**CST-339 Programming in Java III**

**Project Status and Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | *4* | |
| **Date:** | ***October 30, 2022*** | |
| **Revision:** | *3.0* | |
| **Team:** | 1. *Matthew R McCormack* | |
| 1. *David “Nate” Cole* | |
|  | |
|  | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | | *Prepare MySql-ddl.sql info* | *Matthew McCormack* | *1* |  | | *Update/system to utilize SQL* | *Nate* | *3* |  | | *Modify EventModel to work with SQL* | *Nate* | *0.5* |  | | *Update design report* | *Matthew McCormack* | *0.25* |  | |  |  |  |  | | *Implemented MongoDB compatibility, and update, create, and delete methods.* | *Matthew McCormack* | *3* |  | | *Implemented Security update* |  | *1.5-2* |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| **GIT URL:** | *https://github.com/ColeDNate/CST-339.git \*please note that ‘Branch A’ is the current working branch, as opposed to the ‘main’ branch.* | |
| **Screencast URL:** | *\*Please note that current issues are persisting with screen recording, leading to a zip of code screenshots being provided in the GIT under the branch TopicScreenshots. - Matthew* | |
| **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.*

No planning performed. Built piecewise based on the topic assignment

**Retrospective Results:**

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

|  |
| --- |
| **What Went Well** |
|  |
|  |
|  |

*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** |
| Due to the fact that the scheduler is designed to only produce a specific, whole data set, returning \*all products\* or \*specific product\* is not truly applicable. | Apologize for any inconvenience. | **Oct. 30, 22** |
|  |  |  |
|  |  |  |

**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.*

AddEvent section does not produce desired results;

Because the program automatically displays the schedule, it was decided to focus on the update and delete methods;

The nature of the application precludes getting *everything* or getting a *part of something*.

**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. Also, Spring Boot should be monitored for any significant, disruptive updates.

**ER Diagram:**

*Image file of your ER database diagram.*

A

**DDL Scripts:**

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

A

**Sitemap Diagram:**

*Image file of your Sitemap diagram.*

*Home à [Login OR Register] à Schedule 🡪 Add Event 🡪 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.*

*A*

**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.*

A

**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.*

**CLCApplication**

Spring Boot App.

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **main** | /com.gcu.CLCApplication |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| main | SpringApplication.run | void |

**CLCConfig**

Schedule Service **Bean**

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **@Bean getSchedule** | /com.gcu.CLCConfig |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| ScheduleServiceInterface | return | new |

**EventRepository**

Interface that extends MongoRepository<EventEntity, String>

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **insert** | /com.gcu.EventRepository |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| void | String (x3), int (x5) | string, id, name, year, month, day, hour, minute |

**SecurityConfig**

Enables security

//Autowired (PasswordEncoder, passEnc) (UserBusinessService, uBServe)

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **Bean bpassEnc** | /com.gcu.SecurityConfig |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| BCryptPasswordEncoder | Return new | BCryptPasswordEncoder() |

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **Autowired configure** | /com.gcu.SecurityConfig |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| AuthenticationManagerBuilder auth | Auth.  -.UserDetailsService  -.passwordEncoder | -uBServe  -passEnc |

**info**

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | **void** |
| 400 | Exception |

**HomeController**

Primary Controller, /scheduler mapping; (ScheduleServiceInterface service)

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **displayHome (Model model)** | /com.gcu.controller (/scheduler/home/) |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| String | Model.addAttribute | (“Title”, “home”) |

**info**

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | **home** |

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **displayRegister (Model model)** | /com.gcu.controller (/scheduler/register/) |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| String | -Model.addAttribute  -model.addAttribute | -(“Title”, “home”)  -(“loginModel”, new LoginModel()) |

**info**

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | **login** |

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **doLogin (Loginmodel loginmodel, BindingResult bindingResult, Model model)** | /com.gcu.controller (/scheduler/events/) |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| String | -Model.addAttribute  -model.addAttribute | -(“Title”, “Login”)  -(“loginModel”, new LoginModel()) |

**info**

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | **login** |

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **displayLogin** | /com.gcu.controller (/scheduler/events/) |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| String | -Model.addAttribute  -model.addAttribute | -(“MyEvents”, “EventModel”)  -(“events”, service.getEvents()) |

**info**

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | **events** |
| 400 | **login** |

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **addEvent (EventModel eventModel, BindingResult bindingResult, Model model)** | /com.gcu.controller (/scheduler/addEvent) |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| String | -Model.addAttribute  -model.addAttribute | -(“SomeEvent”, “EventModel”)  -(“newEvent”, service.addEvent()) |

**info**

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | **addEvent** |
| 400 | **addEvent** |

**LoginController**

Login controller,

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **X** | / |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| T1  T2 | P1  P2 | Type  Type |

**info**

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | **What is returned**  Example response:-  EXAMPLE |
| 400 | Error1 |

**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 **Section**

Description (this highlights the general layout)

## Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| **X** | / |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| T1  T2 | P1  P2 | Type  Type |

**info**

## Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | **What is returned**  Example response:-  EXAMPLE |
| 400 | Error1 |
| 400 | Error2 |
| 401 | Error3 |
| 500 | Error4 |

**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