Project Plan

SEG2012GP9

|  |  |
| --- | --- |
| Seabrook, Edward J F | [ejfs1g10@ecs.soton.ac.uk](mailto:ejfs1g10@ecs.soton.ac.uk) |
| Elliott, Kristian | [ke1g10@ ecs.soton.ac.uk](mailto:ke1g10@soton.ac.uk) |
| Yu, Brian | [by2g10@ ecs.soton.ac.uk](mailto:by2g10@soton.ac.uk) |
| Mariani Elola, Oscar N | [onme1g10@ ecs.soton.ac.uk](mailto:onme1g10@soton.ac.uk) |
| Chan, Yun Chak | [ycc1g11@ ecs.soton.ac.uk](mailto:ycc1g11@soton.ac.uk) |

# 0 Meta Pages

## 0.1 Document Control

|  |  |  |
| --- | --- | --- |
| **Version** | **Author** | **Changes** |
| 1 | by2g10 | Initial document |
| 2 | ejfs1g10 | Adjusted style to match ‘House Style’ |
| 3 | ejfs1g10 | Added preliminary Gantt chart |
| 4 | by2g10 | Added version control section |
| 5 | onme1g10 | Added risk analysis section |
| 6 | ycc1g11 | Added time budgeting & recording section |
| 7 | ke1g10 | Added work breakdown table |
| 8 | by2g10 | Rewrote version management section of ProjPlan.docx so it fit better with what is required |

## 0.2 Table of Contents

0 Meta Pages 2

0.1 Document Control 2

0.2 Table of Contents 3

1 Work Breakdown & Allocation 4

1.1 Gantt Chart 4

1.2 Description 5

2 Risk Analysis 6

3 Time Budgeting & Recording 7

3.1 Budgeting 7

3.2 Time recording and management system 7

4 Version management 8

4.1 Code and Document repository 8

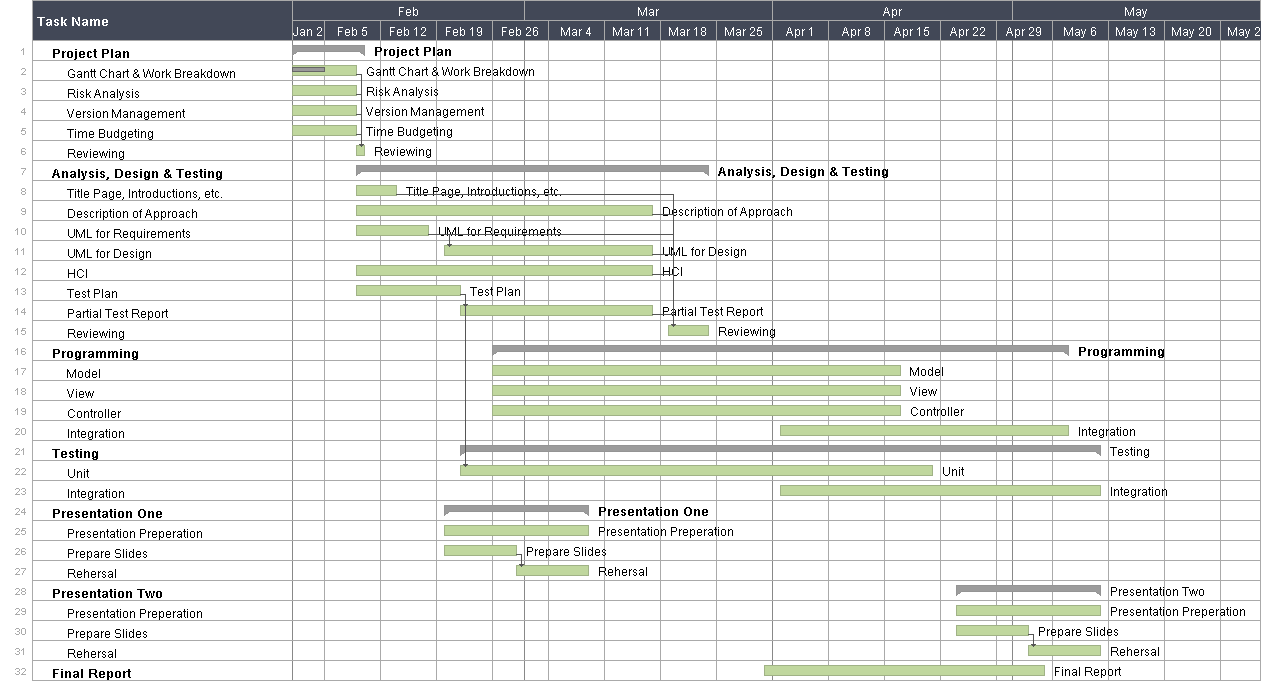
4.2 Programming 8

4.3 Documentation and Project Presentations/Demos 8

4.4 Team meetings 8

# 1 Work Breakdown & Allocation

## 1.1 Gantt Chart



## 1.2 Description

|  |  |  |
| --- | --- | --- |
| **Name** | **Time Estimation** | **Assignment** |
| **Project Plan** | | |
| Gantt Chart and Work Breakdown | 6 hours | Edward, Kristian |
| Risk Analysis | 2 hours | Oscar |
| Budget | 2 hours | Kelvin |
| Project Management | 3 hours | Brian |
| Reviewing | 5 hours | All |
| **Analysis, Design and Testing** | | |
| UML, Use Case diagrams | 2 hours |  |
| Description of Approach | 3 hours |  |
| UML, Class and Sequence | 5 hours |  |
| Wireframes and UI flow diagrams | 5 hours |  |
| Title, Introduction and Revision History | 3 hours |  |
| Test Plan | 5 hours |  |
| Partial Test Report | 10 hours |  |
| Reviewing | 5 hours | All |
| **Programming** | | |
| Model | 40 hours |  |
| View | 60 hours |  |
| Controller | 40 hours |  |
| Integration | 40 hours | All |
| **Testing** | | |
| Unit |  |  |
| Integration |  |  |
| **Presentation (x2)** | | |
| Discussion and Group Preparation | 20 hours | All |
| Prepare Slides | 5 hours | All |
| Rehearsal | 5 hours | All |
| **Final Report** | | |
| Tasks Needed | 50 hours |  |

