Student Registration System – Play! Framework 2.2.0

Design and Functional Specification

Project Requirements and Architecture

<November 13, 2013>

Owners and List of Contacts

Name	Email	Phone	Role
Antony Drew			Developer, QA, Docs
Azizullah Parsa			Developer, QA, Docs
Juan Antonio Fuentes			Developer, QA, Docs

Signoffs

Phase	Name	Date	Signature
Initial Doc.	Juan Antonio Fuentes	11/13/2013	
Revisions	Antony Drew	12/1/2013	

Revision History

Date	Reason for change(s)	Author(s)
12/1/2013	Updated flow charts and SQL info	

Table of Contents

About this Document

Purpose and Contents

What this specification does

This document describes functionality of the Student Registration System – Play! 2.2.0 web interphase application for end-users and owners.

What this specification does not do

This is not a project plan. It is a guide for system architecture and development, not for phasing, timelines or deliverables.

A 'Living Document'

Finally, this specification will change, continuously, as the project proceeds. We will add details and edit existing information as the database structure and functionality evolve in the course of the project.

Assumptions

The user has some general knowledge about computing overall.

Questions and Comments

If you have questions or comments regarding this document, contact: Antony Drew

Software Overview

Product description

This MVC / Play! Framework application is an all-in-one student registration system (URS). The system was programmed in Java and implemented using the Play! Framework application with persistent data stored in a MySQL server.

Product functional capabilities

This product provides a comprehensive registrar system that can be run on a local server or deployed on the cloud using platform as a service (PaaS) tools such as Heroku or Cloudbees.

User characteristics

Users will typically be students, faculty, staff or registrars in a university with their respective authorizations as deemed by the registrar or system administrators.

User operations and practices

This is a friendly, straightforward web application has slightly different interphases for students, faculty and registrars. Currently enrolled students are allowed to look at their grades and courses, as well as register for available classes for the upcoming semester according to their assigned registration date. Faculty can keep track of the courses that they will be teaching as well as the students that are currently enrolled in such courses. The registrar will have access and authorization to add, deactivate and override users as well as to set the new courses to teachers and assign enrollment times. There are also several validators that will provide any feedback in case of user error.

General constraints

User must have access to a smart phone or computer with internet access for web browsing.

Assumptions

The user has some general knowledge of how to navigate the internet via a web browser.

Other software

If a user or developer want to **re-compile** code, then other software will be necessary:

- **JAVA(JDK 6)** or higher
- **ECLIPSE IDE** (helpful, but not required)
- **PLAY 2.2.0** framework
- **HEROKU** package (including SCALA) and registration

System Architecture

Description:

The student registration system developed encompassed three core projects within the total system. Each project was divided in different layers, and could build upon itself by following the same iteration and order. The layer that handles all of the processes the system will handle are located in the business layer. The layer that deals with the persistent data is the database layer, and the layer that is responsible for displaying all of the information in a safe and neat fashion is the view layer. Between these three layers there are certain services that enable the three of them to work together.

Utilizing the Play! Framework allowed for the fast development of the application by implementing model-view-controller (MVC) methods. The Play! Frameworks allowed us to employ convention over configuration so that the code could be easily configured and corrected in a hot-swappable environment.

The core areas and data structures utilized in the Play! Framework within the university registration system include the following – for **more precise details on classes and functions**, please see the HTML folder called "**JavaDocs**":

i. **JAVA**
   a. **Controllers**
      i. Teachers.java
      ii. Students.java
      iii. Employees.java
      iv. Application
   b. **Models**
      i. Teacher.java
      ii. Student.java
      iii. Registrar.java
      iv. User.java
      v. Course.java
      vi. StudentOrder.java
   c. **Views**
      i. Index.scala.html
      ii. Main.scala.html
      iii. Student.scala.html
      iv. Teacher.scala.html
      v. Course.scala.html
      vi. User.scala.html
   d. **Dal**
      i. DAO.java
      ii. DBHelper.java
      iii. Service.java

    e. **Test**
        i. ApplicationTest.java
        ii. IntegrationTest.java
        iii. RoutesTest.java

ii. **HTML (URL)**
    a. URL is hosted/provided by HEROKU

iii. **Web Client**
    a. Web server is hosted/provided by HEROKU

iv. **General Data Structures Utilized:**
    a. Arrays
    b. Lists
    c. Array Lists

Logical View

The architecture is divided into three logical layers in order to maintain highly cohesive projects/classes within the system, increase code reusability/scalability and ease of code maintenance.

- **Business Layer**
  The business layer is the core area within the system that handles that functionality and logic behind the registrar system.

- **Database Layer**
  The database layer is layer that takes care of storing persistent data. This project utilizes MySQL v5.6 to store and retrieve the data entered by the users. The database structure revolves around the user being the central part of the system, and the relationships that exist between the user and the resources available.

- **Presentation Layer**
  The view layer in Play! Framework encompasses a full set of tools that are offered on the framework itself. It sets the layout using scala.html files that are accessible once the application is running on the web, or on a local server. It then uses the controller to coordinate the actions available and fetch data to the user.

View Layer

Figure 0 – Login GUI Diagram

Below is the deployed system's log-in for all users. It is the first page displayed no matter what route the user takes ("**admin"** user **& "admin"** password is default). It establishes a session based on the user's authentication.

Figure 1 – Home Page GUI Diagram

Below is the deployed system's main / home GUI for the admin user.

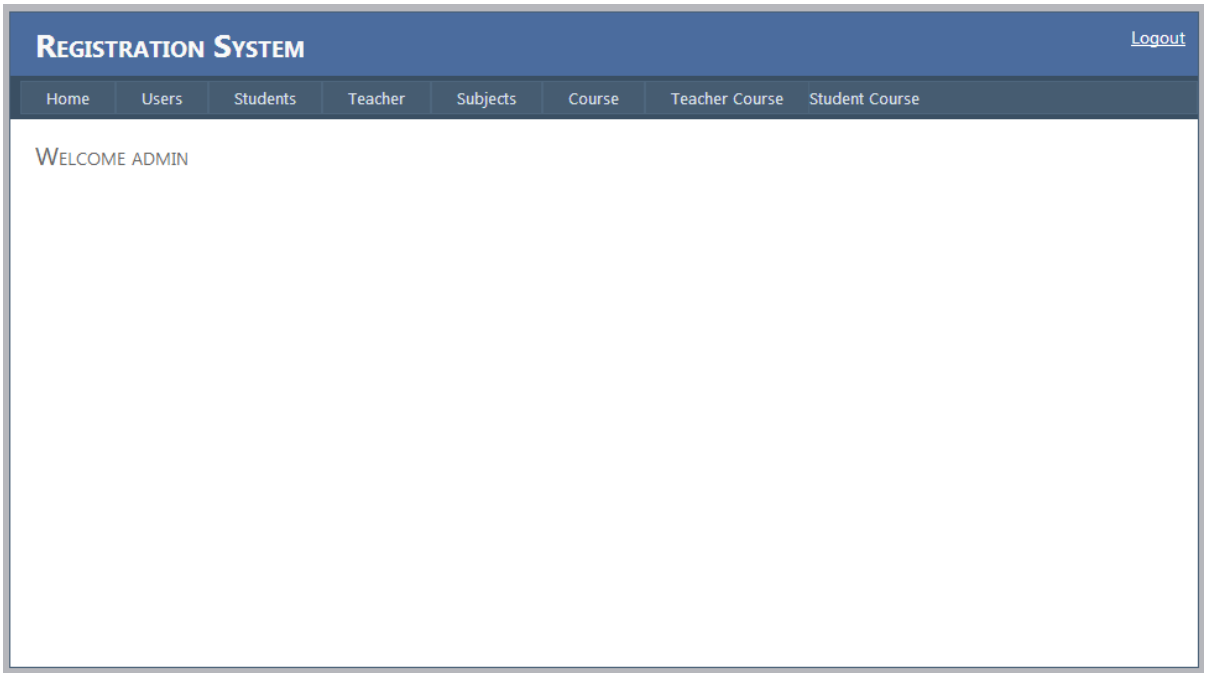


Figure 2 – User GUI Diagram

Below is the deployed system's user GUI for the admin user.

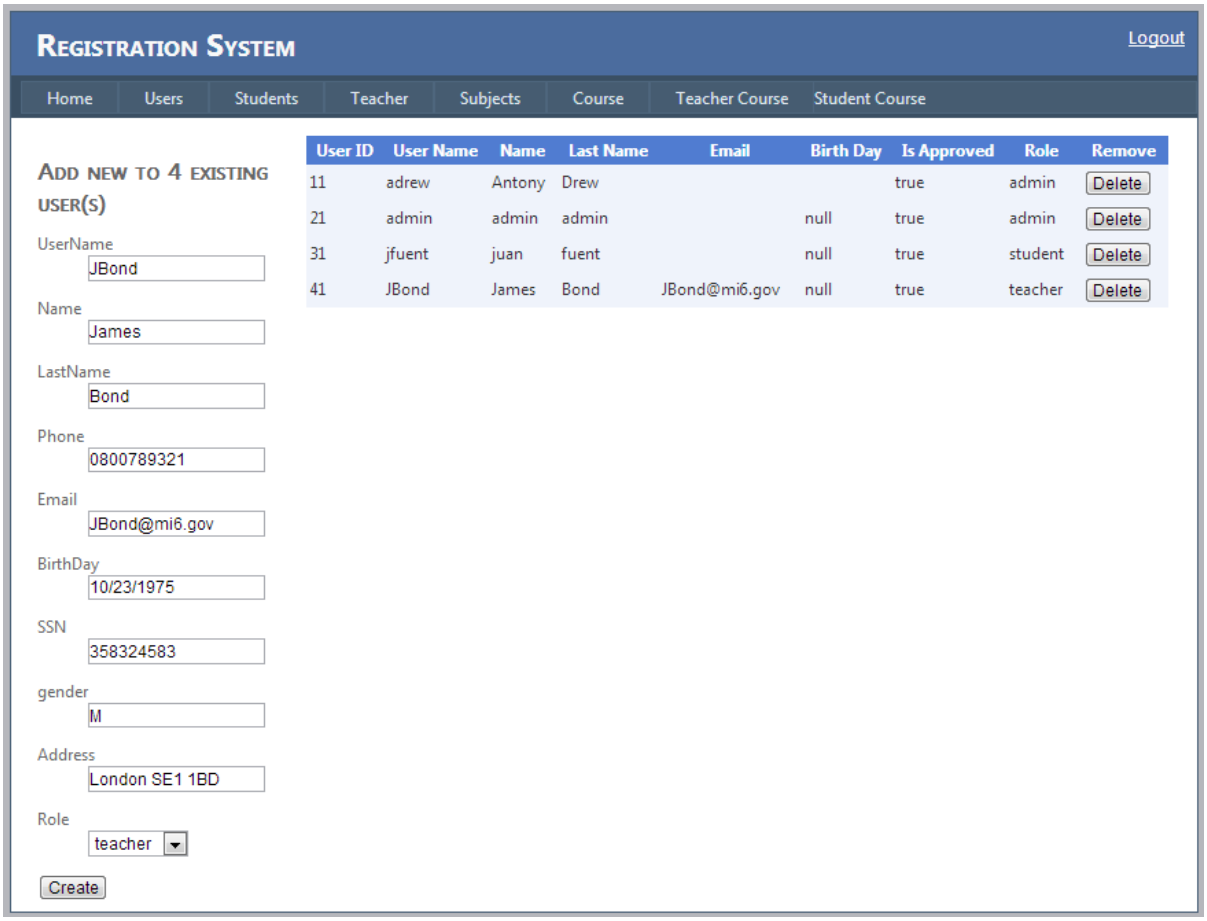


Figure 3 – Student GUI Diagram

Below is the deployed system's student GUI for the admin user.

