ECSE 321 - Intro to Software Engineering
Deliverable 1

Harley Wiltzer
Camilo Garcia La Rotta
Jake Shnaidman
Robert Attard
Matthew Lesko

February 12, 2017

Contents

Chapter 1

Requirements Document

1.1 Functional Requirements

Please Note:

- The requirement's ID is its list number, i.e. 1.1.1.
- The priority of requirements starts with the General Requirements from 1.1.1 to 1.1.6, followed by the Non-functional Requirements. More specifically:
    1. Total completion of Desktop app requirements in numerical order.
    2. Total completion of Web app in requirements numerical order.
    3. Total completion of Mobile app in requirements numerical order.
    4. Implementation of XML persistence across all platforms.
    5. Implementation of database persistence across all platforms.
    6. Intercommunication between all platforms under a centralized persistence system.
    7. Total completion of Non-functional Requirements in requirements numerical order.

1.1.1 General Requirements

1.1.1.1 The Teaching Assistant Management System shall include a desktop application.

1.1.1.2 The Teaching Assistant Management System shall include a web application.

1.1.1.3 The Teaching Assistant Management System include a mobile application.

1.1.1.4 All applications (desktop, web, and mobile) shall have an XML persistence layer.

1.1.1.5 All applications (desktop, web, and mobile) shall have database persistence.

1.1.1.6 The three applications (desktop, web, and mobile) may be capable of communicating with one another.

1.1.2 Desktop Application Requirements

1.1.2.1 The desktop application shall only allow be accessible to users with administrator credentials.

1.1.2.2 The desktop application shall be written in Java with the Java Swing library.

1.1.2.3 The desktop application shall be capable of storing in its persistence layer a list of courses with their hours and credits.

1.1.2.4 The desktop application shall be capable of manipulating a list of courses with their hours and credits from its persistence layer.

1.1.2.5 The desktop application shall provide the ability to store a list of courses and their attributes listed in (1.2.3).

1.1.2.6 The desktop application shall be capable of storing the student enrollment data in its persistence layer.

1.1.2.7 The desktop application shall be capable of storing the TA/Grader salaries of all McGill Departments from a CSV file onto its persistence layer.

1.1.2.8 The application shall be capable of accessing TA/Grader schedules from its persistence layer.

1.1.2.9 The scheduling algorithm shall never appoint work hours for prospective students that are beyond the students' available hours.

1.1.2.10 The scheduling algorithm shall hire a certain TA for as many time-slots as possible for a given class with multiple lab or tutorial sessions.

1.1.2.11 The scheduling algorithm shall limit individual TA/Grader hours to within the range of a minimum of 45 for each course to 180 hours total per semester for all courses.

1.1.2.12 The scheduling algorithm shall prefer graduate students to undergraduate students.

1.1.2.13 The scheduling algorithm shall account for students' indicated priorities when assigning job placements.

1.1.2.14 The desktop application shall provide administrators with the opportunity to review instructor modifications to the TA/Grader hours.

1.1.2.15 The desktop application shall provide administrators with the opportunity to accept or reject the instructors' modifications.

1.1.2.16 The desktop application shall be capable of sending job offers to the selected TA's and Graders *once the administrator has explicitly accepted the placements.*

1.1.2.17 The desktop application shall allow the user to perform the instructor actions described in section 1.2.

1.1.2.18 The desktop application shall allow the user to perform the TA/Grader actions described in section 1.3.

1.1.3 Web Application Requirements

1.1.3.1 The web application shall only allow be accessible to users with instructor credentials.

1.1.3.2 The web application shall be programmed in PHP with the use of HTML and CSS.

1.1.3.3 The web application shall be capable of retrieving the course data from its persistence layer.

1.1.3.4 The web application shall be capable of displaying the course data from its persistence layer.

1.1.3.5 The web application shall be capable of retrieving the student enrollment data from its persistence layer.

1.1.3.6 The web application shall be capable of displaying the student enrollment data from its persistence layer.

1.1.3.7 The web application shall be capable of creating job postings with the attributes requisite skills and previous experience.

1.1.3.8 The web application shall be able to save job postings in the persistence layer.

1.1.3.9 The web application shall be capable of retrieving the course data from its persistence layer.

1.1.3.10 The application shall be capable of displaying the list of TA/Grader placements from its persistence layer.

1.1.3.11 The application shall allow the modification of TA/Grader placements without allowing the modifications to cause budget issues.

1.1.4 Mobile Application Requirements

1.1.4.1 The mobile application shall only allow be accessible to users with student credentials.

1.1.4.2 The mobile application shall be programmed in Java for the Android operating system.

1.1.4.3 The mobile application shall be created using only the tools provided in the Android UI library in Android Studio.

