Design and Testing Documentation

Cosc 499 - Group 6: TA Scheduler

Table of Contents

[**1.0 Document Introduction**](#_j92pjwatg2wk) **2**

[**2.0 Project Introduction**](#_b7syl8etoga3) **2**

[**3.0 Terminology**](#_cmze3umrtmno) **2**

[**4.0 User Groups**](#_nc8fj22kp3xd) **3**

[**5.0 Use Cases**](#_oszazpbw82xk) **3**

[5.1 Use Case Conditions](#_donl2z2qe3tc) 3

[5.2 Use Case Diagram](#_xhcdjfmmjfry) 14

[5.3 Activity Diagram](#_8793km3udazq) 15

[5.4 Component Diagram](#_m5oq7u6jrmv7) 16

[**6.0 System Architecture**](#_owm2epp5c2lr) **17**

[6.0.1 System Architecture Diagram](#_yur7nyzdjyk3) 17

[6.1 MVC Framework](#_2jolvix522s8) 17

[6.1.1 Overview](#_ify2t0jqi80h) 17

[6.1.2 View](#_3gvrd5rysp0) 17

[6.1.3 Controller](#_62annw7r31xh) 18

[6.1.4 Model](#_vob1hhr70s0z) 18

[6.1.5 PHP - Python Communication](#_iogprzec8knj) 18

[6.2 Sequence Diagrams](#_rnjk3u7udazg) 19

[6.2.1 Users](#_s2soviejtaqi) 19

[6.2.2 Student](#_1jyqwqg6ky3u) 20

[6.2.3 Administrator](#_wj2trfiioe6l) 21

[6.2.4 Administrator - ScheduleTAs](#_7j6cjfixaej) 22

[6.3 Database ER Diagram](#_d7atgxwfc325) 23

[6.4 Data flow Diagram](#_os9qhn2ajroj) 24

[**7.0 Technical Specifications**](#_gso9mfpotul7) **25**

[7.1 Client Side](#_bmd2ehj5rrm5) 25

[7.2 Server Side](#_ldt3fgrjagng) 25

[7.3 MySQL Database](#_mpgad3i10wc8) 25

[7.4 Feasibility of Schedule](#_n5wfo6e5jwbc) 25

[**8.0 Testing**](#_j9i1f1cyzcow) **26**

[8.1 Overview](#_ddd6vkuqahpv) 26

[8.1.2 Unit Testing](#_y3n5lu6x18g8) 28

[8.1.2 Unimplemented Tests](#_qbzcxm217gr3) 28

[8.2 Software/Technologies](#_ddd6vkuqahpv) 29

[8.2.1 PHPUnit](#_ly13wqkq7x14) 29

[8.3 Continuous Integration/Deployment](#_cquaguncdrsg) 29

[**9 Licenses**](#_owawcwndo3d9) **29**

# 

# 1.0 Document Introduction

The purpose of this document is to provide the design details of the TA Allocation Scheduler project. This document will be used as a guideline for the development team and will require verification from the client to ensure the project design is consistent with the project requirements. To ensure the document is comprehensive it includes terminology, use cases, user groups, system architecture, UI mockups, technical specifications, testing and licenses.

# 2.0 Project Introduction

Our client for this Project is Abdallah Mohamed who works for UBCO and has the task of manually assigning TAs to Labs. A time consuming process which this project aims to solve. We aim to build a website where Administrators can make Admin accounts and TA candidates can create an account to apply for a position. TAs will upload their information which administrators will manually and/or automatically sort into the specified time slots in a given course. This sorting can be optimized and displayed. We have a number of stretch goals such as Admins sending emails to professors of each course with their assigned TAs, Admin can email contracts to TAs with their assignments, and TAs can sign in and upload their TA contracts.

# 3.0 Terminology

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Student | A person attending the University in question (in the scope of this project UBCO) |
| Teaching Assistant (TA) | An undergraduate or graduate student that has been hired by the university to assist with a course. The assistance may take place during a lab, lecture, seminar, tutorial or additional office hours. |
| Undergraduate | An undergraduate student is a colloquial term for person taking undergraduate courses. These programs are typically 4 years in length but may vary from person to person. |
| Graduate (Student) | An graduate student is a person continuing their education in a graduate studies program after having completed an undergraduate degree. This typically takes the form of a Masters study or a PhD. |
| Administrator | Any persons working for UBCO given the task of allocating TAs. |
| Professor | Any person that works as an official instructor at UBCO. |
| Seminar, Tutorial, Lab | A Seminar, Tutorial, and Lab are all classified as Sections which belong to a Course. TAs are assigned to Sections. |
| Database (DB) | The online persistent storage used to store any data used by the software. MySQL DB will be used for the project. |
| Google OR-Tools | A free constraint problem solver provided by google. |
| MVC | Model-View-Controller framework for web design. Separates out the code to 3 layers to promote modularity, reusability, and scalability. |