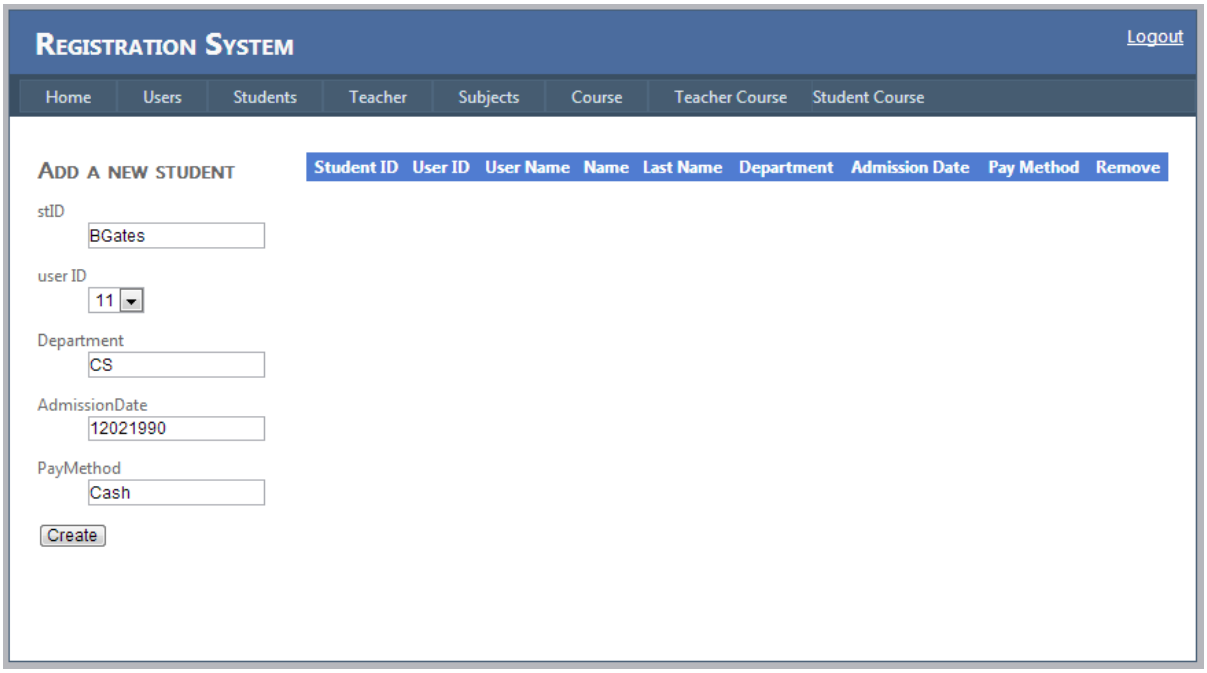


Figure 4 – Teacher GUI Diagram

Below is the deployed system's teacher GUI for the admin user.

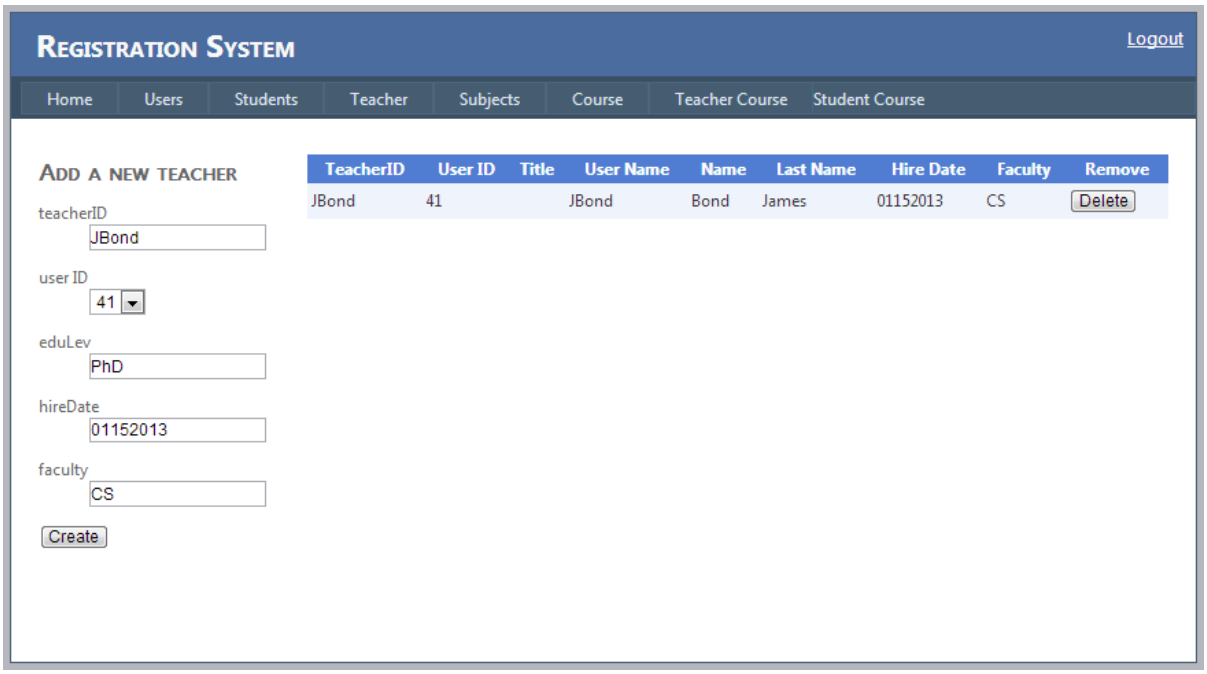


Figure 5 – Subjects GUI

Below is the deployed system's subject GUI for the admin user.

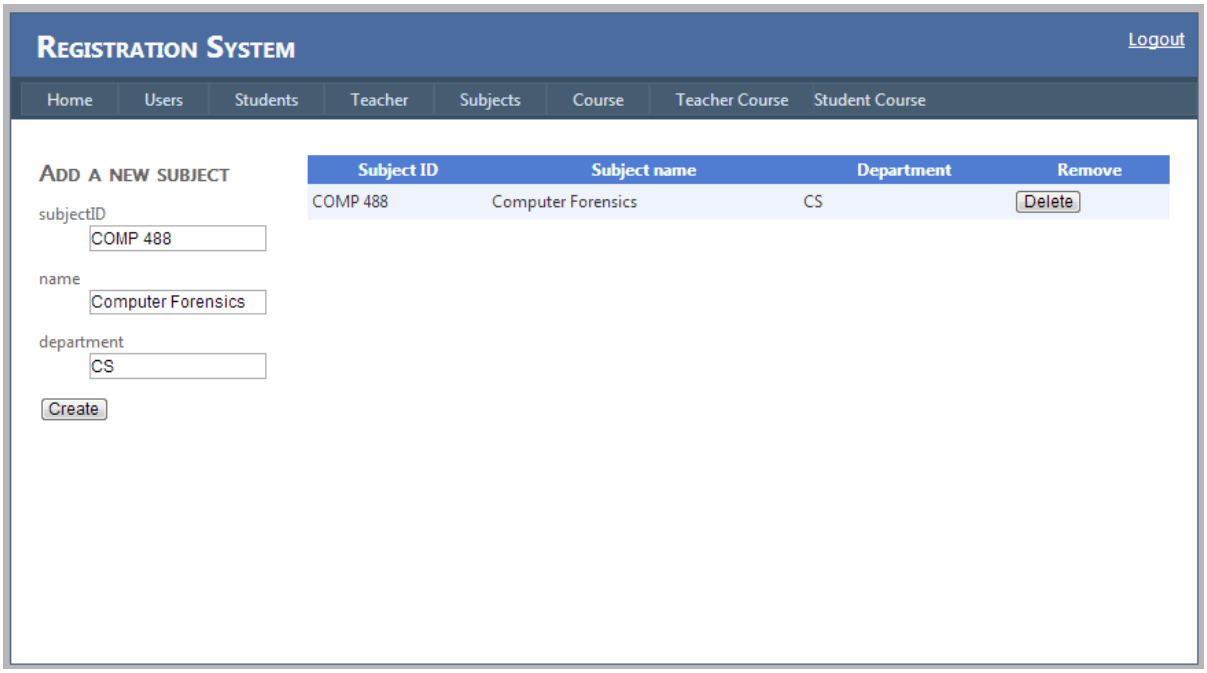


Figure 6 – Course GUI

Below is the deployed system's course GUI for the admin user.