1.1.4.4 The mobile application shall be able to create a profile that contains the users' student ID.

1.1.4.5 The mobile application shall be capable of retrieving a list of job postings from its persistence layer.

1.1.4.6 The mobile application shall be capable of displaying a list of job postings from its persistence layer.

1.1.4.7 The mobile application shall limit the amount of applications of the user to a maximum of three.

1.1.4.8 The mobile application shall allow the arbitrary ranking of job applications by the user.

1.1.4.9 The mobile application shall be capable of retrieving a list of job offers from its persistence layer.

1.1.4.10 The mobile application shall be capable of displaying a list of job offers from its persistence layer.

1.1.4.11 The mobile application shall be capable of submitting acceptance or denial of the aforementioned job offers.

1.2 Non-functional Requirements

1.2.1 Performance Requirements

1.2.1.1 The three applications provided with the product (desktop, web, mobile) shall limit RAM usage to within 750MB.

1.2.1.2 The scheduling algorithm of the desktop application shall not take longer than one minute to complete.

1.2.1.3 The three applications shall provide error messages to handle unexpected behavior.

1.2.1.4 The three applications shall, in case of a system crash, restart the application with the last saved persistence file.

1.2.2 Security Requirements

1.2.2.1 The desktop application shall include a secure authentication procedure to ensure that only administrators gain access.

1.2.2.2 The web application shall include a secure authentication procedure to ensure that only instructors gain access.

1.2.2.3 The web application shall be able to identify the current user and associate users with their modification histories.

1.2.2.4 The mobile application shall include a secure authentication procedure to identify which student is currently using the software.

1.2.2.5 The mobile application shall be capable of associating student security information with the respective student account information.

1.2.2.6 The persistence of administrator, instructor, and student passwords shall be achieved cryptographically, using RSA/NTRU cryptosystems.

1.2.3 Compatibility Requirements

1.2.3.1 The desktop application shall work on Windows 10, Macintosh OS X 10.5 Leopard., GNU/Linux 4.9.8, and BSD 10.3 systems.

1.2.3.2 The web application shall be compatible with Google Chrome version 25, Mozilla Firefox version 50.0.0, Safari 10, Microsoft Edge, and Internet Explorer 9.

1.2.3.3 The mobile application shall be compatible with Android phones running Android 4.0 (Ice Cream Sandwich) or a more recent version.

1.2.4 Graphical Requirements

1.2.4.1 The three applications provided with the product (desktop, web, mobile) shall share a common logo (app icon, desktop shortcut, web logo).

1.2.4.2 The three applications shall have consistent (i.e. the same) color palettes.

1.2.4.3 The three applications shall have the ability to select between Light and Dark themes for optimal comfort in a variety of lighting conditions.

1.2.4.4 The color palettes (light theme and dark theme) shall be designed in a color-blind-friendly manner, to promote a common experience to all users.

