**MSc Project Idea Management Web Application.**

# Introduction

This project looks at developing a Web Application that is to be used to manage Project Ideas for an MSc Degree at the University of Portsmouth. The application is built using Java EE 7 technology.

The application will be used to submit, delete and edit ideas, look up submitted project ideas, and a student will be able to select a project idea that they will be undertaking as their project. As well as giving the user a nice experience by making it responsive for all devices and having a clean user interface.

This report, therefore, explains the design and development process of building the web application.

# Design

From analysing the requirements of the application, I broke down the system into needing the following components:

* Person
* Idea

The person component will hold any data for a user. This will hold what type of user they are such as Staff or Student. I wanted to name this component User, but User is a reserved SQL keyword, therefore, had to use the word Person to define this component as. This meant that this name wasn’t very self-explanatory. I decide to combine an organisation into the user component as this allows for better management functionality.

The Idea component will hold any data for the project ideas.

I have defined two relationships between these components. An Idea has a many-to-one relationship with Person. This relationship defines what Person (User) submitted the idea. This is to allow a person to add many ideas and be able to manage each of them when they want. The other relationship is a one-to-one between Person and Idea, this to maintain what Student User is implementing an idea. One-to-one is used because one student can only implement one idea. The relationships will also have the reversed relationships mentioned between the components.

The user interface follows the same colour scheme and navigation throughout the whole of the application as this will stop users from getting confused on the pages.

## Changes

The design of the application from the initial design has changed as when developing I had found an attribute for Java EE called “Rendered” that allowed the developer to choose what to show based an if statement. For example, as shown in Figure 3 I used the rendered attribute to only render the Edit button on Idea Page if the current displayed exists in the database and if the user is a staff member or the user who submitted the idea. This attribute meant that I could cut down the number of pages needed, and combine similar functioning pages together. For example, I initially had a separate page for submitting an idea and editing an idea but these two were combined as they need the same inputs, the only difference is the submit button. One needs an Add Button and the other needed an Update Button.

