Software Process

Analysis and Design

Vendor:

ZSFJ Ltd - Zac Holmes, Sheridan Boyd, Fili Kurene & Jethro Thomas

Created on: 18/07/2020

Contents

[1. Executive Summary 3](#_Toc54709838)

[2. Requirements 3](#_Toc54709839)

[3. Solution Design 3](#_Toc54709840)

[Entity Relationship Diagram (ERD) 3](#_Toc54709841)

[Database Structure 4](#_Toc54709842)

[4. Process Modelling 4](#_Toc54709843)

[User Stories 4](#_Toc54709844)

[UML Diagrams & Use Cases 6](#_Toc54709845)

# Executive Summary

ZSFJ Ltd have been engaged to develop an ASP.Net Core web application to aid BICT Lecturers and students in booking Test dates in the BICT computer labs.

It is expected that the project team will produce a prototype application that will allow Lecturers to set a Test Schedule for students to book the time they will take the test.

# Requirements

The Requirements of the prototype are as follows: -

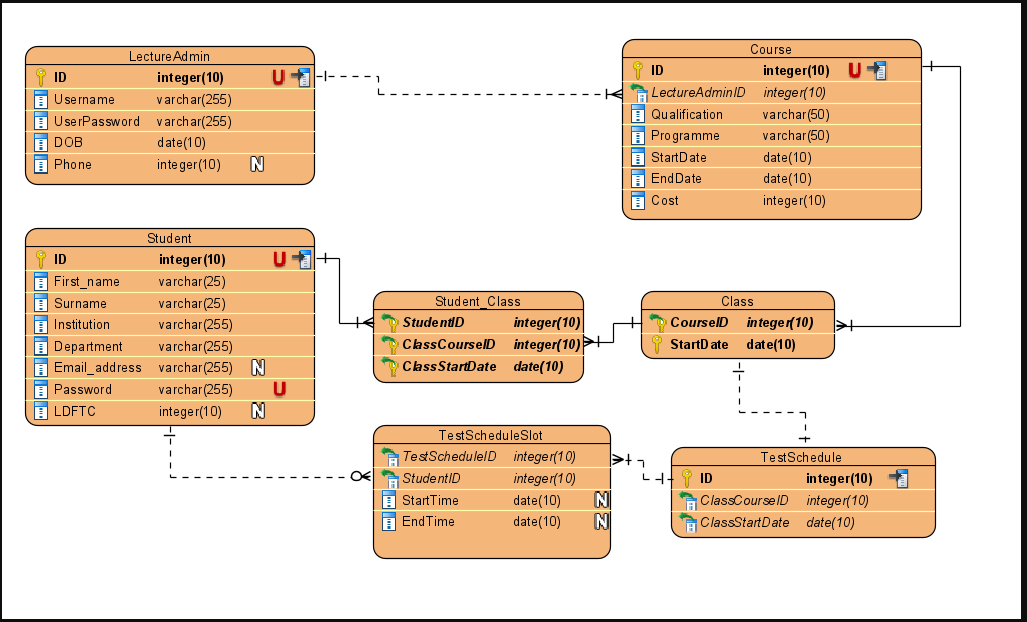
* The application should support multiple users and have two main user types (‘student’ and ‘lecturer admin’).
* The application should have a secure login.
* A class should be created for all courses.
* A class should have a lecturer assigned to it.
* A class should be populated with enrolled students.
* A student should be able to be in many classes (currently on Active Directory you belong to a D202 group, and also other groups for your other courses).
* Lecturers should be able to create a Test Schedule for a class.
* The Test Schedule should be made up of customisable time slots defined by the lecturer at time of creation.
* Students in the class should be able to access the Test Schedule and book an available time slot.
* Lecturers should be able to remove a student from a time slot if they have made a mistake choosing their time.
* Lecturers should be able to use the application to reset any student’s lab password.
* A student should be able to reset their own password.

Throughout the Project, it is possible for the requirements to change after the Product Review with the Product Owner.

# Solution Design

### Entity Relationship Diagram (ERD)

Below is the Entity Relationship Design (ERD) Diagram outlining the entity relationships for the solution design of the application.



### Database Structure

Run the SQL queries in the SQLQueries folder to initialise the database prior to first run of the application. SQLQuery has been included to add Admin User Profile.