Chapter 2

TAMAS Domain Model

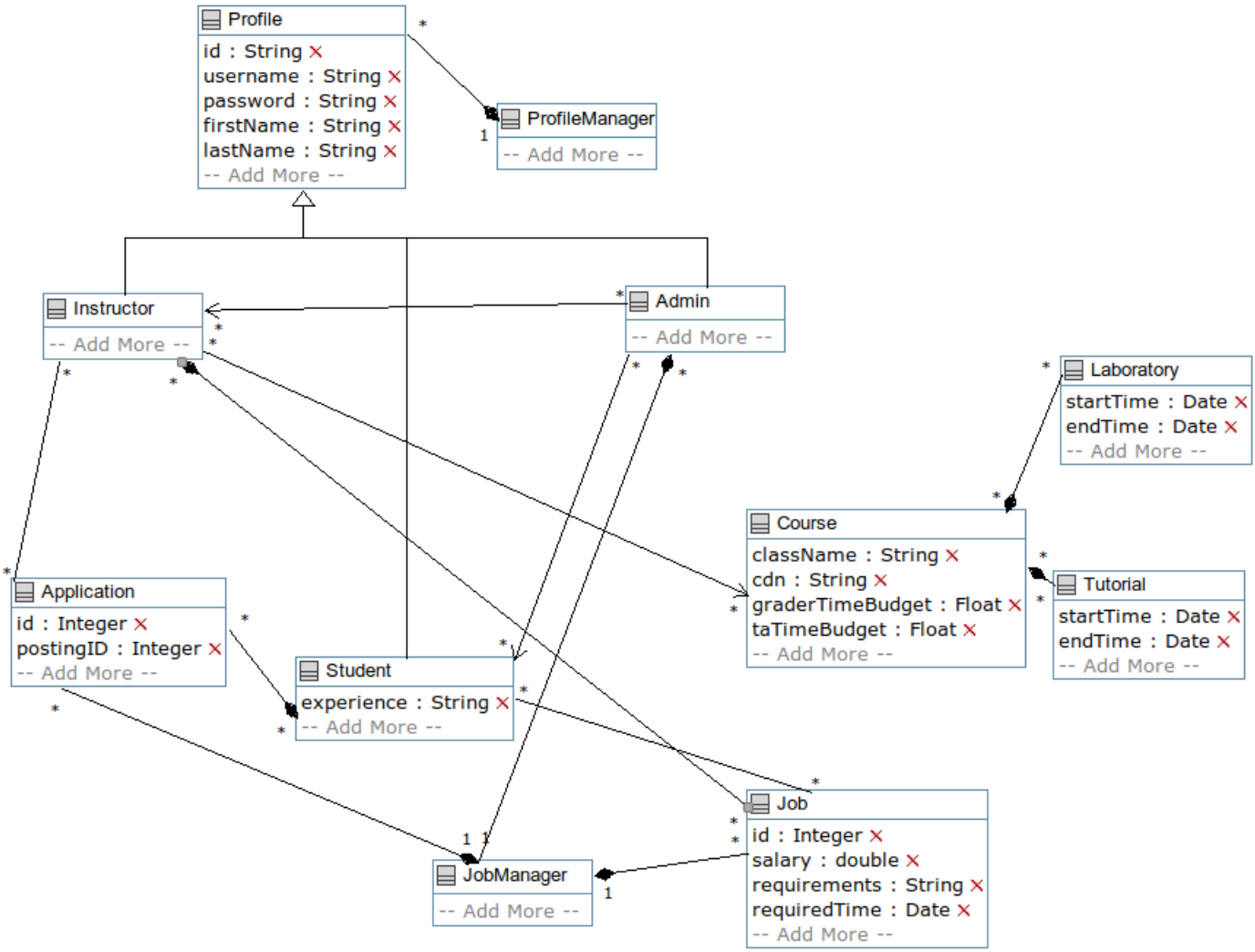
The following UMPLE code was written for the generation of the TAMAS domain model:

```
namespace ca.mcgill.ecse321.group10.TAMAS;

class Course {
  className;
  immutable cdn;
  Float graderTimeBudget; // time budget
  Float taTimeBudget;      // time budget
  * <@>- 0..* Tutorial;
  * <@>- 0..* Laboratory;
}

class Tutorial {
  Date startTime;
  Date endTime;
}

class Laboratory {
  Date startTime;
  Date endTime;
}

class Instructor {
  isA Profile;
  * -> * Course;
  * <@>- * Job;
  * -- * Application;
}

class Student {
  isA Profile;
  experience;
  degree { // enum, wont show in diagram, shows in code
    UNDERGRAD{}
```

```
    GRADUATE{}
  }
  * -- * Job;
  * <@>- * Application;
}

class Profile {
  immutable id; // either student or employee ID, for the securities yo
  username;
  password;
  firstName;
  lastName;
}

class Job {
  autounique id;
  position {  // enum, wont show in diagram, shows in code
    TA {}
    GRADER {}
  }
  immutable double salary; // cuz the pay is so big we need a double
  immutable requirements;
  immutable Date requiredTime;
}

class Application {
  autounique id;
  Integer postingID;
}

// unsure if class should create instances of Intrsuctor/Student
// or rather have all the methods of Instructor/Student
class Admin{
  isA Profile;
  * -> * Instructor;
  * -> * Student;
  * <@>- 1 JobManager;
}

// mimic the registrationManager,
// oversee all job posting/application transactions
class JobManager {
  singleton;
  1 <@>- * Job;
  1 <@>- * Application;
}
```

```
class ProfileManager {
  singleton;
  1 <@>- * Profile;
}
```

The corresponding class diagram generated by UMPLE may be found below:



