**SUMS**

**Project Ideas**

**for**

**MSc Students**

**Final Report**

**By**

**UP687776**

**Zain Ali**

1. **Introduction**

SUMS website allows MSc students to create, view, apply for ideas but not allowing them to make changes to idea the students own. The primary goal of this coursework is to develop a better version of SUMS website. This web application will allow users to submit, update, and withdraw from ideas and better searching functionality has been provided

The secondary purpose of this coursework is to look at latest Java technologies like Java EE 7 using MVC for building enterprise web application and providing users with richer user experience by using latest responsive technologies like CSS3 and BootsFaces.

The purpose of this report is to explain design, development experience as well as any hurdles during development.

1. **Design**
   1. **Model View Controller (MVC)**

The requirement to create this application was using MVC design pattern. The reason being is that it can be simple implementation of complex projects by keep logics and code separate. Using MVC helped allow to keep business logic separate to view by using controller which act as “man in the middle”. This will help future developer to understand architecture and code flows and is maintainable.

Java Packages were created to create MVC patterns then those packaged were called Persistence, Entity, Business (Model), Controllers and View (Web). The business package also contained sub package call Exceptions which would catch any exception should application breaks.

(Liu, S., & Chen, P., 2009, July) stated that “The Java EE platform introduces a simplified programming model and eliminates much of boilerplate that earlier releases required. ” Allowing developers to work keep the code clean and maintainable.

Please refer to figure 1 to see how MVC has been implemented for SUMS.

Entities

Persistence

**Controller**

(ContactAdminController, ProjectController, OrganisationController, OrganisationConveter, UserConverter, UsesrController

DB

Figure 1: MVC Architecture Design Patter for SUMS web application

**Views**

**Model**

Business

1. **Implementation**

Various Tools and technologies were used to build SUMS web application like NetBeans 8.2. IDE, sublime Text, Glassfish 4.1 Server, Java DB (Derby) and BootsFaces.

* 1. **Development Tools**

NetBeans IDE seemed suitable choice for building this application and based on previous experience using this IDE. Overall, there was rather bizaar and somewhat frustrating experience due to NetBeans keep having to use more than 200% processor resources causing the machine to restart.

After adding new functionality, NetBeans would give errors when trying to run the application but not providing specific errors. However, all it needed was to glassfish to be restarted which is time consuming and newbie would struggle with this as not knowing why it needed to be restarted.

Other frustrating experience was debugging, NetBeans debugging tools aren’t very powerful and doesn’t provide richer debugging experience compare to Visual Studio.

On other hand, there were some features which made life easier, like auto generating getters and setters, Javadocs and documentation. (Harshad Oak 2003 (Ref 4)) mentioned JavaDdoc is a nature, it is used every day and developer cannot think of developing Java application without it Javadoc can be referred when working with API and how easy it is to create provided program is documented properly.

The other good point was auto database and table generation. After making changes to Entities framework, NetBeans would automatically make necessary changes when running the server providing previous tables are manually deleted.

* 1. **Implementation Tools**

BootsFaces was implemented in JFS pages for richer user experience allowed to develop front-end application fast and easily without too many complications. BootsFaces logical choice compare to Bootstrap. Because BootsFaces doesn’t require designer to download all code. In fact, all designer need to do is get dependences and add it to Maven xml files and then start referencing it. That’s all. Whereas, Bootstrap would have required to download all code and packaged which could affect code quality and can cause performance issues.

* 1. **Code Quality**

There is no doubt that code quality could have been much better implemented. Some of the code logic was not implemented properly. Function are too big and messy, logical choice should have been to break down function for code quality and readability purpose. Controller some code which should have been in business logic. Entity classes aren’t consistent when defining columns for db.

There was a common code in every single page which was too much duplication causing taking up to 500 lines of code in JSF pages. Then JSF templating was discovered where all duplicate code was implanted in single template file and then was referenced in all pages saving and improving code quality.

Stack overflow was referred when debugging and looking at some strange error messages. There has been some code used from different website which has been referenced in code and link to where it is taken from (CSS and Java).

* 1. **Missing Functionality**

There was some missing functionality at the end, this is due to time constraints and less experience with MVC and Java EE technologies.

Firstly, a user cannot apply for idea submitted by staff because while changing the code architecture, it became quite complicated to implement apply functionality mainly due to time constraints. This is due to previous code flow wouldn’t support new functionality. So, when making new changes apply functionality could not be complicated. However, an ugly hack and quick solution was to link apply button to contact admin form where user can fill in form to request to take project. But user can submit their own ideas to be considered.

The other missing functionality was making some web pages’ user friendly, this is due to time constraint and not having full understanding of BootsFaces. Some web pages would view raw data and wasn’t user friendly.

