**CST-339 Programming in Java III**

**Project Status and Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | *3* | |
| **Date:** | ***October 2, 2022*** | |
| **Revision:** | *3.0* | |
| **Team:** | 1. *Matthew R McCormack* | |
| 1. *David “Nate” Cole* | |
|  | |
|  | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Total Hours**  **Worked** | **Hours Remaining** | | *Updated Design Report, implemented dependency injection, and collaborated to complete code.* | *Matthew McCormack* | *16* |  | | *Debugged and cleaned code, and created/implemented Product Creation Module.* | *David Cole* | *42* |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| **GIT URL:** | *https://github.com/ColeDNate/CST-339.git* | |
| **Screencast URL:** | *https://www.loom.com/share/decc2424b81f44d992cfaef38519935b* | |
| **Peer Review:** | *Y/N* | We acknowledge that our team has reviewed this Report and we agree to the approach we are all taking. |

**Planning Documentation**

**Initial Planning:**

*This can be any task lists or sprint planning you completed to complete this assignment.*

A

**Retrospective Results:**

*The following table should be completed after each Retrospective on things that went well (keep doing).*

|  |
| --- |
| **What Went Well** |
| Team was able to collaborate and troubleshoot issues, allowing the code to be updated properly. |
| A |
|  |

*The following table should be completed after each Retrospective on things that didn’t go well (stop doing) and what would be done differently next time with an action plan to improve (try doing and continuous improvement).*

|  |  |  |
| --- | --- | --- |
| **What Did Not Go Well** | **Action Plan** | **Due Date** |
| Difficulties with last phase significantly delayed start. | a | **Oct. 3, 22** |
| a | a |  |
|  |  |  |

**Design Documentation**

**Install Instructions:**

*Step-by-step instructions for setting up your database, and configuring and deploying/installing your application. This section should also include detailed instructions for what configuration files are required by your application, what configuration settings need to be adjusted for various runtime (development or production) environments, and where the files need to be deployed to. This section should also contain detailed instructions for how to clone your application source code from GIT and deploy the application to an externally hosted site.*

A

**General Technical Approach:**

*You should, in words, describe your approach and design here. You should also summarize any meeting notes, brainstorming sessions, etc. that you want to retain through the design of your project.*

Due to a lack of resources (primarily workers), the overall approach is to use Springboot and re-used code from individual assignments to create a scheduling website. Group members used Discord server, GitHub Repository, and GitHub Desktop to share and modify code and hold discussions/meeting plans.

**Key Technical Design Decisions:**

*Any final technical design decisions, such as framework decisions, should be documented here. This should list the technology/framework, its purpose in the design, and why it was chosen.*

Firstly, the technology and framework chosen is Spring Boot with Maven, due to the project member’s unfamiliarity with application and website design. It should also be known that, with cross-site scripting, SQL injections, and data breaches being a possibility, various security measures will be hardwired in, including anti-injection measures and passwords being stored as hashes.

**Known Issues:**

*Any anomalies or known issues in the code or functionality should be documented here.*

A

**Risks:**

*Any risks, unknowns, or general project elements that need to be tracked for risk management should be documented here.*

One of the primary risks is the possibility of XSS and data breaches. Thus passwords should be stored as hashes, and access controls should be enforced.

**ER Diagram:**

*Image file of your ER database diagram.*

Login Data = [Username, Password]; Register Data = [FirstName, LastName, Email, Address, PhoneNumber, Username, Password]; Schedule Data = [EventName, Year, Month, Day, Hour, Minute].

**DDL Scripts:**

*This should contain a link to Bitbucket wherefrom the DDL script can be downloaded.*

Not currently applicable

**Sitemap Diagram:**

*Image file of your Sitemap diagram.*

*Home 🡪 [Login OR Register] 🡪 Schedule*

**User Interface Diagrams:**

*You should insert any wireframe drawings or whiteboard concepts that were developed to support your application. If you have no supporting documentation, please explain the rationale for why you are able to leave this section as N/A.*

*Overall, the design is fairly basic, and was completed before any concept art could be created.*

**Class Diagrams:**

*You should insert any class diagrams here. Your class diagrams should be drawn correctly with the 3 appropriate class compartments, + and – minus to indicate accessibility, and also the data types for the state/properties as well as method arguments and return types. If you have no supporting documentation, please explain the rationale for why you are able to leave this section as N/A.*

+HomeController

+dispHome

+shows Scheduler home “/scheduler/”

+dispLog

+shows /scheduler/login/

+dispReg

+shows /scheduler/register/

+doLogin

+shows /scheduler/events/

+LoginController

+display

+shows /login/

+doLogin

+shows /login/dologin/

+RegisterController

+display

+shows /register/

+doLogin

+if error shows /login/

+else shows /events/

+EventModel

+EventModel

+sets EventName, Year, Month, Day

+EventModel

+sets EventName, Year, Month, Day, Hour, Minute

+getter/setter [EventName, EventDate, Date, Year, Month, Day, Hour, Minute]

+sets and gets specified objects

+LoginModel

+getter/setter [username, password]

+sets and gets specified objects

+RegisterModel

+getter/setter [FirstName, LastName, Email, Address, PhoneNumber, Username, Password]

+gets and sets specified objects

+ScheduleRestService

+currently unused, but is set up to show events as JSON *or* XML file

+ScheduleService

+create List of EventModel titled *eventList*

+gets eventList

+addEvent

+adds an event with name, year, month, and day

+init

+flavor text to console

+creates default events

+destroy

+currently unused, but base for future expansion

+ScheduleServiceInterface

+getEvents

+dependency injection

+init

+dependency injection

+destroy

+dependency injection

**Service API Design:**

*This section should fully document any service API’s (like REST API’s) that are being published, how to access the service, what parameters are required by the API, and the detailed JSON data format specification that could be used by a third party developer to integrate with the service and API. The design can also be captured with tools such as Swagger.*

A

**Security Design:**

*This section should outline the design for how authentication and authorization was supported. This section should also contain all of the roles and privileges that are supported by the design.*

A

**Other Documentation:**

*You should insert any additional drawings, storyboards, whiteboard pictures, project schedules, tasks lists, etc. that support your approach, design, and project. If you have no supporting documentation, please explain the rationale for why you are able to leave this section as N/A.*

A