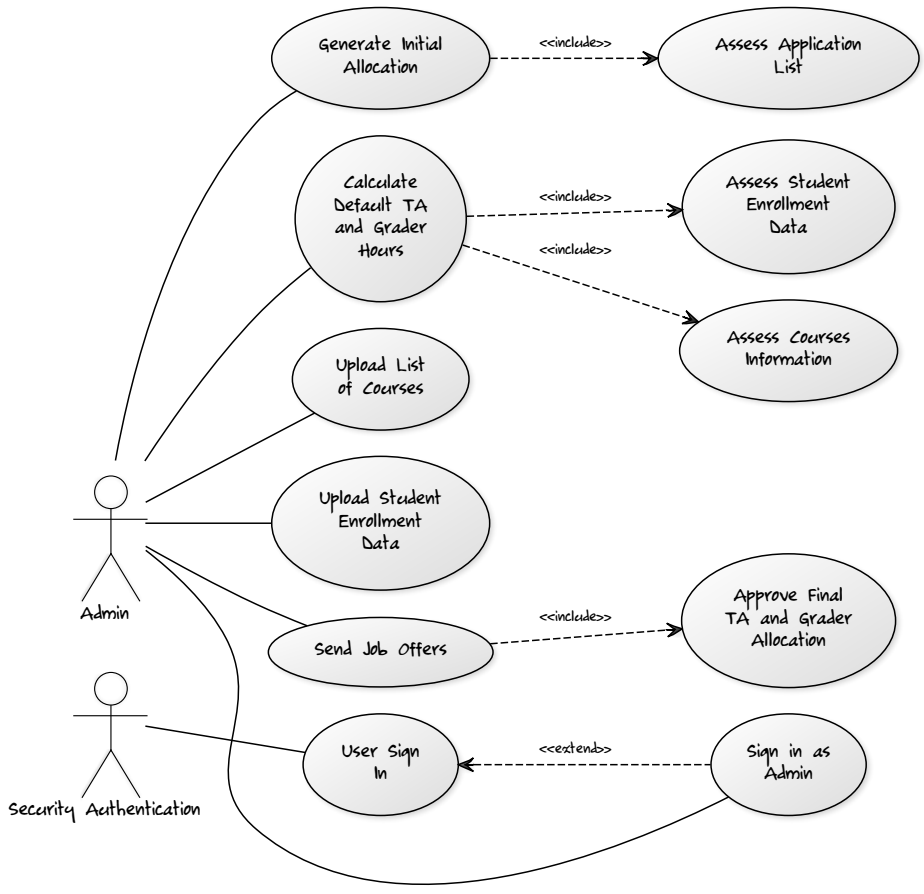
Chapter 3

Use Cases

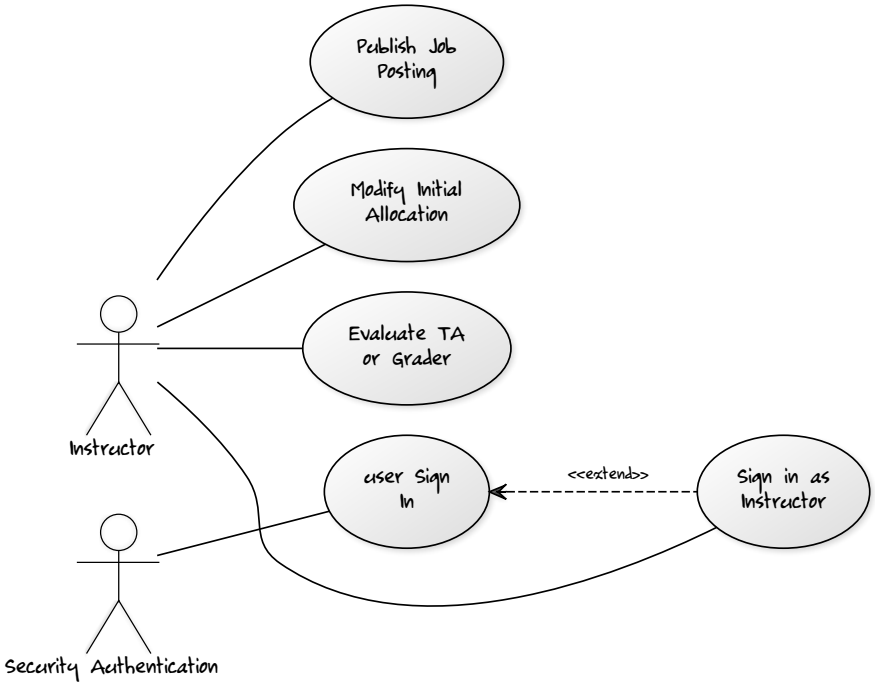
After designing an original use case diagram consisting of all actors and use cases, it was decided that the diagram was too crowded and thus hard to read. Therefore, three separate use case diagrams were created in order to alleviate eye strain and clarify the actions. The UML code that was used to generate the use case diagrams may be found in Appendix A.

