Task Roulette

Software Requirements Specification

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	02/18/14	C. McMahon	Change Formatting
v1.1	02/20/14	M. McGivney	Event Table, Data Requirements
v1.2	02/20/14	J. Carter	Functional Requirements, Interface Requirements
v1.3	02/21/14	J. Carter	Documentation Requirements, Interface Requirements, Software to be Produced, Assumptions, Functional Requirements
v1.4	2/24/14	M. McGivney	Event Table, Security Requirements
v1.5	2/25/14	G. Skotnicki	User and Human Factors Requirements, Quality Assurance Requirements, Resource Requirements
v1.6	2/27/14	J. Carter	Introduction, Stakeholders, Functional Requirements, Physical Environment

			Requirements, Documentation Requirements
v1.7	2/27/14	S. Lo	Assumptions, Stakeholders, Data Requirements, Quality Assurance Requirements
v1.8	2/27/14	M. McGivney	Data Requirements, Security Requirements, Use Case Description
v1.9	2/27/14	C. McMahon	Assumptions, Data Requirements, Security Requirements, Stakeholders, Use Case Diagram

Contents of this Document

1. Introduction
    a. Software to be Produced
    b. Reference Documents
    c. Applicable Standards
2. Definition, Acronyms, and Abbreviations
3. Product Overview
    a. Assumptions
    b. Stakeholders
    c. Event Table
    d. Use Case Diagram
    e. Use Case Descriptions
4. Specific Requirements
    a. Functional Requirements
    b. Interface Requirements
    c. Physical Environment Requirements
    d. Users and Human Factors Requirements
    e. Documentation Requirements
    f. Data Requirements
    g. Resource Requirements
    h. Security Requirements
    i. Quality Assurance Requirements
5. Supporting Material

Section 1: Introduction

Software to be Produced:

The goal of our project is to produce a simple to use mobile friendly web application for task management. The user will start off by logging into Task Roulette and adding tasks that they want to get done. Whenever the user has free time, he/she will tell the app how much time they have, and then a task will be given to them. The user then has the option to do the task, or get a new task. The system will potentially (not a required feature) learn which task is best to give at which time from the users interactions with the app.

For reference on System Requirements, Team Organization, etc. look at Concept of Operations, and Project Management Plan.

Reference Documents:

- Concept of Operations
- Project Management Plan
- Software Requirements Specifications
- High Level Design
- OWASP Top 10 (website)

Applicable Standards:

● No: <unique requirement number>
● Statement: <the "shall" statement of the requirement>
● Source: <source of the requirement. (Documentation or technology concerned)>
● Dependency: <list each other requirement on which this requirement depends. (May be "None")>
● Conflicts: <list each other requirements or systems with which this requirement conflicts. (May be "None")>
● Supporting Materials: <list any supporting diagrams, lists, memos, etc.>
● Evaluation Method: <How can you tell if the completed system satisfies this requirement? >
● Revision History: <who, when, what>

Definitions, Acronyms, and Abbreviations:

- DB: Database
- HTML: HyperText Markup Language
- VPS: Virtual Private Server
- JS: JavaScript
- CSS: Cascading Style Sheet
- TR: Task Roulette
- COTS: Commercial Off The Shelf (Software)
- SQL: Structured Query Language
- OWASP: Open Web Application Security Project

Section 2: Product Overview

Assumptions:

- The user has access to a current web browser that can load Bootstrap. (Bootstrap does not work with Firefox on Android phones. Also, Opera is not supported on iOS or Android).
- The application works on the assumption that the user's device, being mobile or desktop, will have appropriate power/battery life to access our site.
- The user will have to have a stable internet connection in order to connect to the website. (This service could be provided from the user's ISP or the user's cellular provider)
- The VPS, hosted by DigitalOcean, will be running consistently and reliably up.

Stakeholders**:**

- **Users**: The users want to use TR to add and receive tasks to complete.
- **Developers:** TR must first be developed in order to provide an initial product to the users.
- **IT Support**: TR must be supported once it is released, in order to fix bugs or add new features that the user wants.
- **Supporters**: TR must be funded by either those working on it or outside benefactors.
- **Customers**: The customers for our project are the professors/Teachers' Assistants and students that we're trying to sell TR to.
- **Lawyers**: The lawyers are concerned with cover any liabilities in the event of a lawsuit against the TR team.