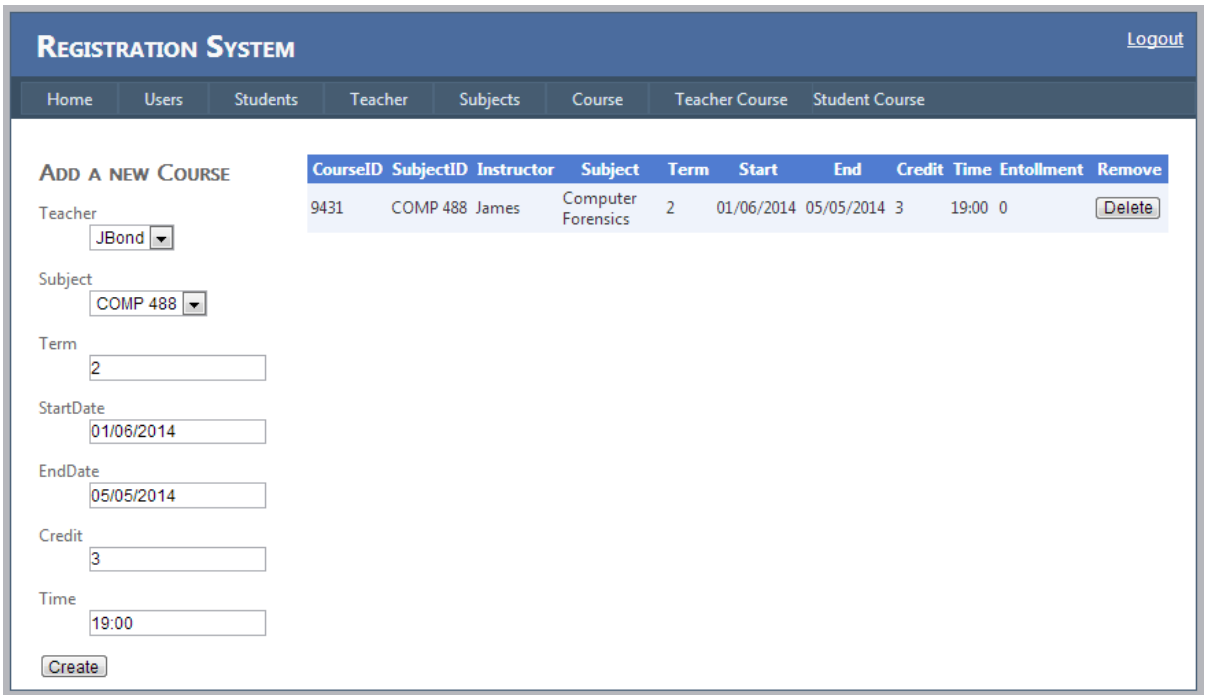


Figure 7 – Teacher Course

Below is the deployed system's teacher course selection GUI for the admin user.

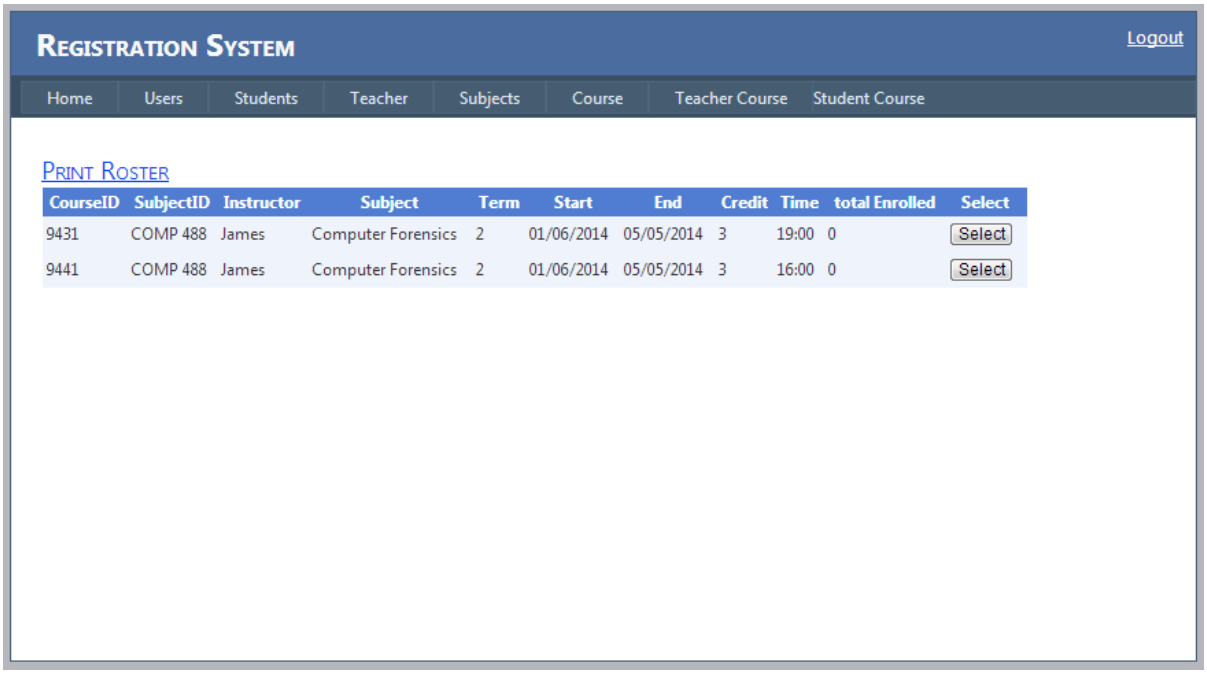


Figure 8 – Student Course

Below is the deployed system's student course selection GUI for the admin user.

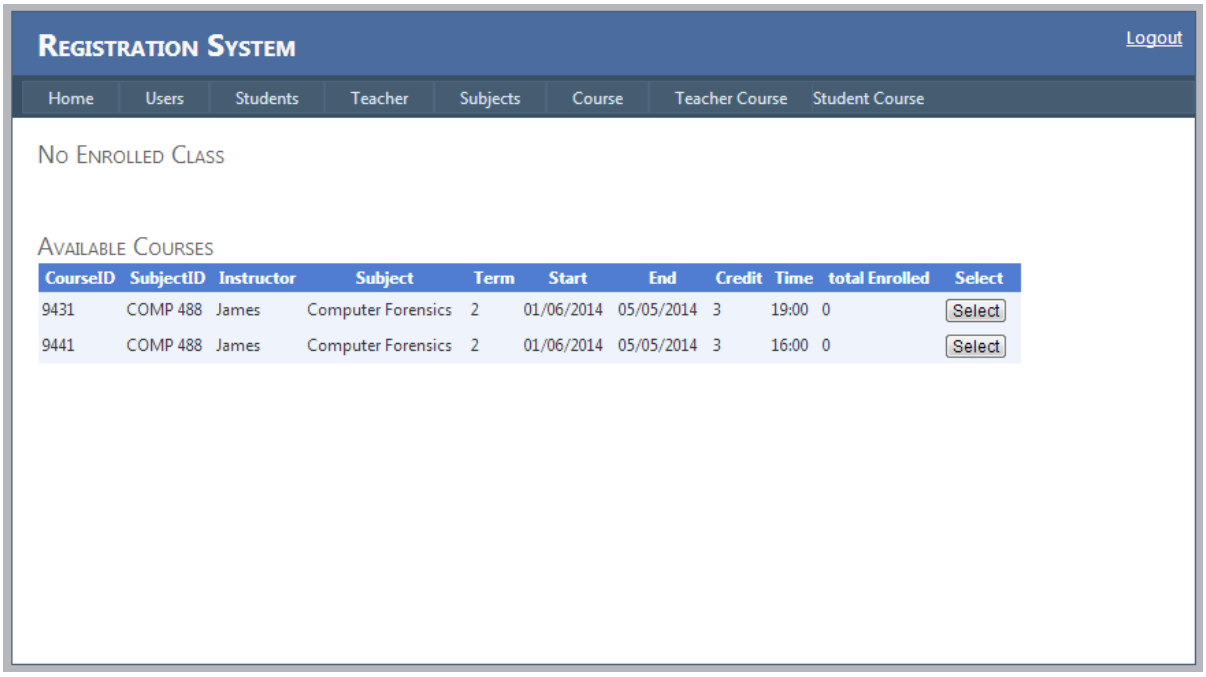
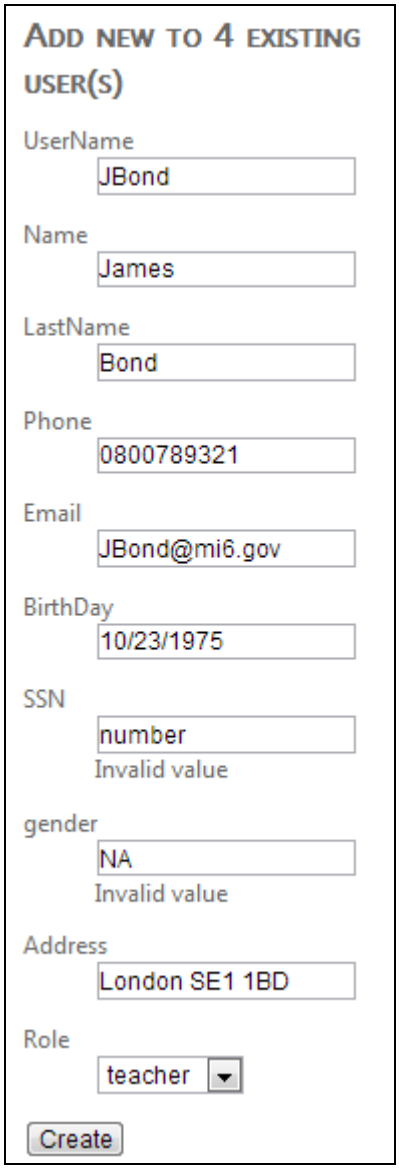


Figure 9 – User Validation

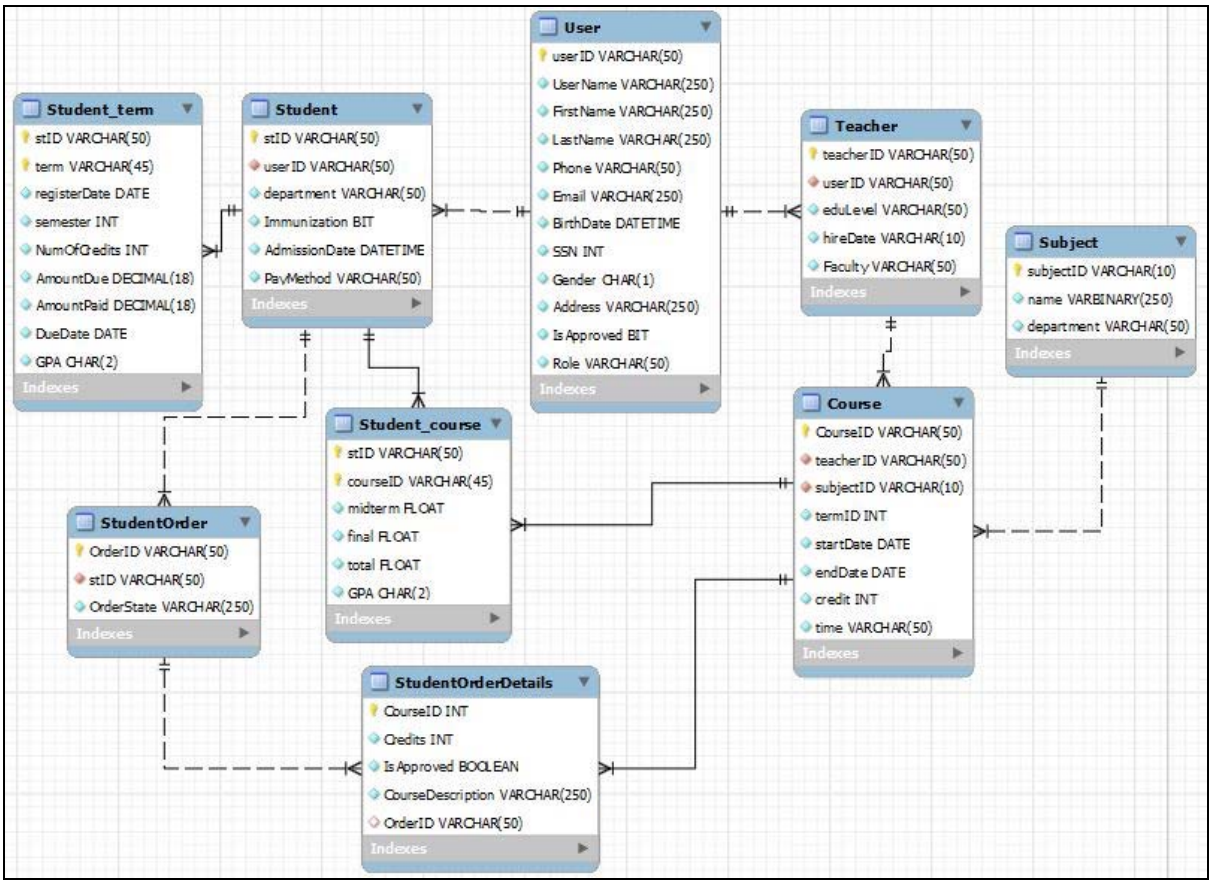
Below is an error example for the system's user GUI validation for the registrar / admin user.