# Process Modelling

The following modelling resources have been created to provide details of the system structure and system behaviour: -

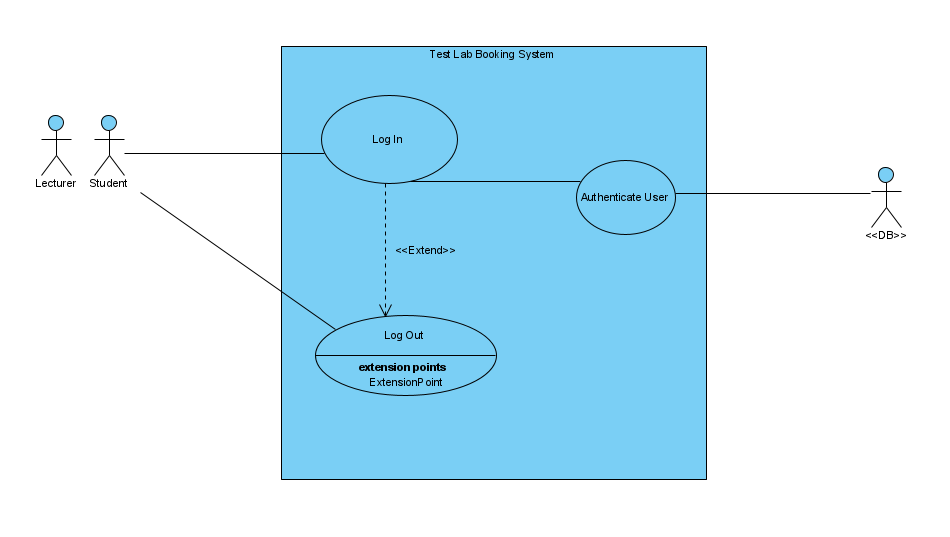
* Product Backlog User Stories
* UML diagrams & Use Cases

### User Stories

|  |  |  |  |
| --- | --- | --- | --- |
| **User Story No.** | **User Story** | | |
| **1** | **As…**  a Lecturer Admin | **I want…**  to be able to log in | **So That…**  I can use the system. |
| **2** | **As…**  a Student | **I want…**  to log in | **So That…**  I can use the system. |
| **3** | **As…**  a Lecturer Admin | **I want…**  to be able to split students into multiple classes | **So That…**  I can have more manageable class sizes (less that 20). |
| **4a** | **As…**  a Lecturer | **I want…**  to be able to create a Test Schedule | **So That…**  I can have students book for a test. |
| **4b** | **As…**  a Student | **I want…**  to be assigned my class | **So That…**  I can access class information. |
| **5a** | **As…**  a Student | **I want…**  to be able to book an available time slot on the Test Schedule | **So That…**  I can choose when I demonstrate to my Lecturer. |
| **5b** | **As…**  a Lecturer | **I want…**  to be able to remove accidental student timeslot errors | **So That…**  I can correctly plan my classes. |
| **6** | **As…**  a Lecturer | **I want…**  to reset Student Passwords | **So That…**  they (students) can log in when they have forgotten their Passwords. |
| **7** | **As…**  a Student | **I want…**  to reset my own Password | **So That…**  I can log into the system when I’ve forgotten my Password. |
| **8** | **As…**  a Lecturer | **I want…**  to reset my Password | **So That…**  I can log into the system (when I forget my Password). |