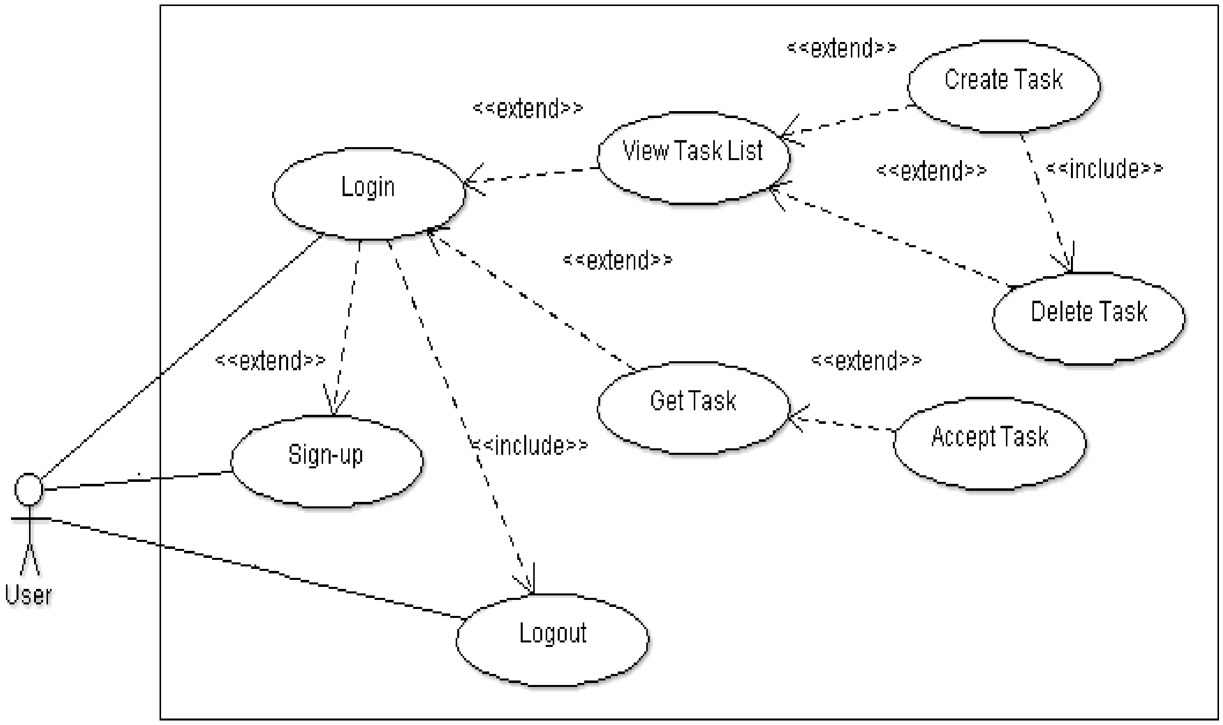
Event Table:

Event Name	**External Stimuli**	**External Responses**	**Internal data and state**	**Use Case Description**
Login	User presses “Login”	Request username and password, notify user of successful or unsuccessful login	Data: Frontend layout, User login credentials State: Login page Next State: Sign-up page or Main page	User enters their personal login information.
Sign-up	User presses "Sign-up"	The account is either created successfully or fails if the email or username is in use.	Data: Frontend layout, User login credentials State: Sign-up page Next State: Login page	User creates an account with their email, username, and password preference.
Main page	User logs in successfully	Displays a menu for navigating the application. Buttons “Get Task”, “View Tasks”, and “Add Task” are presented to the user.	Data: Frontend layout State: Main page Next State: Get task page, View tasks page, or Add task page	The user enters the main page where they have many different options on what to do.
Add Task	User presses “Add Task”	The user may create another task afterwards, or return to the main page.	Data: Frontend layout, New task (inserted into task table) State: Add task page Next State: Add task page or Main page	Request task name, time needed and description, notify if creation succeeds.
Get Task	User presses “Get Task”	This function may fail if the time available is less than what any task needs. The system moves to the Assigned Task page.	Data: Frontend layout, Existing tasks from task table State: Get task page	The user waits while the application queries the database for

