ECSE 321 - Intro to Software Engineering
Detailed Design

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

February 24, 2017

Contents

1 Description **1**
1.1 Detailed Domain Model 1

2 Rationale **1**
2.1 Detailed Domain Model 1

3 Class Diagrams **2**
3.1 Detailed Domain Model Diagram 2
3.2 Java and Android Controller Class Diagram 3
3.3 PHP Controller Class Diagram 3
3.4 Java View Class Diagram 4
3.5 PHP View Class Diagram 5
3.6 Android View Class Diagram 5

1 Description

1.1 Detailed Domain Model

The Detail Design Diagram consists of the following entities: ApplicationManager, ProfileManager, CourseManager, Application, Profile, Course, Job, Instructor, Admin, Student, Laboratory, and Tutorial. It consists of a Controller, called Controller, a Boundary, called View, and a Persistence, called Persistence XStream. The Controller uses the entities ApplicationManager, ProfileManager, and CourseManager to save, edit, and modify data within the model, which are then saved within a persistence layer. The functionalities of the three "Manager" classes are listed below.

- ApplicationManager is in charge of Application, the job application created and submitted by the student for a job. It is associated with Application, Job, and ProfileManager.
- ProfileManager creates Admin, Instructor, and Student entities, all of which inherit from the Profile class.
- CourseManager creates Course entities.

In total there will be three controller classes in the Controller Packages with an additional class for input exceptions or input validation. The three controller classes will each use at least one of the manager classes.

2 Rationale

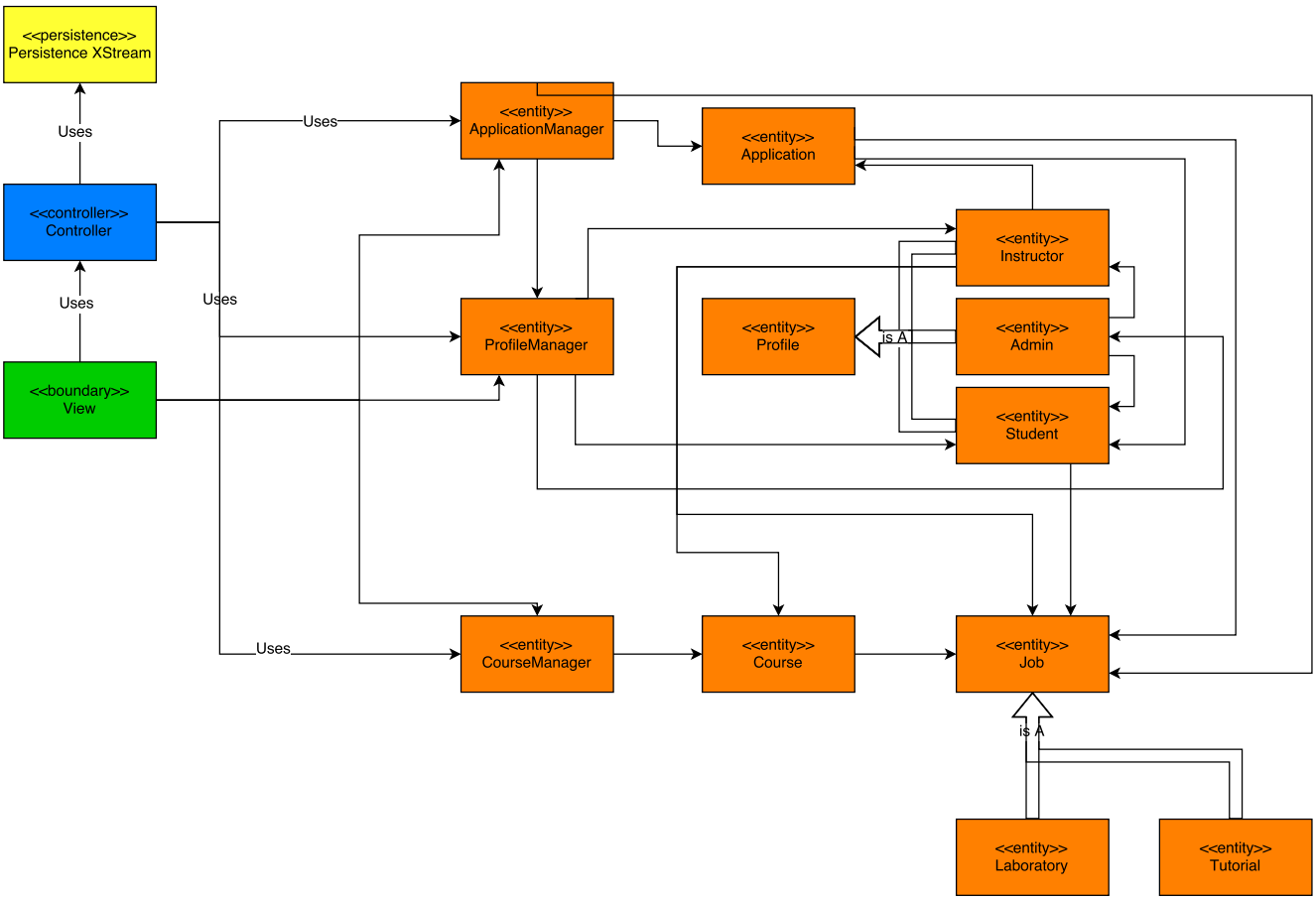
2.1 Detailed Domain Model

The three Manager classes were needed in order to give functionality to the user to create the entities associated with the manager classes, except for ApplicationManager creating a ProfileManager. The reason ApplicationManager is associated to ProfileManager is because otherwise the only way for ApplicationManager to have access to Student is by a direct association to it; this would cause a redundancy as now two manager classes are able to create Student entities, which