3.1 Use Case Diagrams

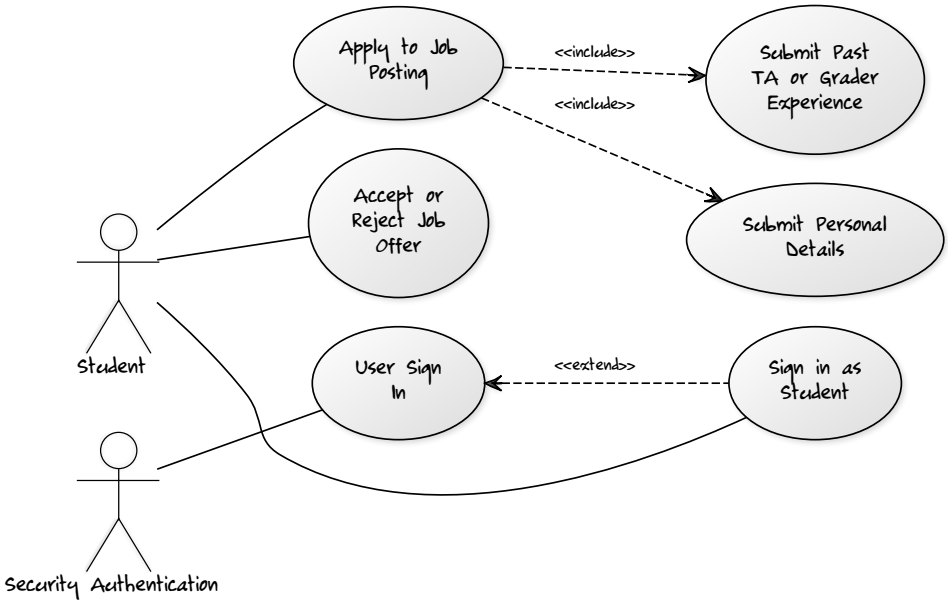
3.1.1 Administrator Use Case Diagram



3.1.2 Instructor Use Case Diagram



3.1.3 Student Use Case Diagram



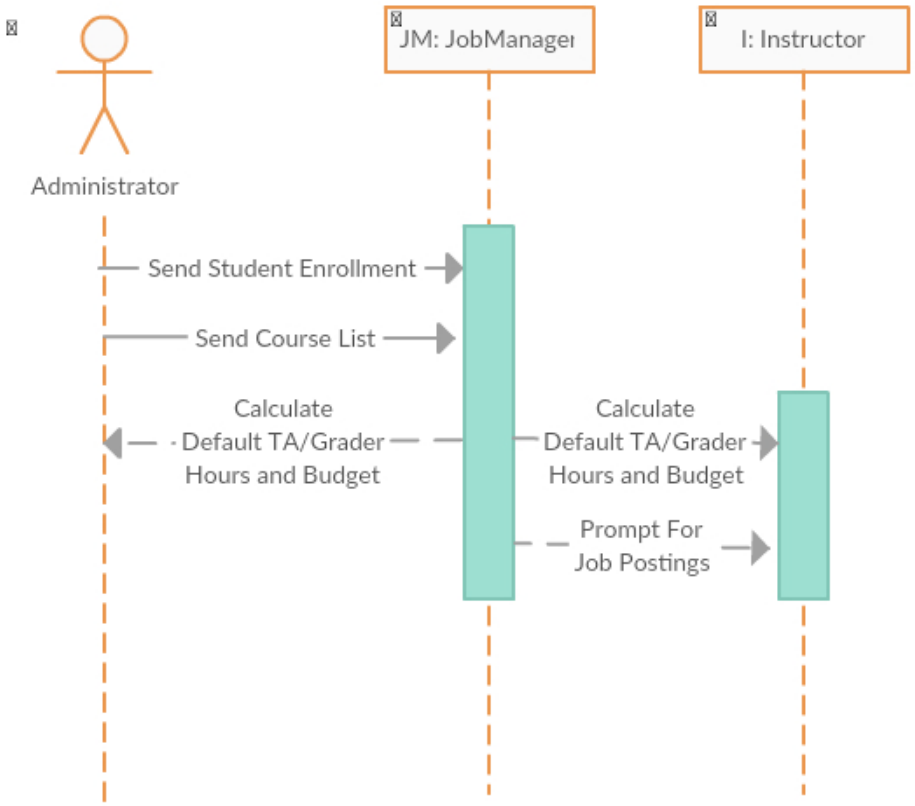
3.2 Descriptions of Use Case Diagrams

Chapter 4

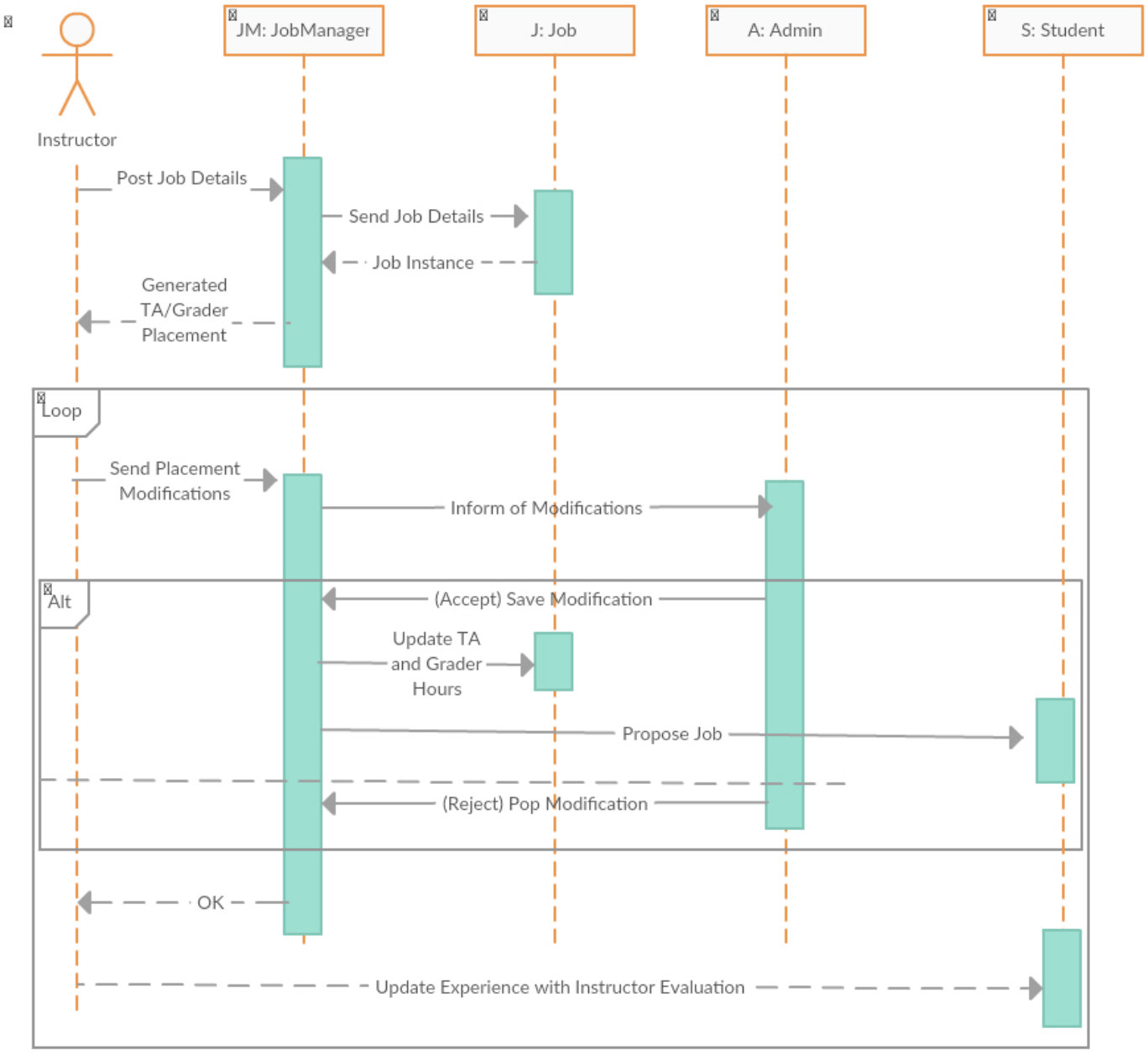
Sequence Diagrams

The following sequence diagrams were designed to display the sequence of actions that will occur in the use of the applications. One sequence diagram was designed per actor, and another was designed to demonstrate the authentication process. It is important to note that before authentication, the user has no identity, and is thus neither a student, an instructor, or an administrator, but just a default Profile.