# 4.0 User Groups

1. University of British Columbia
2. Students / TAs
3. Administrators

# 5.0 Use Cases

## 5.1 Use Case Conditions

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | **Use Case Name** | **Actor(s)** |
| 1 | Register Account | Student, Administrator |
| 2 | Login Account | Student, Administrator |
| 3 | Forgot Password | Student, Administrator |
| 4 | Create Application | Student |
| 4.a | Upload Transcript | Student |
| 4.b | Upload Schedule | Student |
| 4.c | Application Form | Student |
| 5 | View Account Info | Student |
| 5.a | Change Info | Student |
| 6 | Add/Remove Sections | Administrator |
| 7 | View List of Courses | Administrator |
| 8 | Upload Course List | Administrator |
| 9 | Create Administrators | Administrator |
| 10 | Manually Assign | Administrator |
| 10.a | View Recommended TA for Section | Administrator |
| 11 | Automatically Schedule | Administrator |
| 11.a | Set Course Restrictions | Administrator |
| 12 | View List of TAs | Administrator |
| 12.a | View TA Info | Administrator |
| 12.b | Change TA Info | Administrator |
| 13 | Review Applications | Administrator |
| 13.a | Accept or Decline | Administrator |
| 14 | Print Reports | Administrator |
| 15 | Email Offers | Administrator |

|  |  |
| --- | --- |
| **ID:** | 1 |
| **Name:** | Register Account |
| **Actor(s):** | Student, Administrator |
| **Flow of Events:** | 1. Navigate to sign up page 2. Enter valid account information 3. Upon creation login and proceed into site |
| **Pre-Conditions:** | 1. Must be logged out 2. Account doesn’t already exist |
| **Post-Conditions:** | 1. User account is created and stored in DB 2. User redirected to login |
| **Description:** | The sign-up/register process for Administrators and TAs to create accounts.  TAs have a basic sign up page.  Administrators have a create admin account page to create more Administrator accounts (Secured) |

|  |  |
| --- | --- |
| **ID:** | 2 |
| **Name:** | Login Account |
| **Actor(s):** | Student, Administrator |
| **Flow of Events:** | 1. Navigate to login page 2. Enter account information 3. Redirect into website upon successful login |
| **Pre-Conditions:** | 1. Must be logged out 2. Account information is valid |
| **Post-Conditions:** | 1. User is redirected inside of site 2. User login stored in Session |
| **Description:** | The login process for Administrators and TAs to login into the website |

|  |  |
| --- | --- |
| **ID:** | 3 |
| **Name:** | Forgot Password |
| **Actor(s):** | Student, Administrator |
| **Flow of Events:** | 1. Click ‘Forgot My Password’ on the login page |
| **Pre-Conditions:** | 1. Must be logged out 2. User fills out recovery form |
| **Post-Conditions:** | 1. User is emailed login recovery steps |
| **Description:** | Password recovery process |

|  |  |
| --- | --- |
| **ID:** | 4 |
| **Name:** | Create Application |
| **Actor(s):** | Student |
| **Flow of Events:** | 1. Navigate to begin application 2. Enter preferred working times 3. Upload Transcript 4. Fill out application |
| **Pre-Conditions:** | 1. User must be logged in 2. User must not have a application started |
| **Post-Conditions:** | 1. Application is complete 2. Application is ready to be submitted |
| **Description:** | The process for creating a TA application and submitting it. This encompases the subtasks of 3.a, 3.b, and 3.c. Only 1 application is allowed to be submitted per term/session. The application is a form with sections for each sub task. |

|  |  |
| --- | --- |
| **ID:** | 4.a |
| **Name:** | Upload Transcript |
| **Actor(s):** | Student |
| **Flow of Events:** | 1. User hits button to bring up file explorer 2. User selects their transcript csv |
| **Pre-Conditions:** | 1. User is logged in 2. User has begun and is in their Application |
| **Post-Conditions:** | 1. CSV transcript file added to form data |
| **Description:** | Upload csv transcript in application |

|  |  |
| --- | --- |
| **ID:** | 4.b |
| **Name:** | Upload Schedule |
| **Actor(s):** | Student |
| **Flow of Events:** | 1. User hits button to bring up file explorer 2. User selects their transcript csv |
| **Pre-Conditions:** | 1. User is logged in 2. User has begun and is in their Application |
| **Post-Conditions:** | 1. Schedule ical added to form data |
| **Description:** | Upload the users ical schedule file |

|  |  |
| --- | --- |
| **ID:** | 4.c |
| **Name:** | Application Form |
| **Actor(s):** | Student |
| **Flow of Events:** | 1. Navigate to questions of the application 2. Answer questions |
| **Pre-Conditions:** | 1. User is logged in 2. User has begun and is in their Application |
| **Post-Conditions:** | 1. Answers are saved in application |
| **Description:** | This section of the application is the questions that the applicant must answer to be evaluated for hire by the administrator. This will be a form that the user can enter their answers into. |

|  |  |
| --- | --- |
| **ID:** | 5 |
| **Name:** | View Account Info |
| **Actor(s):** | Student |
| **Flow of Events:** | 1. User navigates to View Account |
| **Pre-Conditions:** | 1. User is logged in |
| **Post-Conditions:** | 1. Users account info is displayed |
| **Description:** | The view Account info page for users |

|  |  |
| --- | --- |
| **ID:** | 5.a |
| **Name:** | Change Info |
| **Actor(s):** | Student or Admin |
| **Flow of Events:** | 1. User navigates to View Account 2. User select Edit Info |
| **Pre-Conditions:** | 1. User is logged in |
| **Post-Conditions:** | 1. Changes are reflected to the users account |
| **Description:** | A page to change various account information |