Database Diagrams

Figure 10 – EER Database Diagram – Database Layer

Below is the EER for the database layer – please see document "**EER MySQL**" for more:



Class Diagrams

The overall class diagram for the entire project is below – please see the separate PDF doc "**Class Diagram0**" for more detail:

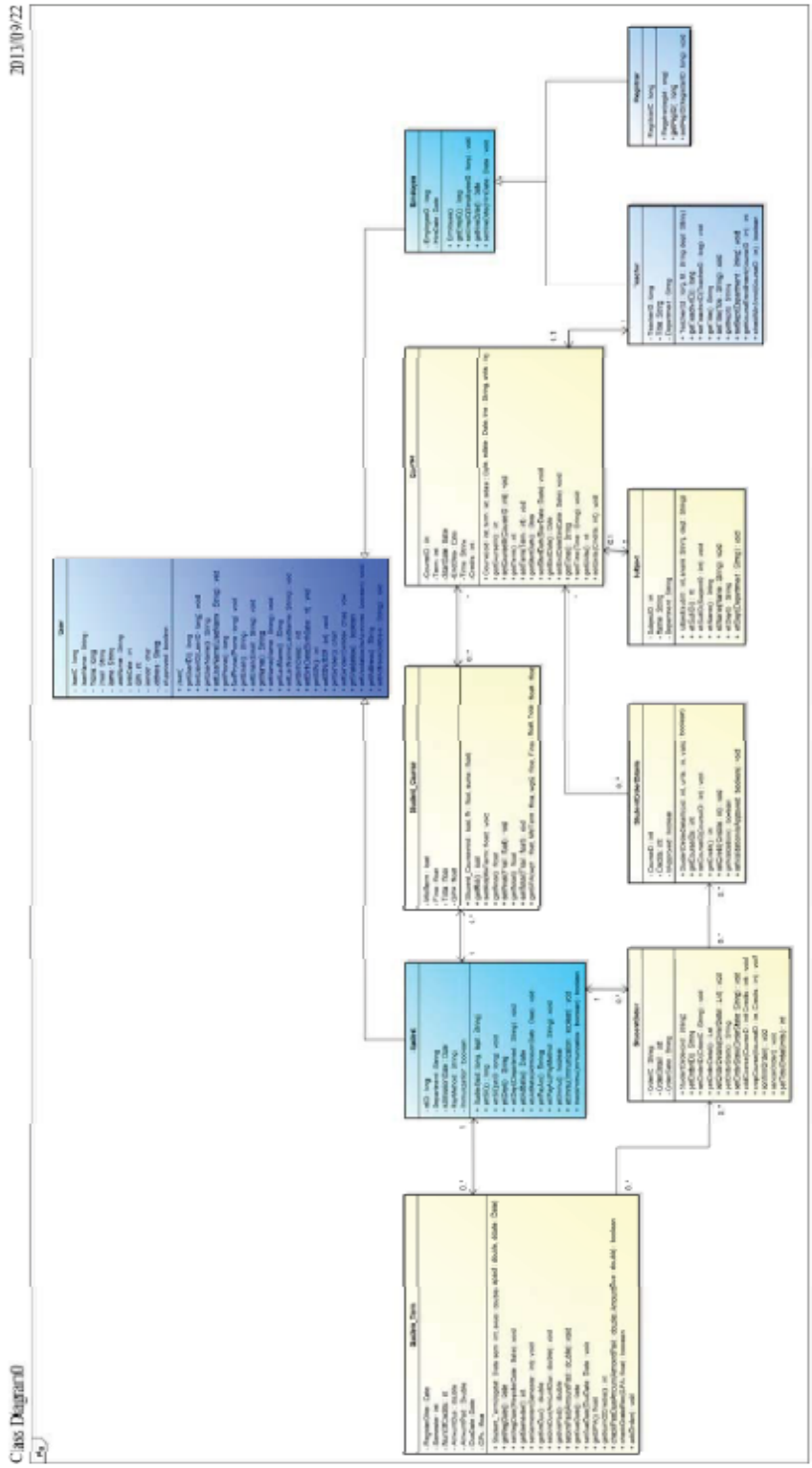


Figure 11 – Class Diagrams – User

Below is the class diagram for the User Class.

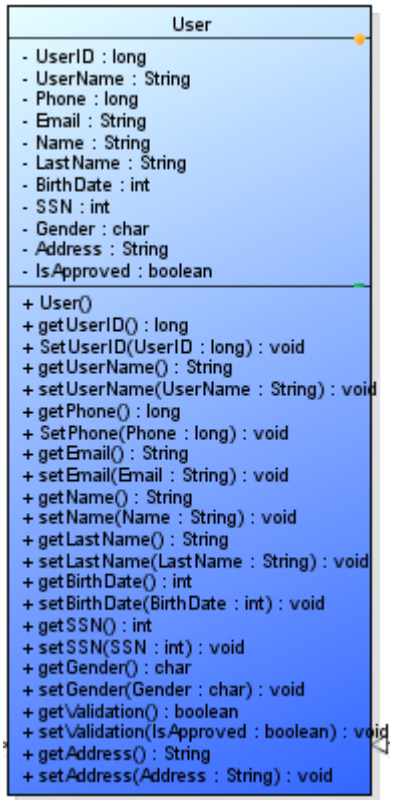


Figure 12 – Class Diagrams – Student

Below is the class diagram for the Student.

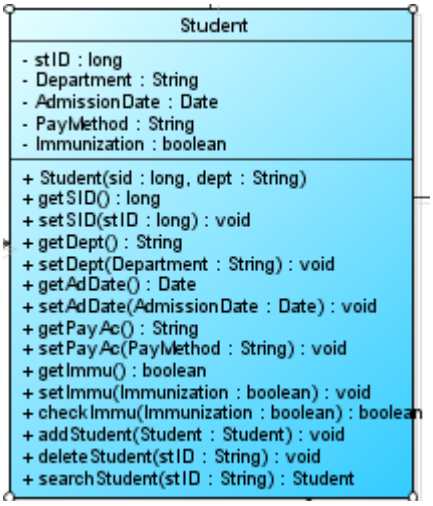


Figure 13 – Class Diagram – Student_Term

Below is the class diagram for the Student_Term.

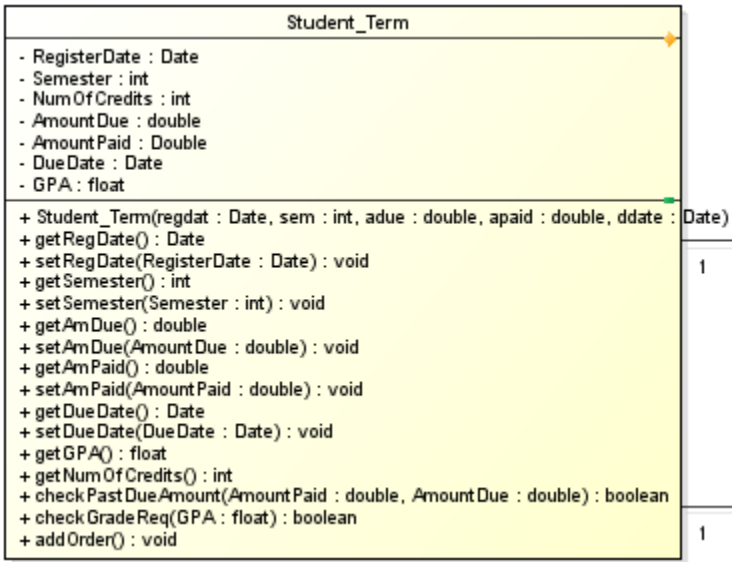


Figure 14 – Class Diagram – Student Order

Below is the class diagram for the Student Order.

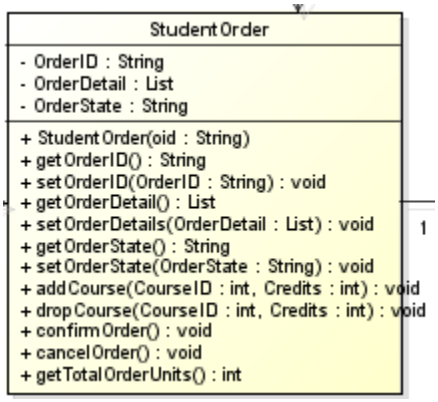


Figure 15 – Class Diagram – StudentOrderDetails

Below is the class diagram for the StudentOrderDetails.

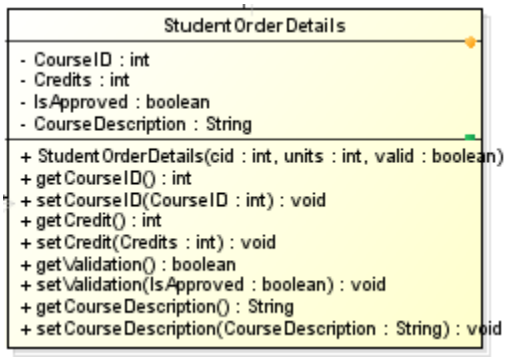


Figure 16 – Class Diagram – Student_Course

Below is the class diagram for the Student_Course.

