Project Management Plan

for

Assignment 1 of

Applied Software Project Management

Prepared by:

Group I

November 17, 2014

1. Overview

1.1. Introduction

This is a project management plan document for a desktop application designed specifically for Windows 8 and Windows 8.1 platforms. It is an interactive application aimed at teaching users function concepts by making use of a simple maze game. Similar applications can be found plenty in number. To name a few, online games such as 'Dungeons of Kithgard' [1] serve the same purpose as our proposed application.

About the application: A brief description of the application's functionality and behaviour is necessary to better explain the intended purpose of the application and the value proposition that could be gained by a typical user. The following labelled diagram (Figure 1) is a predictive screenshot of the application during implementation.

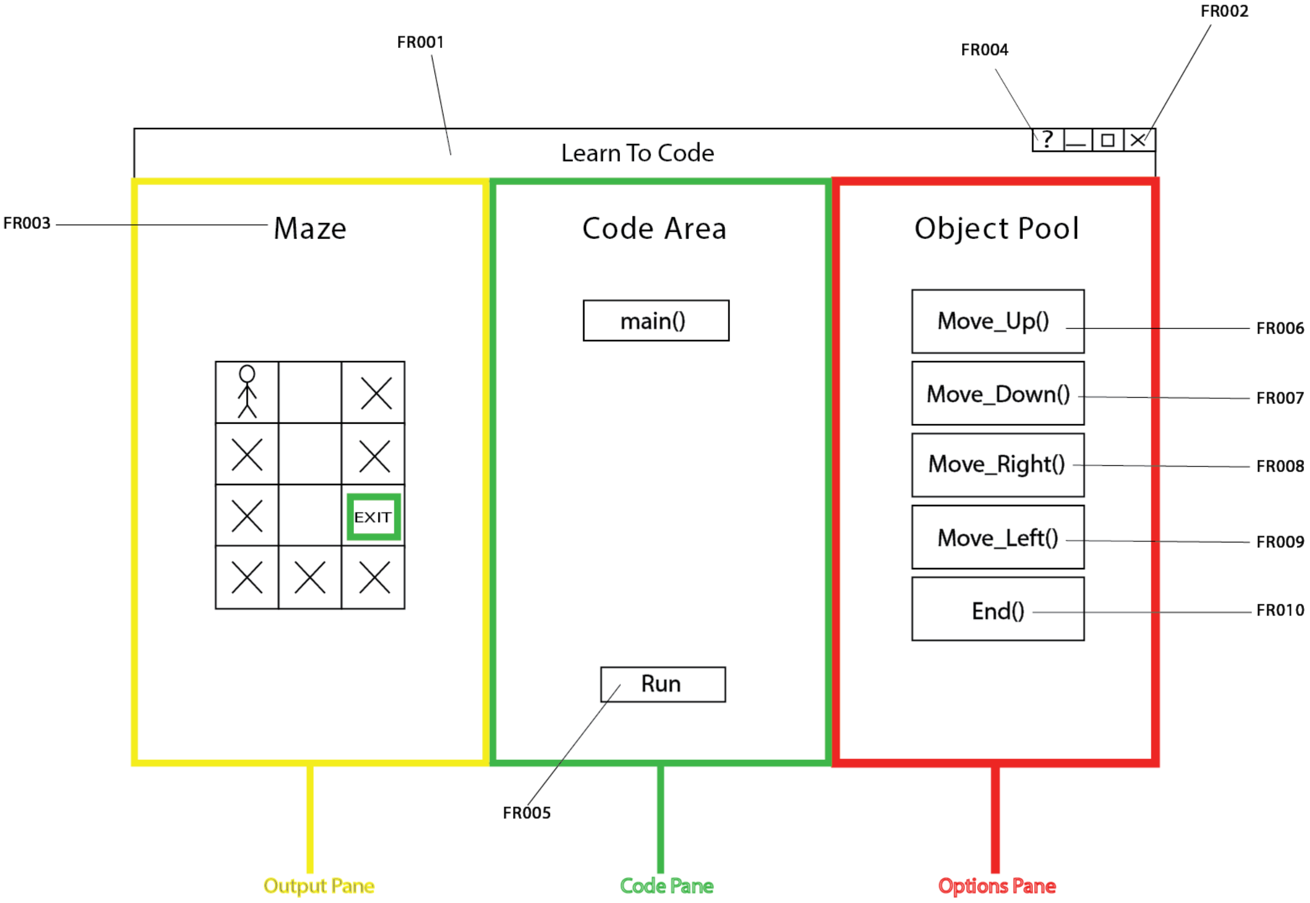


Figure 1 Screen shot of the proposed application

Note: The parts given labels that start with 'FR' are described in detail in section 1.2.1.

