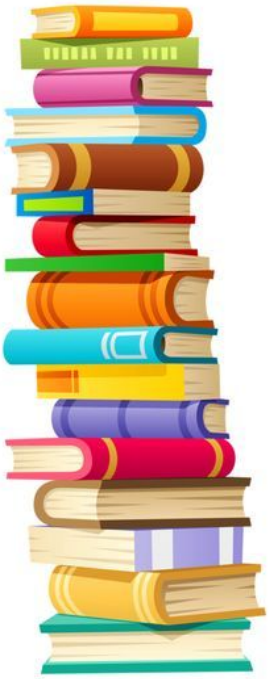


TOWER

Project Plan

Youssef Elmougy, Michael Cappeller, Liam Carlton, Raj Bedi

Version 2

Spring 2020

Table Of Contents

1. Introduction

This project plan will contain an overview of how the TOWER platform will meet the demands specified in the Requirements Specification document. This will be accomplished by grouping requirements together into specific tasks. These tasks are then analyzed for potential risks where it is then discussed as a team how these risks will be mitigated. The tasks are then split up among the group members based on their strengths and weaknesses. Finally, a project schedule is created by estimating the time it will take to complete each task and organizing it in a way to meet the April 28th deadline. The schedule is aided by the activity chart diagram (Section 5.2) which will indicate both the critical path and the areas which contain slack. The TOWER platform will be written on Android Studio using the Java language and the application will utilize outside API's from Amazon. All configuration management will be done through GitHub.

2. Project Organization

- Youssef Elmougy is the Analyst, creating diagrams and charts so other members are better able to keep on track for their tasks.
- Raj Bedi is the Innovator, thinking up ideas outside of the box and doing the research to see if it can fit in the project skill set of the group.
- Michael Cappeller is the Coder, tying it all together and helping the members in their required tasks.
- Liam Carlton is the Debugger, making sure every aspect of the project is functional in its own right.

3. Risk Analysis

3.1. Potential Risks

3.1.1. High Risk

a. Database creation gets delayed due to project members lacking prior knowledge. (High likelihood)
b. The project members lack previous knowledge of implementing API's. (High likelihood)

c. As application nears completion, there is a chance of certain features being unusable due to lack of time in trying to debug the source code. (Medium likelihood)
d. There is a reasonable assumption of Murphy's Law taking into effect regardless of how prepared the project members are. (Definitive likelihood)

3.1.2. Medium Risk

a. The integration of the GUI into the source code of the main application may get delayed due to project members lacking experience with Android Studio. (Medium likelihood)
b. Revisions to the original requirements specification in order to accommodate changes to the overall project may lead to delays. (Low likelihood)
c. Standalone GUI development gets delayed due to project members lacking prior knowledge. (High likelihood)

3.1.3. Low Risk

a. Delay to project start due to learning curve of Github and Android Studio Application (Medium Likelihood)
b. Delay due to tasks due to academic reasons as project members being full time students and cannot give complete attention to the project. (Low Likelihood)
c. Complications in meeting deadlines due to the Covid-19 Virus. (High Likelihood)

3.2. Risk Mitigation Strategies

3.2.1. High Risk

a. Since delays in database creation are relatively likely, we have decided to give priority to the task so that we will know early on whether or not the overall project plan needs to be changed to accommodate these delays.

b. On the project plan we have allowed enough for the project members to get familiar with android and discover how to use external APIs and libraries.
c. We have calculated the critical path to finish a week ahead of the presenting schedule. While putting extra pressure on members to accommodate the application faster, it gives members more time to work out any kinks or bugs that need to be fixed.
d. We have taken certain measures for each member to have a downloaded master copy of the code every four days in case something were to happen to cloud site Github in the case of inaccessibility.

3.2.2. Medium Risk

a. In order to avoid delays in integrating the new GUI with already existing source code, each team member will utilize online lecture videos that will ideally familiarize them with how to appropriately complete integrate the GUI using Android Studio.
b. Group members will be able to adapt to new project requirements, if any changes end up being made, chances are these changes will involve removing less necessary requirements as opposed to adding entirely new ones, the latter potentially being more time consuming.
c. Although the likelihood of delays in GUI development is quite high, the task is not on the critical path, meaning there will be several days of slack during which these delays can occur and have no harmful impact on the rest of the project.

