ECSE 321 - Intro to Software Engineering
Backlog Week 1

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

February 17, 2017

1 Deliverable 1

1.1 Main Tasks

- Harley Wiltzer
  - Co-formulated the prototype Requirements Document.
  - Drafted the first Requirements Document.
  - Designed Sequence Diagrams for Instructor Actions, Student Actions, Admin Actions.
  - Made small changes to Class Diagram.
- Camilo Garcia La Rotta
  - Co-formulated the prototype Requirements Document
  - Designed the prototype Class Diagram
  - Structured the GitHub file system
- Jacob Shnaidman
  - Designed Use Case Diagrams
  - Wrote Use Case Descriptions
  - Helped with the design of Sequence Diagrams
  - Wrote the Work Plan
- Robert Attard
  - Assisted in preliminary design of the use case diagrams
  - Designed and improved on the state diagram
  - Made wording and grammatical changes to Requirements Document
- Matthew Lesko
  - Designed the prototype Use Case Diagram
  - Co-designed Sequence Diagrams

1.2 Key Decisions

1. **Requirement Document**
2. **Use Cases**
   - Included four actors in the diagram, three of which are users and one of which is a service. The three users are: Student, Instructor and Department (has administrator privileges). The service is a Security Authentication.
   - Each user in the diagram acts upon multiple use cases respective to their functionalities mentioned in the Requirements and Specifications documents. Each use case describes functionality that its respective actor has.

3. **Class Diagram**

    - Trying to mimic the architecture of the event registration app, an early choice was to have a JobManager oversee the job transactions between Instructors and Students
    - Each user, whether it has administrator, instructor or student access rights, is obliged to have profile instance to facilitate identity access management
    - Later, it was also decided to have a ProfileManager to handle the actors' profiles and their authentication.
    - To enable an administrator to recreate actions available to instructors or student, we give the class administrator the power to create instances of the aforementioned users

4. **Sequence Diagram**

    - For convenience and concision, it was decided to design sequence diagrams for each main user action.
    - The sequence diagrams may contain actions from multiple user classes to emphasize the sequence. For example, in the Instructor sequence diagram, the action of an administrator verifying Instructor modifications to the TA/Grader hours is seen.
    - The current sequence diagrams include Student actions, Instructor actions, Administrator actions, and the authentication process.

5. **State Chart**

    - The state chart tracks the possible states of the job posting between being created and the position being filled.
    - The chart also indicates the triggers required to transition between job posting states.

1.3 Work Plan

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.

1.4 Work Hours

- Harley Wiltzer
    - Requirements document: 1 hours
    - Sequence diagrams: 3 hours
    - Class diagram: 30 minutes
    - Compilation of report: 4 hours
- Camilo Garcia La Rotta
    - Requirements document: 3 hours

  - Class diagram: 2 hours
  - State diagram: 2 hour

- Jacob Shnaidman
  - Use Case Diagrams: 4 hours
  - Use Case Description 1.5 hours
  - Helped with Sequence Diagrams 0.5 hours
  - Work Plan 0.5 hours

- Robert Attard
  - Requirements document: 0.5 hours
  - Use Case diagram: 3.5 hours
  - State diagram: 3 hours

- Matthew Lesko
  - Use case diagram: 4 hours
  - Co-design sequence diagrams: 4 hours

2 Deliverable 2

2.1 Main Tasks

- Harley Wiltzer
  - Aided in the design of the revamped domain model
  - Designed and developed desktop application
  - Aided in the design of the mobile application
  - Created the Sequence diagram for the Desktop - Apply to Job Use Case
  - Helped create the Sequence Diagram for the Desktop - Publish Job Posting Use Case
  - Compiled the final report
- Camilo Garcia La Rotta
  - Designed and documented Sequence Diagrams for the Web and Desktop implementation of "Publish Job Posting"
  - Derived the PHP code implementation of "Publish Job Posting" and "Apply to Job Posting" based on the aforementioned design-level Class Diagram
- Jacob Shnaidman
  - Co-designed preliminary Architecture Block Diagram
  - Helped with design of Detailed Design Diagram
  - Wrote the Java code in Android Studio for the implementation of "Apply to Job Posting"
  - Designed and documented Sequence Diagrams for the Mobile implementation of "Apply To Job Posting"
- Robert Attard
  - Co-designed design level class diagram
  - Wrote natural language descriptions for publish use-case workflow
  - Created view package class diagrams
- Matthew Lesko
  - Co-designed preliminary Architecture Block Diagram
  - Wrote draft for Architecture's Description and Rationale
  - Co-designed Detailed Domain Model Diagram
  - Wrote Detailed Design report
  - Made Controller Package Class Diagrams