Typical scenario: There are three panes in the application: *output pane*, *code pane* and *options pane*. A player is given a set of functions in the 'options pane' to choose from. A response in the 'output pane' is generated when the player drags and drops a particular function from the 'options pane' into the 'code pane' and clicking the 'Run' button. A maze is displayed on the output pane, the solution to which can only be answered by making basic steps of moving up, down, left and right by the player. These moves are the functions made available to the player in the 'options pane'. So, whenever a player decides to select a particular move, the corresponding function must be dragged and dropped to the 'code pane' that would trigger the appropriate response in the 'output pane'. Game is finished when the maze is solved.

1.2. Team members and Initial Organization

Table 1 describes the team members involved in our project. Also stated are the programming languages each team member is acquainted with.

Member ID	Member Name	Programming Languages Known
TM001	Quentin Prevost	C, C++, C#, Java
TM002	Łukasz Swiatek	C, C++, C#
TM003	Yiran Zhou	C#, Java
TM004	Yuan Zhou	C, C++
TM005	Yanpeng Zhang	C, C++, Python
TM006	Priyanudeep Eada	C, C++, C#, Java
TM007	Srinivasu Akkineni	C, C++
TM008	Lavanya Pampana Vinod	C, C++

Table 1 Team Composition

Deciding on a common programming language is the first step in our project. This later on led to a brainstorming session to decide on the project to work on. Varied suggestions from team members spanned from creating a price comparator website to developing an open source project. However, based on the choice of the majority, the present application was selected as the project to be developed. In spite of team members being comfortable with C, C++ languages, we decided on C# as the intricacies in developing GUI is difficult in C, C++. The advantages of C# shadowed C, C++ for the kind of application we are developing.

There are four members in team who are good in C# and the rest have no knowledge in C#. To utilize the available human resource efficiently, we decided on pair programming. Pair programming, in the sense, one member good with C# will be paired with another member who has no good programming skills in C#.

1.3. Project Scope

1.3.1. Functional Requirements

The following are the functional requirements pertaining our project (see Table 2). These are the requirements to be implemented, the overall completion of which will implicitly lead to project completion.

Functional Requirement ID	Description
FR001	Start
FR002	Quit
FR003	Maze
FR004	Help
FR005	Run
FR006	Move_Up()
FR007	Move_Down()
FR008	Move_Right()
FR009	Move_Left()
FR010	End()

Table 2 Functional Requirements

FR001, FR002, FR004 and FR005 are the basic functions of the game typical to any desktop application. A detailed description of the rest of the requirements is provided in section 3.

1.3.2. Non Functional Requirements

At a broad level, in the view of the nature of the proposed application, the non-functional requirements as stated in ‘Table’ are chosen to review the final development.

Non Functional Requirement ID	Description
NFR001	Response time
NFR002	Testability
NFR003	Maintainability

Table 3 Non-Functional Requirements

Response Time: Games are typically designed according to the expectation of the end users, thus meeting the expected responsiveness. Therefore, the quality attribute ‘response time’ as a non-functional requirement is essential from a game's perspective. As our application is aimed to teaching concepts of functions by employing a game, it is necessary that the response time to the users' actions is well timed and accurate. It is also important that the corresponding actions selected by the user execute the intended functionality. We measure the response time in the units of msec (milliseconds).

Testability: The proposed application must be suitable to conduct functional testing of UI elements and player actions. Therefore, it is necessary that the project plan includes steps to ensure that the final application is testable. A test-driven approach of designing test cases at the start of the design phase of the project is deemed the most viable option to ensure that such a requirement is fulfilled.

Maintainability: Depending on the project's progress, there can be instances when new features have to be incorporated to the product in order to improve the overall quality of the end product. When such issues arise, it is expected that an ideal project to be adaptable to the changes made in the project in terms of architecture, design or implementation without causing problems to the previous stages of the application. Thus, in order to ensure that the application is maintainable, it is necessary that all project related decisions are documented and traced. This will ease the process of making changes to the project during its lifetime.

1.4. Assumptions and Constraints

Assumptions: For every software project, it is crucial that a set of assumptions are made before the start of the project. This is usually included as a part of the project management plan. The risks of these assumptions either going wrong or not meeting standards must be documented. As a part of our project, the following assumptions are made in order to achieve a successful completion:

- Each team member has access to software tools required to develop this application. For example, Microsoft Visual Studio 2013 is used to code the application and is assumed to be available via Dream Spark account of individual members.
- Team members will work with the available resources in hand. Thus, it is assumed that there is no budget in terms of software and hardware costs to the project.
- The majority of the team members are available throughout the project. This is a reasonable assumption made due to the commonness of the inevitable staff attrition that is inherent to every software project. Therefore, it is at least expected that most of the team members are available for the entire project. However, the risk of lack of human resources is reflected in section 6.
- It is assumed that the end users are well versed with basic skills required to interact with the proposed application. At times when users' understanding is impeded, it is assumed that the help document provided as a deliverable is sufficient to gain thorough insight.
- The project is non-profit in nature aimed as a utilitarian enhancement in teaching concepts of functions.

