**Java III Application Programming**

**Project Status and Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | Milestone 7 | |
| **Date:** | 12/4/17 | |
| **Revision:** | 1.7 | |
| **Team:** | 1. Brendan Brooks | |
| 1. Will Bierer | |
|  | |
|  | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | | *Implemented the User REST API* | *Brendan Brooks* | *4* |  | | *Implemented the Lobbies REST API, Updated documentation* | *Will Bierer* | *5* |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| **GIT URL:** | <https://github.com/wbierer/CST-235-Will_Bierer-Brendan_Brooks> | |
| **Peer Review:** | Y | We acknowledge that our team has reviewed this Report and we agree to the approach we are all taking. |

**Design Documentation**

**General Technical Approach:**

* **A Checkers Web Application**
  + Have a login/registration system
  + Main page will be list of available games to join, friends displayed first.
  + Gameplay will be turn based with two players
    - Gameplay data will be stored in database
      * Will use gameboard objects that contain information on location and if there is a checker on it
    - Ajax calls every couple of seconds will keep everything *almost* real-time
    - Logic will have to account for out of bounds exceptions and ensure that only legal moves are allowed
      * Includes having a function that will calculate if a user’s move is legal or illegal (checkers can only go diagonal, not forward)
  + Number of wins will be a statistic, stored in the database as part of the user table
  + A professional looking UI
    - Bootstrap CSS as a foundation
    - Custom CSS layered on top of Bootstrap
  + Users can add friends so they can quickly find games with their friends
* **Backup plan: API Statistic tracker for a video game**
  + In case the other idea won’t work out, our backup idea is to have an application that takes data from an API for a video game
    - The reason is that there is a video game (PlayerUnknown’s Battlegrounds) which actually wipes stats from their databases every few months, making it impossible to have a lifetime track record. Our plan is to prevent that problem with this.
  + This will include login/registration, as well as inclusion of the user’s Steam ID (for use with the API)
  + Users will have the ability to create password-protected groups that friends can join and see total stats for the whole group
    - Groups will have message boards, most likely run with Ajax
  + Possible inclusion of Google’s Analytics API to provide visuals
  + A professional looking UI
  + Every time a user logs in, their data will be stored in the database
    - We may want to look into timed events for updating the data (i.e. every 5 min) or having a separate server (Personal server, probably using AWS) constantly checking for new stats. This *would* also require having a constantly running database over a locally hosted one.

**Key Technical Design Decisions:**

* Ajax-based gameboard with partial page refreshes every couple seconds
  + Will keep gameplay near real-time for the players
* Multiplayer gameplay
  + Creating some kind of artificial intelligence to make this 1-player would have been harder and less interesting, so we decided to take on the idea of 2-player-only gameplay
* Two stylesheets
  + Foundation: Bootstrap (bootstrap.css)
    - This is so that we will have a strong mobile-friendly UI almost out of the box, as well as not needing to do extra simple CSS legwork since it is already done with Bootstrap
  + Layered on top of Bootstrap: Custom CSS sheet (stylesheet.css)
    - This is so that we can have our own customizes look to the site. Bootstrap is great but we don’t want this application to look like every other bootstrap-based web application out there
    - Bootstrap alone will not fit our vision for the look of the site

**Known Issues:**

-jQuery module slide-down-on-hover functionality can be clunky at times.

-Deleting lobbies has some functionality issues.

**Risks:**

We won’t be learning ajax and partial page refreshes as part of the course, so we will have to figure it out on our own. Also, because we will have to figure out game-logic while also learning a difficult framework, it may become more time-consuming than we expect.

**Service API Design:**

**Rest API’s**

Path = “/rest”

Description: Use the /rest URI to call all other rest based URI’s

**LobbyRestService**

Path = “/rest/lobby”

Description: Gets data from the list of lobbies from the LobbyService and displays the data in JSON format

GetAllLobbies

Path = “/rest/lobby/getlobby

Description: Returns data from all lobbies in JSON format

GetLobbyByName

Path = ”/rest/lobby/getlobby/{lobbyname}”

Description: Returns data from one lobby in JSON format based on what the user put as the lobby name in the URI

**UserRestService**

Path = “/rest/user”

Description: Gets data from the list of users from the UserService and displays the requested data in JSON format