			Next State: Get task page or Main page	a task from the user's task list and the user's time specification.
Assign Task	User received a task	Presents a button to accept the task and a button to get a new task. The button to get a new task uses the same selection function as the "Get Task" page.	Data: Frontend layout, Existing tasks from task table State: Assigned task page Next State: Assigned task page or Accepted task page	The system will present the user with a task from their list.
Accepted Task	User presses "Accept"	Presents a button for when the task is completed and a button to have a new task assigned instead. The button to get a new task uses the same selection function as the Get Task page. Navigates to Main page on completion.	Data: Frontend layout, Existing tasks from task table State: Accepted task page Next State: Main page	Presents the user with information about their current task after they've accepted said task. When a task is complete, it is removed from the list of tasks for that user.
View Tasks	User presses "View Tasks"	Show tasks the user has created in a list format.	Data: Frontend layout, Existing tasks from task table State: View tasks page Next State: Main page or Delete task page	Shows the user's task list. Allows deletion of tasks.
Delete task	User presses "Delete"	Shows a message displaying that the task was deleted or an error if deletion fails.	Data: Frontend layout, Existing tasks from task table State: Delete task page	Allows the user to delete any task they would like to delete.

			Next State: View tasks page	
Logout	User presses "Log Out"	Shows a message stating that the user has logged out successfully. Returns to login page.	Data: Frontend layout, User credentials State: Logout page Next State: Login page	Logs the user out.

Use Case Diagram:



Section 3: Specific Requirements

3.1 Functional Requirements

No: 10
Statement: The user shall be able to create an account.
Source: Concepts of Operation - Operational Features (1)

Dependency: None
Conflicts: The username or email may already be taken.
Supporting Materials: Concepts of Operation - Operational Scenarios (1st paragraph)
Evaluation Method: The username will be added to the database and the user will be forwarded to the main page.
Revision History: Jessica Carter, 2/20/14, Initial Jessica Carter, 2/27/14, Revise

No: 11
Statement: The user shall be able to login.
Source: Concepts of Operation - Operational Features (1)
Dependency: Functional Requirement No: 10 - The user shall be able to create an account.
Conflicts: We do not have a recovery option for login credentials.
Supporting Materials: Concepts of Operation - Operational Scenarios (2nd paragraph)
Evaluation Method: The user will be sent to the main page.
Revision History: Jessica Carter, 2/20/14, Initial Jessica Carter, 2/27/14, Revise

No: 12
Statement: The user shall be able to create a task.
Source: Concepts of Operation - Operational Features (2)
Dependency: Functional Requirement No: 11 - The user shall be able to login. Data Requirement No: 62
Conflicts: The task has already been added.
Supporting Materials: Concepts of Operation - Operational Scenarios (3rd paragraph)

Evaluation Method: The user will see their task in the task list and will show up in database.
Revision History: Jessica Carter, 2/20/14, Initial Jessica Carter, 2/27/14, Revise

No: 13
Statement: The user shall be able to view their task list.
Source: Concepts of Operation - Operational Features (3)
Dependency: Functional Requirement No: 12 - The user shall be able to create a task. Data Requirement No: 64 - Users should be able to view a list of their own tasks.
Conflicts: None
Supporting Materials: None
Evaluation Method: The user will see their list of tasks.
Revision History: Jessica Carter, 2/27/14, Initial

No: 14
Statement: The user shall be able to get a task to do.
Source: Concepts of Operation - Operational Features (4)
Dependency: Functional Requirement No: 12 - The user shall be able to create a task. Data Requirement No: 61 - A pseudorandom number generator will be used for selecting tasks from tasks created by the user.
Conflicts: The user does not have any tasks on their task list.
Supporting Materials: Concepts of Operation - Operational Scenarios (4th paragraph)
Evaluation Method: The user will be presented with a random task.
Revision History: Jessica Carter, 2/20/14, Initial Jessica Carter, 2/27/14, Revise