As described earlier, the risk of not fulfilling the above stated assumptions are mentioned as a part of the 'Risk Management Plan' and can be seen in section 6.

Constraints: Owing to the team's composition, knowledge in programming languages of individual team members and the lack of prior work experience, our project suffers from few limitations. These limitations are described as the constraints coupled to our project and are mentioned as follows:

- Not all the team members are well acquainted with the programming language C#, which was chosen to code the application during development phase of the project. This would result in the unavoidable training activities to ensure that every individual in the team is competent. However, in our project training is not provided to any member since all of them are well versed with the programming concepts. Further, everybody is expected to learn syntax on their own.
- The duration of the project is limited in the way that the time spent on the project by individual team members is limited. This is fundamentally because everybody is involved in other projects and can spare limited working hours for the proposed project.
- Communication issues arise due to variation in cultural backgrounds of individual team members. This is attributed to the fact the team is composed of members from different nationalities. However, as suggested in scientific and empirical studies, as suggested in [2], the diversity in team members can also bring in discipline during work.

Unlike the ‘project assumptions’, ‘project constraints’ are not serious potential risks of the project’s progress. They must be viewed as boundaries within which the best strategy must be adapted to implement the project. When properly planned, the constraints can be changed into the project’s major strengths.

1.5. Deliverables

The following components are the deliverables to be expected after the successful completion of the project undertaken:

- A working game application that can be implemented on Windows platform.
- Documentation related to the software. This includes artefacts such as:
  - User manual
  - Help document
- Documents related to version control, risk control and quality control. Choices made during project execution are documented for future reference.

2. Project Organization

2.1. Roles and Responsibilities

Team comprises of only eight members and everyone has to play multiple roles in the project. The different roles in our project are:

1. Analysts: There are 8 analysts in our team. Each of them contribute in identifying the requirements for the project. The functional and non-functional requirements are listed down in section 1.3.1 and 1.3.2, respectively. Peer-review is performed to select the final set of requirements to be implemented from the candidate requirements.
2. Designers: There are 3 designers in our team. TM001, TM006 and TM008 will contribute in modelling UI elements and designing architecture of the project. Using IBM Rational Rose tool the use-case diagram, class diagram, activity diagram, sequence diagram and deployment diagram are designed. Adobe Illustrator is used to design the templates for UI elements. Only the above three members have good knowledge over UML concepts. TM006 is good at modelling UI and has a history of developing prototypes. While TM001 has a good vision of the project.
3. Developers: All the team members will be taking up the role of developers. Thus, there are 8 developers in our team. However, there are three modules in the project namely *Output pane*, *Code pane* and *Options pane* that requires division of work. As discussed in Table 1, only 4 team members have good practical knowledge on coding in C# while the rest do not. Pair programming is considered the best solution to resolve the steep learning curve involved for a novice developer. Roles for each module are discussed below:
   Experienced programmers = Set A = {TM001, TM002, TM003 and TM006}
   Novice programmers = Set B= {TM004, TM005, TM007 and TM008}
   Each module will be developed by two members who are mapped one from each set A and B, respectively.
   a. Module 1 (Output pane):
      Developers for module 1: TM001 and TM008
      Functional requirements FR001 and FR003 are to be implemented by team members TM001 and TM008. The final window visible to the player has to be developed that comprises of integrating module 2 and module 3 to display consistent output.
   b. Module 2 (Code pane):
      Developers for module 2: TM006 and TM007
      Generating the relevant code corresponding to the functions selected by the user is the main component to be developed by the assigned members. This is achieved by implementing the functional requirement FR005. Additionally, the basic features including application termination, help document are handled by implementing FR002 and FR004.
   c. Module 3 (Options pane):
      Developers for module 3: TM002, TM003 and TM004, TM005

Functions available to the player during the launch of the application must be developed by the stated team members. These functions are implemented by FR006, FR007, FR008, FR009 and FR010.

4. Testers: Testing involves test planning and test execution. The role of testers is again classified as test planners and test executers. There are 3 test planners and 5 test executers in the project.
   a. Test planners: TM001, TM003 and TM008
   b. Test executers: TM002, TM004, TM005, TM006 and TM007

