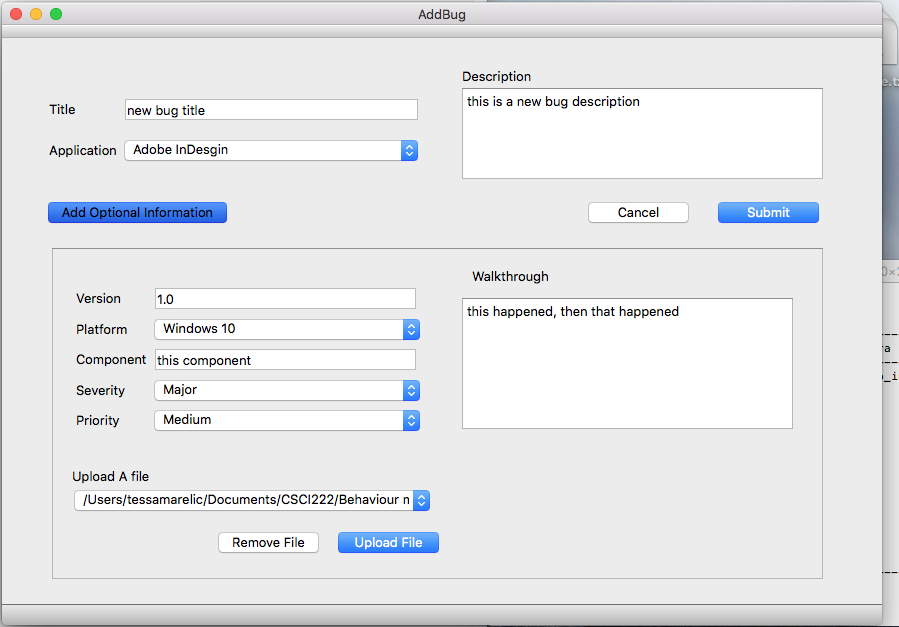
**Iteration 5:**

Functionality has been added to extra options toggle button, it now opens on click then closes on click.

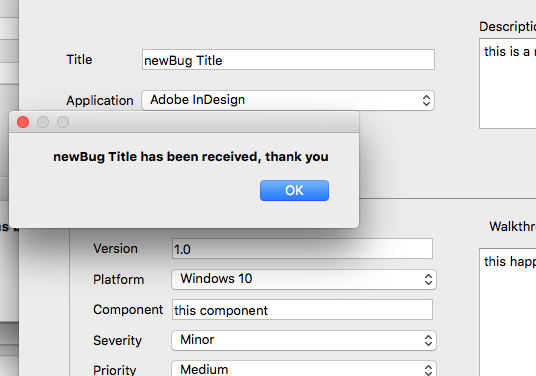
The interface clears when submit is clicked.

**Test Performed**

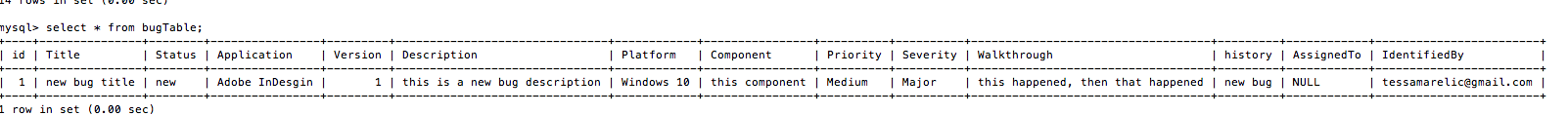
Reporter Adds a new bug:



Message displayed once submit button clicked:



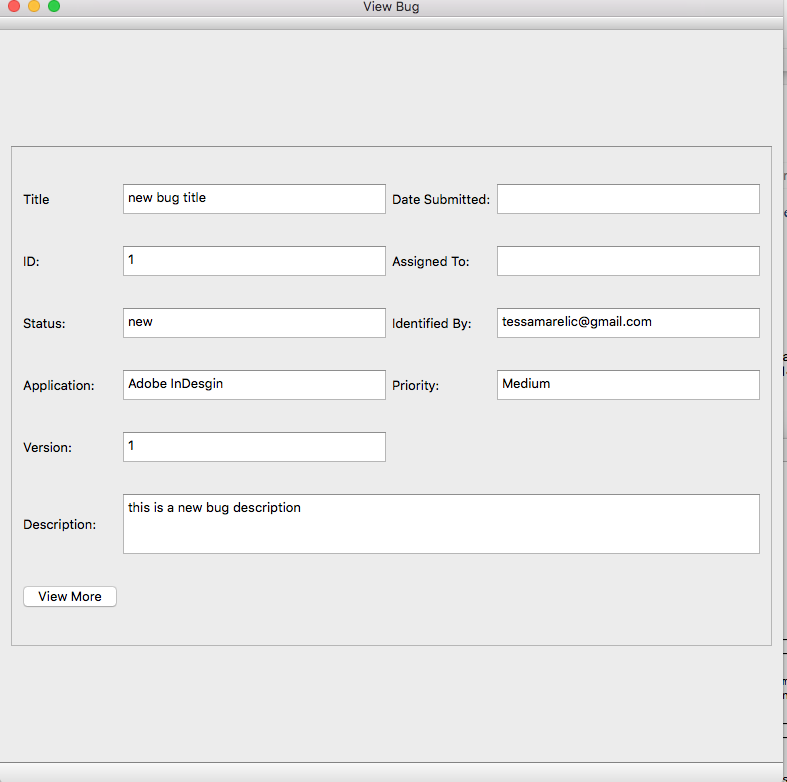
Data added to Database successfully



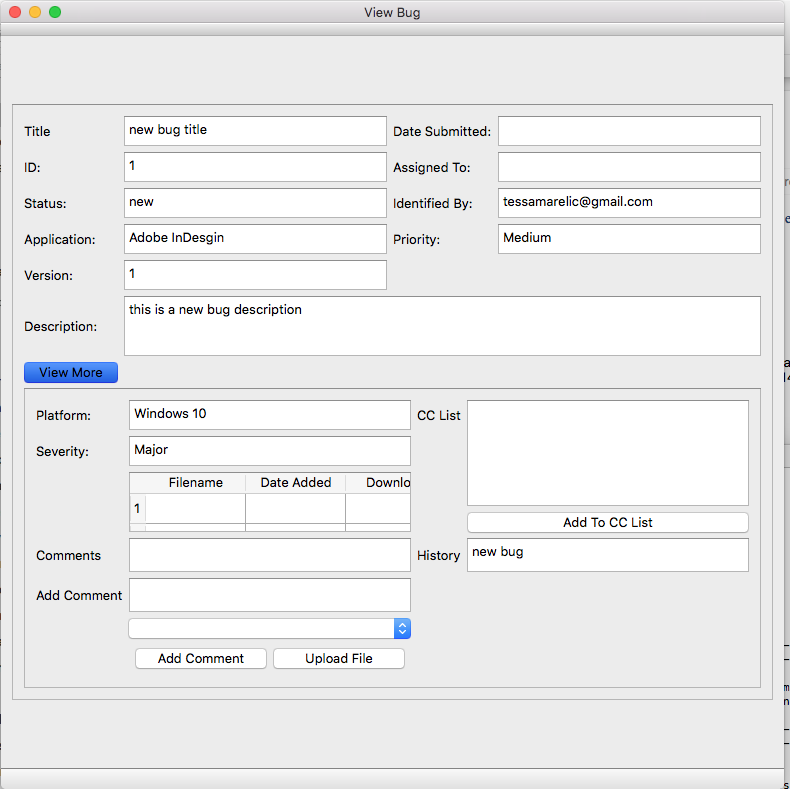
**Iteration 6:**

**Creation of View Bug Interface:**

The interface uses the gridlayout so it changes on window expansion and reduction. Initial Interface:



Interface Expands when View More Toggle Button Clicked.



**Functionality Included So far:**

Retrieves Id, Title, Status, Application, Version, Identified By, Priority, Description, Assigned To (at this stage no assignment), Platform, Severity and History from the database.

**Functionality Still To Complete**:

Retrieve Comments, Date Submitted, files and list of subscribed users from the database.

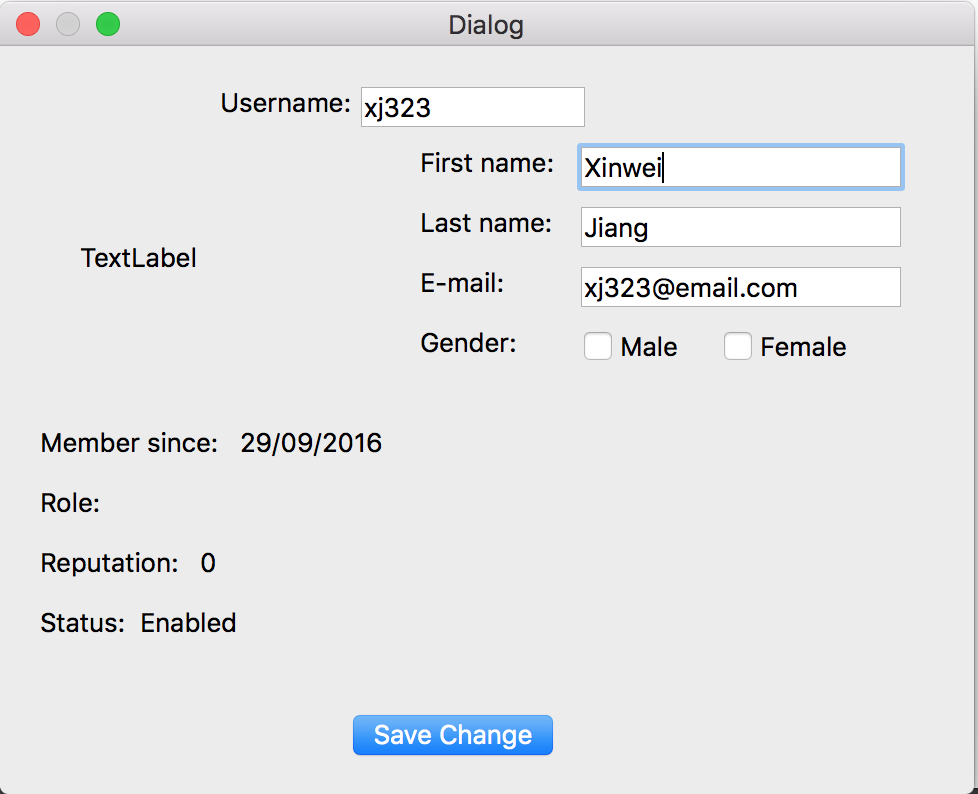
Functionality to add comments and upload a file.