|  |  |
| --- | --- |
| **ID:** | 6 |
| **Name:** | Add/Remove Sections |
| **Actor(s):** | Admin |
| **Flow of Events:** | 1. Admin navigates to Add/Remove Course Section 2. Admin fills out desired information 3. Admin accepts changes |
| **Pre-Conditions:** | 1. Admin is logged in |
| **Post-Conditions:** | 1. Course changes are reflected |
| **Description:** | This process allows Admins to add or remove course data |

|  |  |
| --- | --- |
| **ID:** | 7 |
| **Name:** | View List of Courses |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to Course List 2. Admin can apply filter settings 3. Course information displayed |
| **Pre-Conditions:** | 1. Admin is logged in 2. Course data is present for displaying |
| **Post-Conditions:** | 1. Selected course information is displayed |
| **Description:** | This usage case is for displaying course information currently stored in the system for scheduling purposes |

|  |  |
| --- | --- |
| **ID:** | 8 |
| **Name:** | Upload Course List |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to Upload Class Schedule 2. Admin selects a csv file to upload 3. Admin reviews and accepts class schedule |
| **Pre-Conditions:** | 1. Admin is logged in 2. CSV data is valid |
| **Post-Conditions:** | 1. Course data reflected into system |
| **Description:** | This process is for admins to add in course data for scheduling. |

|  |  |
| --- | --- |
| **ID:** | 9 |
| **Name:** | Create Administrators |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to Create Admin Account 2. Admin enters account information 3. Admin hits create |
| **Pre-Conditions:** | 1. Admin is logged in 2. Account info does not exist |
| **Post-Conditions:** | 1. New Admin account created |
| **Description:** | This process creates new administrator accounts from an admin account. |

|  |  |
| --- | --- |
| **ID:** | 10 |
| **Name:** | Manually Assign |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to Course List |
| **Pre-Conditions:** | 1. Admin is logged in 2. Admin has courses to schedule 3. Admin has TAs to schedule 4. Section does not contain a TA |
| **Post-Conditions:** | 1. TA added to Section |
| **Description:** | This process handles assigned a TA to a given section via the recommendation matrix |

|  |  |
| --- | --- |
| **ID:** | 10.a |
| **Name:** | View Recommended TA for Section |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to Course List 2. Admin selects a section to view TA recommendations |
| **Pre-Conditions:** | 1. Admin is logged in 2. Admin has sections to view 3. TA available for assignment |
| **Post-Conditions:** | 1. Recommended TA’s are displayed for selected section |
| **Description:** | This process recommends TAs for the selected section. |

|  |  |
| --- | --- |
| **ID:** | 11 |
| **Name:** | Automatically Schedule |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to Course List |
| **Pre-Conditions:** | 1. Admin is logged in 2. Courses available for scheduling 3. TAs available for scheduling 4. Selection and Restriction set |
| **Post-Conditions:** | 1. Assigned TAs and Sections are shown for review 2. Admin Accepts/Declines schedule |
| **Description:** | This process takes the courses/tas/restrictions and formulates a constraint model for use by Python to solve with Google OR-Tools as a constraint problem, giving a feasible schedule |

|  |  |
| --- | --- |
| **ID:** | 11.a |
| **Name:** | Set Course Restrictions |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to Course List 2. Admin selects Set Restrictions for a given section 3. Admin sets any restrictions per course |
| **Pre-Conditions:** | 1. Admin is logged in 2. Course data available |
| **Post-Conditions:** | 1. Course restrictions reflected into system |
| **Description:** | This process allows admins to view and edit the course restrictions set and saved per course for scheduling. |

|  |  |
| --- | --- |
| **ID:** | 12 |
| **Name:** | List All TAs |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to List TAs |
| **Pre-Conditions:** | 1. Admin is logged in 2. TAs are hired for current term |
| **Post-Conditions:** | 1. TAs are displayed as a list |
| **Description:** | This page allows the administrator to view all currently hired TAs and displays information about them. |

|  |  |
| --- | --- |
| **ID:** | 12.a |
| **Name:** | View TA Info |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to List TAs 2. Admin selects a user listed, clicking their name 3. Admin is redirected to an account page |
| **Pre-Conditions:** | 1. Admin is logged in 2. TAs are hired |
| **Post-Conditions:** | 1. Account info page displayed for selected user |
| **Description:** | This process allows admins to view a TAs profile |

|  |  |
| --- | --- |
| **ID:** | 12.b |
| **Name:** | Change TA Info |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to Edit TA 2. Admin enters account info to edit |
| **Pre-Conditions:** | 1. Admin is logged in 2. TAs are hired |
| **Post-Conditions:** | 1. Edits reflected to users account |
| **Description:** | This process allows admins to edit a TAs account |

|  |  |
| --- | --- |
| **ID:** | 13 |
| **Name:** | Review Applications |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to View Applications |
| **Pre-Conditions:** | 1. Admin is logged in 2. Applications available |
| **Post-Conditions:** | 1. Applications displayed |
| **Description:** | This process allows admins to view applications for the given session/year |

|  |  |
| --- | --- |
| **ID:** | 13.a |
| **Name:** | Accept or Decline |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to View Applications 2. Admin accepts/declines a given applications via the buttons |
| **Pre-Conditions:** | 1. Admin is logged in 2. Applications available |
| **Post-Conditions:** | 1. Application status reflected |
| **Description:** | This process allows admins to accept/decline a given application |