No: 15
Statement: The user shall be able to stop doing a current task.
Source: Concepts of Operation - Operational Features (5)
Dependency: Functional Requirement No 14 - The user shall be able to get a task to do.
Conflicts: None
Supporting Materials: Concepts of Operation - Operational Scenarios (4th paragraph)
Evaluation Method: The user will be asked to choose a new task or go back to the main page.
Revision History: Jessica Carter, 2/20/14, Initial Jessica Carter, 2/27/14, Revise

No: 16
Statement: The user shall be able to get a new task after starting another task.
Source: Concepts of Operation - Operational Features (5)
Dependency: Functional Requirement No: 15 - The user shall be able to stop doing a current task.
Conflicts: There are no more tasks left.
Supporting Materials: Concepts of Operation - Operational Scenarios (4th & 5th paragraph)
Evaluation Method: The user will be given a new task.
Revision History: Jessica Carter, 2/20/14, Initial Jessica Carter, 2/27/14, Revise

No: 17
Statement: The user shall be able to logout.
Source: Concepts of Operation - Operational Features (6)

Dependency: Functional Requirement No: 11 - The user shall be able to login.
Conflicts: None
Supporting Materials: None
Evaluation Method: The user will be brought back to the login screen.
Revision History: Jessica Carter, 2/27/14, Initial

3.2 Interface Requirements

No: 20
Statement: The user shall be able to view their task list.
Source: Concept of Operations - Operational Features (4)
Dependency: Requirement No: 10 - The user shall be able to create an account and login. Requirement No: 11 - The user shall be able to add a task.
Conflicts: The user has no tasks.
Supporting Materials: None
Evaluation Method: The user will see their tasks left to do on the View tasks page.
Revision History: Jessica Carter, 2/20/14, Initial

No: 21
Statement: The user shall be able to see that they are logged in.
Source: Concept of Operations - Operational Features (4)
Dependency: Requirement No: 10 - The user shall be able to create an account and login.
Conflicts: None
Supporting Materials: None
Evaluation Method: If the user is logged in, they will see their username. Otherwise, they will be at the login screen.

Revision History: Jessica Carter, 2/21/14, Initial

No: 22
Statement: The user shall be able to see that they are not logged in.
Source: Concept of Operations - Operational Features (4)
Dependency: Requirement No: 10 - The user shall be able to create an account and login.
Conflicts: None
Supporting Materials: None
Evaluation Method: The user will see the login page
Revision History: Jessica Carter, 2/27/14, Initial

3.3 Physical Environment Requirements

No: 30
Statement: User shall have browser installed on personal device.
Source: Concepts of Operation - Users and Modes of Operation
Dependency: Browsers (Section 1: Assumptions)
Conflicts: None
Supporting Materials: Concepts of Operation - Implementation (1st paragraph)
Evaluation Method: The user will see the application in their browser.
Revision History: Jessica Carter, 2/27/14, Initial

3.4 User and Human Factors Requirements

No: 40
Statement: Every user is the same level of account

Source: Concepts of Operations - Users and Modes of Operation
Dependency: Functional Requirements No:10 - The user shall be able to create an account
Conflicts: None
Supporting Materials: None
Evaluation Method: No user has more features or less features on the site.
Revision History: Gunnar Skotnicki, 2/27/14, Initial

No: 41
Statement: Should be able to run cross platform (mobile and non-mobile)
Source: Concepts of Operations - Users and Modes of Operation
Dependency: Internet browser (Section 1: Assumptions)
Conflicts: None
Supporting Materials: None
Evaluation Method: Testing across platforms
Revision History: Gunnar Skotnicki, 2/27/14, Initial

No: 42
Statement: User could attempt to misuse input
Source: Concepts of Operations - Users and Modes of Operation
Dependency: User input
Conflicts: None
Supporting Materials: OWASP Top 10
Evaluation Method: Cross site scripting and SQL injection does not work
Revision History: Gunnar Skotnicki, 2/27/14, Initial

No: 43
Statement: Webpage format and size depend on the size of the screen.
Source: Bootstrap
Dependency: Bootstrap
Conflicts: Non-supported browsers
Supporting Materials: None
Evaluation Method: The user will see the webpage re-sized depending on their screen size.
Revision History: Jessica Carter, 2/27/14, Initial

