**CS3733-D21 Project Part C**

**Requirements, Analysis and Design Document**

**Prototype Application**

Due: W 4-14 at 11:59pm

**Submission**

You will submit a zipped file to Canvas that includes

1. A Word document containing a list of sized user stories for your JavaFX prototype application
2. a JAR file of your JavaFX prototype application
3. a copy of the **Requirements, Analysis and Design Document** below, in Word or PDF format
4. (optional) a copy of a spreadsheet containing the survey responses if this is not already included in an Appendix in the above document
5. bwXnodes.csv and bwXedges.csv

This project assignment is complex and has many components. Use Trello, Airtable, Jira, etc. to keep track of who is doing what and how work is progressing for this week. A suggested start would be to create the following categories: Product Backlog, Sprint Backlog, Assigned, In Progress, Blocked/Paused, Done. Some teams may also want a Review category before Done.

**Prototype Application** (Second Practice Sprint)

Create **sized** user stories for the prototype application, and put them in this week’s sprint backlog. If you break down the user stories into smaller tasks, treat these tasks like informal user stories – size them and include them in the sprint backlog. This will permit the entire team to track what everyone is doing and the status of every task by looking at the Trello boards (or something equivalent).

Using the starter project file, write a JavaFX program that demonstrates reading your CSV files from Project B and creating an embedded database table from it. The following JavaFX screens are in the application:

1. A default page with buttons that, when clicked, take you to the other pages.
2. A second page that displays the data from your MapXNodes.csv (X representing your team letter) and permit users to modify the data.
3. In a third page, do the same thing for MapXEdges.csv. You should create Java classes that simplify the operations on a table’s record and on a record’s field. Permit the user to load or save node and edge CSV files.
4. In a fourth page, demonstrate a simple A\* pathfinding program with the parking map that you created in Project B.
5. In the remaining pages, take at least one or two of the service request systems from Project B and use JFoenix (jfoenix.com) Material Design user interface components. Note that Iteration 1 next week will be created using JFoenix.
6. Make sure to include a way to exit the application.

Please note, the structure of this program is strictly to make sure you can move between windows and should not be considered a good user interface design.

Using Gradle, generate a JAR file. Make sure that the program correctly accesses the database if it is run from a computer other than a team member’s computer! Include some JUnit tests for your application. One of the leads or assistant leads should make sure everyone knows how to write JUnit tests.

**Description of the Minimal Application** (Sprint 1)  
This will be implemented in the Iteration 1 development sprint **next week**. This description is given you so can create the Requirements, Analysis and Design document in the next part. The minimal application is to be created using JFoenix Material Design for the user interface.

1. Path finding application for hospital patients and employees. Given a starting and ending location, your application will demonstrate a path drawn on floor 1 of the hospital that includes the two parking lots from Project B and the inside of the 75 Francis building, but does not include the Shapiro building or the Building for Transformative Medicine.
2. Map editor that allows a hospital administrator to create an undirected graph. It should be possible to add, modify and remove nodes; and add, modify and remove edges. The application should be able to reset the map by loading a CSV text file with nodes and edges. The nodes and edges should be able to be backed up into a CSV text file. Improve upon Project C by adding a minor graphical component to the map editor. Note that the application does not have to read the data into the database whenever it starts up. Databases have persistent data between application invocations.
3. Create at one or two service request components that permit an employee to create a request, assign a person to fulfill it and mark it completed when it is done. The request should include field(s) so an employee can fill in the location(s) of the request (nodes on the campus map). Refer to Project B for a list of possible service components.
4. A menu option or button to exit the application gracefully.
5. For the minimal application, you do NOT have to create an employee database and you do NOT have to create different categories of users (patient, employee, administrator). However, this will be required in sprint 2, so if you want to, you can implement it in sprint 1.
6. Add a feature or two that enhances your application.

Note: Your application will utilize an embedded database (Apache Derby) for persistent storage between program invocations. The application should be able to back up the data into CSV files, and load new maps with accompanying CSV files. You are not limited to the CSV files that I have provided but can add supplementary data in separate files that are linked to the existing files using a foreign key.

**Requirements, Analysis and Design Document**

This document should be **written as a professional report** and is to include the following items in order (you will probably include some items that you did in Project B):

1. Cover page including
   1. WPI CS3733-D21 Software Engineering, Prof. Wong
   2. Team name
   3. Team coach
   4. In a table, list each person, their position, and their GitHub account name.
   5. GitHub link to your organization. Make sure repositories are viewable to me and to your team coach!
2. Table of Contents
3. Introduction. This should mention the application you will be developing in this course, for whom, and for what purpose.
4. Software development environment. List all of the software tools and programming environments your team is currently using, most of which is dictated by the class.
5. Functional requirements (for entire application, not just the minimal application)
   1. Requirements gathering – brainstorming
   2. Requirements gathering – summary, graphs and discussion of survey results
   3. Requirements gathering – summary and discussion of interview results
6. Functional Model
   1. Epics for the full application. With guidance from your survey results and interviews, determine as best as possible the application features that your team would like to implement by the end of the term.
   2. User stories for just the minimal application (sprint 1). Prioritize them, and place them in the **sprint backlog** using Trello, AirTable or GitHub Issues. You will size the user stories next week. Create a **product backlog** that contains the remaining prioritized epics and user stories that will not be implemented in sprint 1. List your sprint backlog and product backlog in this section of the paper. You may include screen shots of the backlog instead provided that it is readable.
   3. UI mockups using Figma or Adobe XD for the minimal application. Just include screenshots in the document
   4. Use case diagram only for your minimal application (path finding, map builder, and service request components). Textual use case descriptions are required only for the minimal application’s path finding component. If you choose to do the use case descriptions for the other components, include them here as well.
   5. (Optional) Scenarios
   6. (Optional) Storyboards
7. Design Models just for the minimal application.
   1. Object model – FXML pages, controllers, and entity classes. Use rectangles to represent JavaFX pages. The class diagram should include **class associations**, methods, and attributes.
   2. Dynamic model – sequence and activity diagrams just for the path finding component. Statechart diagram just for the scene switching part of the application.
   3. ERD – database entity-relationship diagram.   
        
      **Important**: the creation of the design models should be conducted in a way that everyone, especially the more junior developers, understand why the classes were created and how they should be used. Keep in mind that since this is the first time you are attempting to create a design that you will probably need to change your design after you start programming!
8. Project management
   1. Any changes in team positions since Project Part B
   2. Indicate what team events were done to improve team bonding.
   3. Acknowledge significant contributions by your team members in a personal manner. Allocation of $60 WPI dollars to team members. Justify your **rationale and method** for the final salary distribution. In a table, list the team members and their salary in **alphabetical order by last name** using the following format:

|  |  |  |
| --- | --- | --- |
| Last name, first name | Salary | Rationale |
|  |  |  |
|  |  |  |

1. Appendix
   1. Survey data (you may include this as a separate spreadsheet file)
   2. Interview data

Do not copy this outline and fill it in verbatim – it is not professional. For example, you would NOT have a heading of **1) Cover Page** on the front page. However, you will want to number your sections starting with the Introduction so the Table of Contents is enumerated.

**CSV Files**

You will now merge your parking lot CSV file with the existing files for the building. Your parking lot nodes are connected to the building map only through the two entrances on floor 1 for now. Name your files bwXnodes.csv and bwXedges.csv where X is your team letter. See Assignments -> Project -> Project Maps and Data for the building’s nodes and edges files.