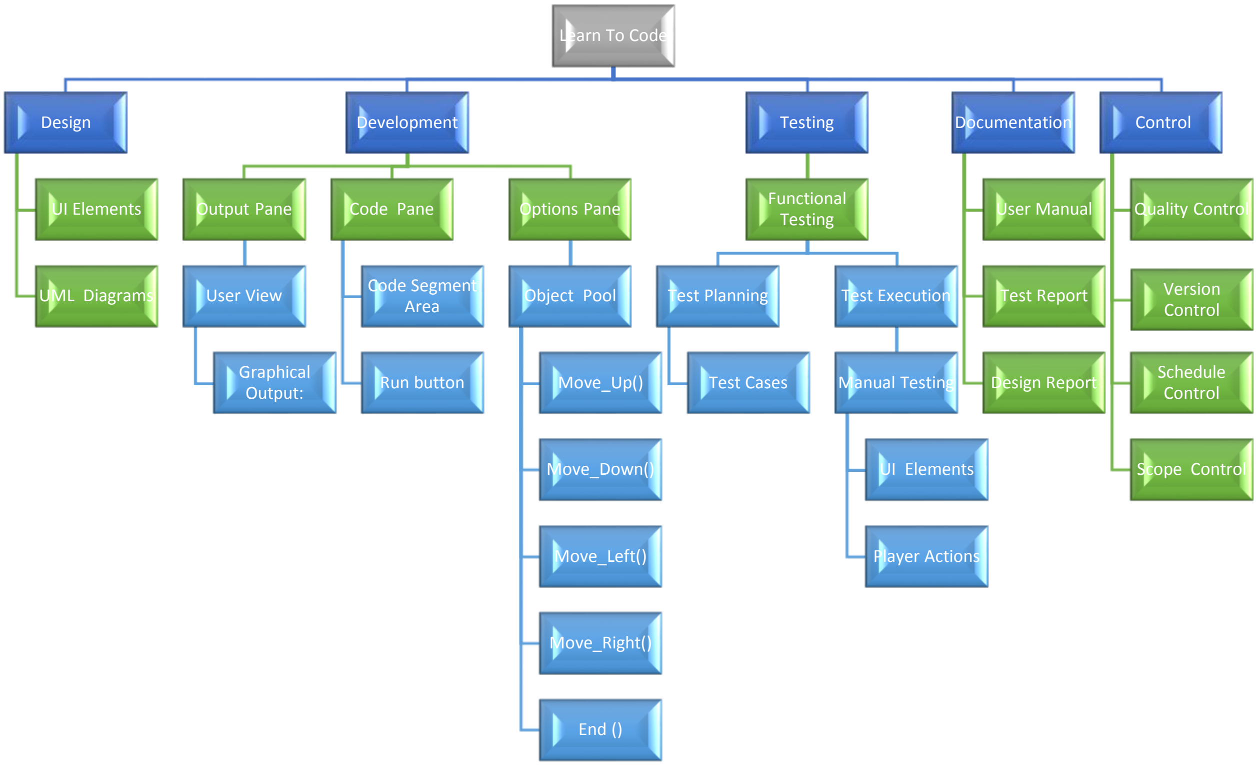
   Test planners will have to plan the test cases. A test report will also be drafted by TM001, TM003 and TM008. Test executers will follow the test plan and execute the test cases and report the bugs found in the test report.

5. Document Reviewers: People proficient in English are given the role of document reviewers. It was unanimously decided that TM002, TM006 and TM008 are good in documenting. Thus, the document reviewers' task is taken up by the above three members. Description for functional requirement FR004: *User help* in the application, test report and design report will be finally reviewed by the document reviewers. The documents drafted for each phase will be finally proofread by the document reviewers. The team members associated in each phase will draft the document. However, the document reviewer will only facilitate in reviewing document drafted in various other phases.
6. Control Board: The entire team is responsible for monitoring control over the project. The description for scope control, quality control, version control and schedule control is provided in section 5. Weekly meetings are conducted to observe any deviation from the defined control plan.

3. Work Plan

3.1. Work Breakdown Structure (WBS)

The Work break down structure describes an increasingly detailed definition of project work [PMBOK]. The first layer of division reflects the process model lifecycle and also the other activities involved. WBS dictionary is supported to describe and understand the terms used in work activities and packages.



WBS Dictionary

Level: The hierarchical level within WBS.

WBS ID: It is a unique identifier for each task that represents hierarchical location within the WBS.

Description: The description for each WBS ID is provided in the table. Most of the level 2 tasks are descriptive and description of level 2 tasks are provided only when needed.

Estimation: The amount of time taken to complete each activity. The effort is set to number of days (8 hours/day).