|  |  |
| --- | --- |
| **Name:** | Print Report |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to List TAs 2. Admin selects a TA 3. Admin clicks Print Report |
| **Pre-Conditions:** | 1. Admin is logged in 2. TAs data available |
| **Post-Conditions:** | 1. A printable report is displayed |
| **Description:** | This process generates a printable report for a given TA of their assignments and info |
| **Name:** | Print Report of TA Assignments |

|  |  |
| --- | --- |
| **ID:** | 15 |
| **Name:** | Email Offers |
| **Actor(s):** | Administrator |
| **Flow of Events:** | 1. Admin navigates to List TAs 2. Admin selects TAs to email offers via checkboxes 3. Admin hits Email Offers |
| **Pre-Conditions:** | 1. Admin is logged in 2. TAs data available |
| **Post-Conditions:** | 1. An email is sent to the TA with an offer of their assigned sections |
| **Description:** | This process allows admins to email TAs their assignment offers |

## 5.2 Use Case Diagram

This use case diagram shows the expected usage scenarios of the Students (TAs) and the employer.

|  |
| --- |
|  |

Figure 5.1

## 

## 5.3 Activity Diagram

This activity diagram shows the flow of actions for the different user groups: students and administrators. The center dotted line denotes where the actions crossover from students to administrators.

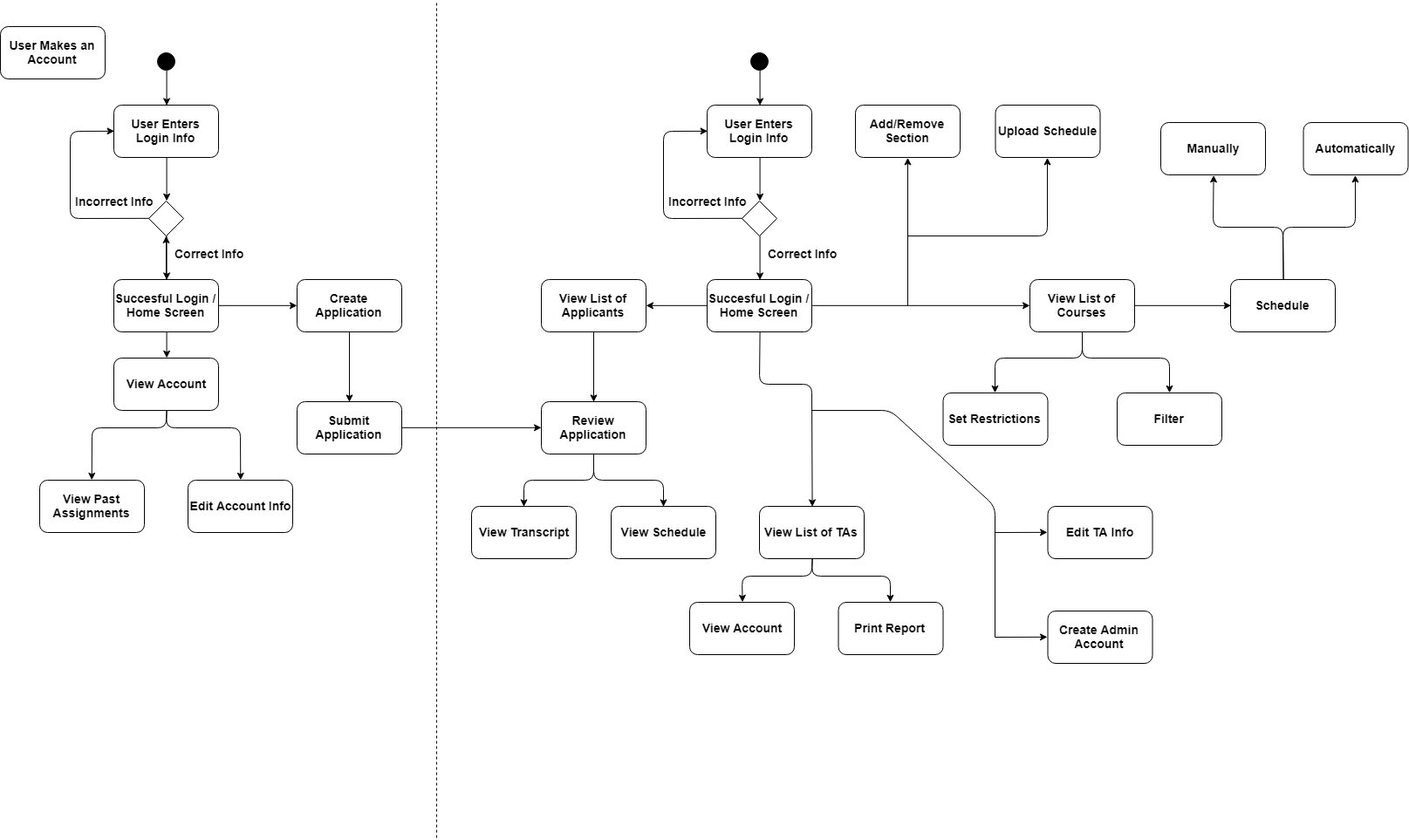


Figure 5.2

## 5.4 Component Diagram

This component diagram demonstrates how the system sends the formulated optimization problem to Google OR-Tools, which determines a feasible schedule.

