

# **Report Title**

Candidate No: 105936

17th May 2019

# Contents

|   |                            |   |
|---|----------------------------|---|
| 1 | 3-Tier Architectural Model | 3 |
| 2 | Securing the Application   | 3 |

## **1 3-Tier Architectural Model**

All of the Java code is in three different folders with it being split between data/entities services/logic and controllers for the views therefore it is split quite nicely. Ideally they would also be split further depending on their uses to make the code cleaner. It makes use of the Model-view-controller pattern by using the xhtml pages as the views, the jsf backing beans as the controllers and the entities as the models. However I feel that the three tier could be better implemented by splitting the application across different programs. For example if we were to implement a relational database manually with an ORM framework it would be a lot simpler and have performance and understanding bonuses. We could also use a different web server which focuses on using controllers to provide a RESTful API, and then a separate frontend which would communicate with the API. This has a number of benefits such as being able to create multiple applications which communicate with the API, being able to split the code across multiple projects meaning that its usages are clearer.

## **2 Securing the Application**

The application has been secured using realms and then restrictions have been put on access available to the pages using the web.xml deployment descriptor. However I feel that this is an overly complicated setup and that a much more simpler solution is to manage the restrictions separately as it would be far more functional and quicker. Using cookies with a sessionId would also mean you can include other forms of authentication such as OAuth.

## **3 Extending Design to Avoid a Single Point of failure**

The main way would be to extract the database to a different server then ideally dockerise the application and then use Kubernetes/Docker Swarm to deploy the application on multiple servers or deploy multiple containers of the application on one server. To do this you need to make sure that none of the requests are session based as different servers can pick up different requests. This is where a RESTful API also has benefits.