**Iteration 7:**

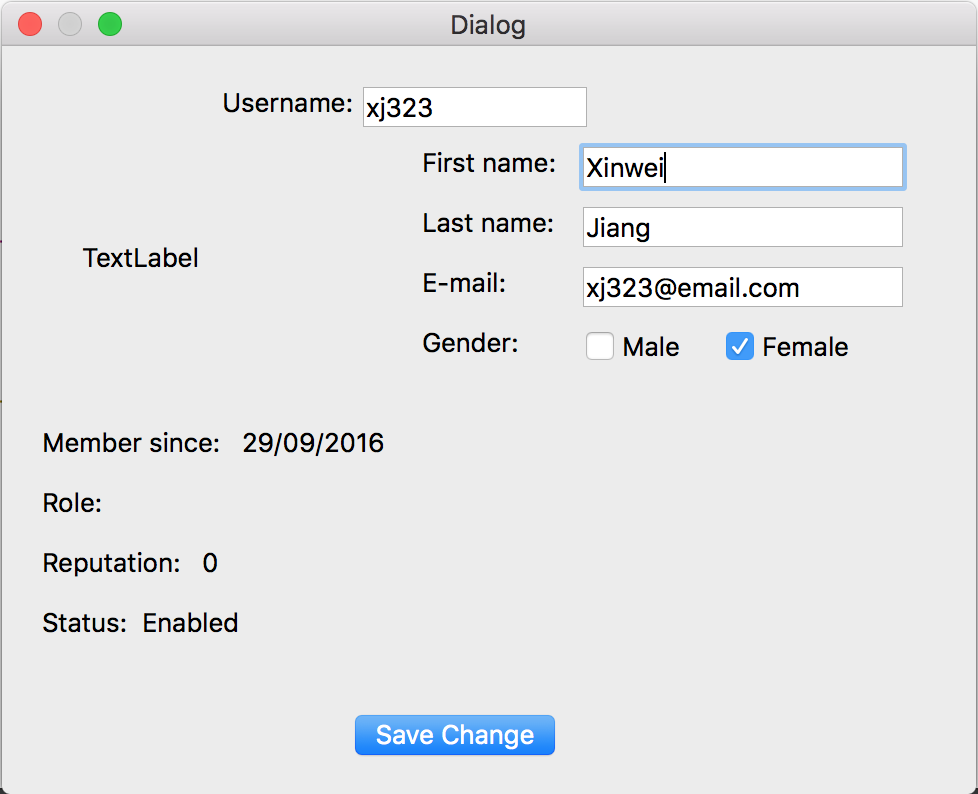
**Creation of Edit Profile interface:**

**Initial interface:**

When the user wants to edit their profile, the initial interface will show the information which is stored in the database, and some information the user can edit.



Then, the user wants to edit their profile, for example, the user wants to set the gender:

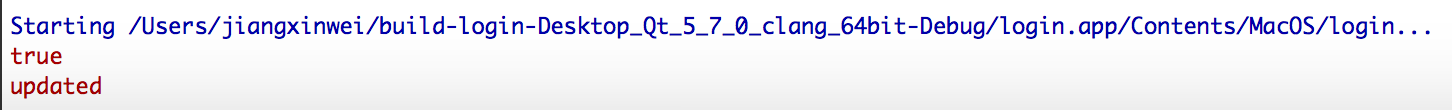


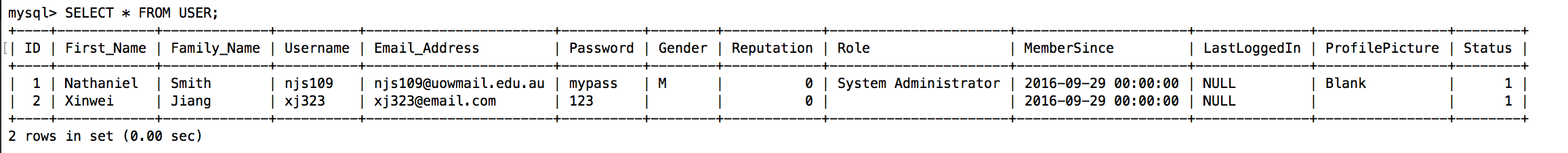
Users can click the save change button to update their profile.

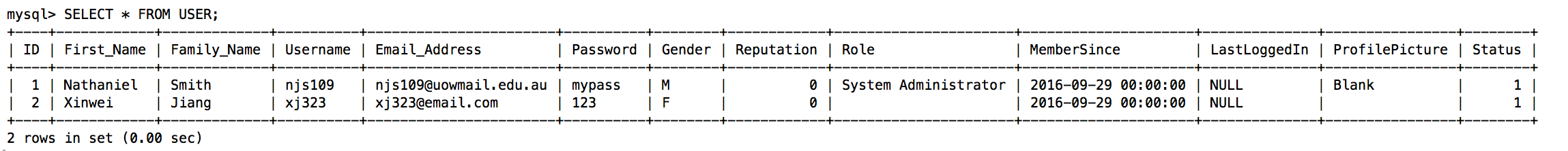
**Still To Be Completed:**

* Functionality to show the subscription of the user.
* Functionality to show the profile picture, and upload the file of profile picture.