Level	WBS ID	Task	Description	Estimation (in number of days)	Acceptance Criteria
1	1	Learn To Code Application		45	
2	1.1	Design	Contains the whole system diagram, deployment diagram and design of UI elements.	8	
2	1.1.1	UI Elements	Designing an attractive user interface.	4	Usage of simple colors and easy to relate UI elements.
2	1.1.2	UML Diagrams	Design use-case diagram, activity diagram, sequence diagram, main class diagram, and deployment diagram.	4	Each diagram can be traced to other diagram.
2	1.2	Development	Development of three different modules namely output pane, code pane and options pane.	15	
3	1.2.1	Output Pane		5	
4	1.2.1.1	User View	Provides the view of user actions.	3	
5	1.2.1.2	Graphical Output	Graphical output includes the maze and player actions in the maze.	2	Functioning maze is displayed on the output pane.
3	1.2.2	Code Pane		5	
4	1.2.2.1	Code Segment Area	The available function options in the object pool are implemented on the Code Pane canvas. The functions dragged and dropped reflect user action.	3	A successful display of the corresponding options selected.
4	1.2.2.2	Run Button	Clicking on Run button will trigger the user action in maze. (function code is run)	2	Triggering the appropriate event upon clicking the button.
3	1.2.3	Options Pane	The options of functions that can be called are listed.	5	
4	1.2.3.1	Object Pool	Collection of functions.	2	
5	1.2.3.1.1	Move_Up()	Function definition of moving one cell up.	1	Should serve the functionality of moving one cell up.
5	1.2.3.1.2	Move_Down()	Function definition of moving one cell down.	1	Should serve the functionality of moving one cell down.
5	1.2.3.1.3	Move_Left()	Function definition of moving one cell left.	1	Should serve the functionality of moving one cell towards left.
5	1.2.3.1.4	Move_Right()	Function definition of moving once cell right.	1	Should serve the functionality of moving one cell towards right.
5	1.2.3.1.5	End()	Function definition of end of actions (functions).	1	Should serve the functionality of end of actions.
2	1.3	Testing		13	
3	1.3.1	Functional Testing	Functional testing of the application is achieved.	13	

4	1.3.1.1	Test Planning		8	
5	1.3.1.1.1	Test Cases	Test cases are developed.	8	Acceptable test cases drafted in test report.
4	1.3.1.2	Test Execution	Test cases are implemented.	5	Bugs identified reported in test report.
5	1.3.1.2.1	Manual Testing		2	
6	1.3.1.2.1.1	UI Elements	User Interface elements such as buttons, drag and drop and so on.	2	Desired functionality provided by UI elements.
6	1.3.1.2.1.2	Player Actions	The functions such as Move_Up(), Move_Down(), Move_Left(), Move_Right() are tested.	2	Desired functionalities are implemented in the functions.
2	1.3	Documentation		6	
3	1.3.1	User Manual	Description for user to understand game application with ease.	2	Document easy to understand and useful for the users containing FAQ.
3	1.3.2	Test Report	Bugs generated are reported timely in Jira tool and then status of fixing the bug is also noted to keep track of bugs.	2	
3	1.3.3	Design Report	Design of the project architecture using UML diagrams is reported for traceability reasons.	2	
2	1.4	Control		45	
3	1.4.1	Quality Control			
3	1.4.2	Version Control			
3	1.4.3	Schedule Control			
3	1.4.4	Scope Control			

Table 4 WBS Dictionary

Schedule Allocation:

The tasks discussed in the WBS are implemented within the limited time available. These are depicted in the Gantt chart as shown in Figure 2.

