**Why certain approaches were used, why others were not selected?**

*Technology stack used:*

* Java 1.8 with Spring Boot, Spring Security, JPA
* Maven as a dependency management tool
* Log4j for logging
* Junit for integration testing
* JSP, Bootstrap, HTML5 and CSS3 for frontend

The main reason for choosing the above-mentioned technology is that it is entirely open source, and it decreases the amount of time spent on development and enhances the overall efficiency of the software development. Another reason is, I am already familiar with these technologies. Also, Spring Boot is an open-source micro-framework for creating microservice-based Spring applications. It can run standalone Java applications as well as traditional WAR deployments. Spring Boot allows developers to get right to work coding instead of wasting time setting up and configuring the environment. To test web applications, it has embedded HTTP servers such as Jetty and Tomcat.

**Design pattern used.**

* *Model View and controller*: Model-View-Controller (MVC) is a software design pattern that separates concerns within a software program. In theory, the application logic (or controller) is isolated from the technology (or view layer) used to present information to the user. The model acts as a channel for communication between the controller and view levels.
* *Observer Design Pattern:* The event mechanism in ApplicationContext uses this pattern. Create a one-to-many relationship between objects so that when one item changes state, all of its dependents are automatically notified and changed.
* *Dependency Injection:* The Spring framework features an IOC container that is in charge of creating objects, wiring them together, configuring them, and managing their whole life cycle from creation to destruction. Dependency Injection (DI) is a feature of the Spring container that is used to manage the components in an application. Spring Beans are a type of object like this.
* *Factory design pattern:* This pattern allows the initialization of an object through a public static method, called the factory method. The factory design pattern is used by the Spring framework for the generation of bean objects using BeanFactory Container and ApplicationContext Container.
* *Singleton Design Pattern:* The singleton design pattern ensures that in memory, there will be only one instance of an object that can deliver services. Singleton is the default scope in the spring framework, and the IOC container creates exactly one instance of the object every spring IOC container.

Apart from this I have spent more time on writing the test cases for service layer which are helpful to detect defects in application. Test user is created and can be accessed using username “Testuser” and password “pwd123”