|  |
| --- |
|  |

Figure 5.3

# 6.0 System Architecture

### 6.0.1 System Architecture Diagram

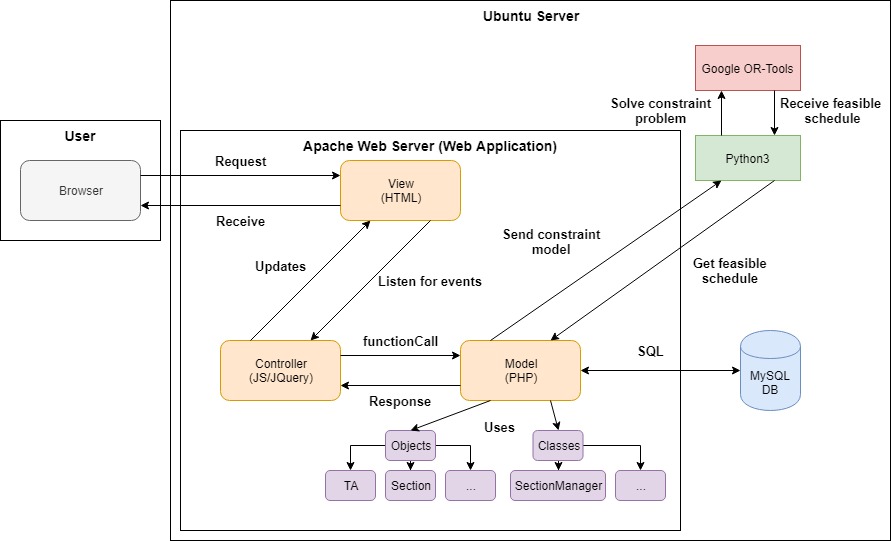


Figure 6.1

## 6.1 MVC Framework

### 6.1.1 Overview

The MVC framework is shown below, this framework will be utilized to structure the code of the website and software being built. Following an MVC pattern ensures a modular system that is updatable, expandable, and scalable.

### 6.1.2 View

The View will be HTML files that contain the DOM layout and CSS styling of the website pages. These views will be the webpages of the TA portal, allowing the user access to the website. The view does not communicate to the Model directly, the Controller will update the view from requests made in the view.

### 6.1.3 Controller

The controller will be Javascript and JQuery files written for the views that contain functions to listen to events on their respective views and to make AJAX calls to the Model functions, passing arguments if required and receiving formatted data. The respective view is then updated with the data received from the Model via the Controller.

### 6.1.4 Model

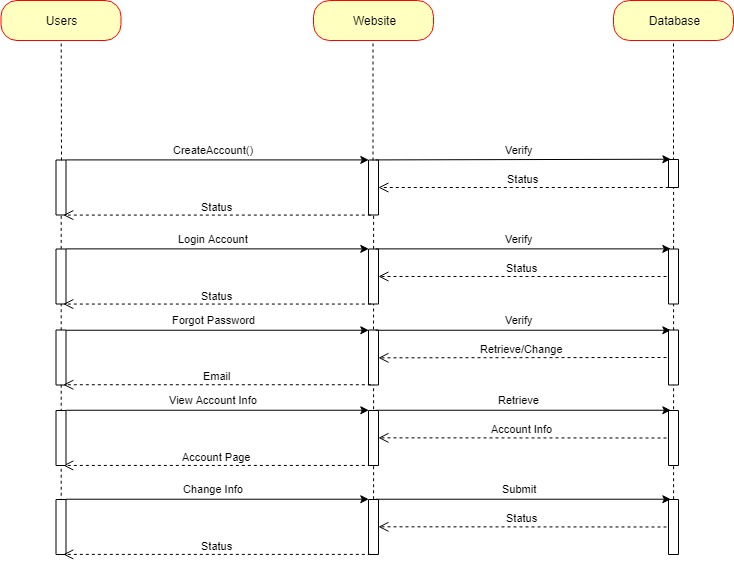
The Model is the PHP code that makes calls to the SQL database. The PHP functions are called via AJAX through the Controller and may pass arguments which are used in the SQL queries. The data is formatted into a JSON object or a as a String object to be interpreted by the Controller.6.2 Sequence Diagrams

### 6.1.5 PHP - Python Communication

When an administrator begins the automatic scheduling process PHP will write the scheduling data to a common .csv filetype and launch a Python script to read in the data. Python then formulates the data into a constraint problem and solves it with Google-OR Tools. The solved constraint problems contains a feasible schedule of TA assignments and that data is written to common .csv file types and read back in through PHP. A statusFile is used to track the status of the optimization as its running.

## 6.2 Sequence Diagrams

### 6.2.1 Users

The following sequence diagrams show the processes being carried out by different users and components of the software and how they interact.