### UML Diagrams & Use Cases

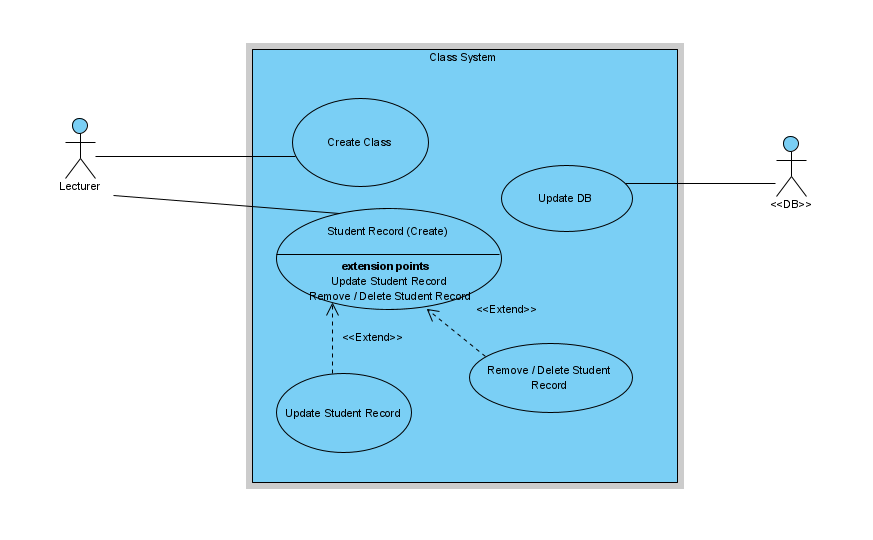
**Log In / Log Out Use Case**



|  |  |
| --- | --- |
| **Name** | **Log In** |
| **Identifier** | **UC 01** |
| **Description** | This use case describes how a user Logs Into the System |
| **Goal** | Log User into the system |
| **Preconditions** | 1. User must exist in the ARION database (as Lecturer or current Student) prior to the DB Migration. 2. DB Migration has been completed. |
| **Frequency** | More than 20 times. |
| **Basic Flow** | 1. The use case begins when the user wishes to log into the system. 2. The system requests that the user enter identification details. 3. The User enters their User ID and Password. 4. The system validates the details and presents default page of the system to the User. 5. The use case ends when the User is presented with the welcome page or system default page. |
| **Alternative Flow** | **Invalid ID**   1. If in step 3 of the Basic Flow the user enters an invalid ID or invalid Password. 2. The system displays an error message. 3. The use case ends when the user chooses to navigate to another part of the system or log out (either by selecting Log out or Closing the window). |
| **Post Conditions** | 1. User has successfully logged into the system. |
| **Actors** | UCOL Lecturer, UCOL currently enrolled Student |
| **Included Use Cases** | N/A |
| **Excluded Use Cases** | N/A |
| **Notes** | User can log in more than once. |

|  |  |
| --- | --- |
| **Name** | **Log Out** |
| **Identifier** | **UC 02** |
| **Description** | This use case describes how a user Logs Out of the System |
| **Goal** | Log User out of the system |
| **Preconditions** | 1. User must exist in the ARION database (as Lecturer or current Student) prior to the DB Migration. 2. DB Migration has been completed. 3. User is Logged in the system. |
| **Frequency** | If the User is logged into the system they are only able to / need to Log Out once. |
| **Basic Flow** | 1. The use case begins when the user wishes to log out of the system. 2. User selects the X icon of the Window or selects “Log Out”. 3. The use case ends when the User is logged out of the system. |
| **Alternative Flow** | N/A |
| **Post Conditions** | 1. User has successfully logged out. |
| **Actors** | UCOL Lecturer, UCOL currently enrolled Student |
| **Included Use Cases** | N/A |
| **Excluded Use Cases** | N/A |
| **Notes** | N/A |

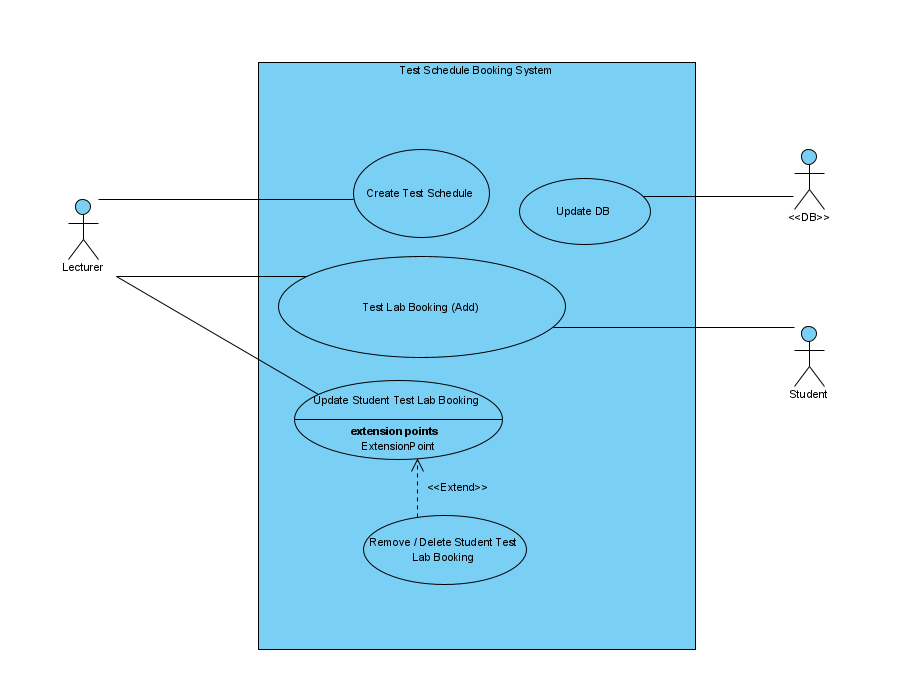
**Class Setup & Student Record (CRUD) Use Cases**