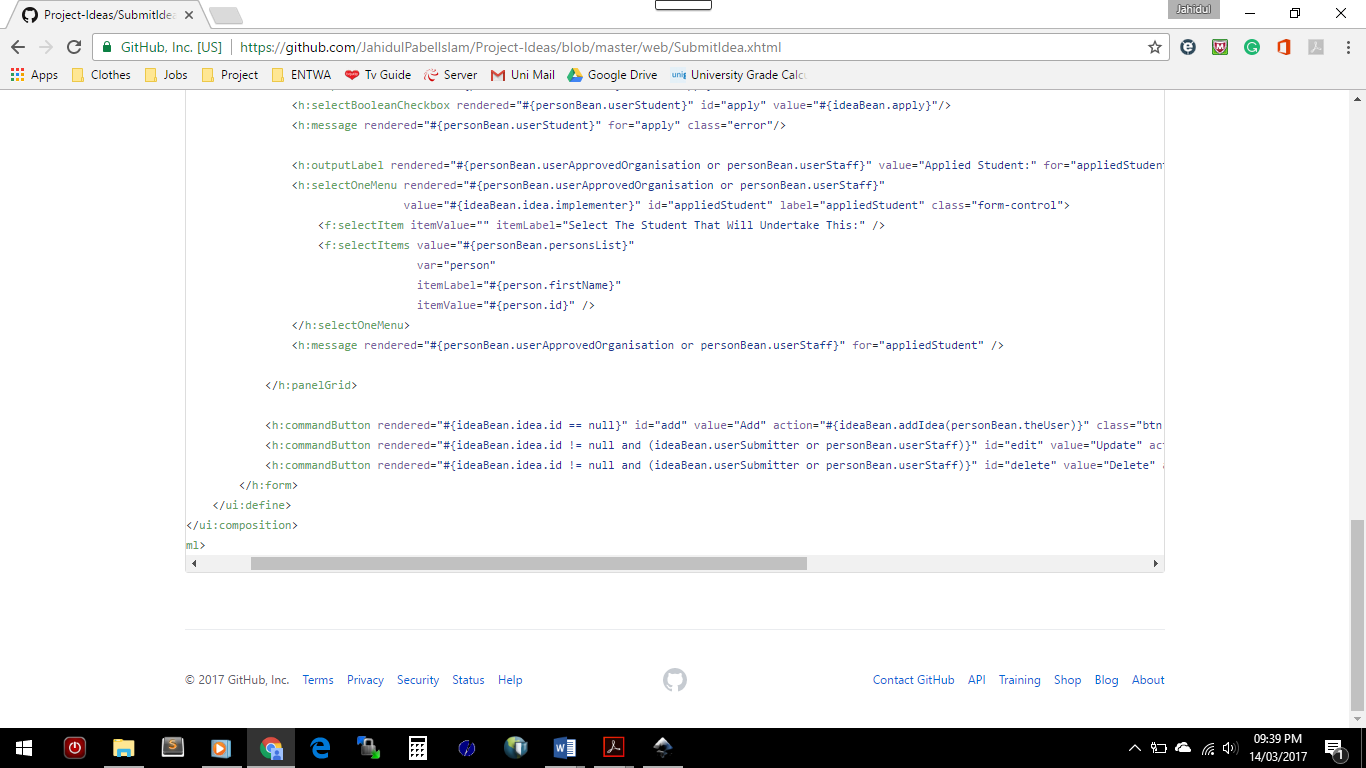


Figure 2 Rendered Example

The structure of the application had slightly changed from the initial MVC pattern as I had put some of the business logic in the controllers. This was because I already had methods for some of the business logic needed in the controller which is used for the rendered attribute in the View. Therefore, I had thought it was unnecessary to have repetitive code in both the controller and the business logic. As illustrated in Figure 3 the “isUserSubmitter” method was already there for use for the Views, therefore, it is used again in the controller.

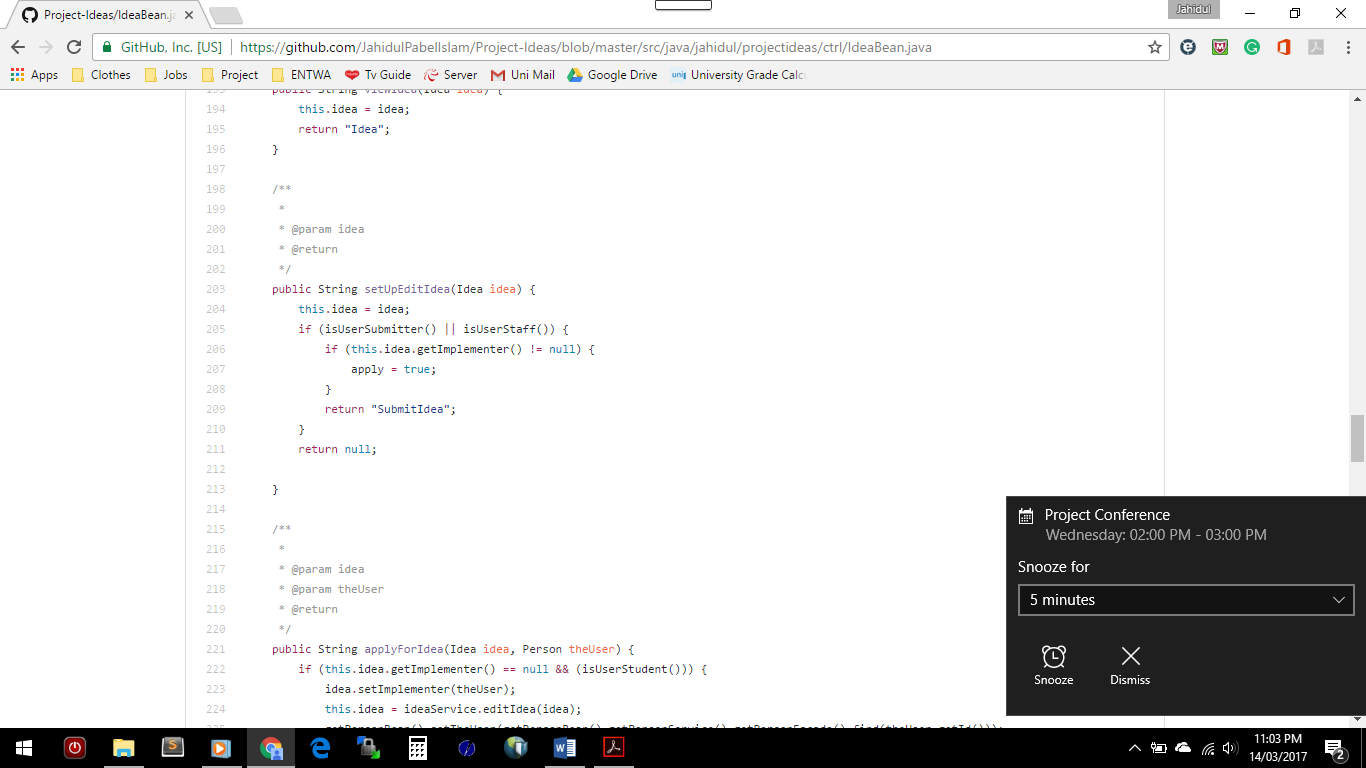
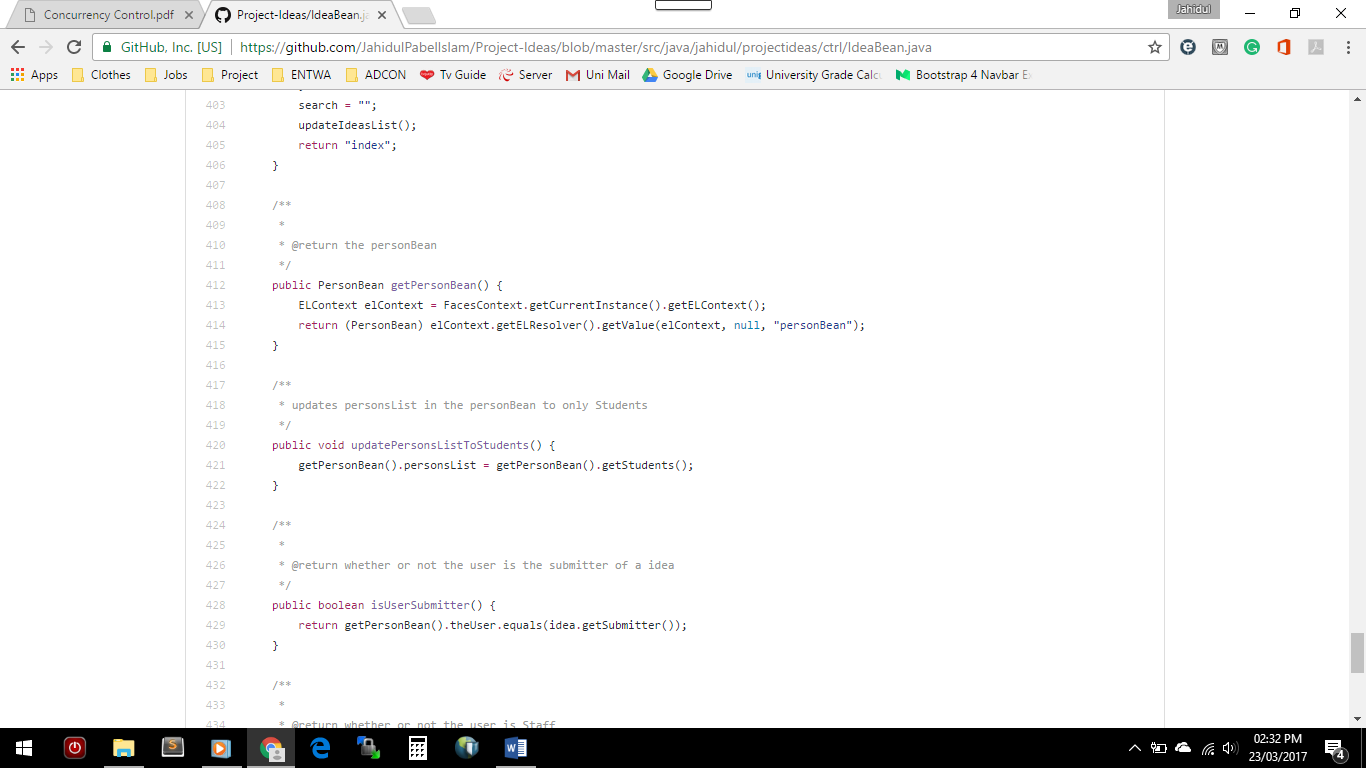
 