2.2 Key Decisions

1. **Use Cases**

   - In order to correctly implement the Entity-Control-Boundary pattern and symbolism in the Use Cases, the naming of the cases from the first deliverable were reviewed and adjusted. Moreover, the relationships between each action was refined so as to reflect the following rules:
     - Actor only interacts with <<Boundary>>
     - <<Entity>> only interacts with <<Control>>
     - <<Control>> manages the interaction flow between <<Boundary>> and <<Entity>>
   - The overall structure of the Use Case for deliverable #2 focused solely on the Profile subclasses Admin and Instructor. To highlight this structure the Use Cases made a deliberate use of Generalization.
   - The "Publish Job Posting" Sequence diagram had to be clear about the alternative actions based on the approval or disapproval of the Posting instance. As such, an ALT pane was implemented only for the Web Use Case, which didn't necessarily imply the posting would be automatically approved. As opposed to the Desktop application, only used by Admin which ensures the posting is automatically approved at the moment of submitting a publish form.
   - It was decided that the Desktop app can handle the creation of all major entities. Starting with no data, the Desktop app has the power to create all necessary actors for all features of the program to be possible.
   - For the purpose of Deliverable 2, the mobile app does not have the ability to create student profiles. Students register for jobs by specifying their username, as authentication has not been implemented yet.
   - Since the mobile app does not support the creation of Students or Jobs, it was decided that it should programmatically create Students and Jobs if no current persistence XML file is found. This was decided solely for the convenience of the graders of this deliverable. If this is not desired, the grader may input his/her own XML to the appropriate directory.

2. **Domain Model**

   - ProfileManager, CourseManager, and ApplicationManager were added to the domain model to assist in the creation and persistence of Profiles, Courses, Jobs, and Applications.

3. **Architecture**

   - The architecture was designed using two different architecture design patterns: a Model/View/Controller pattern and a Layered Architecture patter. The MVC allows one to make changes to a certain component, say the Model component, without affecting the two others. This allows for modular design. The team is also experienced with the MVC pattern, hence another reason why it was chosen.

- Since an authorization and authentication service is required of the system, the architecture consists of an Authorization and Authentication layer above the MVC layer. The user first interacts with the Authorization and Authentication layer, then once access is granted, the user has access to the MVC layer, which is the TAMAS registration system.

2.3 Work Plan

In the following Deliverable, thorough testing needs to be completed. The following are deadlines and the estimated effort it will take to complete these tasks on a scale of 1 to 10.

- March 10 - First iteration of unit tests for Apply to Job and Publish Job posting on web, desktop and mobile. Estimated Effort: 8/10
- March 13 - Second iteration of unit tests for Apply to Job and Publish Job posting on web, desktop and mobile. Estimated Effort: 6/10
- March 13 - First iteration of System and Component tests. Estimated Effort: 6/10
- March 15 - First iteration of Stress Testing and Final iteration of unit tests. Estimated Effort: 6/10
- March 16 - First iteration of Descriptions of all tests done. Estimated Effort: 5/10
- March 10 - Deadline for Deliverable 3, including editing of backlog and update of workplan. Estimated Effort: 4/10

2.4 Work Hours

- Harley Wiltzer
  - Domain Model Modification: 2 hours
  - Java Desktop App: 9 hours
  - Android App: 2 hours
  - Sequence Diagrams: 2 hours
  - Compilation and Editing of Final Report: 3 hours
- Camilo Garcia La Rotta
  - Use Case Diagrams: 3 hours
  - Sequence Diagrams: 2 hours
  - PHP: 9 hours
- Jacob Shnaidman
  - Architecture Block Diagram: 2 hours
  - Sequence Diagram: .66 hours
  - Android Studio: 8 hours

- Robert Attard
  - Detailed Domain Model Diagram: 2 hours
  - View package class diagrams: 4 hours
  - Publish workflow description: 3 hours
- Matthew Lesko
  - Architecture Block Diagram and Description and Rationale: 3 hours
  - Detailed Domain Model Diagram: 1 hour
  - Detailed Design Report and Class Diagrams: 3 hours