GetAllUsers

Path = “/rest/user/getuser

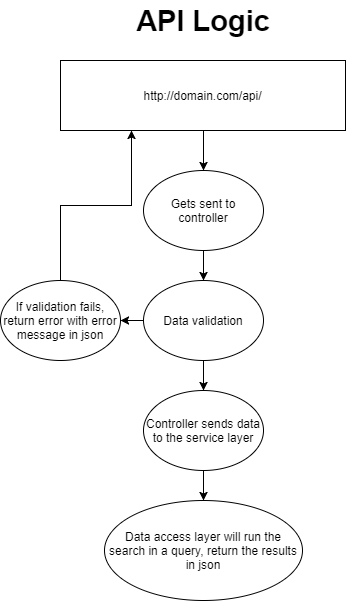
Description: Returns data from all users in JSON format

GetUserByName

Path = “/rest/user/getuser/{firstname}

Description: Returns data from one lobby in JSON format based on what the user put as the firstname in the URI

**REST API Diagrams**

****

*API Example JSON Output*

*/rest/lobby/getlobby/[lobbyname]*

{

"name": "Lobby Name Here",

"name\_description": "The name of the lobby",

"host": "User name here",

"host\_description": "User name of whoever created the lobby",

"players": "1/2",

"players\_description": "The number of players in the lobby out of 2"

}

Description: This API will output information about a lobby called through the search term

*/rest/user/getuser/[firstname]*

{

"username": "testuser123",

"username\_description": "This is the username of the user. This is the only option when retrieving users for sake of privacy"

}

Description: This API will output information about a User called through the search term

*Errors*

If [SearchTerm] in /searchUsers and /searchLobbies does not correlate with a User or Lobby, Then print out an error message saying that the specified user/lobby does not exist.

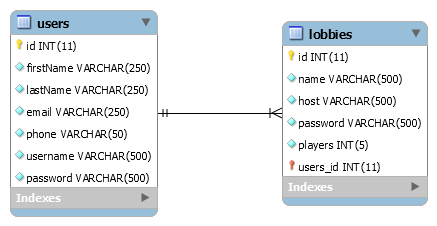
If [SearchTerm] in /searchUsers and /searchLobbies is null than print out a message saying that your search was empty.

If [limit] in /getUsers and /getLobbies is greater than the amount of lobbies/Users, then show the maximum amount of users/lobbies.