3.2.3. Low Risk

a. Members should take the time to create a test Github sharing an Android Studio Project making changes and pushing it back to the Master branch.

b. Project Members should be aware of the difficulty of tasks and make preparations for assistance or finishing at an earlier time in case of exams or other reasons that would cause members to not finish the task in time.

4. Work Break Down

4.1. Key Milestones and Delivery Dates

1. Tuesday, February 4th: Assignment of Project Teams
2. Tuesday, February 25th: Project Schedule Begins
3. Tuesday, March 3rd: Term Project Status Updates
4. Thursday, March 26th: Term Project Group Implementation I
5. Thursday, April 2nd: Term Project Group Implementation II
6. Thursday, April 2nd: Integration of units of Textbook Application (T8)
7. Wednesday, April 8th: Integration of GUI with the Textbook Application (T10)
8. Thursday, April 9th: Term Project Group Implementation III
9. Saturday, April 11th: Integration of Amazon API to GUI (T12)
10. Tuesday, April 21st: Final testing and presentation preparation (T14)
11. Tuesday, April 28th: Term Project Presentations - Day One
12. Thursday, April 30th: Term Project Presentations - Day Two
13. Tuesday, May 5th: Performance Appraisal Process and Peer Reviews

4.2. Requirements Traceability

Task	Traceability to Requirements Specification
T1	3.1.2, 3.1.5
T2	3.1.3, 3.1.4, 3.2.2, 3.2.3
T3	3.1.3, 3.1.4, 3.2.7, 3.2.9, 3.2.10
T4	3.2.2, 3.2.3, 4.1.1,
T5	3.1.2, 4.1.4, 4.3.1,
T6	3.1.1, 3.1.4, 3.1.5, 3.1.7, 3.2.4
T7	3.2.5, 3.2.6, 3.2.8
T8	3.1.1-5, 3.1.7, 3.2.1-9
T9	3.3.1, 3.3.2, 3.3.3
T10	3.2.11, 3.3.1-3
T11	3.2.12
T12	3.3.4
T13	4.1.1-4, 4.2.1-2, 4.3.1-4.3.4
T14	4.4.1-3, 5.1.1-6, 5.2.1-6

4.3. Task Assignment

Task	Assigned
T1	Youssef
T2	Michael
T3	Youssef, Liam
T4	Michael, Raj
T5	Liam, Raj
T6	Michael, Youssef, Raj
T7	Liam
T8	Michael, Youssef, Raj, Liam
T9	Raj
T10	Michael, Liam
T11	Michael, Youssef
T12	Raj, Youssef
T13	Michael, Youssef, Raj, Liam
T14	Michael, Youssef, Raj, Liam