**Outcome of Test:**



The database before editing the profile:

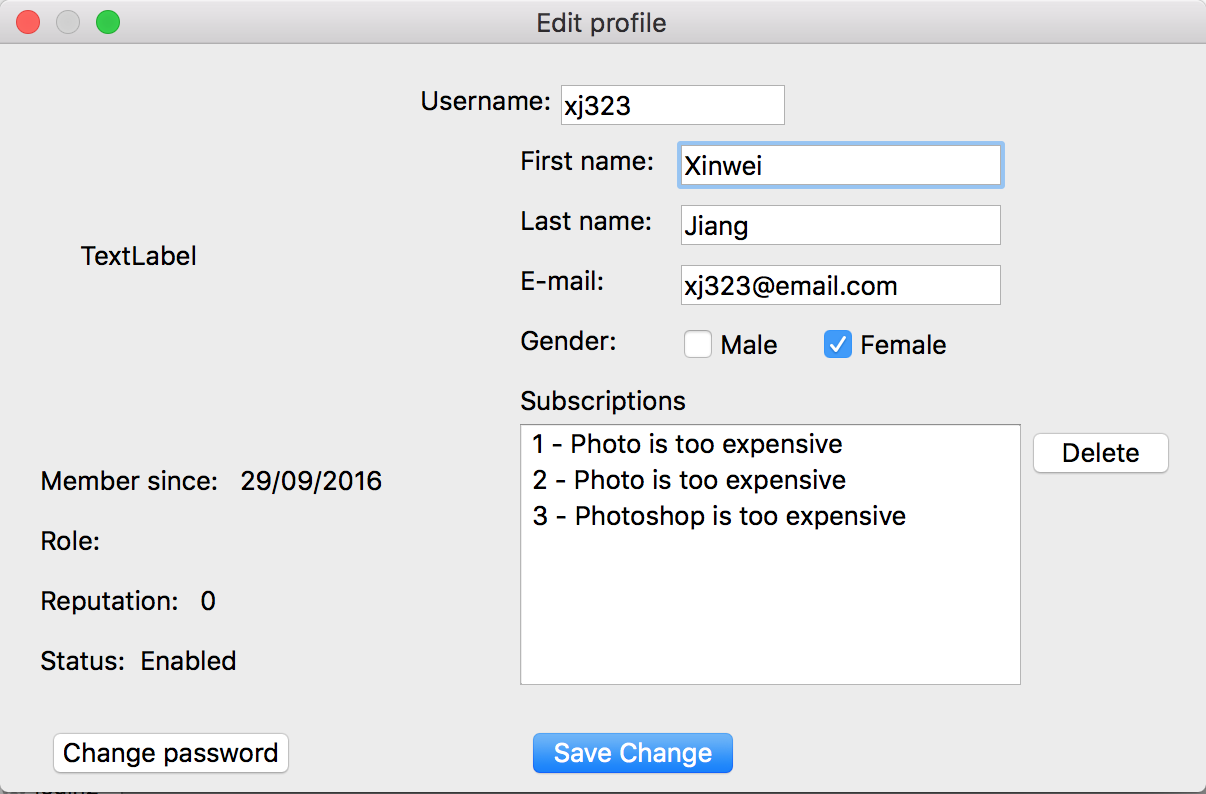
The database after editing the profile:

**Iteration 8:**

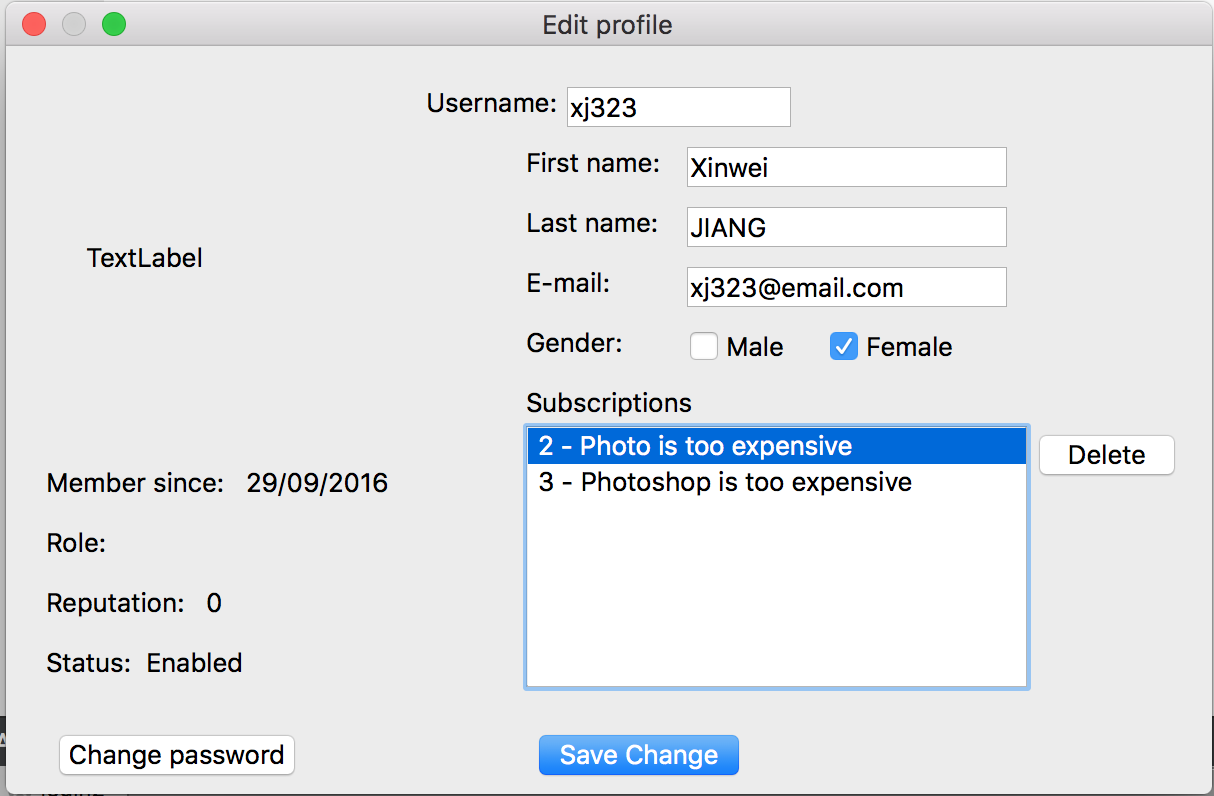
Added the functionality to show the subscriptions of the user, and the user can edit the subscriptions.