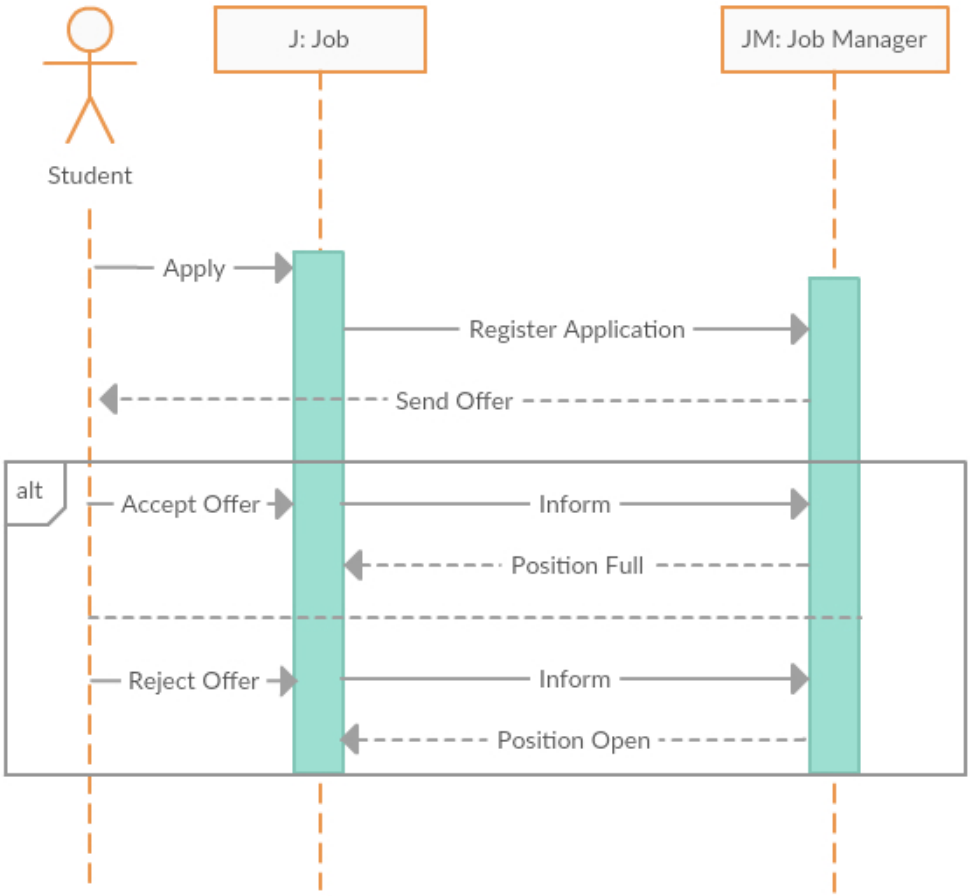
4.1 Administrator Sequence



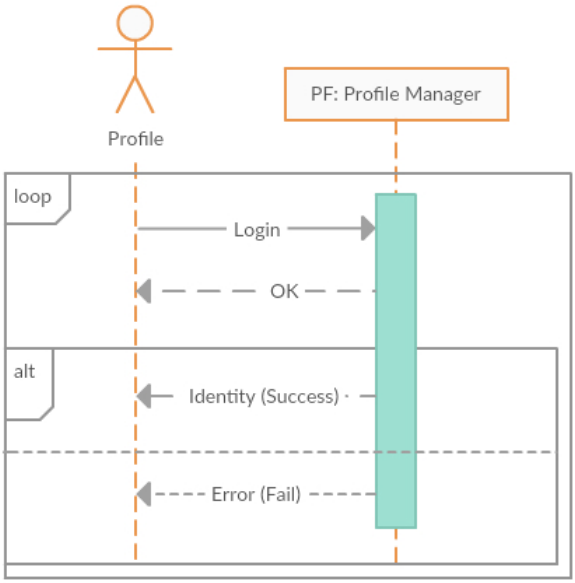
4.2 Instructor Sequence



4.3 Student Sequence



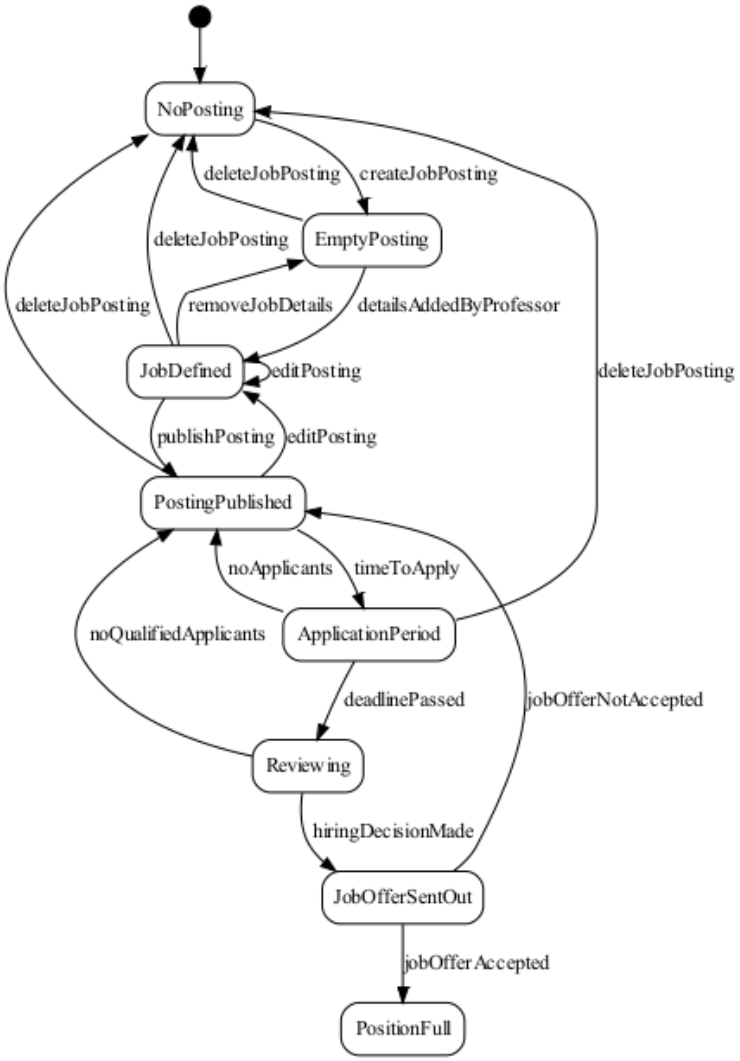
4.4 Authentication Sequence



Chapter 5

Statechart for Job Class

The following statechart was conceived to emphasize the roles and navigations of different states of the applications. The UML code used to generate the statechart diagram may be seen in Appendix B.



Chapter 6

Work Plan for Remaining Iterations

Remaining iterations of the requirements should only modify the nonfunctional requirements. All functional requirements should have been accounted for. Modifications to the nonfunctional requirements should not require any changes to the software architecture after it has been designed. The use cases and actors change due to the addition of new features; new use cases can be generated. The current functionality has been modeled in our use cases. The Domain model is planned to change. We have a profile manager which will authenticate username and passwords; however, the architecture which has yet to be designed may require changes to the domain model in the future. There are currently no plans to change the state chart for the job class. The sequence diagrams currently account for how the different actors interact with other actors and objects. The architectural details of the software should not change this. In the next week, we plan to develop the software architecture and details of the class diagrams which will allow us to move on to the development of source code. In the following week, we will create more detailed sequence diagrams going into the details of the implementations on individual platforms. We will then develop prototype source code for the *Publish Job Posting* and *Apply for Job* use cases for the desktop and mobile applications respectively.

Appendix A - Use Case Diagram UML Code

Administrator Use Case

```
[Admin] - (Generate Initial Allocation)
[Admin] - (Calculate Default TA and Grader Hours)