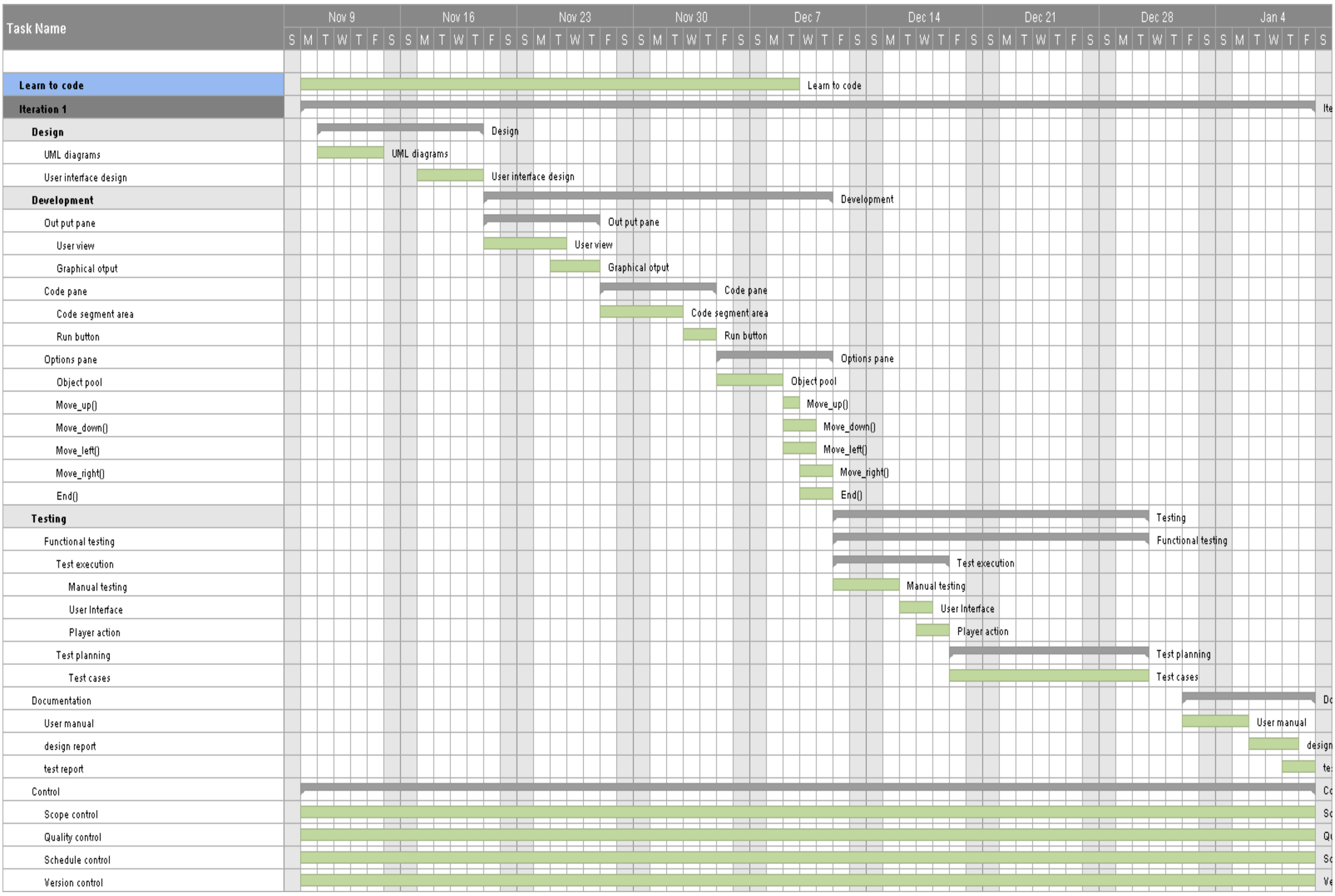


Figure 2 Project Gantt chart

The milestones to be achieved during the project's progress is given in Table 6 along with the indicators required to verify these milestones.

Milestones	Objective Indicators
Creating Architecture	Blue print of the project
Implementing an executable application	A working application
Generating test reports	Test cases
Creating user manuals	Project documentation

Table 5 Milestones

4. Start-Up Plan

4.1. Effort Estimation

Effort estimation is calculated using ISBSG benchmarking-data estimation. ISBSG benchmarking-data estimation includes the past industry data. From the existing tabulated columns of function points and their corresponding work effort, effort estimation of our project is performed. The function points for the project is calculated using function point analysis method. Based on the type and complexity, the predefined values are already known and given below:

Component	Low Complexity	Average Complexity	High Complexity
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	5	6

External Interface Logic	5	7	10
Internal Logical Files	7	10	15

Table 6 Function point analysis pre-defined values

For our project, the external inputs, external outputs, external inquiries, external interface logic, internal logical files for each module is identified and unadjusted function points are calculated for each module using the values in the above table.

Module	Calculation	Total Function Points
1. Output Pane	1EI + 3EO + 3EIF + 2ILF	62UFP
2. Code Pane	2EO+ 3EIF + 2ILF	59UFP
3. Options Pane	2EI + 2EIF + 1ILF	33UFP

Table 7 Function point analysis for the project

Function points for entire project = Function points of (Output pane + Code pane + Options pane) = 62 + 59 + 33 = 154 UFP

A regression model is developed based on the past data and a scatter graph is plotted.

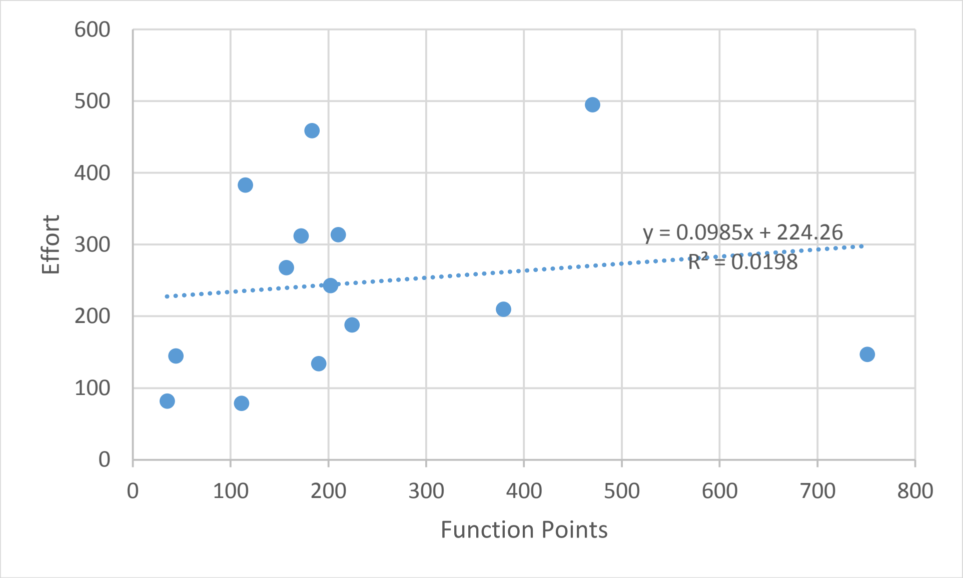


Figure 3 Scatter Plot

The equation identified using the data are:
Y=0.0985X + 224.26
For the UFP of our project, where X= 154 UFP
Effort = Y = 239 person hours
Thus, the effort involved in our project is 239 person hours.