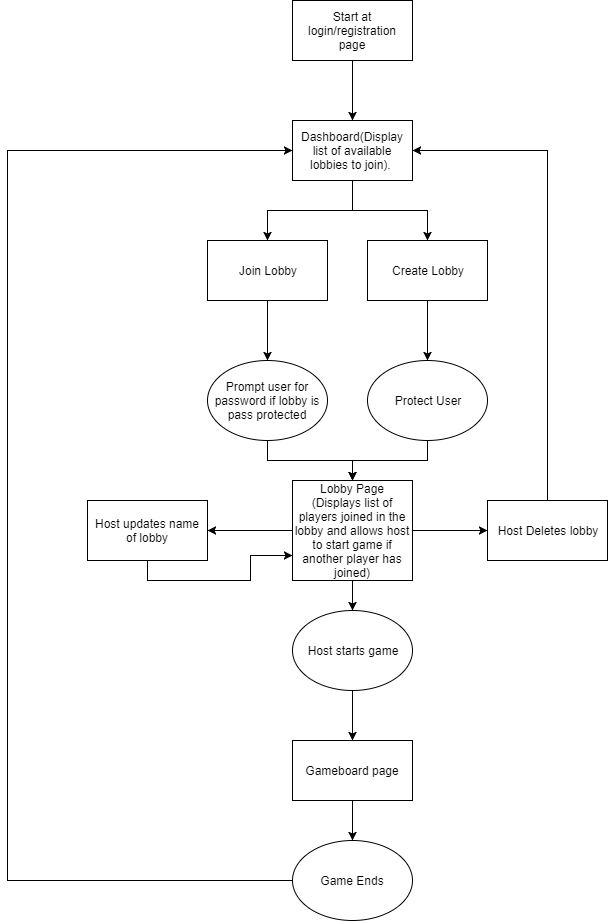
**DDL Scripts:**

**DDL script can be found in the source code and the git repository**

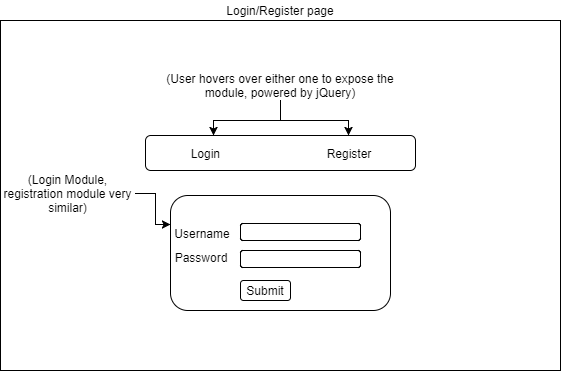
**ER Diagram:**

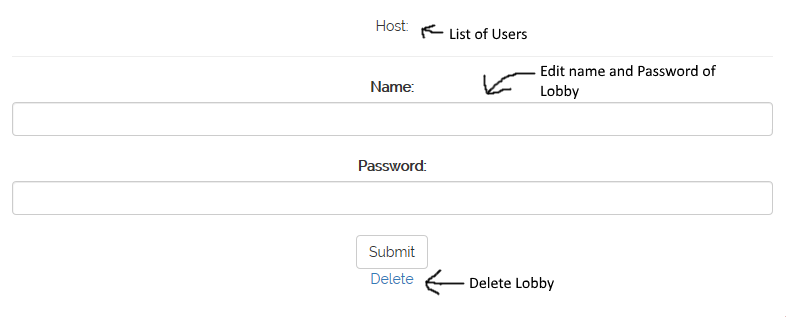


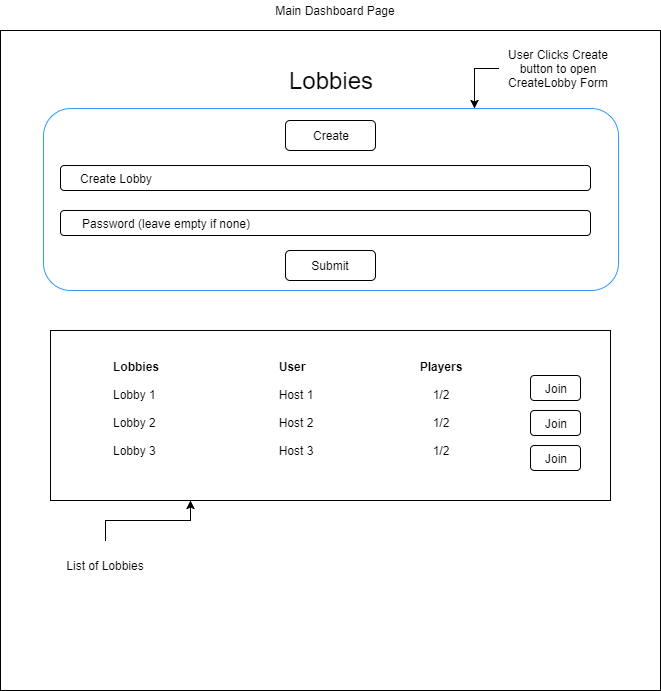
**Sitemap Diagram:**



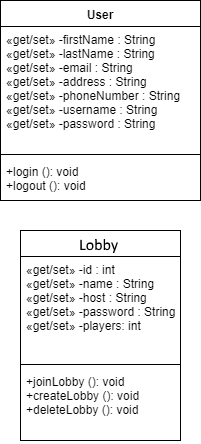
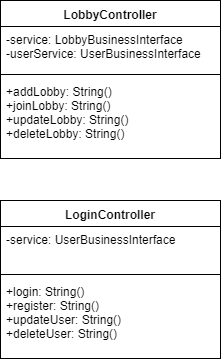
**User Interface Diagrams:**

**

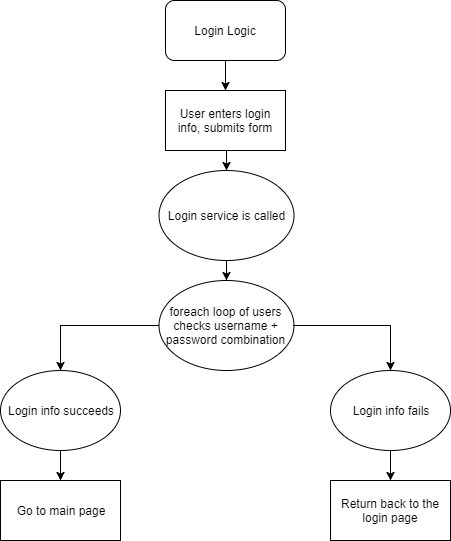


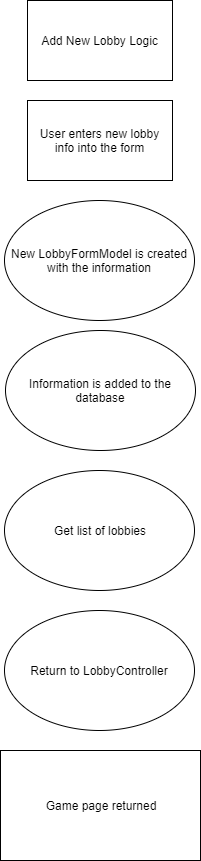


**Class Diagrams:**

**Other Documentation:**

**

**