### 6.2.2 Student

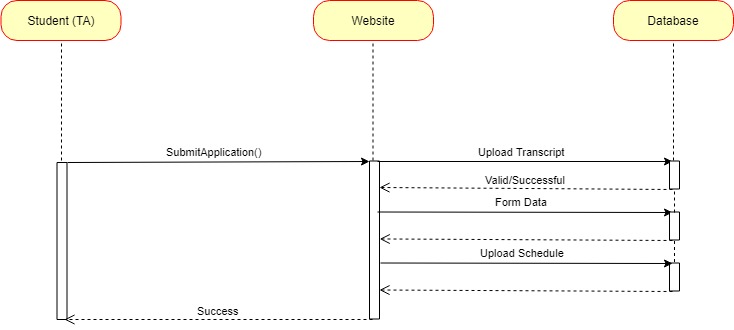
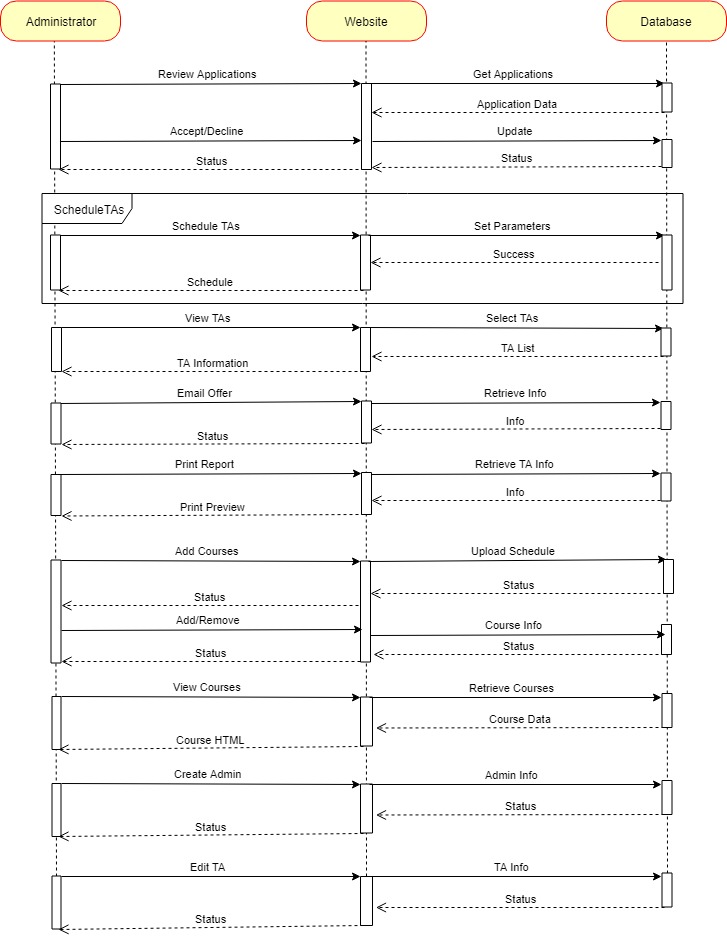
This sequence diagram depicts the communication between the Student (TAs) to the Website and Database

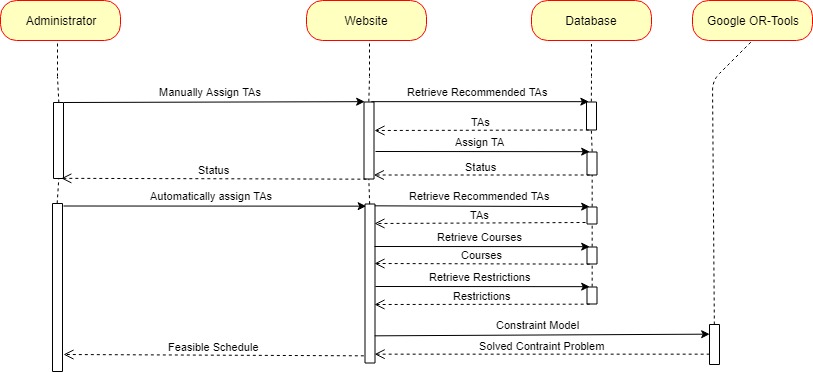
Figure 6.2

### 6.2.3 Administrator

The Employers communication path and processes are shown below here with a generalized subsystem “ScheduleTAs” described further in another sequence diagram.