4.2. Resource Planning

Team members are also treated as effective resources for the project. Tools and Skills required to implement the work packages (WBS ID) are discussed in the table:

WBS ID	Resources	Skills
1.1.1	Adobe Illustrator	Prior experience of working on Adobe Illustrator and similar tools.
1.1.2	IBM Rational Rose	Experienced in designing use-case diagram, main class diagram, activity diagram, sequence diagram, deployment diagram.
1.2.1.2	Microsoft Visual Studio, UI elements (developed in Adobe Illustrator)	C# programming knowledge

1.2.2.1	Microsoft Visual Studio, Code in C#	C# programming knowledge
1.2.2.2	Microsoft Visual Studio, Code in C#	C# programming knowledge
1.2.3.1	Microsoft Visual Studio, Code in C#	C# programming knowledge
1.2.3.2	Microsoft Visual Studio, Code in C#	C# programming knowledge
1.2.3.3	Microsoft Visual Studio, Code in C#	C# programming knowledge
1.2.3.4	Microsoft Visual Studio, Code in C#	C# programming knowledge
1.3.1.1.1	Jira	Experience in developing test cases.
1.3.1.2.1	Manual testing	Experience in using test tools.
1.3	Documentation	Good level of English

Table 8 Resource Allocation

5. Control Plan

5.1. Version Control

We were working as a team and we need some way to coordinate our source code, tests and some other artefacts. To coordinate changes for the files version control system acts as a respiratory and provides a history of changes to the files. For projects without version control, some parts of the code may scattered among developer machines, networked drives. The building process of the application may involve two or more people scrambling to find the latest versions of the several files and trying to put them in right places. To mediate changes to the files version control system will be used. The version control system works in a way that developers get files that they wanted, then run all the tests on these files to confirm that these files works, then check in their changes. This is a continuous integration process, occurs during several times during the project.

5.2. Quality Control

The process quality control is carried throughout the life cycle of the project.

Change control: For most of the time project team should stick to the requirements. In any case new requirements are added, team and product owner should clarify which one should be given more importance and which should be ignored. Product owner is responsible for the change in the requirements. Change in the tools, techniques and methods used need to be approved by the team members, because they will only identify the ease of change**.**

Pair programming: In our project we are using a concept of pair programming. This helps us in evaluating each other's work which ultimately leads to know about the deviations during the development of the project. Pair programming is used where it is necessary for task completion with the help of experienced programmer.

Testing: Identification of errors will be taken place in the whole testing phase which leads to quality control. Different testing techniques will be used to test the software**.**

Team co-ordination: Members in the team are from different backgrounds and cultures, so communication and coordination is very important. To improve the coordination among the team members weekly meetings are held. These meetings helps interacts with each other and reduces the communication barrel among the team members**.**

Quality attributes: We identified the following quality attributes for our project.

Functionality: to attain good functionality quality all the features should be implemented.

Response time: response time to the user's actions is well timed and accurate.

Testability: the application should be suitable to conduct various types of testing.

Code reusability: code modules should be reused in the project.

5.3. Scope Control

Project scope: Every project scope will identify the reason for developing the project and includes the project objectives. Project scope should be outlined in a way that what should be included in the project and what is not. Through project progress analysis project scope may be redefined, but it never differs from its goals and original outlines of the project.

Change control: This change control indicates what is acceptable and what is not. Scope of the project does not changed by the change control. If the scope of the project requires any change, which is acceptable change should be outlined in change control. This outline defines how changes are made and implemented**.**

5.4. Schedule Control

Schedule control plan depends upon the following steps.

1. Team should analyse the schedule and determine which areas may need corrective actions.

2. Then team members should decide what particular corrective actions should be taken.

3. Schedule plan should be revised to incorporate the particular corrective actions.

4. There after schedule should be recalculated to evaluate the effort of the corrective actions.

If any planned corrective actions are not completed according to schedule then the above steps are repeated. Throughout the project if any changes are made to the schedule like adding new requirements and it is necessary to change schedule. For the changed schedule it should be known whether it needs any further attention or not. If any delay or a fall back occurs all the team should know it as soon as possible, this reduces the delay of the actual activities. This delay of actions may show some the effect on overall schedule.

6. Risk Management Plan

The potential risks associated with our project are detailed in Table 9. The mitigation strategies to handle these risks are illustrated by describing the responsibilities of assigned team members.