|  |  |
| --- | --- |
| **Name** | **Class Setup** |
| **Identifier** | **UC 03** |
| **Description** | This use case describes how a Lecturer will setup up the Class with Students. |
| **Goal** | Lecturer is able to setup up the Class with students. |
| **Preconditions** | 1. User must be logged into the system. |
| **Frequency** | Lecturer is able to create as many classes with as many number of students as required. |
| **Basic Flow** | 1. The use case begins when the User (Lecturer) wishes to setup the Class. 2. The User selects the class to be setup and creates the class name. 3. The User can add new students to the class. 4. The use case ends when the new student record has updated successfully to the DB. |
| **Alternative Flow** | N/A |
| **Post Conditions** | 1. Multiple classes are created. |
| **Actors** | UCOL Lecturer |
| **Included Use Cases** | N/A |
| **Excluded Use Cases** | N/A |
| **Notes** | Lecturer can create multiple classes. |

|  |  |
| --- | --- |
| **Name** | **Create, Remove, Update, Delete Students** |
| **Identifier** | **UC 04** |
| **Description** | This use case describes how a Lecturer will perform the following functions against Students in the Class: -   * Create Student(s) * Remove Student(s) * Update Student(s) * Delete Student(s) |
| **Goal** | Lecturer is able to Create, Remove, Update or Delete students from the Class. |
| **Preconditions** | 1. User must be logged into the system. |
| **Frequency** | Lecturer is able to create, remove, update & delete multiple student records in the class as required. |
| **Basic Flow** | 1. The use case begins when the User (Lecturer) wishes to make changes to Student Data record in the class. 2. The User is able to select <Create> to add a new Student Record. 3. On selecting <Save> the new Student Record is saved successfully to the database. 4. The User then logs out of the Class area of the system. 5. The use case ends when the User has completed the User |
| **Alternative Flow** | N/A |
| **Post Conditions** | 1. User (Lecturer) has created & updated various student records in a class. |
| **Actors** | UCOL Lecturer |
| **Included Use Cases** | UC 03 |
| **Excluded Use Cases** | N/A |
| **Notes** | User (Lecturer) is able to create, remove, update and delete a number of student records from the Class. |

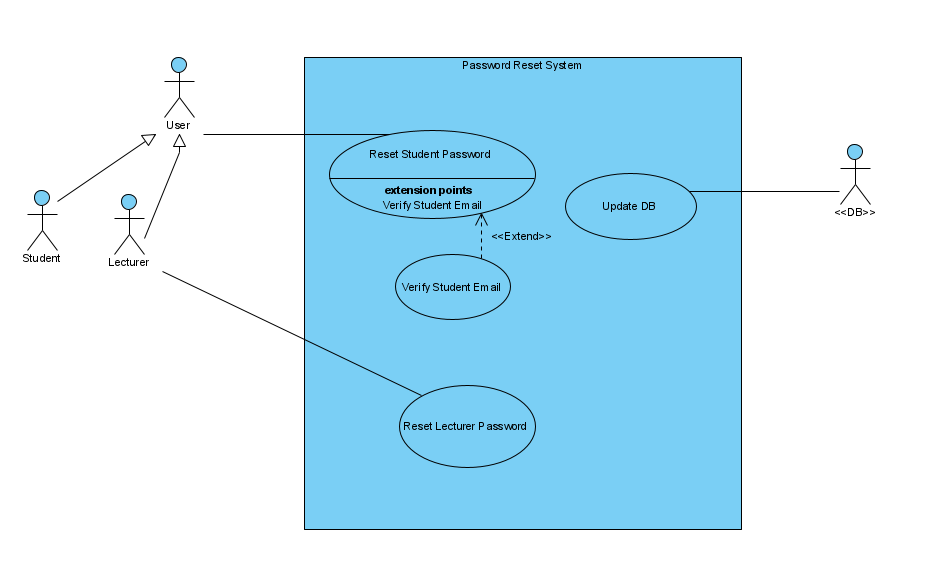
**Test Schedule Create & Booking Test Lab Use Cases**



