**Task Roulette**

**Project Management Plan**

**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 | 01/23/14 | C. McMahon | Initial Project overview and Project Team Description |
| v1.1 | 01/30/14 | C. McMahon | Applicable Standards: Coding Conventions and Document Standard |
| v1.2 | 01/30/14 | C. McMahon | Tools and Computing Environment, and Configuration Management |
| v1.3 | 01/30/14 | C. McMahon | Risk Management and Quality Assurance |
| v1.4 | 01/30/14 | J. Carter | Table of Work Packages, Time Estimates, and Assignments |
| v1.5 | 01/30/14 | G. Skotnicki | Artifacts Size Metric Standard, Technical Progress Metrics, Plan for T,C,andP, Risk Management |
| v1.6 | 1/30/14 | S. Lo | Software Life Cycle Process |
| v1.7 | 1/31/14 | J. Carter | Software Life Cycle Process - Diagram |



**Contents of this Document**

1. **Project Overview**
2. **Reference Documents**
3. **Applicable Standards**
4. **Project Team Organization**
5. **Deliverables**
6. **Software Life Cycle Process**
7. **Tools and Computing Environment**
8. **Configuration Management**
9. **Quality Assurance**
10. **Risk Management**
11. **Table of Work Packages, Time Estimates, and Assignments**
12. **Technical Progress Metrics**
13. **Plan for tracking, control, and reporting of progress**

**Project Overview**

As a team, we hope to develop an intuitive task manager. We are going to focus on simplicity and accessibility. Our application will be a Mobile Friendly webapp that allows users to create a list of mandatory/non-mandatory tasks which they wish to do. After creating the list of tasks, the app will provide the users with an interface where they can ask for a random task given a certain amount of available time. If the user, does not wish to do the given task they can “roll” again to find another task that may fit the time they have given.



**Reference Documents**

* Concept of Operations
* SCRUMDesign.pdf



**Applicable Standards**

* Coding Standard :
  + General:
    - Indentation: 4 spaces (do not tab, change settings in environment if necessary)
    - Comment at the top of every file with a summary of what the file accomplishes
    - Comment before every function
    - Underscore for lengthy variable names
    - Related variables should have similar names
    - All blocks need to enclosed with {} including ones that it may not be necessary for like, single line If statements
  + Javascript (JS):
    - ‘use strict’ conventions
    - end every line with a ‘;’
    - lead every variable declaration with ‘var’
    - Callback function and Object declaration should span multiple lines
    - Always put check cases in for ‘err’, the error variable returned.
    - Following CommonJS modular standards, used in NodeJS, of exporting modules and functions to be required by other files
* Document Standard:
  + Font: Calibri, Font Size: 11, Color: Default Black
  + Language: English (American)
  + Headings: Document Title
    - If the Document is to be submitted by the team, use Team Name
    - Otherwise (internal purposes), use original author’s name
  + Table of Contents
    - Single page
    - Bold
    - Number list
    - Separated by line after Modification table
  + Spelling and grammar: Google Docs Standard, Checking System
  + Modification History Tables created after Document Completion (We have Hojoki)
  + All Figures should have titles, Numeric names, and descriptions
    - ex. Figure 1: Prototype Class Diagram for…
  + Spacing: Single Spaced, Space between paragraphs
  + Bold Titles
  + Any modifications made that are left with questions make with ‘??’
    - easily searchable
* Artifact Size Metric Standard:
  + Sprints will be measured by tasks (webpage/functions)
    - Based on 1-2 weeks depending on the task
    - End of sprint will bring source code and documentation for that sprint
  + Google Analytics for web based stats (time taken, etc.)
  + Database: By number of tables
    - User information table
    - Task Table



**Project Team Organization**

We are the awesome group of: Cody McMahon, Jessica Carter, Matt McGivney, Steven Lo, and Gunnar Skotnicki. We plan on meeting in person at least twice a week on Tuesday and Thursday at 5:30pm. This will give us a good 4 hours per week in person to discuss the project work and phase deliverables each week. Every team member is also required to check Hojoki board daily to see what tasks need to be done and log their plan for the day (can be a no-day). We are currently using Google Drive/Docs to manage to manage collaborative work on the documents aside from code files which will be managed on our shared Dropbox folder.

