**CST-361 - Design Report Template**

|  |  |  |
| --- | --- | --- |
| **Topic:** | *Milestone 7* | |
| **Date:** | *February 2019* | |
| **Revision:** | *7.0* | |
| **Team:** | 1. *Samantha Krall* | |
|  | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | | *Clean and finalize design report* | *Sam* |  |  | | *Refactor and clean code based on instructor feedback* | *Sam* |  |  | | *Project Presentation-*  *Mimic technical design review*  *Present a live demo* | *Sam* |  |  | | |
| **GIT URL:** | [*https://github.com/SamanthaKrall/CST-361*](https://github.com/SamanthaKrall/CST-361) | |
| **Peer Review:** | *Y* | We acknowledge that our team has reviewed this report and we agree to the approach we are all taking. |

**Planning Documentation**

**Agile Scrum Product Backlog:**

<https://github.com/SamanthaKrall/CST-361>

**Agile Scrum Sprint Backlog:**

<https://github.com/SamanthaKrall/CST-361>

**Agile Scrum Burn Down Chart:**

<https://github.com/SamanthaKrall/CST-361>

**Agile Retrospective Results:**

*The following table should be completed after each Retrospective on Things That Went Well (Keep Doing). An alternative to the following table is to use a Mind Mapping tool such as Coggle. If you use a Mind Mapping tool you must include a URL or Image File.*

|  |
| --- |
| **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). An alternative to the following table is to use a Mind Mapping tool such as Coggle. If you use a Mind Mapping tool, you must include a URL or Image File.*

|  |  |  |
| --- | --- | --- |
| **What Did Not Go Well** | **Action Plan** | **Due Date** |
|  |  |  |
|  |  |  |
|  |  |  |

**Design Documentation**

**Install Instructions:**

*This section should include step-by-step instructions for setting up your database, 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 BitBucket and deploy the application to an externally hosted site.*

*Using Red Hat Developer Studio, MAMP, and MySQL for the project.*

**General Technical Approach:**

*You should, in your own 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.*

**Key Technical Design Decisions:**

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

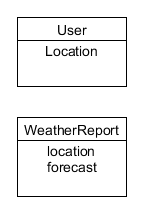
**Known Issues:**

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

**Risks:**

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

**ER Diagram:**



**DDL Scripts:**

[*https://github.com/SamanthaKrall/CST-361*](https://github.com/SamanthaKrall/CST-361)

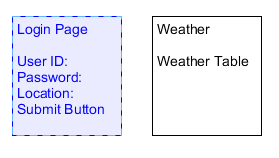
**Flow Charts:**

*You should insert any flowcharts here. Flowcharts should document algorithms or workflow that will be implemented in your program. At a minimum, this should contain a flowchart of the Minesweeper game logic.*

**Sitemap Diagram:**

*Image file of your Sitemap diagram.*

**User Interface Diagrams:**



**Class Diagrams:**

|  |
| --- |
| User |
| +name:String  +location: String  +password:String |
| +User()  +getPassword():String  +setPassword(String)  +getName():String  +setName(String)  +getLocation():String  +setLocation(String)  +User(String, String, String) |

|  |
| --- |
| Weather |
| +day:String  +high:int  +low:int  +forecast:String |
| +Weather()  +getDay()String  +setDay(String)  +getHigh():int  +setHigh(int)  +getLow():int  +setLow(int)  +getForecast():String  +setForecast(String)  +Weather(String, int, int, String) |

|  |
| --- |
| Weathers |
| +theList:ArrayList<Weather> |
| +add(Weather)  +printAll() |

|  |
| --- |
| BusinessService |
| +db:DabaseInterface |
| +deleteOne(String):int  +insertOne(Weather):int  +readAll():ArrayList<Weather>  +updateOne(String, Weather):int |

|  |
| --- |
| BusinessServiceInterface |
|  |
| +deleteOne(String):int  +InsertOne(Weather):int  +readAll():ArrayList<Weather>  +updateOne(String, Weather):int |

|  |
| --- |
| WeatherRestService |
| +bs:BusinessServiceInterface |
| +getAllAsJson():List<Weather>  +getAllAsXml():Weather[] |

|  |
| --- |
| DatabaseInterface |
|  |
| +deleteOne(String):int  +InsertOne(Weather):int  +readAll():ArrayList<Weather>  +updateOne(String, Weather):int |

|  |
| --- |
| FormController |
|  |
| +onSubmitEdit():String  +FormController()  +fetchAll():ArrayList<Weather> |

|  |
| --- |
| DatabaseService |
| +dbURL:String  +user:String  +password:String  +c:Connection  +pstmt:PreparedStatement  +stmt:Statement  +rs:ResultSet  +rowsAffected:int  +numberOfRowsAffected:int |
| +deleteOne(String):int  +InsertOne(Weather):int  +readAll():ArrayList<Weather>  +updateOne(String, Weather):int |

|  |
| --- |
| UserDatabaseService |
| +dbURL:String  +user:String  +password:String  +c:Connection  +pstmt:PreparedStatement  +stmt:Statement  +rs:ResultSet  +rowsAffected:int  +numberOfRowsAffected:int |
| +deleteOne(String):int  +insertOne(User):int  +ReadAll():ArrayList<User>  +updateOne(String, User):int |

**Service API Design:**

*This section should fully document any Third Party Service Interface APIs being consumed or application specific Service APIs 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.*

**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.*

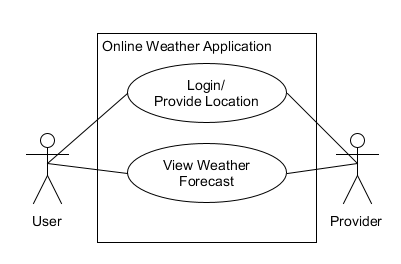
**Pseudo Code:**

[*https://github.com/SamanthaKrall/CST-361*](https://github.com/SamanthaKrall/CST-361)

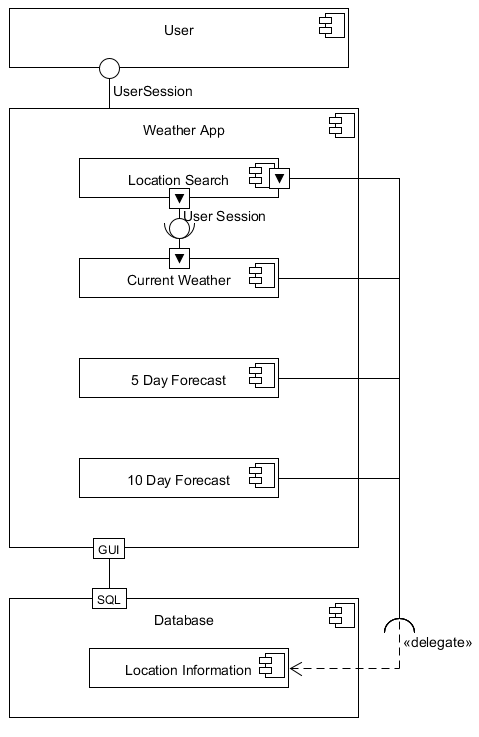
**Other Documentation:**

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

*Use case diagram:*



*Component diagram:*



*Deployment diagram:*