**Tested Performed**

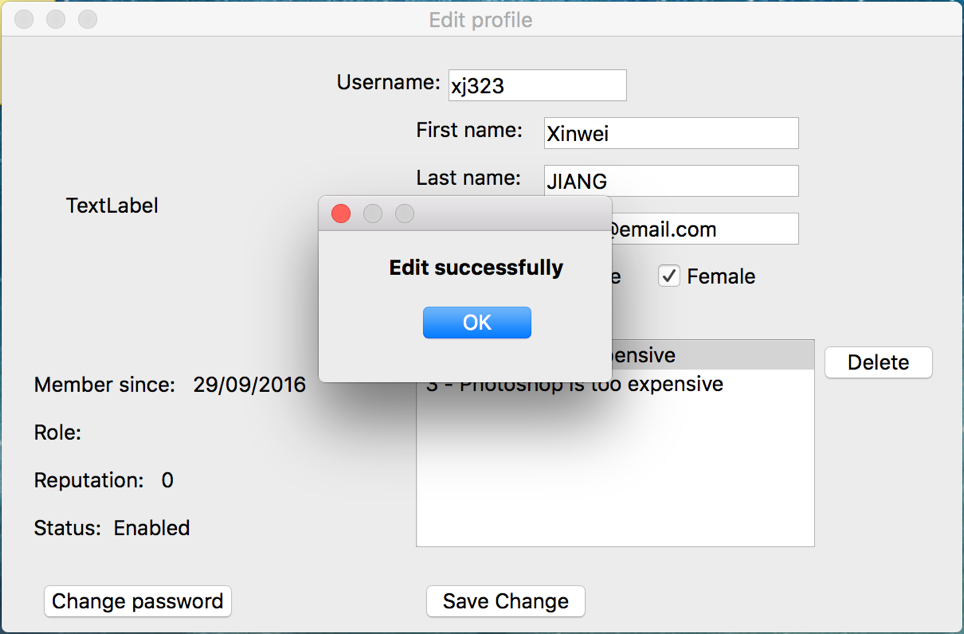
User open the interface of edit profile, and their information and subscriptions will be shown on the interface.

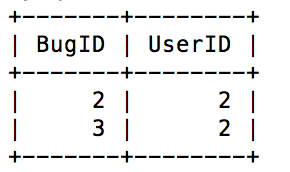


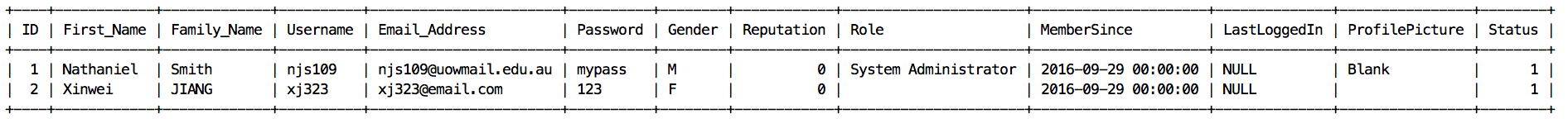
User can change the information and delete the subscriptions.



Message displayed once save change button clicked:



Data changed in the Database successfully.



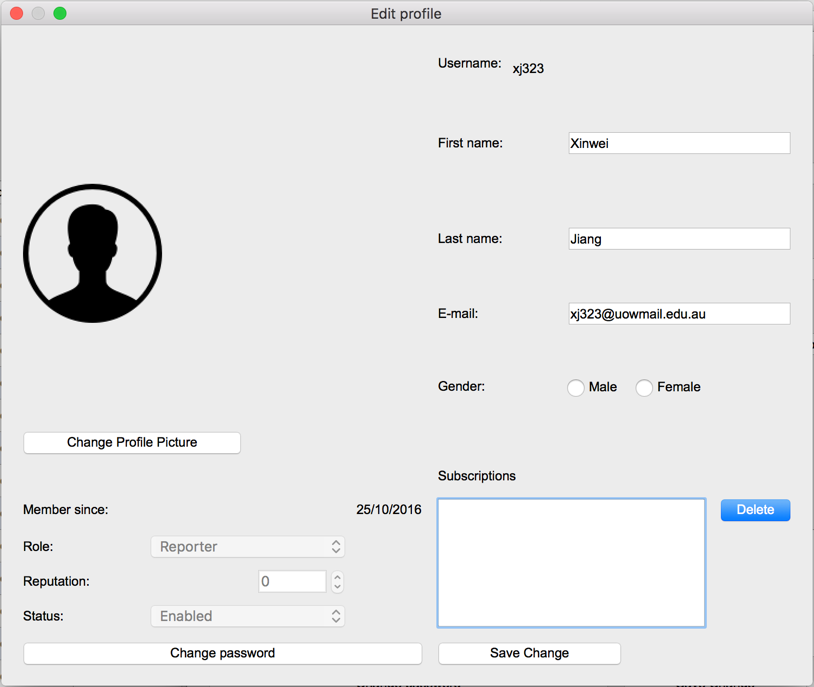
**Still To Be Completed:**

* Functionality to show the profile picture, and upload the file of profile picture.
* Functionality to allow the admin edit the role, reputation and status.

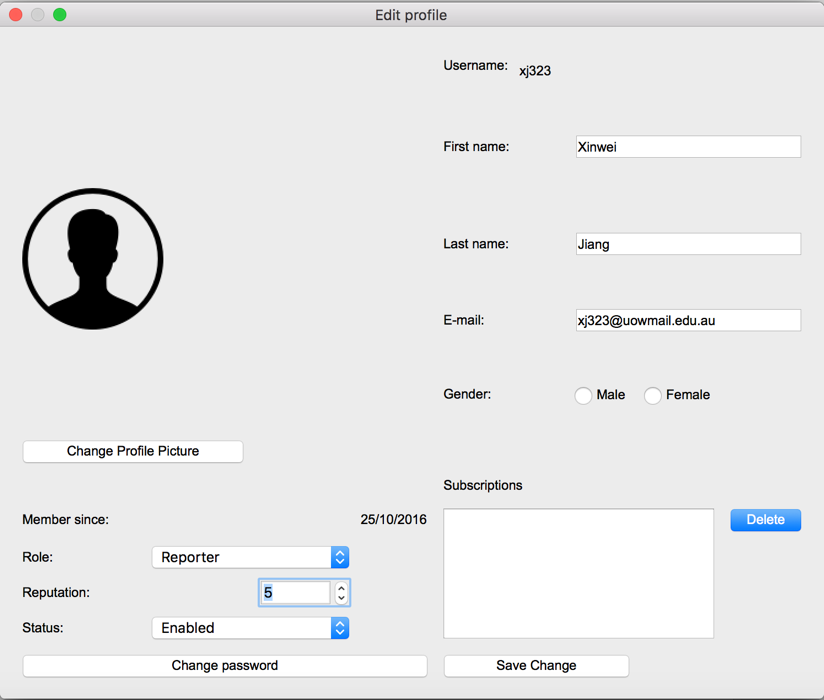
**Iteration 9:**

Added functionality to show the profile picture, and upload the file of profile picture and functionality to allow the admin edit the role, reputation and status.

**Tested Performed**

****User open the interface of edit profile, and their information, profile picture and subscriptions will be shown on the interface.

If login as the admin, he can edit role, reputation and status.



After clicking save change, the reputation of Xinwei Jiang will be changed in the database.



**Creation of Change Password Interface:**

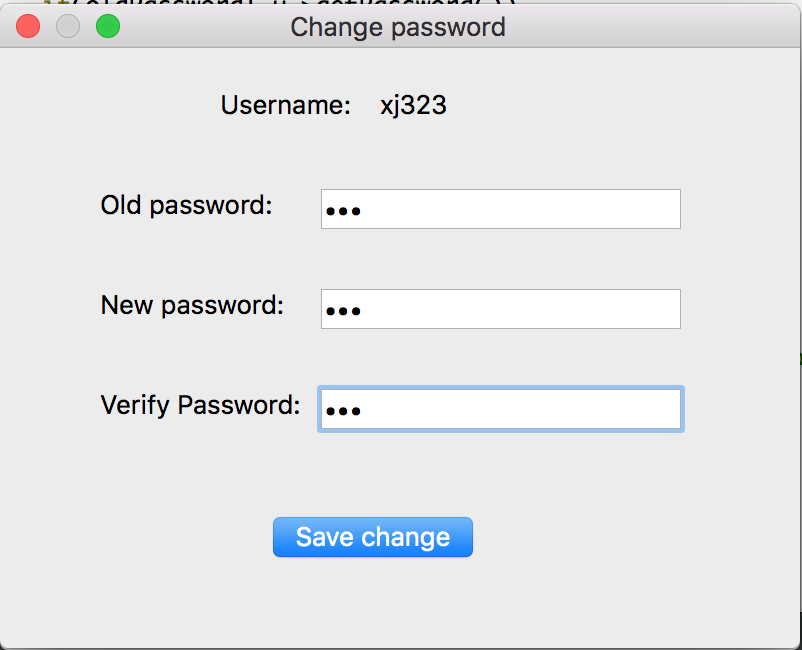
**Iteration 10:**

Completed ChangePassword Interface.