Figure 3 Business Logic in Controller

# Implementation

## Development Tools

Java Enterprise Edition 7, Java Server Faces Framework (JSF), Derby (Java DB), Glassfish 4.1 Server Bootstrap was used to develop the application.

NetBeans was the IDE that was chosen for the development as the IDE includes all the technologies that the project needed as mentioned above. Developing in this environment was easy going as I have previous experience using the IDE. The only downside of NetBeans was that it was slow on load in terms of NetBeans initially starting as well as when running a project. NetBeans has a code generation functionality where it generates common code for you when needed for example the getter and setter for variables which was new to me, but I liked the feature as it made developing easier. NetBeans also provides a feature to auto-generate comments which can be translated into documenting web pages, this makes coding much simpler as the developer doesn’t have to worry about creating separate web pages for the documentation of the code (Duke, 2004).

JSF provides a templating functionality to input common code such as the header and footer into many pages (MKyong, 2012). For example, my project has the same header in all its pages, therefore, I had created a page just for the header and used templating to automatically input the header into the pages I wanted which is all the web pages. This allowed for a consistent design making it easier for users to use the different pages and easier for maintenance.

What I enjoyed during the development of the web application was that the tables in the database were auto-generating on the first run using the Entity classes. And anytime I wanted to make a change to an entity I could make the change on the Entity class and I could delete the current table and when run next time it would auto create again.

