Project Plan
SEG2012GP9

Seabrook, Edward J F	ejfs1g10@ecs.soton.ac.uk
Elliott, Kristian	ke1g10@ ecs.soton.ac.uk
Yu, Brian	by2g10@ ecs.soton.ac.uk
Mariani Elola, Oscar N	onme1g10@ ecs.soton.ac.uk
Chan, Yun Chak	ycc1g11@ ecs.soton.ac.uk

0 Meta Pages

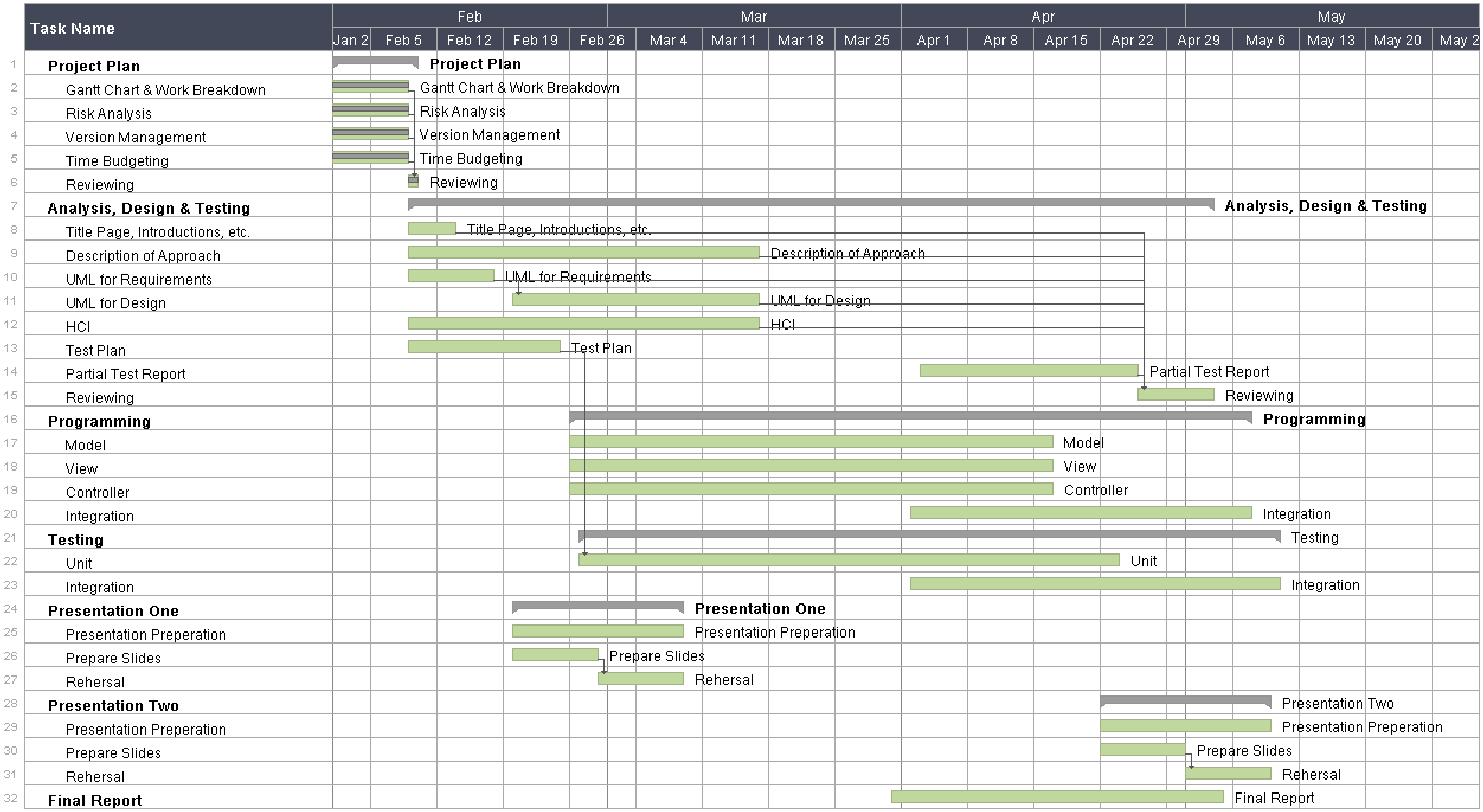
0.1 Document Control

Version	Author	Changes
1	by2g10	Initial document: title page, table of contents page, document control page and section headings.
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
9	by2g10	Added more things I did for the initial document. Added headings to version management section.
10	ycc1g11	Updated Time budgeting section
11	All	Reviewed whole document. Made minor spelling, grammar and formatting changes. Updated the Time and Budgeting table.

0.2 Table of Contents

1 Work Breakdown & Allocation

1.1 Gantt Chart



	Task Name
1	**Project Plan**
2	Gantt Chart & Work Breakdown
3	Risk Analysis
4	Version Management
5	Time Budgeting
6	Reviewing
7	**Analysis, Design & Testing**
8	Title Page, Introductions, etc.
9	Description of Approach
10	UML for Requirements
11	UML for Design
12	HCI
13	Test Plan
14	Partial Test Report
15	Reviewing
16	**Programming**
17	Model
18	View
19	Controller
20	Integration
21	**Testing**
22	Unit
23	Integration
24	**Presentation One**
25	Presentation Preperation
26	Prepare Slides
27	Rehersal
28	**Presentation Two**
29	Presentation Preperation
30	Prepare Slides
31	Rehersal
32	**Final Report**

Figure 1 – Gantt chart (Generated using smartsheet.com)

1.2 Description

We have based the main milestones of our project on the dates of the deliverables, planning to ensure that all the tasks are completed by the hand in dates. For many of the tasks we have 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. We have allocated a total of 356 hours in Table 1, this gives us plenty of leeway if we overspend.