3 Deliverable 3

3.1 Main Tasks

- Harley Wiltzer
  - Wrote the Description section of the System Test Plan
  - Wrote the Rationale section of the System Test Plan
  - Wrote the Test Coverage section of the System Test Plan
  - Wrote and designed the two detailed test cases for the System Test Plan
- Camilo Garcia La Rotta
  - Wrote the Description section of the Unit Test Plan
  - Defined the structure to present the test cases of the Unit Test Plan
  - Wrote half the classes tested of the Unit Test Plan
  - Wrote the Test Coverage section of the System Test Plan
  - Wrote the Test Techniques and Tools section of the Unit Test Plan
  - Derived next workplan
  - Corrected Integration Description, Strategy and Techniques and Tools section
- Jacob Shnaidman
  - Wrote the integration strategy
  - Made the integration diagram
  - Wrote the skeleton for the integration tests
- Robert Attard
  - Detailed the different cases for integration test
  - Compiled and formatted integration test plan section
  - Aided in developing a template for the integration test format
- Matthew Lesko
  - Involved in the Unit Test Plan
  - Wrote Unit Test Cases
  - Involved in Classes to Test and Classes to Not Test
  - Involved in Techniques and Tools
  - Wrote Differences in Unit Testing for the Three Platforms
  - Involved in Coverage Statistics for the Unit Test Plan

3.2 Key Decisions

1. **Unit Test Plan**

   - The main decision was the design of framework of Test Cases. We aimed to have a well structured suite of test cases that could be generalized to the maximum amount of classes. This would minimize the time spent for coding the tests and facilitate the process of verification of such code. The final choice was a set of tests by-attribute rather than by-outcome
   - Tested JaCoCo but decided to continue using EclEmma as sole Desktop code coverage tool due to its use for the course ECSE 321
   - Tested Selenium but decided to continue using Junit as sole Desktop code coverage tool due to its use for the course ECSE 321
   - JaCoCo is available as a plugin for Gradle and will be used as a tool for coverage analysis for the mobile platform
   - PHPUnit library and tool will be used for PHP unit test executions and as a tool for coverage analysis

2. **Integration Strategy**

   - Decided to use a bottom up approach in our integration strategy
   - Decided to focus integration testing on fulfilling functions that incorporate all the relationships between units rather than testing the relationships individually.

3. **System Test Plan**

   - Decided to make system tests for each use case
   - Decided to structure system tests by actor
   - Chose to design detailed system test plan for the Publish Job Posting and Apply To Job use cases, as enough infrastructure has already been programmed for a solid understanding of how those use cases can be tested.
   - Derived the more general logical sequence of actions possible for system Testing based on the Requirement Document of Deliverable #1

3.3 Work Plan

In the following Deliverable, a formal release pipeline plan will be published. The report will contain an overview of the implementation (tools, design and rational) at each phase of the process. The following are deadlines and the estimated effort it will take to complete these tasks on a scale of 1 to 10.

- March 21 - Integration phase section Starts. Estimated Effort: 8/10
- March 24 - Integration phase section completed.
- March 25 - Build phase section starts Estimated Effort: 8/10
- March 28 - Build phase section completed.

- March 29 - Deployment section starts Estimated Effort: 8/10
- April 1 - Deployment phase section completed.
- April 2 - Compiling final report, editing of backlog and update of workplan. Estimated Effort: 5/10
- April 2 - Deadline for Deliverable 4.

3.4 Work Hours

- Harley Wiltzer
  - System Test Plan: 6.5 hours
  - Unit Test Coverage: 0.5 hours
  - Unit Test Techniques and Triggers: 0.5 hours
- Camilo Garcia La Rotta
  - Unit Test Plan: 5 hours
  - System Test Plan Coverage: 1/2 hour
  - Workplan: 1/2 hour
  - Integration Test: 1 hour
- Jacob Shnaidman
  - Integration Strategy: 3 hours
  - Integration Diagram: 1/2 hour
  - Integration testing skeleton: 1/2 hour
- Robert Attard
  - Integration test cases: 1.5 hours
  - Integration test plan section: 2.5 hours
- Matthew Lesko
  - Unit Test Plan Involvement: 5 Hours