3.5 Documentation Requirements

Not applicable due to intuitive site design

3.6 Data Requirements

No: 60
Statement: Data will be stored on a VPS provided by DigitalOcean.
Source: DigitalOcean
Dependency: Digital Ocean, MongoDB
Conflicts: None
Supporting Materials: None
Evaluation Method: We will extensively test between the server and front-end of the project to make sure data is correct.
Revision History: Steven Lo, 2/26/14, Initial Steven Lo, 2/27/14, Revision

No: 61
Statement: A pseudorandom number generator will be used for selecting tasks from tasks created by the user.
Source: - Operational Features/Scenarios (4)
Dependency: Functional Requirement - No: 11, 12, 13, 14, 15, 16. Data Requirement Documentation Requirements - No: 62 - Users should be able to create their own tasks.
Conflicts: None
Supporting Materials: Concept of Operations
Evaluation Method: An user will create and complete many tasks.
Revision History: Steven Lo, 2/26/14, Initial Steven Lo, 2/27/14, Revision Matthew McGivney 2/27/14, Revision

No: 62
Statement: Users should be able to create their own tasks. Created tasks must be retained in the database and linked to the user who created them.
Source: MongoDB, Concept of Operations - Operational Scenarios/Features (2)
Dependency: Functional Requirements - No: 10, 11, 12
Conflicts: Tasks are stored improperly or are not linked correctly.
Supporting Materials: None
Evaluation Method: User will create tasks and see if those tasks are available to them.
Revision History: Steven Lo, 2/26/14, Initial Steven Lo, 2/27/14, Revision

No: 63

Statement: Usernames and passwords must be matched, allowing users to login properly.
Source: MongoDB
Dependency: Security Requirement - No: 80 - Usernames and hashed passwords will be stored in the database.
Conflicts: The username/password is stored to the database incorrectly. There is collision in the hash table for passwords.
Supporting Materials: Concept of Operations - Operational Scenarios/Features (4)
Evaluation Method: Users will create accounts and see if they can login properly once they are created.
Revision History: Steven Lo, 2/26/14, Initial Steven Lo, 2/27/14, Revision Matthew McGivney, 2/27/14, Revision

No: 64
Statement: Users should be able to view a list of their own tasks.
Source: MongoDB
Dependency: Data Requirement - No: 62 -
Conflicts: None
Supporting Materials: Event table - View Tasks
Evaluation Method: User will see tasks are available to them.
Revision History: Matthew McGivney, 2/27/14, Initial

No: 65
Statement: Users should be able to delete their own tasks.
Source: MongoDB
Dependency: Data Requirement - No: 64

Conflicts: None
Supporting Materials: None
Evaluation Method: Users will see an option on their task list to delete unwanted tasks. Deletion is confirmed via a message, and removed tasks will no longer appear in the task list.
Revision History: Matthew McGivney, 2/27/14, Initial

3.7 Resource Requirements

No: 70
Statement: The product will be hosted on a VPS.
Source: DigitalOcean
Dependency: DigitalOcean, MongoDB
Conflicts: Server instability
Supporting Materials: Concept of Operations - Operational Scenarios/Features (4)
Evaluation Method: The users will check the website to see if it loads consistently.
Revision History: Steven Lo, 2/27/14, Initial

No: 71
Statement: We plan on building our site with libraries from AngularJS, Bootstrap, Node.JS, and MongoDB, and using: CSS, HTML, and JavaScript, for our languages. We will build our product through Sublime Text 2.
Source: AngularJS, Bootstrap, Node.JS, MongoDB, CSS, HTML, JavaScript, Sublime Text 2
Dependency: DigitalOcean, MongoDB
Conflicts: Any of the given technologies not working together with each other correctly.
Supporting Materials: Concept of Operations - Operational Scenarios (4)
Evaluation Method: The users will test the implementation of the site once it’s fully created.