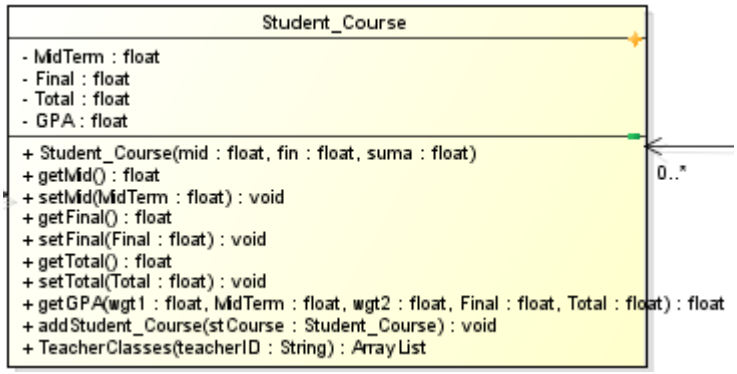


Figure 17 – Class Diagram – Subject

Below is the class diagram for the Subject.

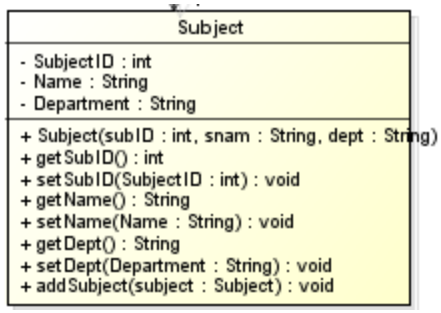


Figure 18 – Class Diagram – Teacher

Below is the class diagram for the Teacher.

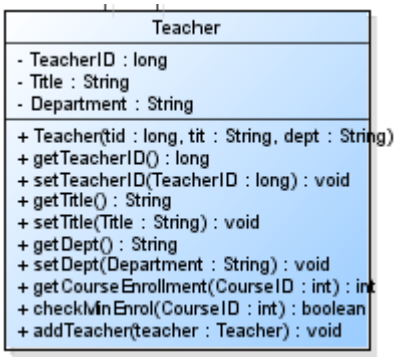


Figure 19 – Class Diagram – Course

Below is the class diagram for the Course.

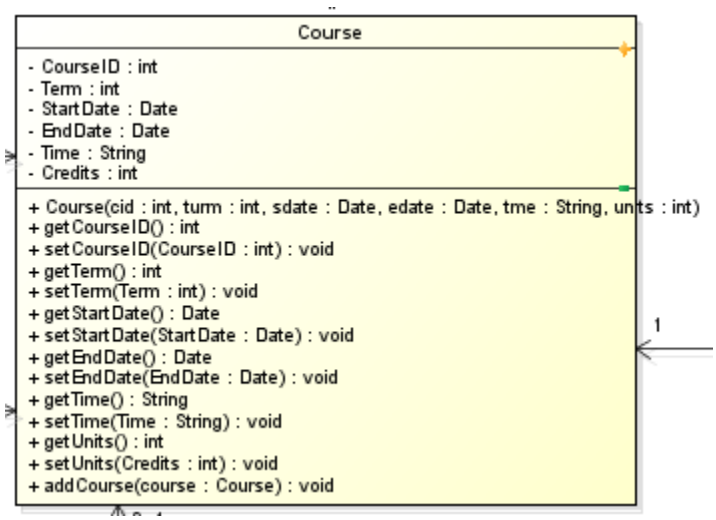


Figure 20 – Class Diagram – Employee

Below is the class diagram for the Employee.

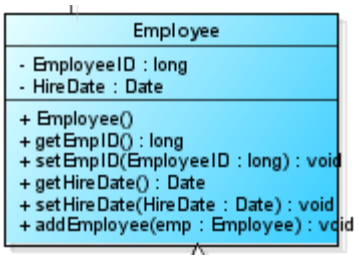


Figure 21 – Class Diagram – Registrar

Below is the class diagram for the Registrar.

Database Schema

Figure 22 – Database Schema

Below is the database schema for the system's table – for more details, please refer to the separate database document "**MySQL DB Specs**" :

Tables

Name	Description
course	Contains course catalog
registrar	Contains registrar employees
student	Contains students
student_course	Contains enrolled student courses
student_term	Contains various enrolled student terms
studentorder	Contains various student orders
studentorderdetails	Contains various student order details to normalize data and break many-to-many relationship between courses and orders
subject	Contains subjects for various courses
teacher	Contains teacher employees
user	Super-class user

Table: course

Columns

Name	Type	Description
CourseID	varchar(50)	Primary key course id
teacherIDb	varchar(50)	Teacher user id
subjectIDb	varchar(10)	Subject id
termID	int	Term id (Fall/Spring or 1-or-2)
startDate	date	Course start date
endDate	date	Course end date
credit	int	Number of credits per course
time	varchar(50)	Meeting time of course

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	CourseID	Primary key course id
subjectID_idx	Non-unique	subjectIDb	Join to Subject table
teacherID_idx	Non-unique	teacherIDb	Join to Teacher table

Table: registrar

Columns

Name	Type	Description
regID	varchar(50)	Primary key registrar employee id
userIDc	varchar(50)	User id

Indexes

Name	Type	Columns	Description
Registrar_User1_idx	Non-unique	userIDc	Join to User table
PRIMARY	Unique	regID	Primary key registrar employee id

Table: student

Columns

Name	Type	Description
stID	varchar(50)	Primary key student id
userIDb	varchar(50)	User id
department	varchar(50)	Dept/School
Immunization	bit	Is immunized? Yes/No or 1-or-0
AdmissionDate	datetime	Admit date
PayMethod	varchar(50)	Credit card or checking account string

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	stID	Primary key student id
userID_idx	Non-unique	userIDb	Join to User table

Table: student_course

Columns

Name	Type	Description
stIDb	varchar(50)	Primary key student id
courseID	varchar(45)	Primary key course id
midterm	float	Midterm course grade
final	float	Final course grade
total	float	Total course grade
GPA	char(2)	Course GPA

Indexes

Name	Type	Columns	Description
courseID_idx	Non-unique	courseID	Join to Course table
PRIMARY	Unique	stIDb, courseID	Primary keys student id and course id

Table: student_term

Columns

Name	Type	Description
stIDe	varchar(50)	Primary key student id
term	varchar(45)	Primary key term id
registerDate	date	Registration date
semester	int	Semester (Fall/Spring or 1-or-0)
NumOfCredits	int	Total number of credits of all courses in term
AmountDue	float	Total amount of money due to school
AmountPaid	float	Total amount paid by student to school
DueDate	date	Due date of student payment
GPA	char(2)	Total GPA of term

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	stIDe, term	Primary keys student id and term id

Table: studentorder

Columns

Name	Type	Description
OrderID	varchar(50)	Primary key order id
stIDd	varchar(50)	Primary key student id
OrderState	varchar(250)	Order state (open,closed, cancelled)
term	varchar(45)	Term

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	OrderID, stIDd	Primary keys order id and student id
stID_idx	Non-unique	stIDd	Join to Student table
term_idx	Non-unique	term	Join to Student_term table

Table: studentorderdetails

Columns

Name	Type	Description
stIDc	varchar(45)	Primary key student id
subOrderID	varchar(45)	Primary Sub-order id
CourseIDb	varchar(50)	Course id
Credits	int	Total Credits of course order
IsApproved	tinyint	Is approved to enroll in course?
CourseDescription	varchar(250)	Description of course
OrderID	varchar(50)	Overall order id

Indexes

Name	Type	Columns	Description
OrderID_idx	Non-unique	OrderID	Join to StudentOrder table
PRIMARY	Unique	stIDc, subOrderID	Primary keys student id and sub-order id

Table: subject

Columns

Name	Type	Description
subjectID	varchar(10)	Subject id
name	varbinary(250)	Name of subject
department	varchar(50)	Department name where subject resides

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	subjectID	Primary key subject id

Table: teacher

Columns

Name	Type	Description
teacherID	varchar(50)	Teacher id
userIDd	varchar(50)	User id
eduLevel	varchar(50)	Highest level of education
hireDate	varchar(10)	Hire date
Faculty	varchar(50)	Faculty

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	teacherID	Primary key teacher id
userID_idx	Non-unique	userIDd	Join to User table

Table: user

Columns

Name	Type	Description
userID	varchar(50)	User id
UserName	varchar(250)	User name
FirstName	varchar(250)	First name
LastName	varchar(250)	Last name
Phone	varchar(50)	Phone number
Email	varchar(250)	Email address
BirthDate	datetime	DOB
SSN	int	Social security number
Gender	char(1)	Male/Female
Address	varchar(250)	Address/Residence
IsApproved	bit	IsAprroved? (for use or enrollment)
Role	varchar(50)	Role

Indexes

Name	Type	Columns	Description
PRIMARY	Unique	userID	Primary key user id

Specific Function Description

1. Product

Description

The general flow of the project can be described by the following flow chart:

