**Task Roulette**

**Detailed Design**

**COP 4331, Spring, 2014**

**Team Name:** MADNESS (aka. Team 14)

**Team Members:**

* Cody McMahon
* Jessica Carter
* Matt McGivney
* Steven Lo
* Gunnar Skotnicki

**Modification history:**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Who | Comment |
| v0.0 | 05/13/13 | S. Applegate | Template |
| v1.0 | 03/09/14 | C. McMahon | Formatting |
| v1.1 | 03/24/14 | G. Skotnicki | Added User and Human Factors to Trace Design and Quality Assurance as well as formatted the trace design |
| v1.2 | 03/24/14 | J. Carter | Trace of Requirements to Design, Design Issues |
| v1.3 | 03/24/14 | M. McGivney | Trace of Requirements to Design |
| v1.4 | 03/28/14 | J. Carter | Trace of Requirements to Design |



**Contents of this Document**

1. **Reference Documents**
2. **Design Issues**
3. **Detailed Design Information**
4. **Trace of Requirements to Design**



**Reference Documents**

* Concept of Operations
* Project Plan
* SRS
* High Level

**Design Issues**

Reusability:

The way we are creating this application with NodeJS, AngularJS, Bootstrap, and MongoDB will make it easy for us to create other applications with this code. TR could easily be transformed into other projects such as a web-based bookkeeping application or a web-based note taking application. Using these technologies also allows for rapid development.

Maintainability:

Separating our application with a MVC (Model, View, and Controller), we can easily maintain the design requirements with the functional requirements. We are using AngularJS for our frontend framework.

Testability:

The independence of our models allows for easy testing. Each module can be extensively tested on its own. Afterwards, components can be combined and tested together until the entire software system is composed and tested as a whole. This method creates easy and thorough testing of the system. Since the scope of the project is so small, we will be testing manually.

Performance:

Performance is not an issue since we are using NodeJS for a web-based application. This framework combined with Bootstrap and AngularJS provides a lightweight system where performance requirements are a minimum.

Portability:

Task Roulette will be able to run on mobile and non-mobile devices since we are using Bootstrap. It configures its display format based on the user’s screen resolution, allowing Task Roulette to display correctly on any screen size. Using Bootstrap overcomes our issue with portability issues.

Safety:

Task Roulette is not responsible for the tasks each user inputs. If a user inputs an illegal task, it is entirely the user’s responsibility.

Connectivity:

The user will need to be connected to the Internet in order to access our application.



**Detailed Design Information**

??

Classes:

1. User
2. Activity
3. Page
4. Login
5. Signup
6. Home
7. List
8. Web Server
9. Database
10. AccountInfo
11. Timer

<In this section, you include diagrams, listings, etc. to provide the complete detail of the design of your product. A class diagram is required. Depending on the tools or development environments you are using, you may be able to automatically generate much or all of this section.>

<NOTE: The design documentation must be well organized, modular, and of sufficient detail so that, using only the concept of operations, SRS, and design documentation (and NOT the code):

* the technical reviewer or software maintainer can fully and easily understand the structure, function, and non-behavioral characteristics of the software
* the coder can develop the final product from the design with NO decisions to make other than syntax
* the maintainer can safely determine how to modify or enhance the code and know what other sections of code may be impacted by the change>



**Trace of Requirements to Design**

*Template:*

|  |
| --- |
| *Location in SRS, Along with Title* |
| ***Statement:*** *From SRS* |
| ***Implementation:*** *Class where this will be implemented (and/or related to)* |

*List:*

|  |
| --- |
| *3.3.1.10 Create Account* |
| **Statement:** The user shall be able to create an account. |
| **Implementation:** Sign Up, AccountInfo, Database, Web Server |

|  |
| --- |
| *3.3.1.11 Login* |
| **Statement:** The user shall be able to login. |
| **Implementation:** Login, WebServer, Database, AccountInfo, User |

|  |
| --- |
| *3.3.1.12 Create a Task* |
| **Statement:** The user shall be able to create a task. |
| **Implementation:** List, AccountInfo, Web Server, Database |

|  |
| --- |
| *3.3.1.13 View Tasks* |
| **Statement:** The user shall be able to view their task lists. |
| **Implementation:** List, AccountInfo |

|  |
| --- |
| *3.3.1.14 Get a Task* |
| **Statement:** The user shall be able to get a task to do. |
| **Implementation:** List, Web Server, Database, AccountInfo, Activity |

|  |
| --- |
| *3.3.1.15 Stop a Task* |
| **Statement:** The user shall be able to stop doing a current task. |
| **Implementation:** List, Web Server, Database, AccountInfo, Activity |