4 Deliverable 4

4.1 Main Tasks

- Harley Wiltzer
  - Wrote the entire Build Pipeline
  - Edited the Integration Pipeline and wrote the Integration Process sections
  - Designed and carried out all System Tests
  - Themed the Desktop Application to make it consistent with the Web Application
  - Implemented the "Hire Applicant", "Modify Job Placement", and "Accept/Reject Job Offers" Use Cases for the Desktop Application
  - Began developing the build system
- Camilo Garcia La Rotta
  - Wrote the description of all the deployment phase sections
  - Wrote the Design section of the deployment phase
  - wrote the Rational section of the deployment phase
  - wrote PHPUnit test cases for Course, Job and Profile classes
  - Fixed bugs found in the Controller classes while writing the unit test cases
  - Made a complete visual overhaul of the web application
  - Debugged Controllers and Validators based on failing PHPUnit tests
- Jacob Shnaidman
  - Wrote the integration tests for the Desktop app
  - Co-wrote the Integration pipeline
  - Modified java controllers based on tests
  - Wrote the Java portion of the testing-state doc
  - Wrote the workplan/edited backlog
- Robert Attard
  - Wrote the integration test for the Web and unit test for desktop Application class (Still a work in progress)
  - Llaborated and added to descritpion and rationale of tools for integration pipeline.
  - Documented the state of the integration testing for the web app.
  - Performed Black-Box testing of the Web app.
- Matthew Lesko
  - Wrote the Description and Rationale for the deployment tools
  - Wrote JUnit test cases for Course, Job, and Profile
  - Implemented certain error messages for controller classes

4.2 Key Decisions

1. **Unit test cases**

   - During the coding of the test cases we realized the uniqueness of the CDN and username attribute of Course and Profile classes was not being handled by the model natively. Thus, we had to make sure the respective controllers made the verification before submitting any modification to the model and to the persistence.

2. **Build System**

   - Decided to have one unified build script that can build targets for all platforms
   - Designed the intended file structure of the overall system

3. **Integration**

   - Decided which tools to use (Documented in Deliverable 4 Section 2)
   - Decided on naming conventions (Documented in Deliverable 4 Section 4-5)

4. **Deployment Phase**

   - Ensuring that the delivery of the product to different platforms worked without manual input by any of the developers proved to be hard to achieve with a single delivery tool. We decided instead to use platform specific tools which would save configuration scripts to handle each platform while allowing the team to use a more appropriate tool for each target platform For further details please refer to the report of Deliverable #4.

5. **Desktop Application**

   - Added the `offerSent` property to the Job class and the `offerAccepted` property to the Application class in the domain model, to allow persistence for job offers and acceptances.
   - Added another persistence file to store global variables to be persisted. This allows the persistence of the Administrators choice to send job offers.

4.3 Work Plan

- April 2nd - Finish Slides for presentation. Estimated Effort: 5/10
- April 3rd - Finish Extra features for presentation. Estimated Effort: 10/10
- April 3rd - Have practice presentation. Estimated Effort: 7/10
- April 4th - Present TAMAS in class. Estimated Effort: 10/10
- April 6th - Finish Tests for all remaining unimplemented use cases. Estimated Effort: 6/10
- April 10th - Finish Writing code for all implementations. Estimated Effort: 9/10
- April 11th - Finish all pipelines and documentation. Estimated Effort: 7/10

4.4 Work Hours

- Harley Wiltzer
  - System Tests: 2 hours
  - Build Pipline: 4 hours
  - Desktop Application Theming: 1 hour
  - Edits to the Integration Pipeline: 0.5 hours
  - Implementation of three Use Cases for the Desktop Application: 6 hours
  - Progress on the build system: 2 hours
- Camilo Garcia La Rotta
  - PHPUnit test cases: 3 hours
  - PHP Controller fixes: 1.5 hours
  - Deployment phase documentation: 4 hours
  - Web app css overhaul: 3 hours
  - Web app controller, validator debugging: 3 hours
- Jacob Shnaidman
  - Integration tests: 4 hours
  - Integration pipeline: 3 hours
  - Modifying code: 1 hour
  - testing state doc: 1 hour
  - workplan/backlog: 1 hour
- Robert Attard
  - Unit testing (desktop Application class): 3 Hours
  - Integration testing (Web): 5 hours
  - state of Integration testing in document: 0.5 hours
  - Integration Pipeline: 2.5 hours
  - Black box testing of the web app: 1 hour
- Matthew Lesko
  - Unit Testing: 12 Hours
  - Deployment Phase: Tools Description and Rationale