ID	Name	Time Estimation	Assignment
1	**Project Plan**		
2	Gantt Chart and Work Breakdown	6 hours	Edward, Kristian
3	Risk Analysis	2 hours	Oscar
4	Budget	2 hours	Kelvin
5	Project Management	3 hours	Brian
6	Reviewing	5 hours	All
7	**Analysis, Design and Testing**		
8	Title, Introduction and Revision History	3 hours	Kelvin
9	Description of Approach	3 hours	Kristian
10	UML for Requirements	2 hours	Brian
11	UML for Design	5 hours	Edward
12	Human Computer Interaction	5 hours	Oscar
13	Test Plan	5 hours	Brian
14	Partial Test Report	10 hours	Edward, Kristian
15	Reviewing	5 hours	All
16	**Programming**		
17	Model	40 hours	Kelvin, Oscar
18	View	60 hours	Edward, Kristian
19	Controller	40 hours	Brian, Oscar
20	Integration	40 hours	All
21	**Testing**		
22	Unit	20 hours	All
23	Integration	20 hours	All
24 & 28	**Presentation (x2)**		
25 & 29	Discussion and Group Preparation	20 hours	All
26 & 30	Prepare Slides	5 hours	All
27 & 31	Rehearsal	5 hours	All
32	**Final Report**		
	Tasks to be assigned	50 hours	All

Table 1 – Task allocation table

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.

Table 2 – Risk Analysis Table

3 Time Budgeting & Recording

3.1 Budgeting

When deciding the hours to allocate, we used the following techniques:

3.1.1 To obtain an initial estimate on the time spent on each task, we considered the weighting of the deliverable it corresponded to. For example we are in a group of five and each member is expected to spend around 80 hours on the project in total and because the project plan is worth 5% of the overall mark of the project, we will make use of approximately 20 hours out of 400 hours on the project plan.

3.1.2 We plan to have a formal meeting every week for at least an hour with our supervisor. We may also conduct some informal meetings in which we shall review what we have done since last formal meeting and resolve any problems that needs solving before the next formal meeting. This should allow us to have a better understanding of our progress.

3.1.3 For many of the tasks we found that assigning around two hours, and then increasing and decreasing the amount of time required based on the nature of the task worked very well for acquiring an estimate.

3.1.4 We allocated the largest proportion of time to the programming and testing sections of the project as these are the fundamental areas of the project.

3.1.5 Before each presentation we shall discuss as a group what we feel need to be in the presentation, we shall then produce some slides for the presentation, and finally we will spend some time rehearsing the presentation.

3.2 Time recording and management system

We shall each be responsible for recording the hours we worked in a time sheet in our log books. When we start working on a task we will record it in a table (like Table 3) to make the record clear and to check whether we have overspent the time on a task.

Format of the Table:

Code / No.	Title	Start time	End time	Duration (h*)	Progress (IP / F)*	Subtotal duration (h*)	Expected Duration (h / NA)*	Notes

Key: h = hours, IP = In Progress, F = Finished, NA = Not Available

Table 3 – Time Sheet

We have assigned Kelvin to be the time manager. He will be assigned the responsibilities of monitoring the time spent, collecting, manipulating and maintaining the data, and summarizing the data for the end-of-project report in order to provide a clear image of this information. Statistics such as mean (average) and total time shall be used along with a variety of graphs shall be used to convey the information. During the weekly meetings the time manager shall compile all of the time spent that week by each of the members into a single table that shall be entered into a spread sheet and held in the version control system. Progress of each task will also be updated on the Gantt chart to help visualise the current project progress.

Half-way through and at the end of the project, the time manager will compare the initial time allocation with the actual time spent to ensure we are on track and make any adjustments to the schedule if necessary.

3.3 Time Expenditure Reporting

Each team member will have to report their time spent and the tasks they are working on and have finished to the time manager:

3.3.1. Show the time spent record in the log book and the effort of the week.
3.3.2. Provide the task code or number and the task name in order to identify the task.
3.3.3. Provide some basic information of the task.
3.3.4. Report the member involved and the time spent for each of them.
3.3.5. Report the starting time, ending time and actual time spent and expected time spent which can be in the form of verbal, words or even graphical format.
3.3.6. Report the progress of the task. (In progress / Finished)
3.3.7. Extra notes for the task. (Delay on finishing a task and the effect on the whole project)
3.3.8. Supervisor will be signing for the record in the log book and time manager will record and accumulate all the data reported by a team member.

4 Version management

Version management will be done using an SVN repository provided by the ugForge suite. We will mainly be accessing this 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
```

4.1 File and directory naming scheme

In the root directory of the SVN will be two directories named /src[1] and /doc[2]. 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.

4.2 Documents

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.

4.2.1 Document producer

Edward is assigned the responsibility of monitoring the folder of each submission for work added by team members; he will incorporate each member’s work into the main document (the one to be submitted) and credit the member’s work in the document control page of the main document.

4.3 Summary

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’ (the differences between to files) 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 trace 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.

[1] Stands for “Source Code”

[2] Stands for “Documents”

0.2 Index