Get Login Splash Screen

Authenticate User and Create Session ←→ Checks DB for Credentials

Get Home Page → Launch Pages User / Teacher / Student

Get User Page

Add New User → Sends Information to DB → Get Refreshed User Page

Get Teacher Page

Add New Teacher → Sends Information to DB → Get Refreshed Teacher Page

Get Student Page

Add New Student → Sends Information to DB → Get Refreshed Student Page

Get Subjects Page

Add New Student → Sends Information to DB → Get Refreshed Student Page

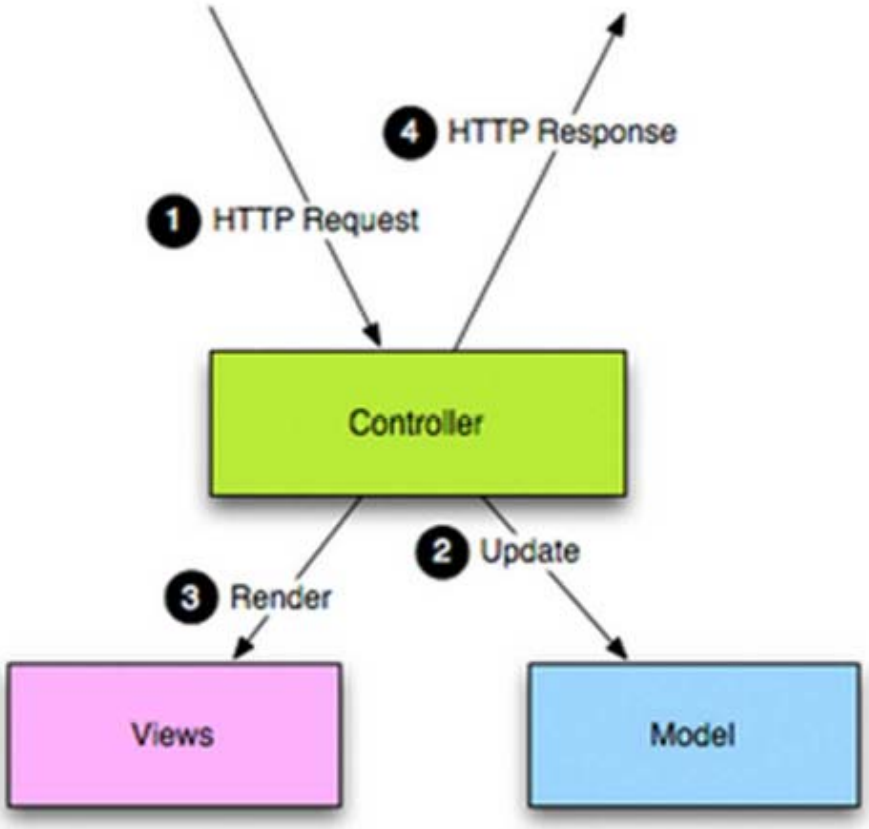
Get Teacher Course

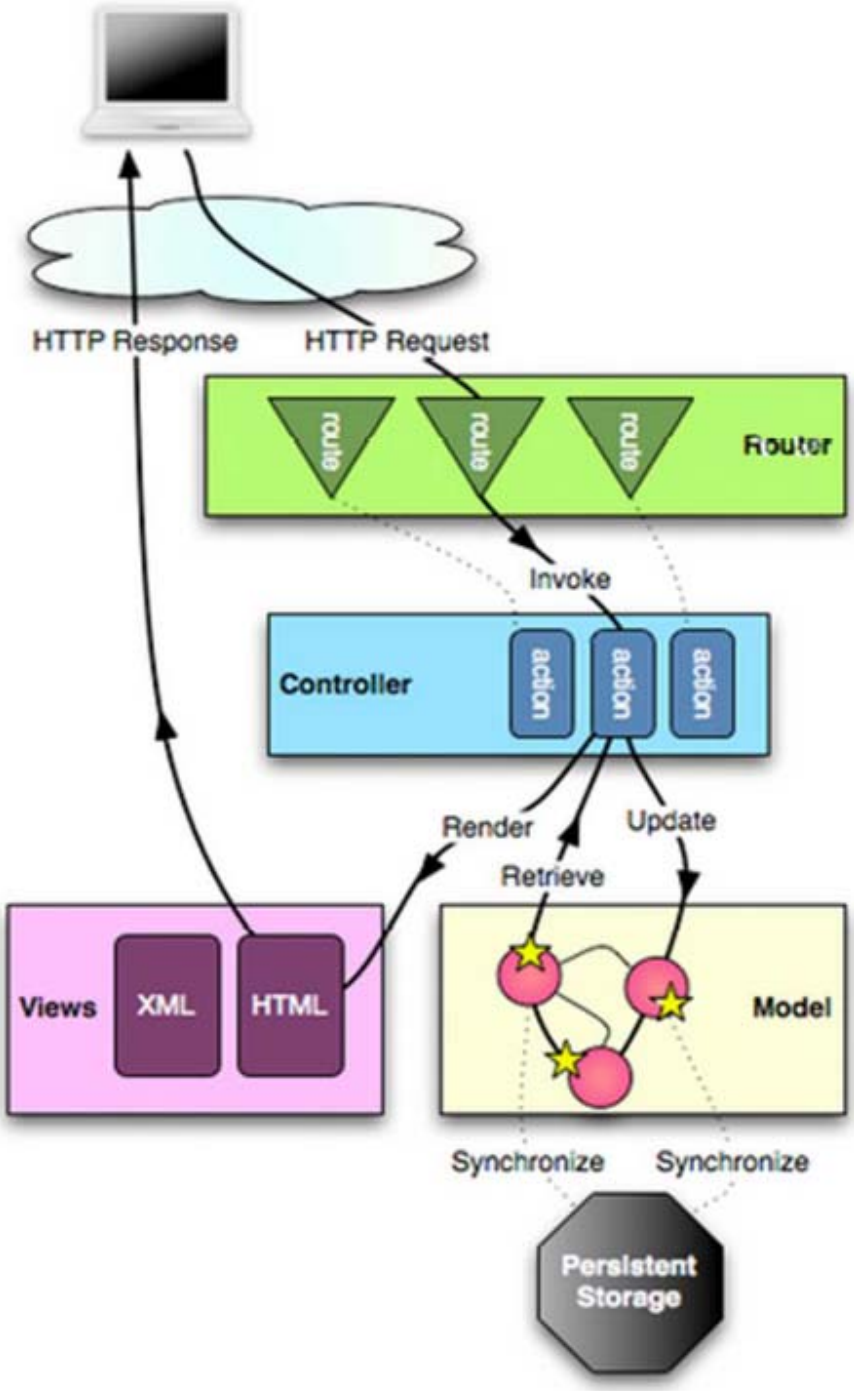
Get Available Course List → Sends Selected Course to DB → Get Refreshed Page

Get Student Course

Get Available Course List → Sends Selected Course to DB → Get Refreshed Page

Figure 23 – MVC Interactions & Work Flow




HTTP Response
HTTP Request
route
route
route
Router
Invoke
Controller
action
action
action
Render
Update
Retrieve
Views
XML
HTML
Model
Synchronize
Synchronize
Persistent Storage

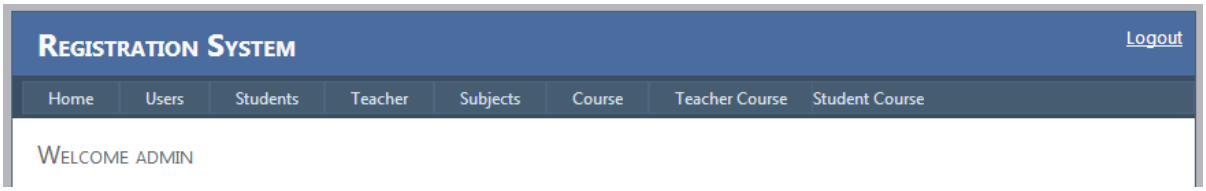
User Interface

When the user navigates to the local host or to the published URL by simply using their browser, the application will be available on the hosted server. The user can then authenticate via the login screen and then access the resources that are available. There are 7 links available that will navigate the user to access various resources based on their authentication (“**admin”** user **& “admin”** password is default):

Figure 24 – Login

A new session will be started based on the users’ credentials. The **7** links are: **Home, Student, Teacher, Subjects, Course, Teacher Course and Student Course:**

Figure 25 – Linked Tab View after Login



The users credentials are displayed in the welcome screen. Figure 25 corresponds to the Admin credentials. Users will have to end their session by clicking on the Logout button on the top right corner of the screen.

Inputs

All user-adjustable inputs based on their credentials are located on the left-side of the browser. The user (based on credentials) can then enter teacher / student / course information and or sign up to teach a class or take a class. See the Figures # 20 - 21 for more details.

The inputs are protected against an illegal input by the user, such as not entering a 8 digit SSN when entering a student or teacher.

Processing

During processing, the page will reload and the mouse will turn into a **loading circle** icon (depending on user configurations) so the user knows that something is occurring. If there are any errors, the user will be redirected to an **error page** based on the error encountered**.** This will serve as a guide to inform the user about what to do next, most likely this will be a matter of simply refreshing the browser window.

Actions

Though we did make STORED PROCEDURES for the MySQL database, HEROKU security restrictions prevent them from running (so we had to **re-factor** in JAVA) – here is a list:

Procedures

Name	Description
GetStudent(varchar)	Returns student info like name and address
GetStudentEnroll(varchar)	Returns student info like enrollment
GetStudentOrder(varchar)	Returns student info like order details
GetTeacher(varchar)	Returns teacher info like name and address
GetTeacherEnroll(varchar)	Returns student info like enrollment of classes taught by professor
AddStudent(varchar)	Inserts new student into database
DeleteStudent(varchar)	Deletes student from database
AddTeacher(varchar)	Inserts new teacher into database
DeleteTeacher(varchar)	Deletes teacher from database
GetCourseCatalog(varchar)	Returns list of courses
AddCourse(varchar)	Inserts new course into database
DeleteCourse(varchar)	Deletes course from database
Insert_Drop_UpdateTables	Inserts, drops or updates all tables

Procedure: GetStudent(varchar)

Parameters

Name	Type	Direction	Description
_stiD	varchar	Input	Student id

Procedure: GetStudentEnroll(varchar)

Parameters

Name	Type	Direction	Description
_stiD	varchar	Input	Student id

Procedure: GetStudentOrder(varchar)

Parameters

Name	Type	Direction	Description
_stiD	varchar	Input	Student id
_OrderID	varchar	Input	Order id

Procedure: GetTeacher(varchar)

Parameters

Name	Type	Direction	Description
_teacherID	varchar	Input	Teacher id

Procedure: GetTeacherEnroll(varchar)

Parameters

Name	Type	Direction	Description
_teacherID	varchar	Input	Teacher id

Procedure: AddStudent(varchar)

Parameters

Name	Type	Direction	Description
_stiD	varchar	Input	Student id

Procedure: DeleteStudent(varchar)

Parameters

Name	Type	Direction	Description
_stiD	varchar	Input	Student id

Procedure: AddTeacher(varchar)

Parameters

Name	Type	Direction	Description
_teacherID	varchar	Input	Teacher id

Procedure: DeleteTeacher(varchar)

Parameters

Name	Type	Direction	Description
_teacherID	varchar	Input	Teacher id

Procedure: GetCourseCatalog(varchar)

Parameters

Name	Type	Direction	Description
_regID	varchar	Input	Registrar id

Procedure: AddCourse(varchar)

Parameters

Name	Type	Direction	Description
_regID	varchar	Input	Student id
_courseID	varchar	Input	Course id

Procedure: DeleteCourse(varchar)

Parameters

Name	Type	Direction	Description
_regID	varchar	Input	Student id
_courseID	varchar	Input	Course id

Procedure: Insert_Drop_UpdateTables(varchar)

Parameters

Name	Type	Direction	Description
_tableName	varchar	Input	Table name

Output Screens:

Figure 26 – Login GUI Diagram

Below is the deployed system's log-in for all users. It is the first page displayed no matter what route the user takes. It establishes a session based on the user's authentication.

Figure 27 – Home Page GUI Diagram

Below is the deployed system's main / home GUI for the admin user.

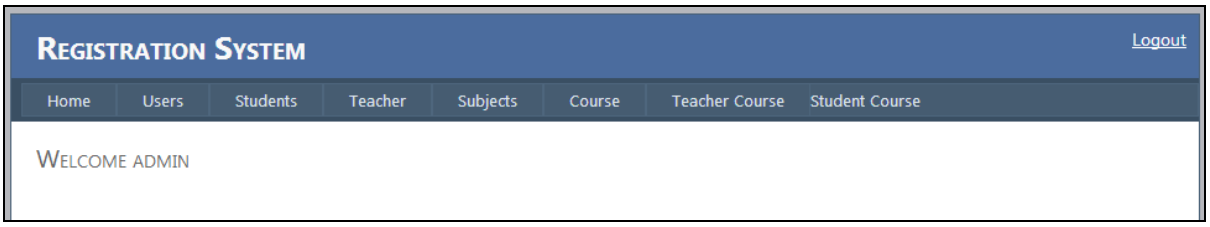


Figure 28 – User GUI Diagram

Below is the deployed system's user GUI for the admin user.