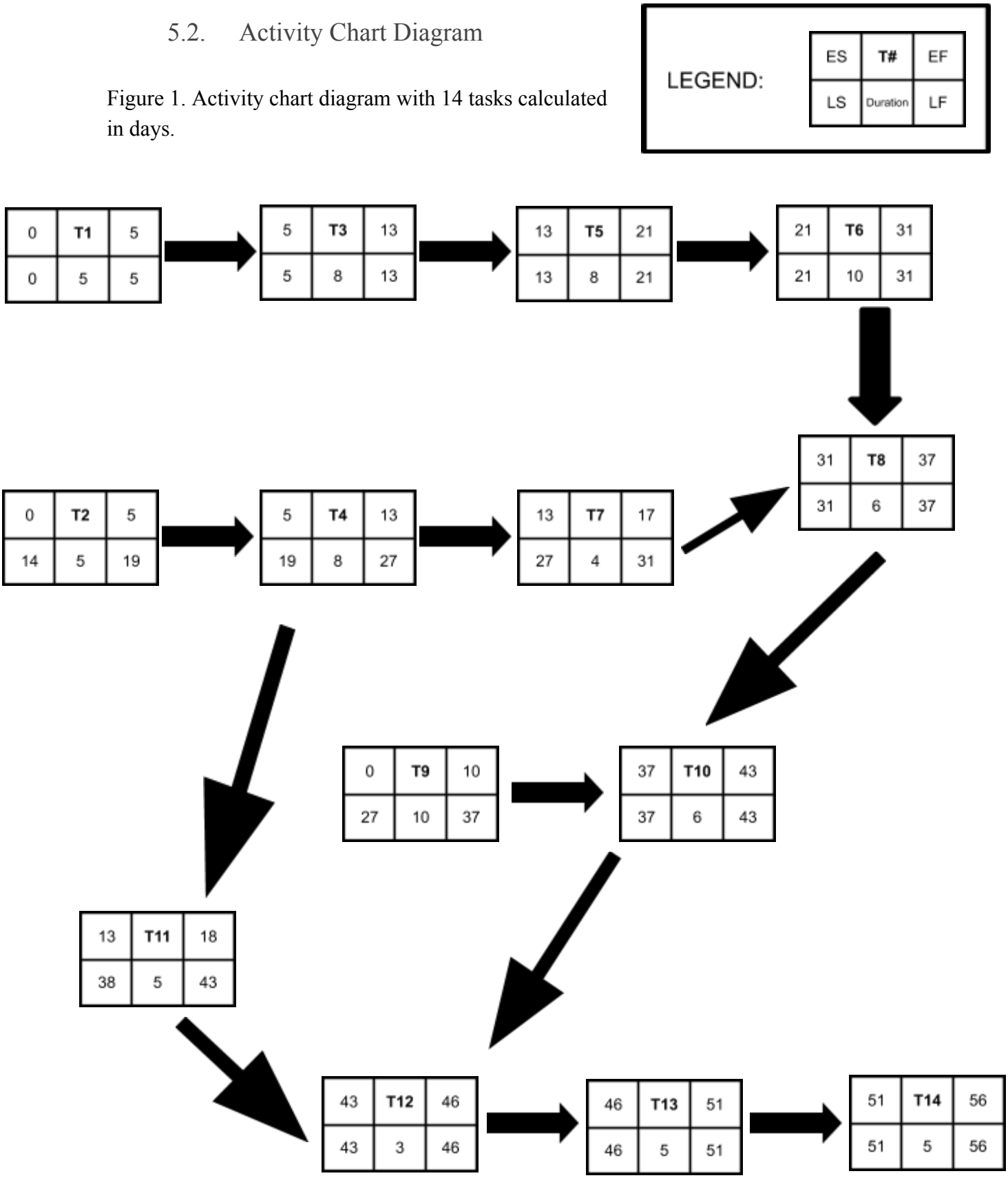
5. Project Schedule

5.1. Dependency Chart

Task	Description	Dependency	Duration
T1	Generate data structures for user profiles.	None	5 days
T2	Generate data structures for textbooks.	None	5 days
T3	Create a database to store user information.	T1	8 days
T4	Create a database to store textbook information.	T2	8 days
T5	Develop a login and sign-up system.	T3	8 Days
T6	Develop user features for buyers and sellers.	T5	10 Days
T7	Design search and sorting algorithm.	T4	4 Days
T8	Integration of units of the Textbook Application.	T6, T7	6 days
T9	Standalone GUI Design.	None	10 Days
T10	Integration of GUI with the Textbook Application.	T8, T9	6 Days
T11	Integration of Amazon API with the database.	T4	5 Days
T12	Integration of Amazon API to GUI.	T10, T11	3 Days
T13	Debugging.	T12	5 Days
T14	Final testing and presentation preparation.	T13	5 Days

5.2. Activity Chart Diagram

Figure 1. Activity chart diagram with 14 tasks calculated in days.



The critical path is T1, T3, T5, T6, T8, T10, T12, T13, T14. Tasks T2, T4, and T7 each have a slack of 14 days. Task T9 has a slack of 27 days. Task T11 has a slack of 25 days.

6. Quality Assurance Planning

6.1. Black-Box Testing

6.1.1. Youssef has been using the app as a user would and is finding any bugs that a client may encounter. He is making a note of every bug he finds and is telling the team in order to address the problem.

6.2. White-Box Testing

6.2.1. A “Helper” class was created that contains various methods useful for testing the capabilities of the app. One method, floodDatabase(), allowed us to “flood” our database with upwards of a thousand textbook objects simultaneously. Doing this helped us determine the number of books that can be added without causing drastic slowdowns.

6.3. This will ensure that tasks are being tested before further development of the project, hence mitigating the potential risk of project components not working when attempting to integrate units.