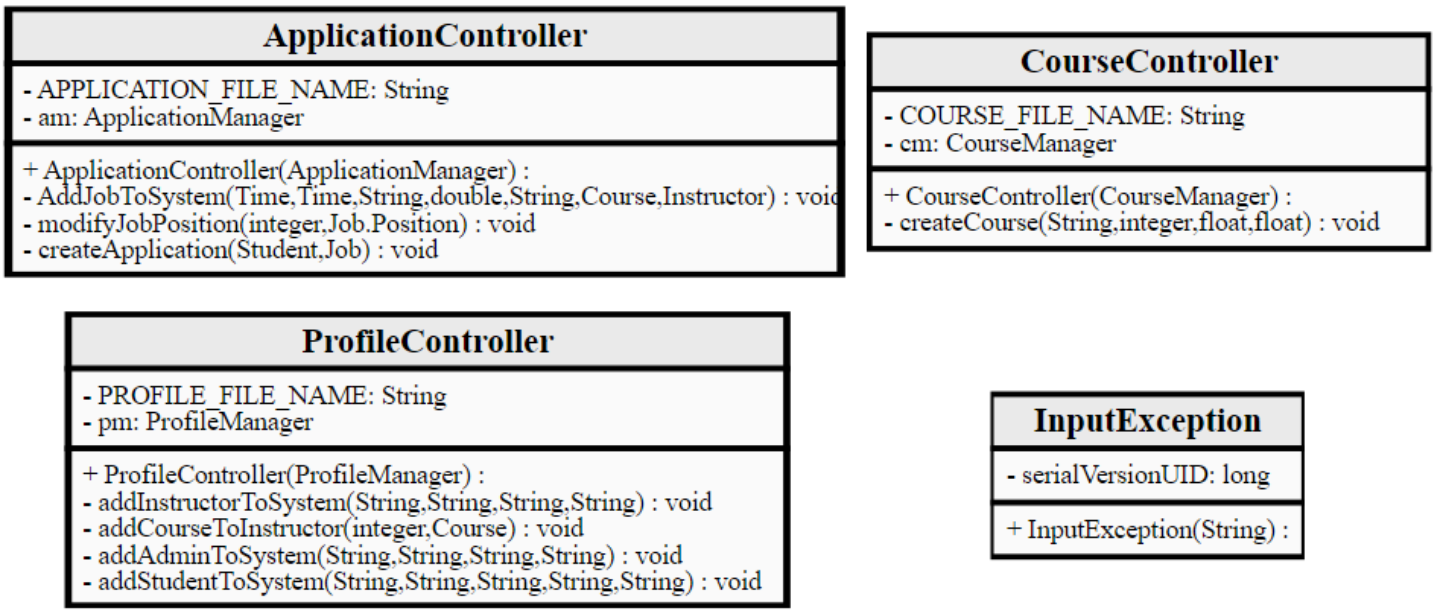
is something we want to avoid. Having a separate controller for each manager class allows one to modify the functionality of one controller class with its respective manager class without it having to affect the other controller classes.