Revision History: Steven Lo, 2/27/14, Initial

3.8 Security Requirements

No: 80
Statement: Usernames and hashed passwords shall be stored in the database.
Source: MongoDB
Dependency: Functional Requirement No 10 - The user shall be able to create an account.
Conflicts: None
Supporting Materials: None
Evaluation Method: Passwords will not appear as plaintext in the database, showing that the passwords are hashed.
Revision History: Matthew McGivney, 2/24/14, Initial Matthew McGivney, 2/27/14, Revision

No: 81
Statement: Usernames and passwords are not recoverable.
Source: MongoDB
Dependency: Security Requirement No 80 - Usernames and hashed passwords will be stored in the database.
Conflicts: None
Supporting Materials: None
Evaluation Method: The password hashing function will be one way. There isn’t a button on the site for a lost password.
Revision History: Matthew McGivney, 2/24/14, Initial Matthew McGivney, 2/27/14, Revision

No: 82
Statement: There is no backup server.
Source: DigitalOcean
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: The server is a VPS and we do not have access to another if it fails.
Revision History: Matthew McGivney, 2/24/14, Initial Matthew McGivney, 2/27/14, Revision

No: 83
Statement: Known insecure functions shall be avoided.
Source: Pre-existing libraries
Dependency: Implementation of software system
Conflicts: May have no alternative depending on implementation
Supporting Materials: None
Evaluation Method: Source code will be reviewed for known insecure functions.
Revision History: Matthew McGivney, 2/24/14, Initial Matthew McGivney, 2/27/14, Revision

No: 84
Statement: Passwords entered upon login attempt shall be hashed and compared to the existing hashed password in the database.
Source: MongoDB
Dependency: Security Requirement No:80 - Usernames and hashed passwords shall be stored in the

database.
Conflicts: None
Supporting Materials: None
Evaluation Method: Passwords will not appear as plaintext in the database, showing that the passwords are hashed.
Revision History: Matthew McGivney, 2/27/14, Initial

No: 85
Statement: Users shall be uniquely identifiable
Source: MongoDB
Dependency: Functional Requirement No: 10 - The user shall be able to create an account.
Conflicts: None
Supporting Materials: None
Evaluation Method: Users will be uniquely identified in the database by their username, which is guaranteed to be unique at registration.
Revision History: Matthew McGivney, 2/27/14, Initial

No: 86
Statement: One user's data shall be isolated from others
Source: MongoDB
Dependency: Security Requirement No: 85 - Users shall be uniquely identifiable
Conflicts: None
Supporting Materials: None
Evaluation Method: All user data will be referenced using the unique identifier username, which works to isolate user data

Revision History: Matthew McGivney, 2/27/14, Initial

No: 87
Statement: Public-key cryptography access for the development server
Source: DigitalOcean
Dependency: Development (need access to server)
Conflicts: None
Supporting Materials: None
Evaluation Method: Making sure we can connect to the server with the key and cannot without the key
Revision History: Cody McMahon, 2/27/14, Initial

3.9 Quality Assurance Requirements

No: 90
Statement: The users should be able to access Task Roulette from many browsers.
Source: Concept of Operations - Users and Modes of Operations (3)
Dependency:
Conflicts: The user tries to access TR from an unsupported browser (Firefox on Android and Opera across all platforms).
Supporting Materials: None
Evaluation Method: Users will access TR through all popular browsers for each system.
Revision History: Steven Lo, 2/27/14, Initial

No: 91

Statement: The users should be able to access Task Roulette from many different systems.
Source: Concept of Operations - Users and Modes of Operation
Dependency: Compatible operating system.
Conflicts: None
Supporting Materials: None
Evaluation Method: Testing to be done cross platform
Revision History: Gunnar Skotnicki, 2/27/14, Initial

No: 92
Statement: The information is accurate cross system/browser
Source: Concept of Operations
Dependency: The technologies (Bootstrap has problems with Firefox on Android, etc)
Conflicts: None
Supporting Materials: None
Evaluation Method: Testing to ensure proper data
Revision History: Gunnar Skotnicki, 2/27/14, Initial

No: 93
Statement: Site recovery after a server crash
Source: Concept of Operations
Dependency: VPS and associated technologies
Conflicts: None
Supporting Materials: None
Evaluation Method: Viewing website post server crash.

Revision History: Gunnar Skotnicki, 2/27/14, Initial

Section 4: Supporting Material

- MongoDB Documentation - http://docs.mongodb.org/manual
- AngularJS Documentation - http://docs.angularjs.org/guide
- Bootstrap Documentation - http://getbootstrap.com
- NodeJS Documenation - http://nodejs.org/api