There was a bug which couldn’t be fixed in fear of break everything which was every time user, project or organisation values are updates, glassfish needs to be restarted for changes to take affects.

However, the above functionalities aren’t obvious and less likely to cause any problem and might not be noticeable naked eyes.

1. **Challenges Faced**

The project was quite difficult due not much experience in Java apart from few coursework’s during 1st and 2nd year at university. Having no experience in Java EE and debugging made it more difficult to work on project. It was very hard to learn Java EE due to nature of language and constraints from other units and time managements.

Very first challenge was trying to run Glassfish server as it was not starting or deploying the project. After some time, it was discovered that due to nature of Mac security which provided some extra security layer and glassfish setting weren’t set properly.

On other hand, using @RequestScoped was big problem as it would allow to create or delete entities but when it comes to updating them it would create new object rather than updating exiting one. The only solution was then to use @SessionScoped which seems to solve the problem but it isn’t a long-term solution and could cause future problems.

The big impact took place day before project submission where to students could apply for as many idea as they want. However, this seemed illogical move and decided to change this so when student apply for a project it will be automatically assigned to a “unassigned admin user”. Once, the idea is approved then either staff member or admin will make student who applied for the owner. However, this was a big mistake day before deadline as it required to change how user create project and how user type handle project which broke most of the code and there wasn’t enough time to fix it. This resulted in some missing functionality which is mentioned above.

Finally, the other problem keep occurring is updating database when updating any entity on web application. There is not current solution to that and only way to get around I to restart web application through NetBeans which would then take affects.

1. **Testing**

Lots of testing was done throughout development life cycle. The prime focus at the start was to do manual testing and sanity testing after each functionality was added which was good choice as it allowed the application to be tested thoroughly and find any bugs. (ViVEK KUMAR, 2012) mentioned that “Manual testing is process of testing software for defects by playing the role of end user and ensuring application behaves correctly”.

GitHub was used to keep track of all changes, only working and tested code / functionality was pushed to GitHub which allowed to revert to changes when things went wrong.

Writing unit tests and doing automation testing would have been better approach for long term scenario. As more functionality was added to SUMS, it became very hard and time consuming to carry on doing manual testing and it slowed development and testing performance and had effect on time management.

Having experience working as QA and Automation developer, it would have been logical choice to write unit tests and carry on create automation tests cases which would have not only tested functionality but also helped reflect on code quality and less time consuming. Unit tests and Automation is industry standard right now. If this project was to be done again, automation and unit test would also have taken high priority.

1. **Summary**

Overall, this coursework was good experience as well as learning activity to new technologies and how to work on

The project provide user with majority of functionality and tried to be completed according to industry standard. The final piece of software included most of the requirements set in requirement specification in v1.

There was good input from client Professor Jim Brigg’s who also played important role by guiding into correct direction when necessary. The was a web based step by step document provided by Jim which was key point to start creating this web application and helping to keep SUMS MVC style (Briggs, J., 2017). The new SUMS application includes better functionality compare to existing one allowing user to create, update, delete project ideas as well as perform live search on any column.

Exceptions were added to business logic based on client requirements. This application should handle exceptions and provide user with useful message should an exception, error or problem arises.

However, there could have been a lot more done to this project expected at the start, this is due to time constraint and some lack of knowledge. The main function that was implemented wasn’t completed was apply button allowing students to apply for project. Also, user interface should have been implemented properly using CSS3. Due to too many problems and bugs time came between developer and interfaces.

Finally, the code logic should have been implemented in proper way. Some of business logic may have been called in Controller. Time management should have been considered. Mainly new technologies should have been researched and investigated rather than half way through the coursework and then implemented poorly.

1. **References**
2. Kumar, V. (2012). Comparison of Manual and Automation Testing. *International Journal of Research in Science And Technology*, *1*.

<http://www.ijrst.com/images/short_pdf/Jun_2012_VIVEK%20KUMAR%201.pdf>

1. BootsFaces (2017). BootFaces, Retrieved From

<https://www.bootsfaces.net/>

1. Briggs MyWeb (2017). Briggs, J., Retrieved From

<http://briggs.myweb.port.ac.uk/WEBP/notes/db/jpasteps.html>

1. TechnRepublic (2017) Retrieved From

<http://www.techrepublic.com/article/the-importance-of-the-humble-javadoc/>

1. TutorialsPoint (2017) Design Patterns – MVC Pattern, Retrieved From

<https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm>

1. Liu, S., & Chen, P. (2009, July). Developing java EE applications based on utilizing design patterns. In *Information Engineering, 2009. ICIE'09. WASE International Conference on* (Vol. 2, pp. 398-401). IEEE.
2. JavaPoint (2017) MVC in JSP, Retrieved From  
   <http://www.javatpoint.com/MVC-in-jsp>