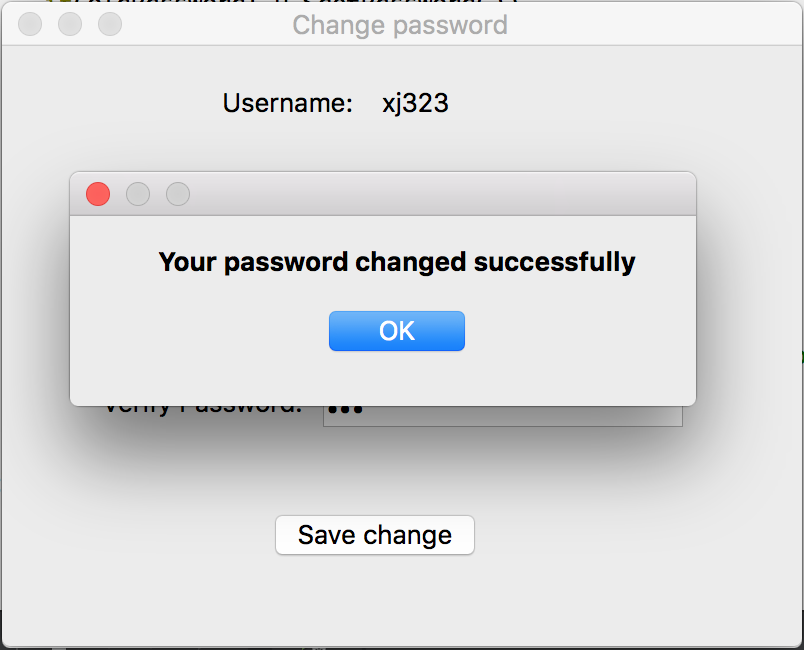
* The user can change the password through the interface.

**Test Performed**

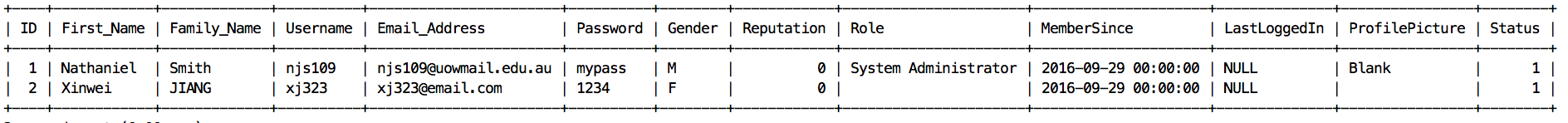
User enter the old password and new password.



Message displayed once save change button clicked:



Data changed in the Database successfully



**Creation of View Profile Interface:**

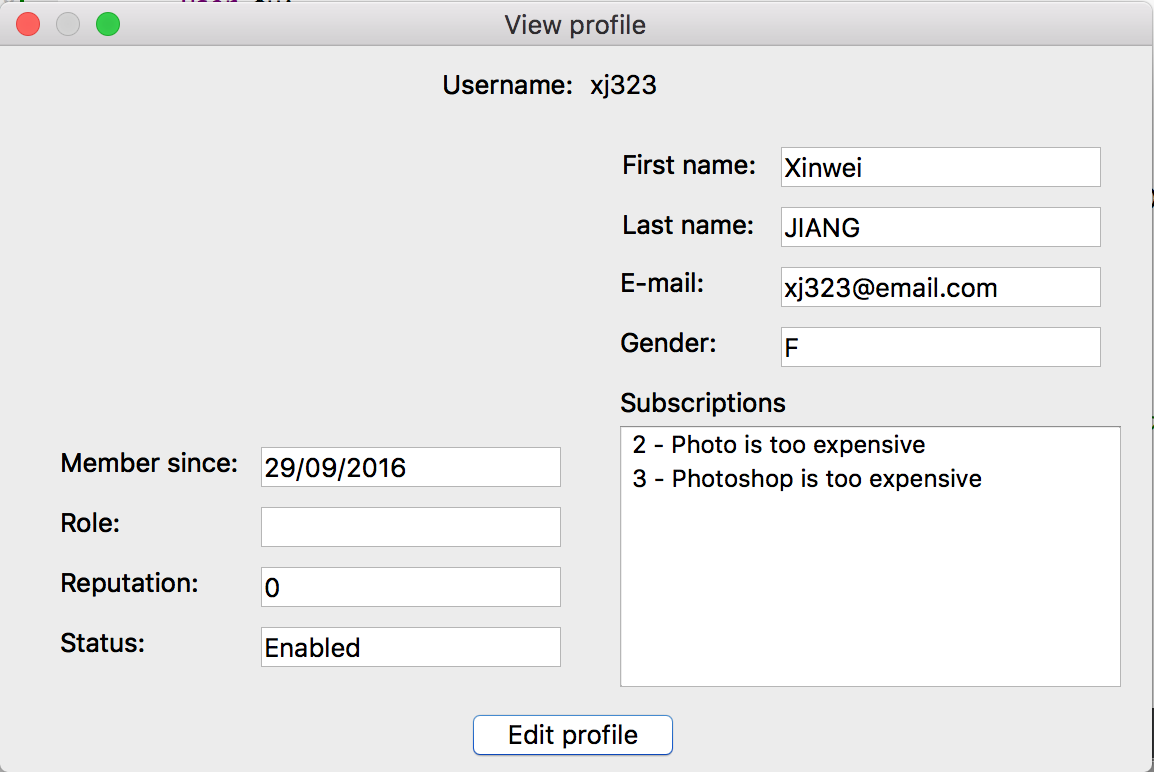
**Iteration 11:**

**Initial interface:**

User can view the information about their profile.