[Admin] - (Upload List of Courses)
[Admin] - (Upload Student Enrollment Data)
[Admin] - (Send Job Offers)
[Admin] - (Sign in as Admin)
[Security Authentication] - (User Sign In)
(User Sign In) < (Sign in as Admin)
(Calculate Default TA and Grader Hours) > (Assess Courses Information)
(Calculate Default TA and Grader Hours) > (Assess Student Enrollment
   Data)
(Send Job Offers) > (Approve Final TA and Grader Allocation)
(Generate Initial Allocation) > (Assess Application List)
```

Instructor Use Case

```
[Instructor] - (Publish Job Posting)
[Instructor] - (Modify Initial Allocation)
[Instructor] - (Evaluate TA or Grader)
[Instructor] - (Sign in as Instructor)
[Security Authentication] - (User Sign In)
(user Sign In) < (Sign in as Instructor)
```

Student Use Case

```
[Student] - (Apply to Job Posting)
[Student] - (Accept or Reject Job Offer)
[Student] - (Sign in as Student)
[Security Authentication] - (User Sign In)
```

```
(User Sign In) < (Sign in as Student)
(Apply to Job Posting) > (Submit Personal Details)
(Apply to Job Posting) > (Submit Past TA or Grader Experience)
```

Appendix B - Statechart UML Code

```
class JobPosting
{
  status {
    NoPosting{ createJobPosting -> EmptyPosting;  }
    EmptyPosting{
      detailsAddedByProfessor -> JobDefined;
      deleteJobPosting-> NoPosting;
    }
    JobDefined{
      editPosting -> JobDefined;
      publishPosting -> PostingPublished;
      deleteJobPosting -> NoPosting;
      removeJobDetails -> EmptyPosting;

    }

    PostingPublished{
      timeToApply -> ApplicationPeriod;
      deleteJobPosting -> NoPosting;
      editPosting -> JobDefined;
    }

    ApplicationPeriod {
      deadlinePassed -> Reviewing;
      deleteJobPosting -> NoPosting;
      noApplicants -> PostingPublished;
    }

    Reviewing{
      hiringDecisionMade -> JobOfferSentOut;
      noQualifiedApplicants -> PostingPublished;
    }

    JobOfferSentOut{
      jobOfferAccepted -> PositionFull;
      jobOfferNotAccepted -> PostingPublished;
    }
```

```
    PositionFull{ }
  }
}
```