RID	Risk	Probability (Scale 0-5)	Impact	Assigned To	Roles and Responsibilities
R001	Unavailability of a team member for a meeting.	3	Roles and Responsibilities	TM007	TM007 will monitor the team members' participation. If one of the team member is unavailable for reasons then TM007 will have to inform him, the assigned task.
R002	Inability to meet the deadlines.	4	Project Schedule	TM006	TM006 will co-ordinate with the entire team and find out each team member's progress and point out to the team members well before-hand to hurry up to meet the deadlines.
R003	Over or under-estimation of effort.	3	Effort Estimation	TM003	Estimating effort for every 10 days can help in providing a more valid and reliable data of effort. TM003 will have to capture the data and estimate effort on timely basis and study factors that affect the changes in effort estimation. If the factors are identified and handled then, project can run smooth.
R004	Inability to perform different roles.	3	Roles and Responsibilities	TM007	Resource constraint forces all the team members to play multiple roles such as developers, testers and so on. TM007 will monitor the team members' participation. Members who are unable to perform triple or more roles will be swapped against roles that require relatively less effort.
R005	Inexperienced team members.	4	Resource Planning	TM004	C# is used as programming language. Four novice programmers are

					inexperienced in C#. TM004 will take the responsibility of monitoring pair programming. Make any modifications in the pair if the pair has any team issues.
R006	Communication issues.	4	Resource Planning	TM004	Since, team members come from different countries, background and education culture. It is difficult to communicate with team members and let everyone associate with the same thought process and approach to work. Regular meetings and meeting schedules will be drawn by TM004 and followed up regularly.
R007	Insufficient time.	3	Project Schedule	TM002	Time constraint is a serious risk to project. To meet the deadline of project, proper planning of schedule is necessary. TM002 will make changes to schedule and accommodate time lag of meeting the deadline if any.
R008	Adapting to change in technology.	4	Resource Planning	TM003	The IDE currently being used is Eclipse. If a decision to migrate to a new IDE is taken then training has to be offered to the entire team. TM003 will plan and handle the learning curve for any technology, tool related shift.
R009	Changes in design identified on developing code.	4	Design	TM001	Design drafted for the project might change through the course of development. TM001 will have to observe any changes made to design and update the changes in the design report.
R010	Incomplete test coverage.	3	Testing	TM005	There is possibility that the testing coverage is not complete. Test plan should be carefully drafted accommodating complete coverage.
R011	Module cohesive and coupling for Output pane, Code pane and Options pane unachieved.	4	Design	TM001	Module cohesive and coupling should be ensured to adapt new features easily on the existing code and also enable four sub-teams to work independently. Design of the three modules should be carefully planned and sketched.
R012	Stress on team members.	3	Roles and Responsibilities	TM008	Stress must be effectively managed by finishing the tasks well before the deadline. TM008 will serve as stress monitor. Also, provide

					assistance to team members to cope with stress in a better way.
R013	Over or under estimation of time spent in the activities may delay the release.	4	Schedule Planning	TM006	The schedule planning must be monitored and altered to meet the deadline. TM006 will facilitate in making modifications to schedule if necessary.

Table 9 Risk management

7. Process Model

We select the "Waterfall Model" as our process model for developing the project. Table 8 also further explains the motivation for selecting our choice.

ID	Process Models	Accept/Reject	Criteria
P001	Waterfall Model	Accept	All the team members have good experience of working with waterfall model. Everyone has developed projects using waterfall model previously. Working on an already known model saves time and avoids the need to familiarize new concepts. Further, the application being developed is primitive and does not require iterations. Thus, a linear approach of development will help in streamlining the process of development and also help focus on the one phase, at a time, completely. Since, time is crucial hindering factor of development. However, pair programming is a borrowed concept from extreme programming that is included in our model for convenience of eliminating C# training provided to novice programmers. Since, novice programmers have good knowledge over C, C++ and coding concepts there is no need for training. Pair programming reduces the learning curve.
P002	Agile Methodology -Scrum -Extreme Programming	Reject	Agile is a customer-centric model. Customer plays an important role in the process model. Since, our project doesn't focus entirely on customer the process model is rejected. Further, it involves stand-up meetings at regular intervals of time. All the team members cannot be present for the meeting at specific and regular intervals of time since one of them is working and other team members have other responsibilities in other courses.
P003	Prototyping -RAD -Spiral -Incremental	Reject	Involves development of prototypes. With the time constraint of 8 weeks. Developing prototypes and further developing code involves more time. Also, involves iterations that are not needed for primitive scoped application. Thus, rejected.

Table 10 Process model acceptance criteria

References

[1] www.codecombat.com

[2] Erran Carmel, *Global Software Teams- Collaborating across borders and Time Zones*.

[3] A Guide To The Project Management Body Of Knowledge (PMBOK Guide). Newtown Square, Pa. : Project Management Institute, Inc., 2004. Print.