**Test Performed**

User open the view profile interface. The information about the user will be shown on the interface.



**Still To Be Completed:**

Functionality to show the profile picture.

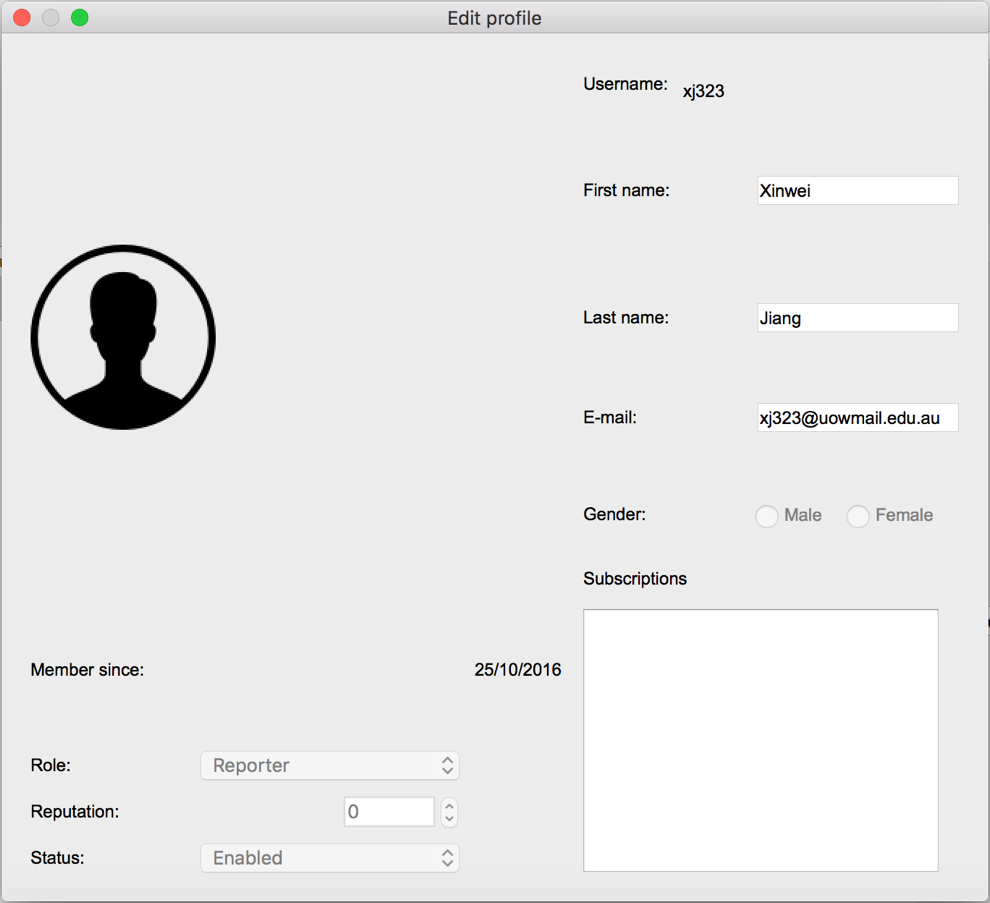
**Iteration 11:**

**Completed interface:**

Added functionality to show the profile picture.

**Test Performe**

User open the view profile interface. The information about the user will be shown on the interface.



# Resources:

Computer with QT Framework and software modelling application

4 Team members.

MySQL databases so each member can test their user interface functionality locally.

# Use Cases

**Use cases covered over the construction phase:**

Use Case 1: The system should provide a user with a gui to register and create an account.

Use Case 2: The system should provide a user with a gui to log in.

Use Case 3: The system should provide any user with a GUI to modify profile details, name, phone, address, and password.

Use Case 4: The system should provide users with a gui form to search for bugs of any type.

Use Case 5: The system should provide a user with a gui to change personal information.

Use Case 6: The system should provide a user with a GUI form to search for users of any type.

Search criteria include: First Name, Last Name, Email, Role, Account

Status and Key word.

Use Case 7: The system should provide a user with a GUI to view any non-administrator user profile. A profile consists of First Name, Last Name, User Name, Password

(Cannot be viewed), Email, Addresses, Account Status, Role (required), Date of Birth

Use Case 11: The system should provide a user of any type (Reporter, Developer, Triager, and Reviewer) with a GUI form to subscribe to the bug.

Use Case 12: The system should provide a user with a gui to add a new bug.

Use Case 13: The system should provide any user with a facility to add comments.

Use Case 18: The system should provide a User with a GUI to view Bugs.

Use Case 19: The system should provide a Triager with a GUI to edit existing Bugs in the System. Details such as title, description, status, severity, priority, assigned developer can be changed

# Evaluation Criteria

All guis must provide the functionality required for its specific use case, as outlined in SRS.docx V1.3.