|  |
| --- |
| *3.3.1.16 Logout* |
| **Statement:** The user shall be able to logout. |
| **Implementation:** Page |

|  |
| --- |
| *3.3.2.20 Viewing Task List* |
| **Statement:** The user shall be able to view their task list. |
| **Implementation:** List |

|  |
| --- |
| *3.3.2.21 “Logged In?”* |
| **Statement:** The user shall be able to see that they are logged in. |
| **Implementation:** On this page - Login,  Signup - Cannot be on this page,  Home - Cannot be on this page,  List - Cannot be on this page |

|  |
| --- |
| *3.3.4.40 Every user has the same level of account* |
| **Statement:** There is only one account type. |
| **Implementation:** AccountInfo |

|  |
| --- |
| *3.3.4.41 Cross Platform Application* |
| **Statement:** The application will be able to run across many systems and devices. |
| **Implementation:** Page |

|  |
| --- |
| *3.3.4.42 Misuse via input* |
| **Statement:** The users could attempt to do malicious things through the input. |
| **Implementation:** Page |

|  |
| --- |
| *3.3.4.43 Webpage format* |
| **Statement:** The webpages format and size depends on the screen. |
| **Implementation:** Page |

|  |
| --- |
| *3.3.2.60 Data will be stored on a VPS* |
| **Statement:** Data will be saved on a server. |
| **Implementation:** Web Server |

|  |
| --- |
| *3.3.6.61 Random Number Generation for getting task* |
| **Statement:** A random number generator will be able to grab a task for a user |
| **Implementation:** Web Server, Database, Account Info |

|  |
| --- |
| *3.3.6.62 Users Can Create Tasks* |
| **Statement:** The user will be able to create their own task and those tasks will be saved in the database |
| **Implementation:** Web Server, Database, AccountInfo, Activity |

|  |
| --- |
| *3.3.6.63 Usernames and passwords must match database records* |
| **Statement:** The usernames and passwords will be hashed and compared in the database |
| **Implementation:** Web Server, Database, Login |

|  |
| --- |
| *3.3.6.64 Users can view their tasks* |
| **Statement:** Users can view all of their tasks in a list. |
| **Implementation:** List, Web Server, Database, AccountInfo, Activity |

|  |
| --- |
| *3.3.6.64 Users can view their tasks* |
| **Statement:** Users can view all of their tasks in a list. |
| **Implementation:** List, Web Server, Database, AccountInfo, Activity |

|  |
| --- |
| *3.3.8.80 Store secure credentials* |
| **Statement:** Usernames and hashed passwords shall be stored in the database. |
| **Implementation:** Login, WebServer, Database |

|  |
| --- |
| *3.3.8.81 Credentials are not recoverable* |
| **Statement:** Usernames and passwords are not recoverable. |
| **Implementation:** Login |

|  |
| --- |
| *3.3.8.82 No backup server* |
| **Statement:** There is no backup server. |
| **Implementation:** Server - not taken care of in a class |

|  |
| --- |
| *3.3.8.83 Avoid insecure code* |
| **Statement:** Known insecure functions shall be avoided. |
| **Implementation:** Page |

|  |
| --- |
| *3.3.8.84 Passwords hashed on login* |
| **Statement:** Passwords entered upon login attempt shall be hashed and compared to the existing hashed password in the database. |
| **Implementation:** Login, Web Server |

|  |
| --- |
| *3.3.8.85 Users are unique* |
| **Statement:** Users shall be uniquely identifiable. |
| **Implementation:** Users, Web Server, Login, Database, AccountInfo |

|  |
| --- |
| *3.3.8.86 Isolate user data* |
| **Statement:** One user’s data shall be isolated from others. |
| **Implementation:** Web Server, Database |

|  |
| --- |
| *3.3.8.87 Public-key required for dev access* |
| **Statement:** Public-key cryptography access for the development server. |
| **Implementation:** Server - not taken care of in a class |

|  |
| --- |
| *3.3.9.90 Support for many browsers* |
| **Statement:** Support many browser versions. |
| **Implementation:** Page |

|  |
| --- |
| *3.3.9.91 Access to TaskRoulette* |
| **Statement:** Access from many systems |
| **Implementation:** Page |

|  |
| --- |
| *3.3.9.92 Accurate information accross platforms* |
| **Statement:** The data should be consistent in each system. |
| **Implementation:** Page, Web Server, Database |

|  |
| --- |
| *3.3.9.93 Site Recovery* |
| **Statement:** If the server crashes the site will recover when the server comes back |
| **Implementation:** Server - not taken care of in a class |