|  |  |
| --- | --- |
| **Name** | **Create Test Schedule** |
| **Identifier** | **UC 05** |
| **Description** | This use case describes how a Lecturer will setup up the Lab Booking Test Schedule. |
| **Goal** | Lecturer is able to setup up the Test Schedule. |
| **Preconditions** | 1. User must be logged into the system. |
| **Frequency** | Lecturer is able to create as many Test Schedules as required. |
| **Basic Flow** | 1. The use case begins when the User (Lecturer) wishes to setup a Test Schedule for Students to book the Lab. 2. The User creates the Booking Test Schedule. 3. The use case ends when the Test Schedule is updated to the DB and the schedule is viewable to the User. |
| **Alternative Flow** | N/A |
| **Post Conditions** | 1. Test Schedule has been created and time slots are available for booking by the student. |
| **Actors** | UCOL Lecturer |
| **Included Use Cases** | N/A |
| **Excluded Use Cases** | N/A |
| **Notes** | N/A |

|  |  |
| --- | --- |
| **Name** | **Remove/Delete, Update Student Test Lab Booking** |
| **Identifier** | **UC 06** |
| **Description** | This use case describes how a Lecturer will perform the following functions against Test Lab Booking: -   * Remove / Delete Student booking * Update Student booking |
| **Goal** | Lecturer is able to change the Test Lab Booking made by the student. |
| **Preconditions** | 1. User must be logged into the system. 2. Students have booked in time against the Test Schedule. |
| **Frequency** | Lecturer is able to change the student booking as required (no limit to the number of changes). |
| **Basic Flow** | 1. The use case begins when the User (Lecturer) wishes to change the Students Lab booking in the Test Schedule. 2. The User is able to select the Student booking to Remove/Delete or Update. 3. The system successfully updates the student record. 4. The student record is updated or removed/deleted from the Test Schedule. 5. The system successfully updates the DB. 6. The use case ends when the Test Schedule has been changed. |
| **Alternative Flow** | N/A |
| **Post Conditions** | 1. Student or Lecturer is able to view the changes made and see that the system has been updated. |
| **Actors** | UCOL Lecturer |
| **Included Use Cases** | N/A |
| **Excluded Use Cases** | N/A |
| **Notes** | User (Lecturer) is able to create, remove, update and delete a number of Student Test Lab bookings. |

**Password Reset (Lecturer / Student) Use Cases**



|  |  |
| --- | --- |
| **Name** | **Password Reset – Lecturer** |
| **Identifier** | **UC 07** |
| **Description** | This use case describes how a Lecturer can Reset the Student Password & their own Password. |
| **Goal** | Lecturer is able to reset a Student’s Forgotten Password or reset their own Password. |
| **Preconditions** | 1. User must be logged into the system. |
| **Frequency** | Lecturer is able to perform this function multiple times as when required. |
| **Basic Flow** | 1. The use case begins when the User (Lecturer) wishes to reset the Student Password or their own Password. 2. The User (Lecturer) reset’s the Student Password. 3. The System sends an email verification to the Student email address to advise the Password has been reset. 4. The Student User is able to enter their User ID and Password with their new password. 5. The User (Lecturer) resets their own Password. 6. The System resets the user Password for the Lecturer. 7. The use case ends when the Student or Lecturer is able to successfully log into the System with the reset password. |
| **Alternative Flow** | N/A |
| **Post Conditions** | 1. User (Lecturer / Student) Password has been reset. |
| **Actors** | UCOL Lecturer, UCOL currently enrolled Student |
| **Included Use Cases** | N/A |
| **Excluded Use Cases** | N/A |
| **Notes** | N/A |

|  |  |
| --- | --- |
| **Name** | **Password Reset – Student** |
| **Identifier** | **UC 08** |
| **Description** | This use case describes how the Student can reset their own password. |
| **Goal** | Student is able to reset their Password. |
| **Preconditions** | 1. User must be logged into the system. |
| **Frequency** | Student is able to perform this function multiple times as when required. |
| **Basic Flow** | 1. The use case begins when the User (Student) wishes to reset their password. 2. The System advises the Lecturer a Password reset is required. 3. The System sends an email verification to the Student email address to advise the Password has been reset. 4. The use case ends when the Student is able to successfully log into the System with the reset password. |
| **Alternative Flow** | N/A |
| **Post Conditions** | 1. Student Password has been successfully reset. |
| **Actors** | UCOL Lecturer, UCOL currently enrolled Student |
| **Included Use Cases** | N/A |
| **Excluded Use Cases** | N/A |
| **Notes** | N/A |