The roles are as follows:

* Project Manager-Cody McMahon
* Lead Designer- Jessica Carter
* Code Developers-Whole Team
* Testers-Gunnar Skotnick , Matt McGivney, Steven Lo
* Librarian-Gunnar Skotnicki
* Progress Analyst-Cody McMahon



**Deliverables**

|  |  |
| --- | --- |
| **Artifact** | **Due Dates**  <some will have multiple deliveries> |
| Meeting Minutes | Daily Meetings |
| Individual Logs | Mid-Sprint |
| Team Reports | Sprint Reviews |
| ConOps | January 31st |
| Project Plan | January 31st |
| SRS | TBD |
| High-Level Design | TBD |
| Detailed Design | TBD |
| Project Management Report | TBD |
| Test Plan | TBD |
| User's Manual | TBD |
| Test Results | TBD |
| Source, Executable, Build Instructions | TBD |
| Project Legacy | TBD |

**Software Life Cycle Process**

We are going to use the SCRUM model, an Agile Method comprised of sprints. We plan on breaking into smaller teams to work on different areas of the project simultaneously. The group will have daily meetings through Hojoki, a webapp, and physical meetings twice a week. At each physical meeting our group will discuss each team's sprint reports and set deadlines for each sprint as well submitting documentation and source code.

We have drawn up an estimated SCRUM diagram: SCRUMDesign.pdf

The first screenshot is of the General Sprints. The second screenshot shows an example of tasks for a given Sprint.

**Tools and Computing Environment**

We are going to be developing and running the application on a Ubuntu 13.10 64-bit VPS hosted by DigitalOcean. Since this will be a web application most of the work will be done HTML, Javascript, and CSS. We plan to include three major front-end libraries: AngularJS (a javascript frameworking library for dynamic HTML), Bootstrap(a CSS and Javascript library used for making webapps mobile friendly), and jQuery(the necessary Javascript library for easy coding).

On the back-end, we will be handling requests and doing server-side scripting using Node.js which allows us to code in Javascript. It runs on the V8 engine made by Google. All of our Users’ information and data will be stored and manipulated with the MongoDB system. Node.js has modules that provide MongoDB connectors. MongoDB uses a JSON like format which is very easy to manipulate in Node.js.

For the design, we will be using ArgoUML. This tool allows use to easily create all of the UML diagrams we will need, Use Case, Activity, Class, etc. As far as coding goes, we will use Sublime Text 2 and upload all of our files to the server using FTP.

**Configuration Management**

For configuration management, we will use Dropbox for all source code. Dropbox allows you to see previous versions of documents and revert them. We have a shared folder that will contain any code needed on the server-end or front-end. Hojoki, a webapp, provides a platform for monitoring when and who monitored what document as well as the ability to make comments on activity. We have the shared Dropbox folder and Google Drive (used for non-code documents) synced. Every person can only the files on Dropbox. Once a code file has been uploaded to the server, write permission will be removed for all users to that file.

We will have a Development database as well as a Production one. Using Cron, we will set up automated backups to be made. Every member of the team will have access to the Development database but only the Project Manager can make changes to the Production database.

**Quality Assurance**

The assigned testers, noted in the Project Team Organization section, will be working manually to discover issues on the website. If later we can find a good automated testing tool for things like Node.JS, then we may end up using that. For the scale of our project it should not be too difficult to simulate our expected user base. Any issue that is found either functionally or visually will be reported as a “task” with the title Bug or Discrepancy on our Hojoki team site. During our Scrum meetings the Project Manager will review the bugs and we will discuss as a team whether they are really an issue. The Project Manager will then assign the bugs via Hojoki to a specific Developer to fix.

