**Java III Application Programming**

**Project Status and Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | Topic 1 | |
| **Date:** | 09/03/17 | |
| **Revision:** | 1.0 | |
| **Team:** | 1. Brendan Brooks | |
| 1. Will Bierer | |
|  | |
|  | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| **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:**

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

**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.

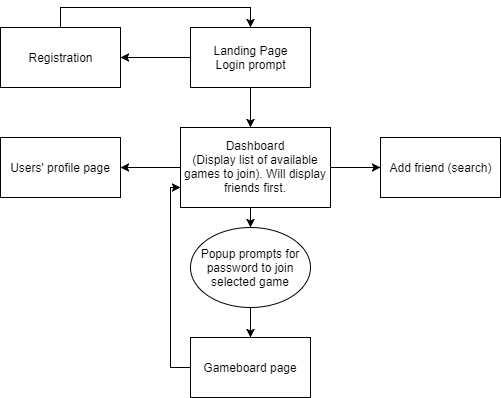
**ER Diagram:**

*Image file of your ER database diagram.*

**DDL Scripts:**

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

**Sitemap Diagram:**

**

**User Interface Diagrams:**

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

**Service API Design:**

*This section should fully document any 3rd Party Service Interface API’s being consumed or application specific Service API’s 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 3rd 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.*

**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 rational why you are able to leave this section as N/A.*