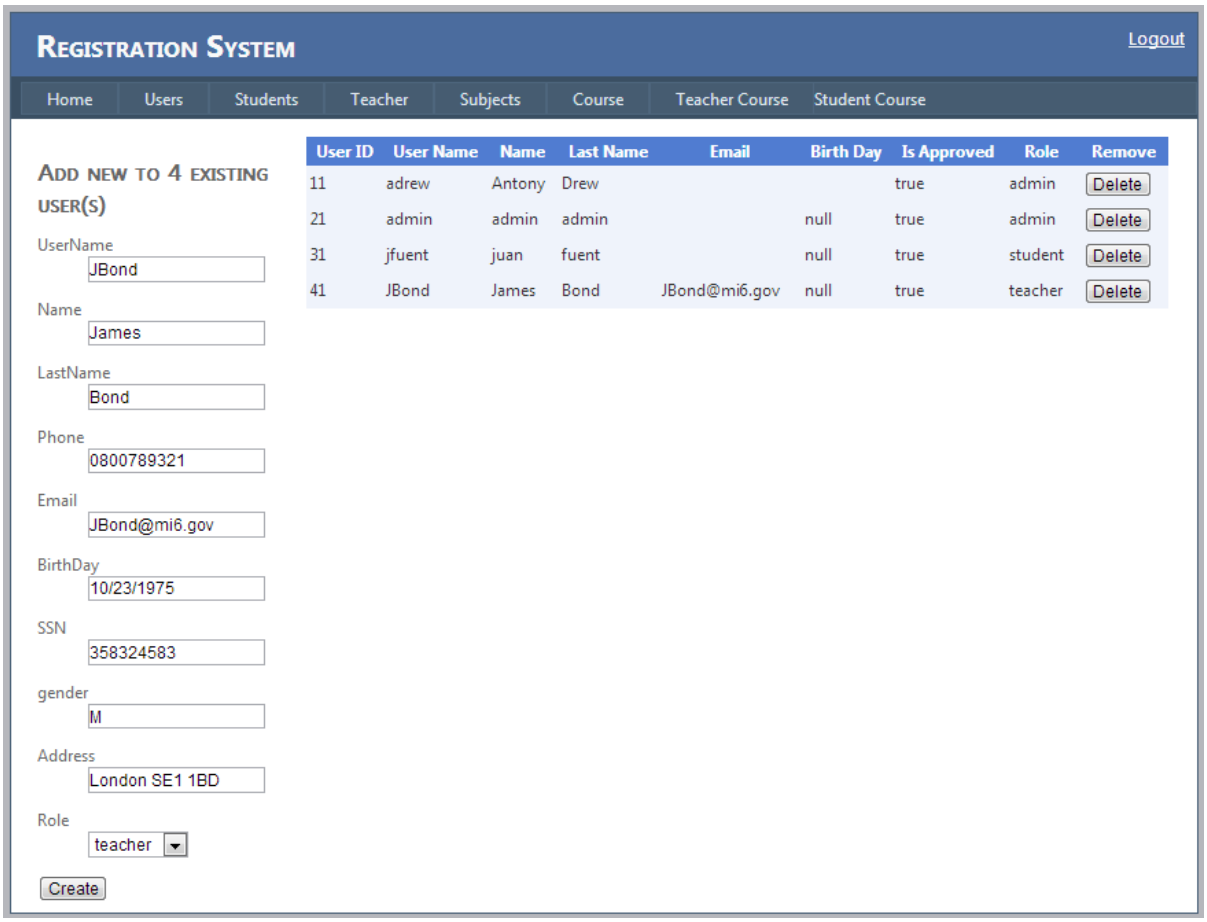


Figure 29 – Student GUI Diagram

Below is the deployed system's student GUI for the admin user.

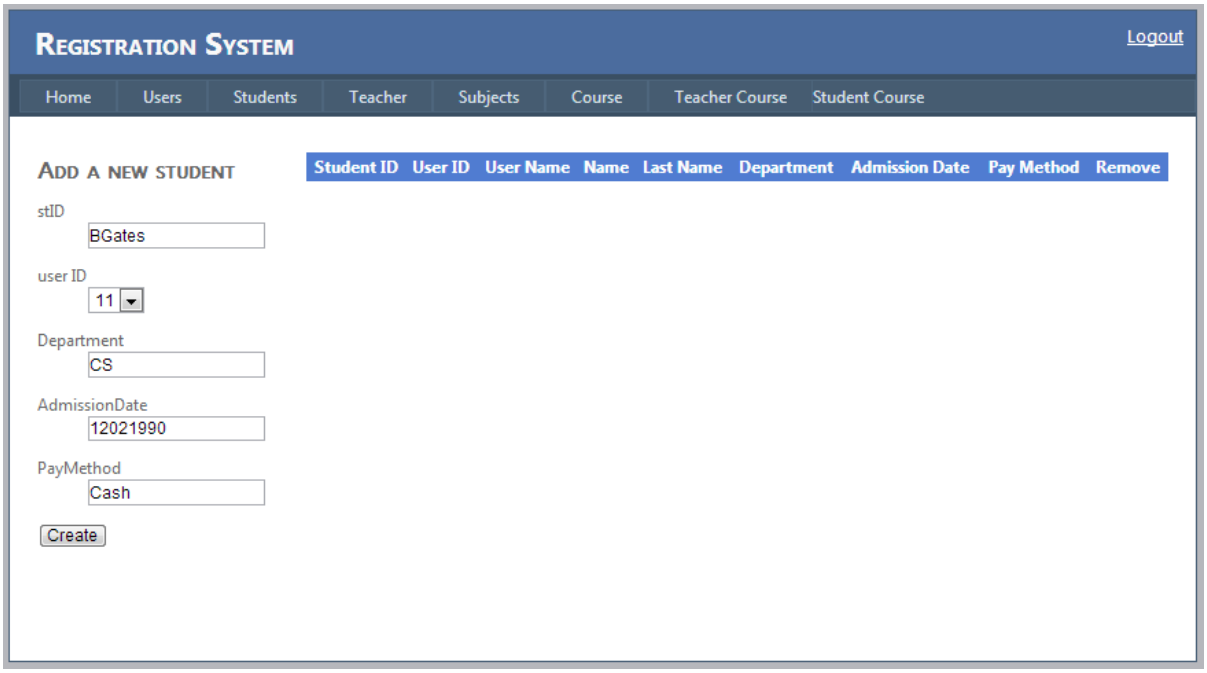


Figure 30 – Teacher GUI Diagram

Below is the deployed system's teacher GUI for the admin user.

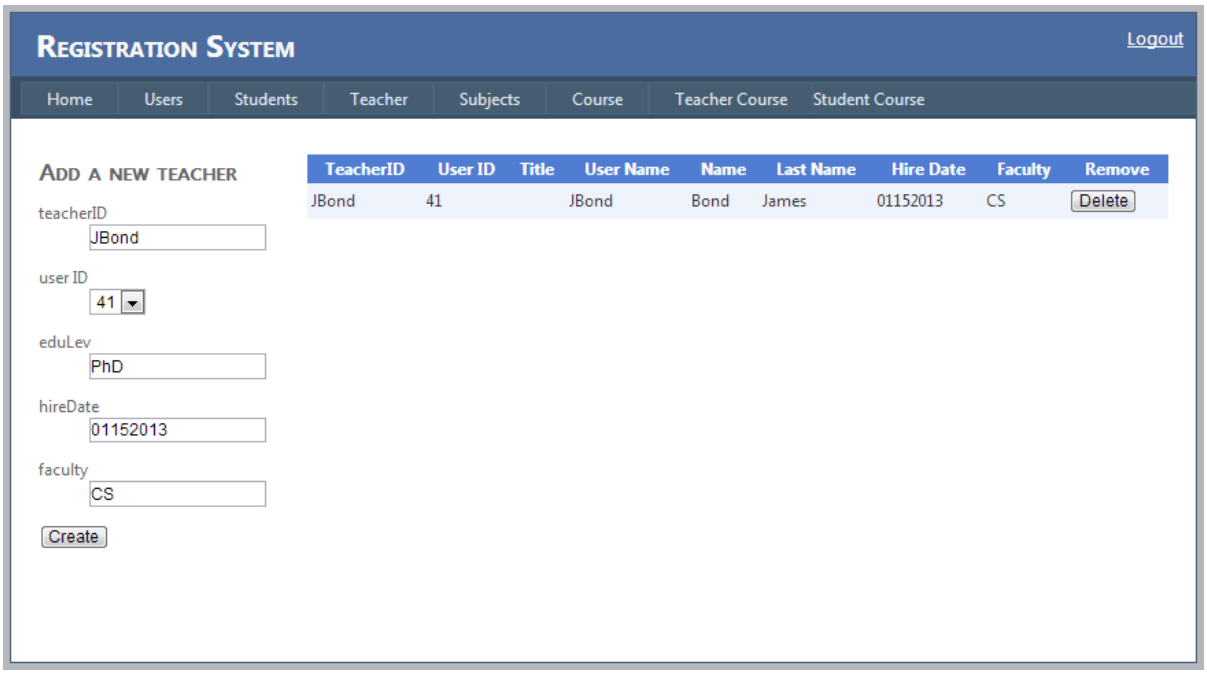


Figure 31 – Subjects GUI

Below is the deployed system's subject GUI for the admin user.