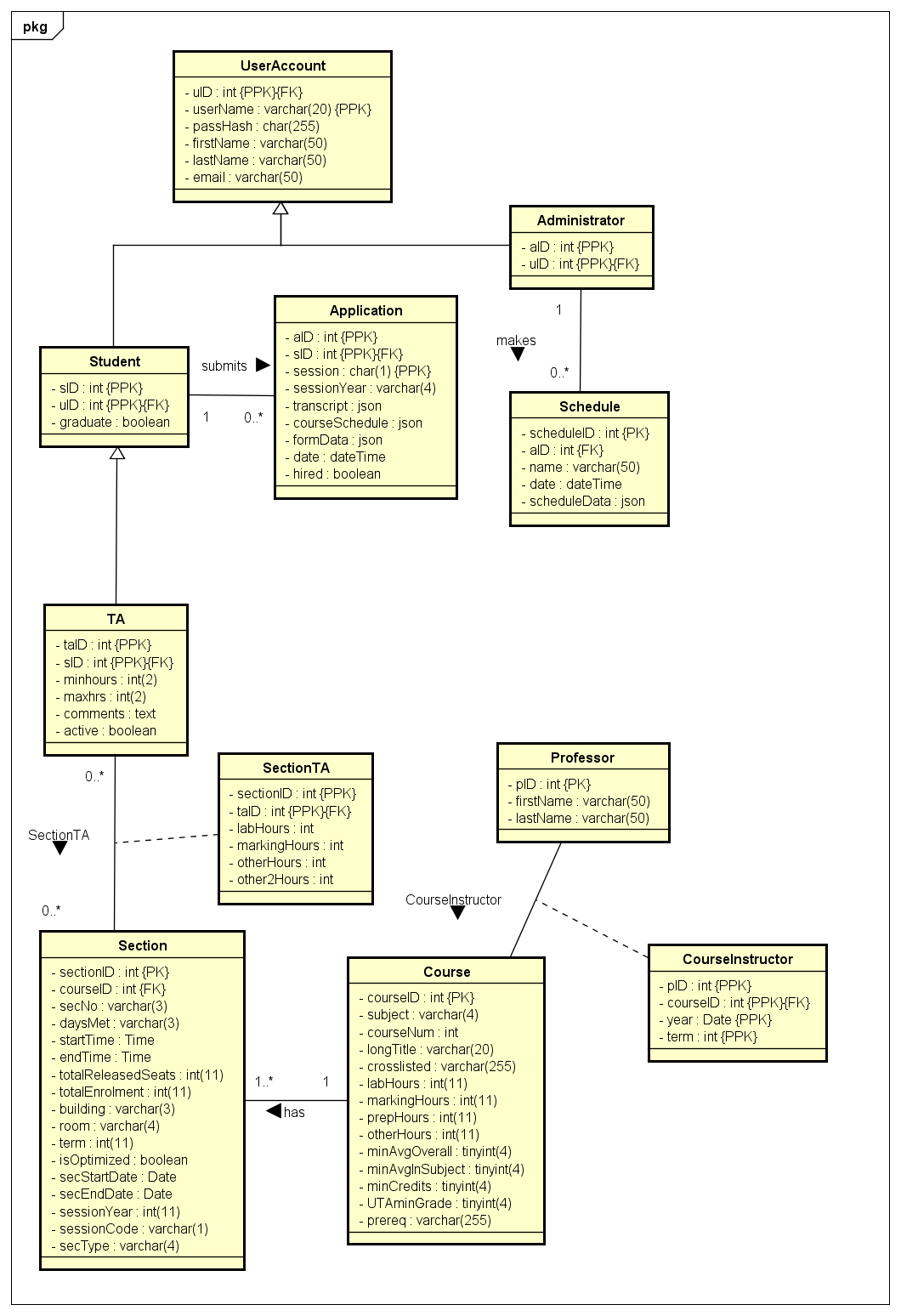


### 6.2.4 Administrator - ScheduleTAs

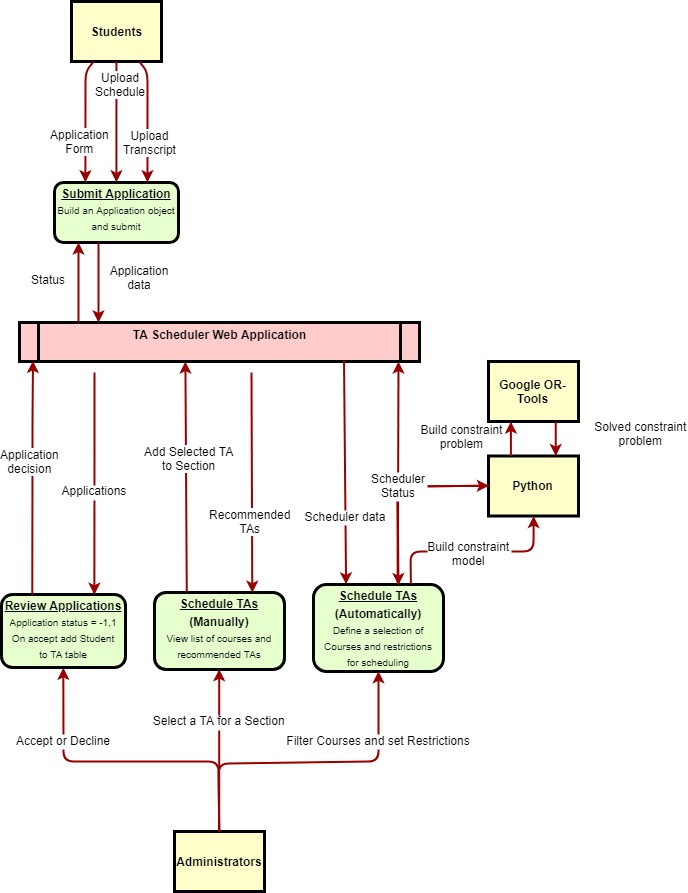
The “ScheduleTAs” communication and process path is shown here with more detail.

## 6.3 Database ER Diagram

This diagram shows how information will be represented as objects in a relational schema.



## 6.4 Data flow Diagram



# 7.0 Technical Specifications

## 7.1 Client Side

Our web application includes technologies such as HTML to show the raw data. CSS for the styling, Javascript and jQuery for our client side scripting. The output data will be displayed after optimization and will be displayed in a HTML table as a list.

## 7.2 Server Side

The main server side scripting language used is PHP. PHP makes up the Model in the MVC framework used, receiving AJAX requests from the Javascript Controller and querying the MySQL database, returning the formatted data back to the Controller. Python3 is used for the Google OR-Tools library to solve the constraint problem of scheduling TAs.

