**Task Roulette**

**Concept of Operations**

**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 | 1/23/14 | C. McMahon | Change document style to match Standards and Project Management Doc. |
| v1.1 | 1/31/14 | C. McMahon | The Purposed System: Operational Features, and The Current System |
| v1.2 | 1/31/14 | M.McGivney | Proposed System: Operational Scenarios, and Proposed System: Needs |
| v1.3 | 1/31/14 | G.Skotnicki, S. Lo | Proposed System: Implementation |
| v1.4 | 1/31/14 | J. Carter | Proposed System: Users and Modes of Operation |



**Contents of this Document**

1. **The Current System**
2. **The Proposed System**
   1. **Needs**
   2. **Users and Modes of Operation**
   3. **Operational Scenarios**
   4. **Operational Features**
   5. **Implementation**

**The Current System**

There are many applications that exist which allow people to make task lists. Typically, these are static listings of tasks that the user needs to manage themselves. Recently, these applications have begun to allow users to move the tasks around and set priorities. Current popular applications in this area are Wunderlist, a web and mobile application, and Any.do, a mobile application. Wunderlist and Any.do allow users to access a simple list interface where they can click and drag tasks they have created to a specific day/ordering. In our opinion, these apps don’t really push people to finish tasks that are not mandatory.

Our idea is an application that allows users to create a list of tasks and randomly grab one to be done at a given time. This is more than just an organized list, but basically an assisting ”get-things-done” type of app. The idea is that we want to help users organize the non-mandatory tasks that they want to do, and give them a push that can sometimes be hard to do by yourself.

**The Proposed System: Needs**

As of now, few scheduling applications exist that assign tasks to the user instead of the user creating a schedule themselves. The proposed system would not only allow users to create a lists of tasks that they want to do, but can dynamically assign the created tasks given an amount of free time that a user has. If a user finishes a task with extra time remaining, a new task can be suggested by the system based on the remaining time. Since this system allows for more flexible scheduling, it can increase the productivity and overall satisfaction of the users.

An ideal situation would be that we could make our dynamic tasker learn users’ habits for certain tasks, like estimated completion time vs real completion time. These will be optional features for if we have time, but the core of our app is dynamically assigning non-mandatory tasks based on available time. Our app will have a very friendly one-click interface for receiving tasks so it will be very accessible via mobile and web.

**The Proposed System: Users and Modes of Operation**

Since we are using Bootstrap, all users will be able to access the application on both web and mobile. There will be only one class of users for the application and each account will be accessible from any platform. The application will be free regardless of platform. Both the mobile and web version of app will have the same functionality since it should be minimal across the board: login, ask for task, create task, view existing list.

**The Proposed System: Operational Scenarios**

Users create an account by providing an username, email address, and a password. Passwords are confirmed by means of a second field that must match the first one to continue. If the passwords do not match, the system will alert the user and request for the user to type it again. If an account already exists with a chosen username or email, the system will alert the user and request different credentials.

Users will be able to login to the system. If the username or password is incorrect, the system will alert the user of an incorrect username or password. If a user forgets their password, there is not a recovery method. Given correct login credentials, the user will be able to either create new tasks or accept them from the system.

Users create tasks to be completed at some later time. Tasks are given a unique title. Each task also includes an estimated time of completion given by the user. If there is a task with the title, then we say there is one that already exists and you can modify it in the list view.

When a user has free time, they can login and request a random task from the system. The user provides the system with how much time they have to work and “roll” for a task to complete. Suggested tasks are chosen from the user’s task list with at most that much estimated time. Once a task is accepted, the user notifies the system later whether or not the task was completed as well as the time needed. If completed, it is marked as complete in the database and a new task may be suggested based on any additional time the user has.

It is possible that when a user requests a task, there are none in the database that are short enough to fit the user’s timeframe. If this happens, the system will alert the user that there is not one that fits the time, and ask if the user wants to create a new task. The new task is then assigned to the user.

**The Proposed System: Operational Features**

Must Have:

1. Create Account/ Login
2. Create a Task (We store task on the backend)
3. View Task List
4. Suggest Task based on time
5. New Task upon Completion in Time
6. Logout

Would Like to Have:

1. Create Categories for Tasks
2. Learn tasking based on Categories and Completion Time
3. Ask for real Completion time to extending task for Recurring items

**The Proposed System: Implementation**

Our system will be a Node based Http server on an Ubuntu 13.10 64-bit linux distribution. This will handle all requests coming in from browsers and serve the correct pages back to the user. MongoDB will be how we store all user information and activities. Node will be integrated to work directly with MongoDB. The server programming will be managed in Javascript and all of the front-end code will be done in HTML, CSS, and Javascript with popular libraries: AngularJs, Bootstrap, and jQuery.

Since we plan on using multiple new technologies, there will be a learning phase for all of the team members. We plan on using our first sprint towards learning the languages/technologies that we will be using. This may impact our development as it will likely increase our individual workloads. Our application will be hosted on a website, thus requiring an internet connection. Our system has the benefit of being cross-platform, as well as the benefit from Bootstrap allowing it to be mobile friendly.

Because the application is web-based there is the disadvantage of not having any localized data or usage. The trade off between local-based and the web-based application is that the web-based app can be accessed across many devices (via User Account) however you cannot access unless you have an internet connection. Our goal is to create an app that can be used in any situation a user might be in, but with a desktop or mobile specific app the user would need that required device. With our implementation the user just needs an internet connection as well as a web-browser and their login information.