We have based the main milestones of our project on the dates of the deliverables, planning to ensure than all the tasks are completed by the hand in dates. For many of the tasks we’ve set ourselves earlier deadlines to ensure that dependent tasks are able to go ahead as scheduled. In many cases the dependent tasks do not require the full completion of its dependencies; in these cases the tasks have been allowed to continue after their dependencies start.  
We have allocated team members to the individual tasks and have made predictions of the time the tasks will take, we aimed to stay within the budget of 400 hours with our predictions:

# 2 Risk Analysis

| **Risk** | **Probability** | **Impact** | **Strategy** |
| --- | --- | --- | --- |
| Wrong time estimation for tasks or stages of the project | High | Delays, unfair distribution of tasks, inaccurate budget calculations, project failure (worst case). | Strict tracking of worked hours and tasks, frequent formal meetings addressing the issue and reevaluation of time estimates at later stages of the project. |
| Failure to identify complex functionalities of the system and time required to develop them | Medium to Low | Delays, inaccurate budget calculations, project failure (worst case). | Very strict and thorough analysis and research of the project at the analysis and design stages. Frequent analysis of the status of the project during formal meetings and constant reevaluation and planning in an iterative manner as in the spiral model of software development. |
| Group member does not contribute as expected | High to Medium | Distribution of tasks would have to change, possible delays, lower quality of product in some areas of the project. | Constant tracking of worked hours by each team member. Document tracking them available widely to the whole team and updates on every weekly formal meeting. In this way problems can be identified and dealt with as soon as they appear. |
| Group member get designated a task that does not match his skills | Medium to Low | Lower quality of the final product. More time taken than needed to perform a task. Delays in the project’s deadlines, possible project failure. | Encouraging honesty in formal meetings. Allowing team members to declare their preferences in terms of task allocation. Engaging in healthy criticism during meetings. Constant support amongst team members. |
| Team member absent for a period of time because of illness or dropping out | High | Delays, inconsistency of tasks completion date and expected task completion date as planned. Missing deliverables, bigger workload on other team members. | Design a task and time allocation system that allows extra time to solve this sort of issues. Consider this as a highly probable problem and include it in the project plan to avoid bad consequences. |

# 3 Time Budgeting & Recording

## 3.1 Budgeting

* Basically, for most of the task0 we assign the time proportionally to the ratio in the whole project. For example we will make use of approximately 20 hours on the project plan.
* We plan to have a formal meeting every week for at least an hour up to two hours with our supervisor. Also we may conduct some informal meetings which should last for an hour to review what we have done after last formal meeting and before next formal meeting in order to have a better tracking on our progress.
* For most of the tasks we planned to assign two hours per person to finish each of them. Time allowance will be increased if the tasks are relatively complicated or have the need of integration with others’ work.
* We decided to spend most of our time in programming as presented in our Gantt chart which included the time for testing for each of the component.
* Before each presentation and demonstration we will spend four hours in discussing what content should be included in the presentation, an hour for preparing on slides and another hour for conducting a rehearsal which could help us to have a better performance for both presentations.

## 3.2 Time recording and management system

* Each of us will make use of our own log book. When start working on a task we will record the title of the task, starting time, ending time, actual duration and the expected duration in a form of table to make the record clear.
* We don’t have a specified person to be the time manager but during our each formal meeting we will report and review our own progress and the time spent with the whole team in order to monitor the team progress construct a better plan for the time and tasks allocation for the upcoming tasks.

# 4 Version management

Version management will be done using TortoiseSVN. A message will be included with each Commit which will describe the difference between the new version and the previous one and the reason for this change: this will usually start with a verb and using a bullet list with \* denoting a bullet point e.g.

\*Added new /src directory for storing source code for making things more organised.

\*Fixed outOfBoundsException thrown when using large values of x in Calc.java

## File and directory naming scheme

In the root directory of the SVN will be two directories named /src and /doc. The /src directory stores all files to do with the programming solution including the source code files, graphics for the GUI etc. The /doc directory stores all files to do with the documentation of the project including all documents needed for submission for marking and other relevant files. In addition, the /doc directory will contain several subdirectories named after each submission (e.g. /Project Plan) and will store all relevant files to do with that submission, /Minutes which will store the minutes of each team meeting.

Every document for submission will include a document control page which outlines the revision history of the document including three pieces of information: version number, the author of this revision and a description of the changes the author(s) have made. This will be presented in a table located between the title page and table of contents.

[are we going to have a document producer?]

Version management using SVN will enable unique, traceable versions of all project elements to be stored and retrieved because all versions of the repository is saved as a ‘diff’ of the last version and so retrieving the previous versions is just a matter of locating the revision to be retrieved and calling revert function in TortoiseSVN (the use of commit messages is useful here to help locating the right version to revert to).

Version management using SVN will enable an end-of-project report to report on the amount of change in various project elements – TortoiseSVN can be used to produce reports on the changes made to a file or directory, who made the changes at what time for all files for the project. This powerful feature in TortoiseSVN is used in combination with the tasking and Gantt chart feature of UgForge to enable different ways to show what the changes (to each file or directory(s)) were and what the cost are for each task in terms of who spent how much time on it.