## 7.3 MySQL Database

A MySQL database is, the website utilizes a MySQL database to store all information that pertains to the function of the system. PHP is used to query and manipulate the database.

## 7.4 Feasibility of Schedule

A feasible schedule will be found through Google OR-Tools, which can can function in a Python environment, Java environment, and some other environments. We utilize the Python environment, but the system will be constructed in such a way that the option for integration Google OR-Tools in another language supported by Google OR-Tools is available. The system will write the optimization problem into a file readable by Google OR-Tools, then call on Google OR-Tools to perform optimization, which, once complete, is written back to a file. Files have unique names so any administrators running optimization at the same time do not override each other.

# 

# 8.0 Testing

## 8.1 Overview

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | Test | Status | Comments |
| **Administrator** | createAdministrator | **PASS** |  |
| loginAdministrator | **PASS** |  |
| getAccountType | **PASS** |  |
| loadAdministrator | **PASS** |  |
| isAdministrator | **PASS** |  |
| printInfo | **PASS** |  |
| deleteAdministrator | **PASS** |  |
| **Application** | testStudent | **PASS** | Travis CI file path bug when executing test files, not run on the travis build |
| formData | **PASS** |
| transcriptJson | **PASS** |
| scheduleJson | **PASS** |
| loadApplications | **PASS** |
| getApplication | **PASS** |
| cleanup | **PASS** |
| **Professor** | nameSplit | **PASS** |  |
| delete | **PASS** |  |
| **Schedule** | test1 | **PASS** | Used to check various conflicts in schedules |
| test2 | **PASS** |
| test3 | **PASS** |
| test4 | **PASS** |
| **SectionManager** | equality | **PASS** | Used to test if two sections are equal for merging labs in crosslisted sections. |
| sectTimes | **PASS** |  |
| termDiff | **PASS** |  |
| **Student** | createStudent | **PASS** |  |
| loginStudent | **PASS** |  |
| loadStudent | **PASS** |  |
| isStudent | **PASS** |  |
| printInfo | **PASS** |  |
| deleteStudent | **PASS** |  |
| **TA** | createTA | **PASS** |  |
| loadTA | **PASS** |  |
| isTA | **PASS** |  |
| printInfo | **PASS** |  |
| deleteTA | **PASS** |  |
| **Transcript** | test1 | **PASS** |  |
| test2 | **PASS** |  |
| **UserAccount** | createUserAccount | **PASS** |  |
| loadUserAccount | **PASS** |  |
| printInfo | **PASS** |  |
| changeFirstName | **PASS** |  |
| changeLastName | **PASS** |  |
| changeEmail | **PASS** |  |
| changePassword | **PASS** |  |
| deleteUser | **PASS** |  |
| getAccountTypeAdministrator | **PASS** |  |
| getAccountTypeStudent | **PASS** |  |
| **UserHandler** | createStudent | **PASS** | This test encompases testing the functionality of the model code. If this set passes it can be assumed any of the models are functioning as expected if their respective business logic code is passing. |
| loginAccountStudent | **PASS** |
| printInfo | **PASS** |
| deleteStudent | **PASS** |
| createAdministrator | **PASS** |
| loginAdministrator | **PASS** |
| deleteAdministrator | **PASS** |

### 8.1.2 Unit Testing

Where possible Objects are tested for their functionality in reflecting some business side logic within the web application usually to the database. Mostly business logic is tested within the objects. One model is tested for its ability to properly execute functionCalls to it’s residing functions and pass the output appropriately, with this test the other models follow the same framework and can be assumed working. This way only the business side logic needs to be tested fully and not the individual models.

### 8.1.2 Unimplemented Tests

The following areas have not had tests built:

* Add/Remove Course **Sections**
* Automatic assignment SQL
* Manual assignment SQL
* StatusFile R/W - used to track status code of automatic scheduler
* Server permissions
* Recommendation Matrix
* Course Restrictions
* Hour Assignment for TAs
* Application status on accept/decline (TA tested)

## 8.2 Software/Technologies

### 8.2.1 PHPUnit

PHPUnit is used to test the server side code that acts as the Model in the MVC framework. The main functions being tested will be the database queries to the MySQL DB, ensuring the output matches the expected output of the tests.

## 8.3 Continuous Integration/Deployment

Travis CI is used to continuously integrate code changes made during the development process. Travis CI is configured to run the different tests constructed through PHPUnit on every commit to the github repo, ensuring that every change made does not break anything in the software. On a successful build travis will launch a script that pushes the changes to the live server.

# 9 Licenses

1. MySQL: GPL v.2.0. May not change/extend MySQL or sublicense. Must disclose source, include copyright, intact license and state changes.
2. Google OR-Tools: Apache License v.2.0. Permissive license.
3. Selenium: Apache License v.2.0. Permissive license.
4. PHP: PHP License v3.01. Redistributions of source code must keep license intact. Restrictions on using ‘PHP’ in name of software not applicable.
5. PHPUnit: Copyright Sebastian Bergmann. Redistributions of source code must retain intact license.
6. Python: Python Software Foundation License. Permissive license.
7. Travis CI: MIT license. Permissive license.
8. Fpdf: Permissive license.
9. PHPMailer: GNU LGPL license.