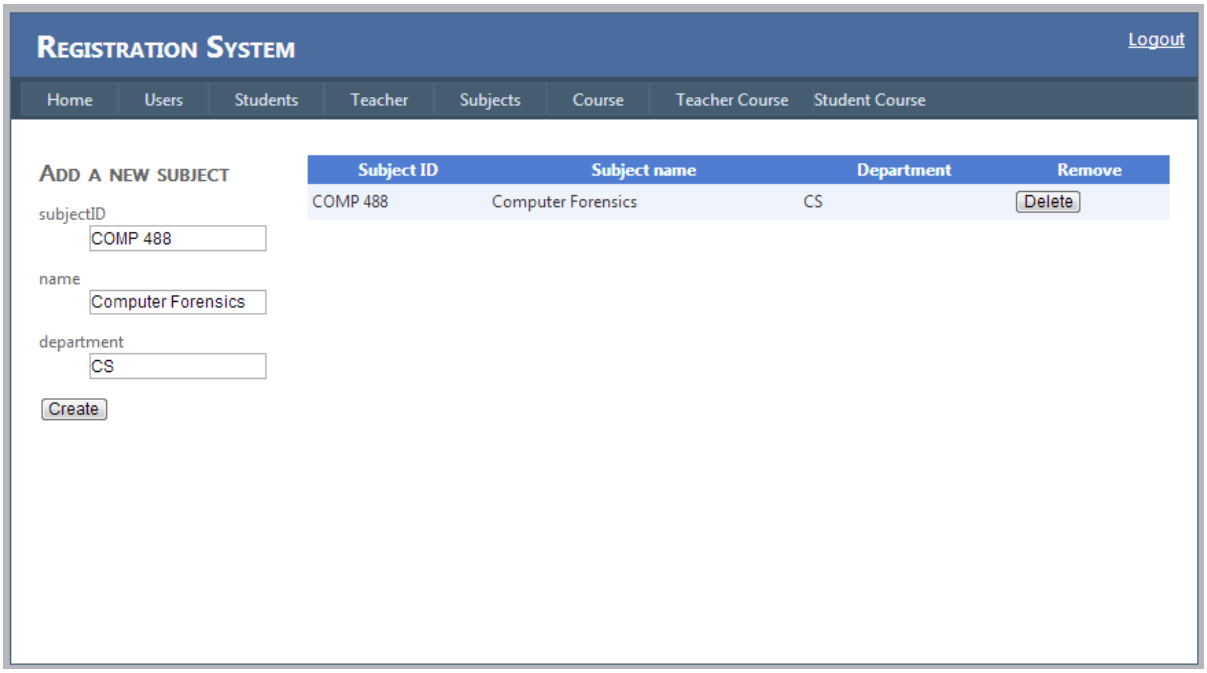


Figure 32 – Course GUI

Below is the deployed system's course GUI for the admin user.

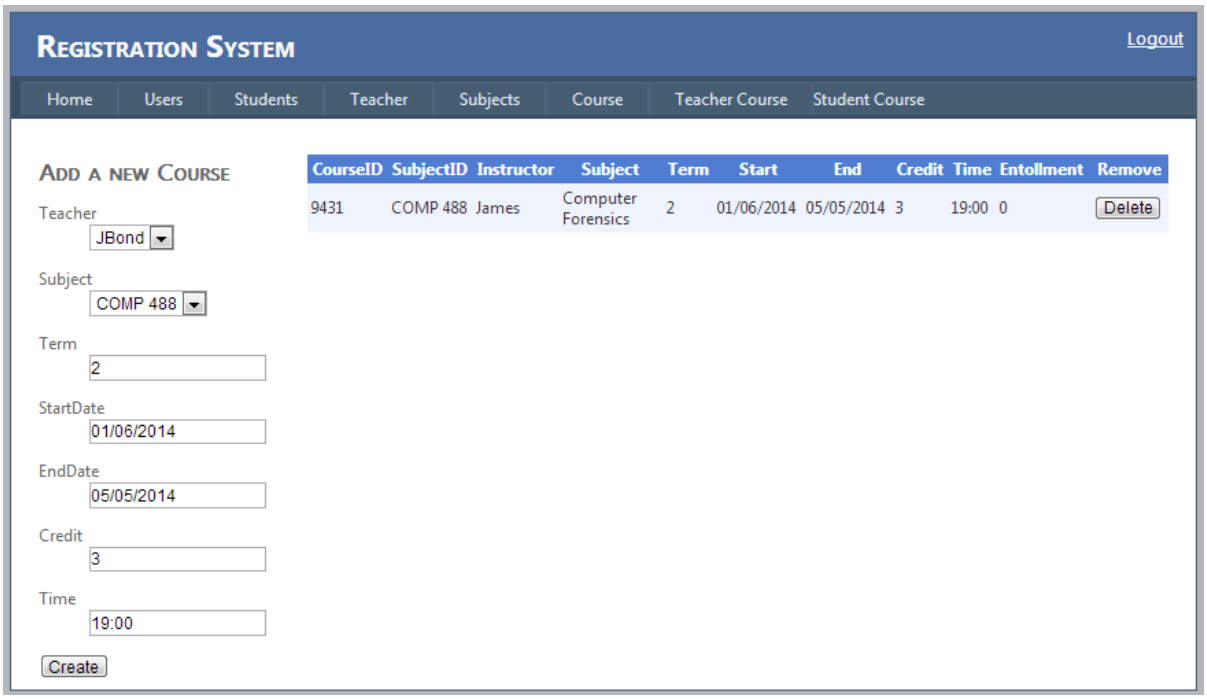


Figure 33 – Teacher Course

Below is the deployed system's teacher course selection GUI for the admin user.

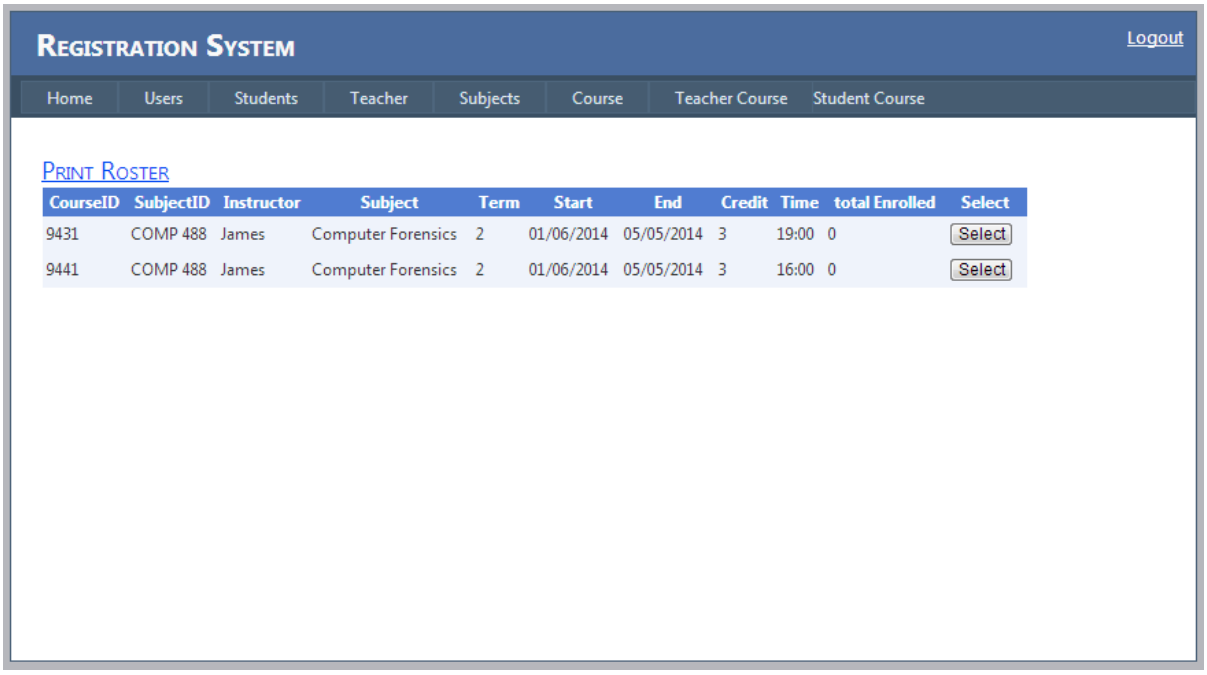
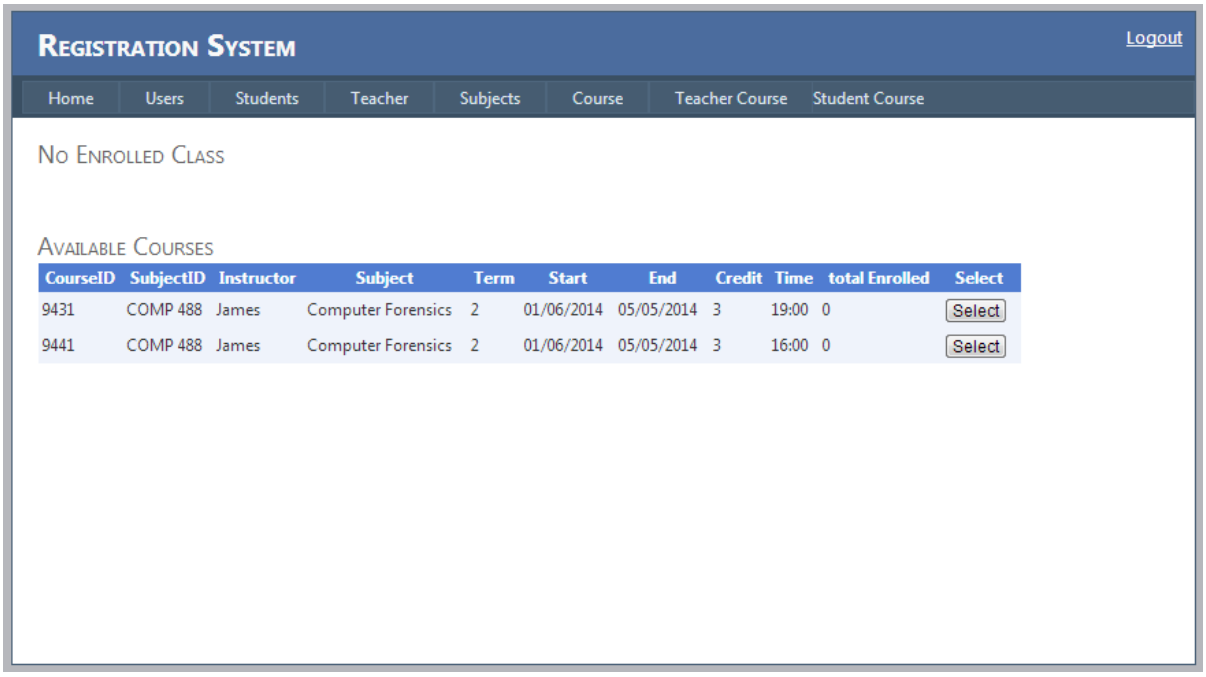


Figure 34 – Student Course

Below is the deployed system's student course selection GUI for the admin user.



Security

Security is handled on the MySQL SERVER side via user name and password prompts. When the system is deployed on the cloud it includes a wide variety of security features based on the service provider or running environment. Security is also inherently provided internally by the user's default web browser.

Heroku Security

Heroku's physical infrastructure is hosted and managed within Amazon's secure data centers and utilize the Amazon Web Service (AWS) technology. Amazon continually manages risk and undergoes recurring assessments to ensure compliance with industry standards. Amazon's data center operations have been accredited under:

- ISO 27001
- SOC 1 and SOC 2/SSAE 16/ISAE 3402 (Previously SAS 70 Type II)
- PCI Level 1
- FISMA Moderate
- Sarbanes-Oxley (SOX)

Further details may be found at https://www.heroku.com/policy/security