After a Bug fix is submitted, we will do Regression tests to make sure nothing else has been affected. The Progress Analyst will create charts and models for the number of bugs that were found as well as analytics on the amount of bugs fixed. These analytics will be reviewed during end of Sprint meetings.

**Risk Management**

The application has multiple levels of risk, ranging from the user level to the system level. The user has a risk of submitting tasks that promote illicit activities so to counteract this situation there will be a disclaimer that states how all tasks are User-Created and the app by default has no tasks to give. As with any web-based application there is a risk of the server crashing or being offline, but this is outside the scope of our mitigation plan. There is risk associated in saving users information in the database in the event of losing data. We plan to counteract this by using Cron to back up our server at given time intervals in order to have a recovery plan in the event of this problem. Since the server will be hosting the main active files, the permissions of the server will only allow team members to upload a new file (as opposed to editing files on the server) since the server has no log of changes. All files will be uploaded to Dropbox first and then to the server, in order to keep team members accountable as well as not break our system. Since user information is so valuable we plan on encrypting their passwords prior to storing their information on our database.



**Table of Work Packages, Time Estimates, and Assignments**

|  |  |  |  |
| --- | --- | --- | --- |
| **Package** | **Responsible Person/s** | **Estimated Completion Time** | **Summary** |
| Documentation for Libraries | J. Carter, C. McMahon | 2 days | node.js, angular.js, Bootstrap |
| Requirements Documentation | Whole team | 2 days | What our project should have |
| Initial UMLs and Wireframes | Whole team | 4 days |  |
| Backend Documentation | C. McMahon | 3 days | What we can do on the backend. |
| Database Models | Whole team | 3 days | MongoDB |
| User Login Page | Whole team | 1 week | Frontend & backend. |
| Add Task Page | Whole team | 2 weeks | Frontend & backend. |
| Get Task Page | Whole team | 2 weeks | Frontend & backend. |
| Testing Documents | M. McGivney, S. Lo, G. Skotnicki | 1 month |  |

**Technical Progress Metrics**

Through the use of the scrum development cycle, our group can use the project’s web pages and backend functions as metrics to see how far along the project is coming. We are currently planning to allot one to two weeks for each sprint, depending on the depth of the particular task. At the end of these sprints we should have the source code and documentation (UMLs) associated with the task. Our current proposed number of users is approximately 100 and we will also be using Google Analytics for the page to keep track of technical metrics (how fast the page is served, etc.). Our backend will be using MongoDB, in which we are planning on using two main table designs, one for Users and another for Tasks.

Our current rough sprint schedule-

- Learning languages and technologies (1 week)

-Login Page (1 week total)

-Page Design

-Functionality

-Get Task Page (2 weeks total)

-Page Design

-askTime() function (1 week)

-giveTask() function (1 week)

-Option Page (2 weeks total)

-Page Design

-skipTask() function (1 week)

-doneTask() function (1 week)

-Add Task Page (2 weeks total)

-Page Design (1 week)

-createTask() function (1 week)

-Testing (4 weeks)



**Plan for tracking, control, and reporting of progress**

At the end of a sprint the group will be collecting the source code and documentation. Collection will all be done digitally through Dropbox/Google Drive, allowing all team members equal access to the files. Each team member will submit the files they are specifically associated with on that sprint. If multiple team members are working on a task they will just submit one copy of the required files. Whenever a team member updates a file, we will be documenting that change in Hojoki’s comment section in order to have an easy access to prior information and version history. Given the style of the scrum development cycle, we will be discussing our needs and next sprint at our physical meetings, analyzing current progress and pending progress. Once the sprint is finished that particular portion of the app should be done, minus testing. When all modules are finished the group will transition into the testing/polishing sprint at the end of this sprint the final reports of the test analysis will be submitted. Aside from the actual deliverables and project’s progress, in addition to the biweekly meetings the group will be in daily contact through Hojoki (Dropbox/Google Drive). Each team member will check Hojoki daily to see what needs to be done, but does not necessarily need to do work that day.