3 Class Diagrams

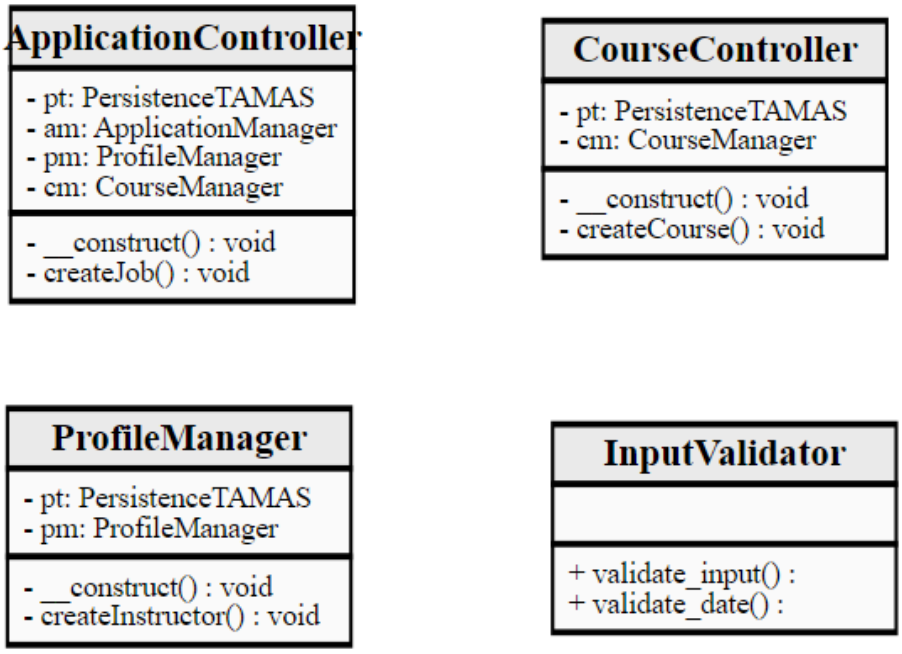
3.1 Detailed Domain Model Diagram



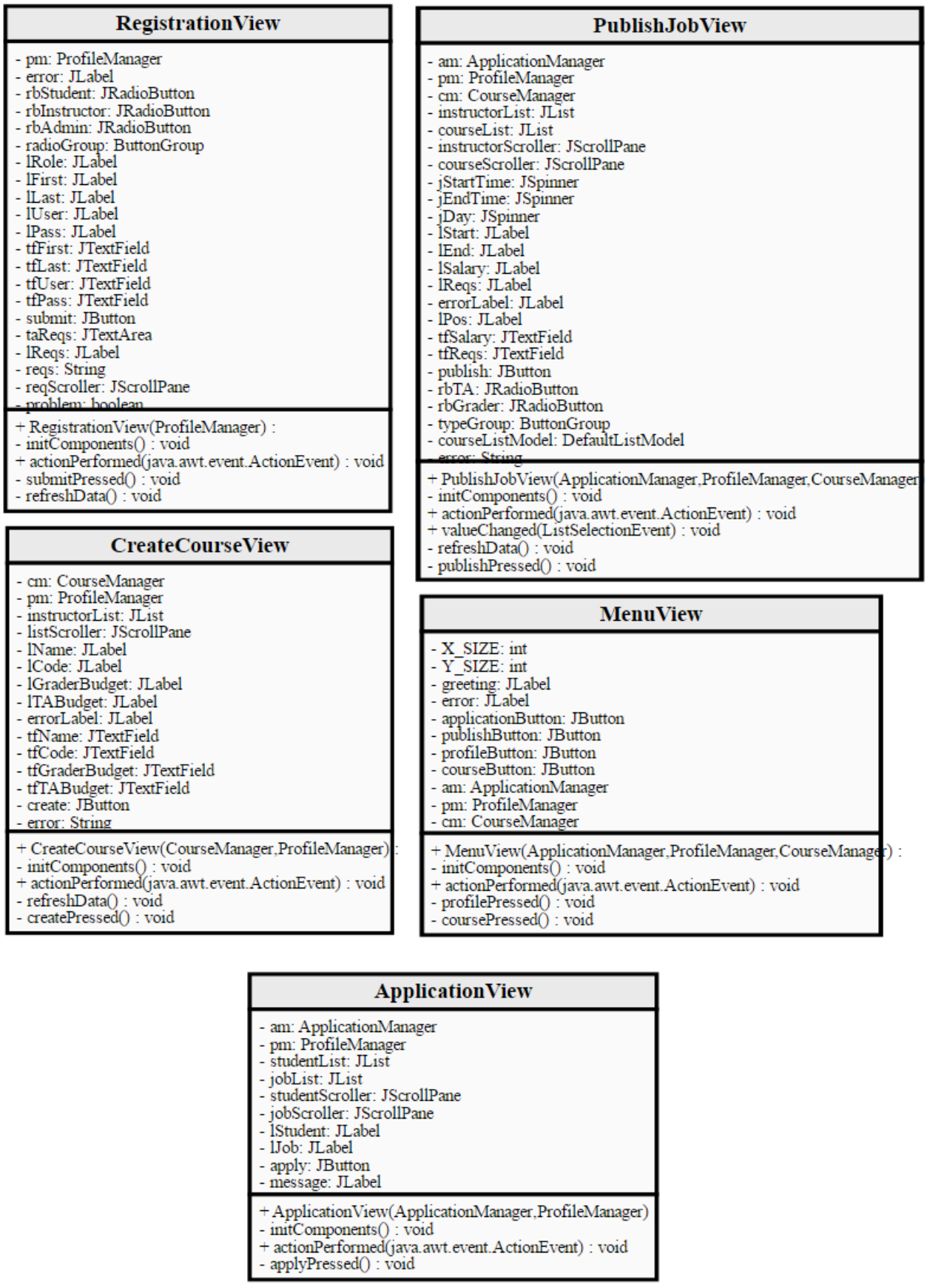
3.2 Java and Android Controller Class Diagram (Desktop and Mobile)



3.3 PHP Controller Class Diagram (Web)



3.4 Java View Class Diagram (Desktop)



3.5 PHP View Class Diagram (Web)

3.6 Android View Class Diagram (Android)