|  |
| --- |
| SE 577: Software Architecture    **Group Homework 3**    Shivani Aggarwal  Jijo George  Vighnajeet Naik  Priya Thotta Jayachandran      21-March-2020 |
| **TrainTicket Quality Attribute Analysis** This section will describe the various quality attributes of TrainTicket application and the way they have been/will  be ensured in the implementation. The major quality attributes are Availability, Interoperability, Performance, Modifiability, Testability, or Security.  Below is the short description as per my understanding about our application for all the above QA’s with respect to  TrainTicket application:   |  |  | | --- | --- | | **Quality Attributes** | **Possible Description** | |  |  | | **Availability** | This relates to software/infrastructure in terms for our TrainTicket Application. As of now the application is running on local system , however once its hosted then the availability would be dependent on that respective provider. | | **Interoperability** | Our application is interoperable for e.g. we would be doing integration with Stripe/Paypal for payment service. | | **Modifiability** | This is open source so there are no restrictions, 100% of code is available and is modifiable.  Also, we are using MVC to adhere to modularity and modifiability. | | **Performance** | This is JVM based, so performance depends on allocated RAM and CPU. | | **Security** | Our application sits behind Spring Security a mature framework for securing web requests and APIs . We may use JWT tokens as well in future for security. | | **Testability** | The application will have automation test script along with unit tests(#To be implemented) to adhere to testability. | | **Usability** | It has a responsive UI which talks about its usability. |   For each of the QA’s, a corresponding QA scenario with the 6 parts have been described.  Below are the three major QA’s that have been covered:   1. Modifiability 2. Security 3. Interoperability   Below are the six parts being included for each of the above QA’s:   1. Stimulus source: software 2. Artifact: processor 3. Stimulus: omission 4. Environment: normal operations 5. Response: notify appropriate entities 6. Response measure: time to detect/repair the fault   Below are the QA scenarios corresponding to each of the above QA’s:   1. **Modifiability**  |  |  | | --- | --- | | **Portion of Scenario** | **Required Values** | |  |  | | **Stimulus Source** | Developer | | **Artifact** | Codebase : TrainTicket codebase is available at location “<https://github.com/sa3664/Group3_TrainTicketingSystem.git>” wherein all modules could be easily found. | | **Stimulus** | A directive to add/delete/modify functionality: TrainTicket application has below features to adhere to this portion of scenario-   * The application is open source so there are no restrictions, 100% of code is available and is modifiable. * The application is feasible to any change in the requirement. * The application is using Spring JPA framework and repository pattern with a loosely coupled application. * The application code is adhering to SOLID principle. | | **Environment** | Design and Build Time : Design/Build of the application. | | **Response** | Response includes the below set of steps for any new change:   * Make the required modifications and changes. * Create Unit tests and integration tests . * Check for code quality and vulnerability with tools such as SonarQube, Facebook Infer. * Deploy the new changed code. | | **Response  Measure** | Cost should be in terms of :   * Effort * Calendar time   As the application is modular so it makes it to be lower risk with less effort and consumption of time as compared to monolithic applications. | | **Code location** | This can be found out in below:  “/TrainDemo\_Group3/src/test”  “/TrainDemo\_Group3/src/main/resources/application.properties” |  1. **Security**  |  |  | | --- | --- | | **Portion of Scenario** | **Required Values** | |  |  | | **Stimulus Source** | Human/User. | | **Artifact** | A component or resources of the system : TrainTicket Login component. | | **Stimulus** | Authorized / Unauthorized attempt is made to display data : User logins into the TrainTicket application as a Traveler or an Admin using the respective credentials as per the permission being authorized for them.  User would have few ways paths to create their account with TrainTicket Application i.e. Github ,Google Mail or can signUp for a new account on our Login Page. | | **Environment** | The system(TrainTicket application) is online connected to a network with being fully operational. | | **Response** | The response is carried out in a fashion such that :   * Unauthorized access : It doesn’t let the user to login with bad authentication. * Authorized access: It will let you into the system as per the authorization i.e. Admin/Traveler and would provide you with the authorities accordingly. * Services are not being manipulated without authorization. * The system may track the authentication via JWT tokens(#To be Implemented). | | **Response  Measure** | TrainTicket would sit behind Spring Security , a mature framework for securing web requests and APIs . We may also use Bearer JWT tokens to have the measured response in future(#To be Implemented). | | **Code location** | The code can be seen at below location:  “/TrainDemo\_Group3/src/main/resources/application.properties”  “/TrainDemo\_Group3/src/main/java/edu/drexel/TrainDemo/configuration/SecurityConfiguration.java”  “/TrainDemo\_Group3/src/main/java/edu/drexel/TrainDemo/Controller/UserController.java” |  1. **Interoperability**  |  |  | | --- | --- | | **Portion of Scenario** | **Required Values** | |  |  | | **Stimulus Source** | A system(TrainTicket Application) | | **Artifact** | The systems that will interoperate : Stripe Payment Server and TrainTicket application. | | **Stimulus** | A request to exchange information among systems: The payment done at the end of the purchase during checkout which comprises of credit/debit card authentication . | | **Environment** | Both the systems are discovered prior to runtime. | | **Response** | The response is carried out in a fashion such that :   * Stripe authentication Failure: The request would be rejected in case of incorrect details and the person would be asked to reenter the details on the TrainTicket application for proceeding to purchase. * Stripe authentication Success: The request would be accepted in case of correct information and the purchase would be successfully completed with a success response being send by “payment\_intent.succeeded” API due to successful information exchange between both the systems(TrainTicket Application and Stripe Payment server). | | **Response  Measure** | TrainTicket would sit behind Spring Security , a mature framework for securing web requests and APIs . We may also use Bearer JWT tokens to have the measured response in future(#To be Implemented). | | **Code location** | The code can be seen at below location:  “/TrainDemo\_Group3/src/main/resources/application.properties”  “/TrainDemo\_Group3/Pom.xml”  “/TrainDemo\_Group3/src/main/java/edu/drexel/TrainDemo/Controller/CheckOutController.java” | |

# 