Bootstrap CSS framework was used and some CSS was written myself to customise to make the login form narrow and centre align it to give the page a cleaner look which Bootstrap doesn’t include. Also, added a custom background colour of white to the tables because bootstrap’s colour was blending in with the main background colour. Bootstrap was used as it provided a quick approach to designing a web application without having to worry about coding the design instead can focus on the functionality of the application. This also allowed for a mobile first approach to the design allowing many users on many different sized devices to go on the application (Bootstrap. 2017).

GitHub was used to manage the code for the project as it allowed me to upload updated code and in case any errors occurred in the future to turn back to a certain point of the project where it was working and find out what went wrong.

## Issues

When developing the editing feature of a ‘Person’ or an ‘Idea’, I had a few issues when trying to update. The problem was that when you press update, it wouldn’t update the data it would create a new row in the database with the new inputs. The scope of the controllers was Request at the time however when changed to SessionScoped it worked even without changing any other piece of code. There this was the solution that was used for the problem as no other fix was found using a RequestScoped Controller.

NetBeans doesn’t automatic refactor any changes that are made to any method or file names unlike other IDE’s such as PHPStorm. This was a big issue as when you make many changes to the names of methods, you would have to manually change the name everywhere where it is used. This took up precious time that I had very little of anyway.

Another problem that was encounted was when trying to run the project sometimes I accidentally opened the project without having an absolute file path for the project without realising. This meant that the project wouldn’t run. This happened a couple of times before I had decided to read the error messages and I finally figured out what was going wrong. Also, a few times when I had opened the project in a new device the project wouldn’t run, however, I also didn’t read the error message properly as the message said to go to the log file to see what the problem is. And it said exactly what was wrong when trying to run the project, the problem was that it couldn’t find a data source, however, using Professor Jim Briggs Configuring the Data source help page (2016a) the problem was solved quickly and easily. Therefore, next time an error occurs, it would help to have a good look at the error/log files.

# Testing

Testing for the project was done manually, using the black box method. This approach was chosen because of the time constraints. I had thought instead of taking time out creating test classes/functions it would be quicker and easier to do testing through manual function testing. The testing was done throughout the whole process of implementation making sure after a feature was developed or changed it was working how it should be. As testing throughout the developing stage was done it saved time at the end of the project as it final testing wasn’t necessary needed. If time allowed it would have been better to have a proper testing mechanism in place.

# Summary

I have successfully completed the project, which includes completed requirements and design specifications, finished artefact which is commented which produces Javadoc for the Application.

The final artefact includes all the requirements that were initially set. This was because Professor Jim Briggs’s Step had set us an exercise to build a basic application. I had used his step by step page of building a basic application to build the exercise (2016b) this provided me with the understating of how to build a java web application. Therefore, the exercise gave me the building blocks for my application and was able to build on it as the exercise made the process easier.

The application isn’t very secure in terms of the registering and logging into the system as well as the block unwanted access to pages that normal users can’t view. The current system only blocks it via the navigation of the application however it doesn’t block any users that directly type in the web address to an unauthorised page.

The application should handle exceptions or bugs better. Code that deals with user inputs should be wrapped in a try-catch block in the controller and validate the data and throw any exceptions if necessary in the business logic and users should be informed accordingly. This will provide the user with a better experience.

The development could have improved if I had a Test-Driven approach to the project. Would’ve also been better if I had written JUnit Tests to accurately test the application as with me manually testing it could mean inaccurate results as I know what works and I could only be doing what I know. Also, would fill the database with loads of dummy data to test the performance.

# References

Bootstrap. (2017). Bootstrap. Retrieved from <http://getbootstrap.com/>

Briggs, J. (2016a). Configuring a JDBC datasource in Glassfish. Retrieved from <http://briggs.myweb.port.ac.uk/WEBP/notes/db/dsconfig.html>

Briggs, J. (2016b). Step by step JPA/JSF webapp 1. Retrieved from <http://briggs.myweb.port.ac.uk/WEBP/notes/db/jpasteps.html>

Duke. (2004). Javadoc. Retrieved from <https://www.cs.duke.edu/courses/spring04/cps004/notes/pdf/4javadoc.pdf>

MKyong. (2012). JSF 2 Templating with Facelets example. Retrieved from <https://www.mkyong.com/jsf2/jsf-2-templating-with-facelets-example/>

stackoverflow. (2014). Get bean values in another bean in prime faces. Retrieved from <http://stackoverflow.com/questions/25209786/get-bean-values-in-another-bean-in-prime-faces>

tutorialspoint. (2017). MVC Framework - Introduction. Retrieved from <https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm>