Object Oriented Analysis and Design of Software Engineering

UE18CS353

Project Title : Photo Editor

Prepared by

Nikhil KR (PES2201800044)

Worked on

Lifecycle and Model (Section 1)

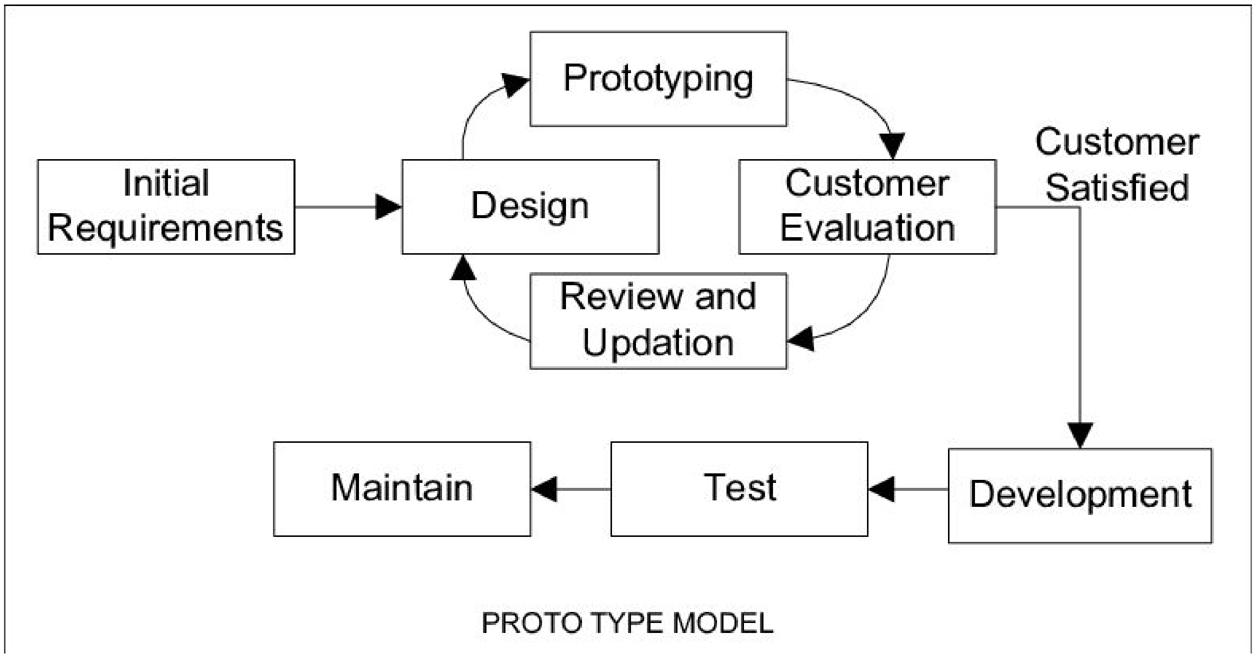
Tools Required (Section 2)

WBS: Execution and Test (Section 4)

Table of Contents

Lifecycle and Model

We propose a prototype model under the legacy Software Development Lifecycles (SDLC) for the photo editor project. This model is also known as a lightweight process model since it does not follow as much rigidity as the waterfall or the V-model. The flow diagram of a generalized prototype model can be seen below :



The model involves gathering a few requirements at the beginning and forming a basic prototype with which the implementation can begin. After building a basic prototype, customers are allowed to evaluate, and their suggestions are taken to refine the requirements. This process is repeated until a final set of requirements is listed down with which the final version of the product can be delivered.

The advantage of this mechanism is that the customers are involved throughout the process and hence a more user-friendly system can be built easily. This model also favours the developers in terms of freezing the requirements early and hence has an upper hand over the other models.

Since the GUI is an important aspect for a photo editor, this model will allow us to revisit the prototype from time to time which can help improve the entire system.

Tools Required

Planning Tool:

Visual Paradigm,Git/Gitlab

Planning is done in two stages, here, we’ve incorporated VP into Git, first by analyzing the requirements in VP and implementing these tasks by creating milestones and objectives through the use of issues in Git/Gitlab.

Design Tool:

StarUML, TeamGantt, Visual Paradigm

Star UML was used for constructing the Use Case Diagram and is going to be used for future UML diagrams that are needed for this project.

Version Control:

Git/Gitlab

A version control tool is used in this project due to its dependency on CI/CD as it deals with a prototype model.

Development Tool:

JavaScript-(NodeJS, Angular)

A web based application with OOP is one of the requirements stated for implementation. NodeJS and Angular, both offer easy integration for creating the system required.

Bug Tracking:

Git/Gitlab

Git provides an easy way of communication through issues and open for comments sections for bug tracking and error reporting.

Testing Tool:

Selenium

Currently, the testing tool that is considered to be ideal for this project is Selenium.

Deliverables

List :

- **Software Requirement Specification**
- **Design Document**
- **Project Plan**
- **Project Design**
- **Management Plan**
- **Testing**
- **Final Build**
- **Final Report**

Build Components:

- **Frontend UI :** A skeleton structure is analyzed and built from scratch. References shall be specified wherever necessary.

- **SRS/Design Document/Plans :** All the necessary documents are written by the team members from scratch. Diagrams and Flowcharts that are drawn are on the basis of the given template.

- **Code Integration :** Integration and merging of code is done using a version control tool to maintain a smooth workflow and easy transfer of code.

- **Scheduling/Milestones/Objectives/Deadlines :** All the important sidelines with respect to the project are considered and implemented from scratch by the team members.
- **Documentation/Report :** A flowing document is passed around to be filled up by the team members which is used to maintain the overall documentation of the

code written , difficulties faced and solutions to the same, and steps to be taken on encountering errors must be jotted down. A final report, that is user-friendly, is done on the project that includes the workflow of the whole application built.

Reuse Components:

- **Customer Requirements :** The list of the requirements such as cropping, rotating etc. of a customer with respect to a photo editor app can be reused.

- **Proof of Concept :** This system can be categorized as a doable one since there are many applications such as Snapseed, InShot, Prisma etc. in the market that serve as a proof of concept.

- **Testing :** Testing is done using a testing tool by feeding in the final deliverable, that is the application, into the testing software. Custom code is written wherever necessary.

- **APIs:** Integrated APIs are used for building up the different use cases wherever libraries are defined.

Reusable components mentioned use existing code, APIs whenever feasible and meet the requirements.

Work Breakdown Structure

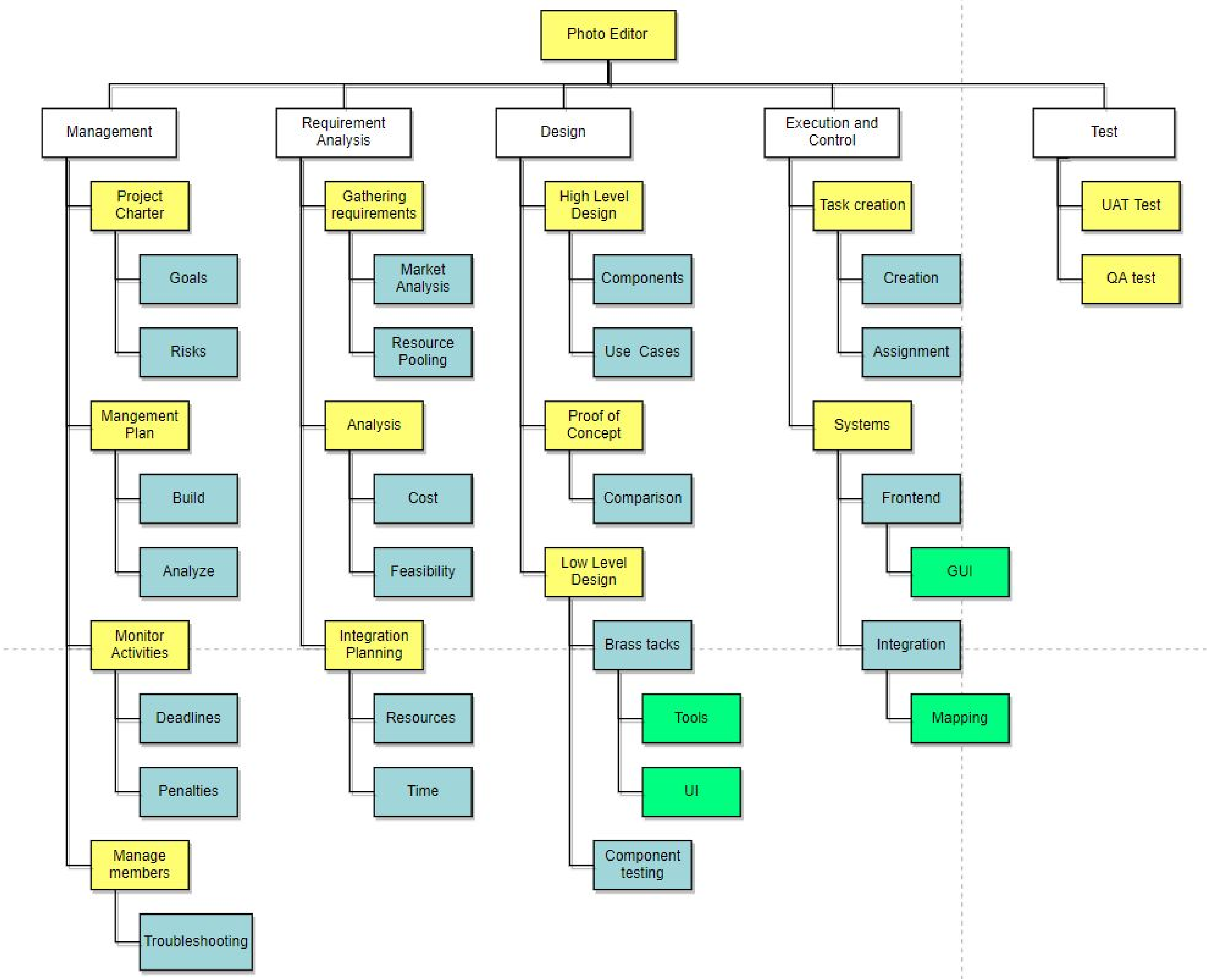


Photo Editor

1. Management

1.1. Project Charter

1.1.1. Goals

The goals of the project, the initial outline with customer fulfilment in mind, is put into action.

1.1.2. Risk

Initial risk with respect to cost, both in person and time, must be analysed before giving the go-ahead.

1.2. Management Plan

1.2.1. Build

Set up a management team to fulfil the goals set. Availability of members and time constraints must be accounted for. Note that the selection of members is not done here and just the availability is checked.

1.2.2 Analyse

In the analysis phase, final deductions about the constraints that are set are analysed and decisions to go-ahead are taken.

1.3. Monitor Activities

1.3.1. Deadlines

Soft deadlines are set and the management must assign project leads to start working on the application. Leads in turn assign members to work on different parts of the project, giving them various hard deadlines to meet.

1.3.2 Penalties

Any deadlines that are not met must be dealt with, by scrutinization/removal of members/lead off the project.

1.4. Monitor Activities

1.4.1 Troubleshooting

Any member or lead that faces problems in completing their assignment must be given sufficient time and help to fix/solve their problems.

2. Requirement Analysis

2.1 Gathering Requirements

2.1.1 Market Analysis

Analysis of market trends on the product being made, with future demand and cost constraints included.

2.1.2 Resource Pooling

Scouring of resources needed for finishing MVP (minimum viable product) and final integration.

2.2 Analysis

2.2.1 Cost

Cost analysis of the pooled resources and cost in terms of manpower and time must be calculated.

2.2.2 Feasibility

Feasibility must include further analysis of other constraints that are not included in cost requirement analysis. This may include scheduling, funding and other off-stream constraints.

2.3 Integration Planning

2.3.1 Resource

Resource integration involves combining effective resources at the right time. This may involve scheduling different integrations at different points in time.

2.3.2 Time

Time sharing of different planned schedules to meet various deadlines must be taken into account for smooth functioning of the project in a version control environment.

3. Design

3.1 High Level Design

3.1.1 Component

Component diagrams are structural diagrams that are used to model the static implementation view of a system. This involves modelling the physical aspects of the system.

3.1.2 Use Case

To understand the behavioral part of our system, a use case diagram is used to depict the various relationships between all the use cases, actors and systems in play.

3.2 Proof of Concept

3.2.1 Comparison

This involves a basic understanding of our system done in model analysis and comparing it with a similar product on the market for further changes to be made.

3.3 Low Level Design

3.3.1 Brass Tacks

3.3.3.1 Tools

All types of tools required to start working on the project is mentioned here. The most important tools that are necessary for the system to work are : planning tool, design tool, version control, development tool, bug tracking, testing tool.

3.3.3.2 UI

A skeleton structure must be done for a rough visualization of what the user will see and work with. This template will serve as a reference for the UI/UX designers.

3.3.2 Component Testing

Component testing must deal with feasibility of working of different components created. This indirectly refers to modulating the features of UI components in our project.

4. Execution and Control

4.1 Task Creation

4.1.1 Creation

Create tasks to meet deadlines. Usage of version control tools and timeline/deadline milestones to communicate the same to all the team members.

4.1.2 Assignment

Assign tasks to team members, which involves issue creation/role assignment with deadlines/milestones set.

4.2 Systems

4.2.1 Frontend

4.2.1.1 GUI

The GUI framework with the right elements and their workings must be made. Developments must be made from the skeleton structure drawn in the Design Phase.

4.2.2 Integration

4.2.2.1 Mapping

Mapping of functionalities to UI components after coding up the features. Features are individually coded up and then integrated after testing their working.

5. Test

5.1 UAT Test

The goals formulated at the start of the project must be satisfied. As an evolutionary prototype model is used, customer evaluation is key for continuous updation of the initial and subsequent prototypes. UAT is done multiple times until the conditions set in the charter are met and the final deliverable is verifiable.

5.1 QA Test

The testing tools and customer feedback give out important results which determine the steps taken by the prototype model to drive the system. Scrutinization is key at this stage as it forces changes to be made in response, which helps in building up the next best prototype.

Effort Required

The Constructive Cost Model aka Cocomo is a procedural cost effective model that is widely used for the approximation of effort and time required for building a project. The model wholly depends on the number of lines of code as its input.

We will use the basic Cocomo model for the calculations. Since the project involves a small team and has already been worked on previously as per literature, we can use the Organic system under Cocomo.

The calculation of effort for a basic organic Cocomo is given by :

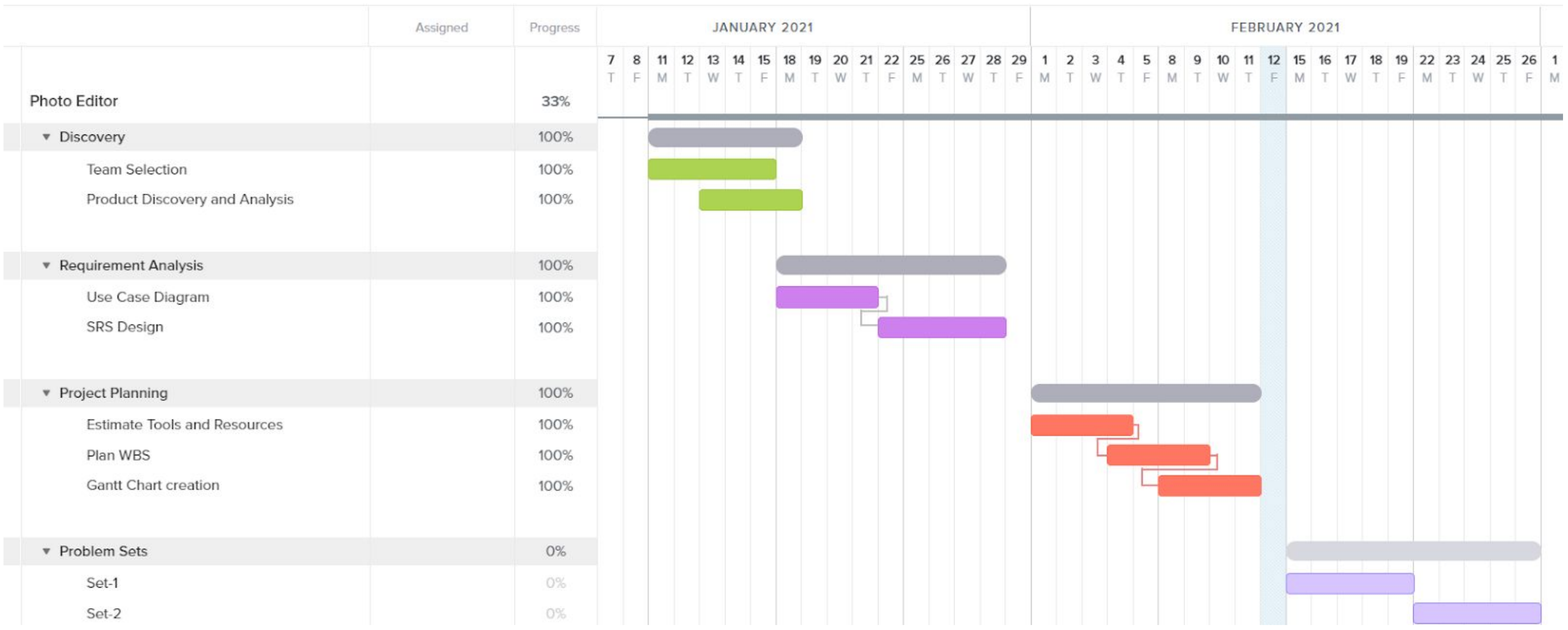
$$\mathbf{E = a(KLOC)^b} \qquad \text{... eq. 1}$$

where KLOC is the number of lines of code in the scale **1 unit = 1000 lines** and the values of **a** and **b** for a basic organic model are 2.4 and 1.05 respectively.
An approximation is made that this project will take **1200 lines** of code.
Substituting all the values in eq. 1, we get,

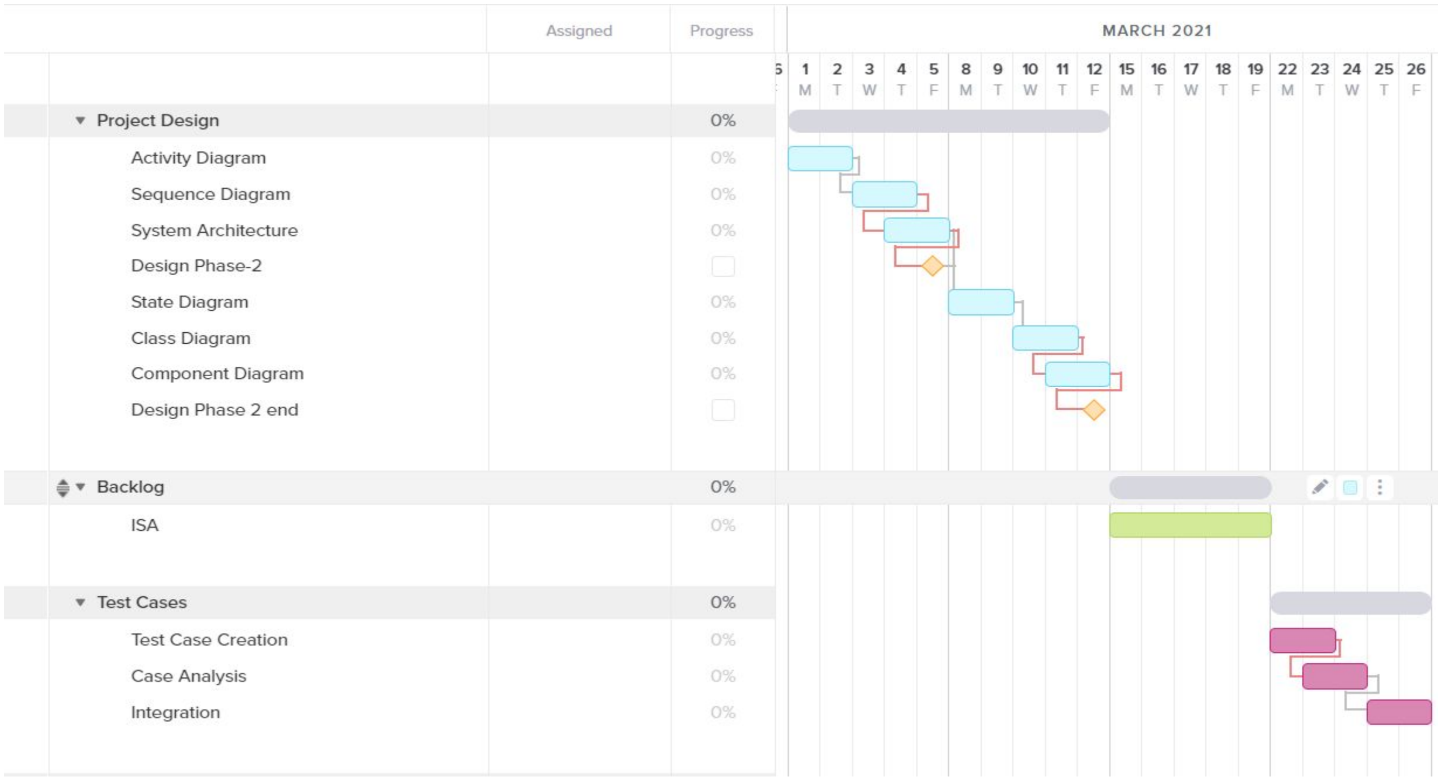
E = **2.90** Person-Months

Hence, the effort required is **2.90 PM**

Gantt Chart for Scheduling

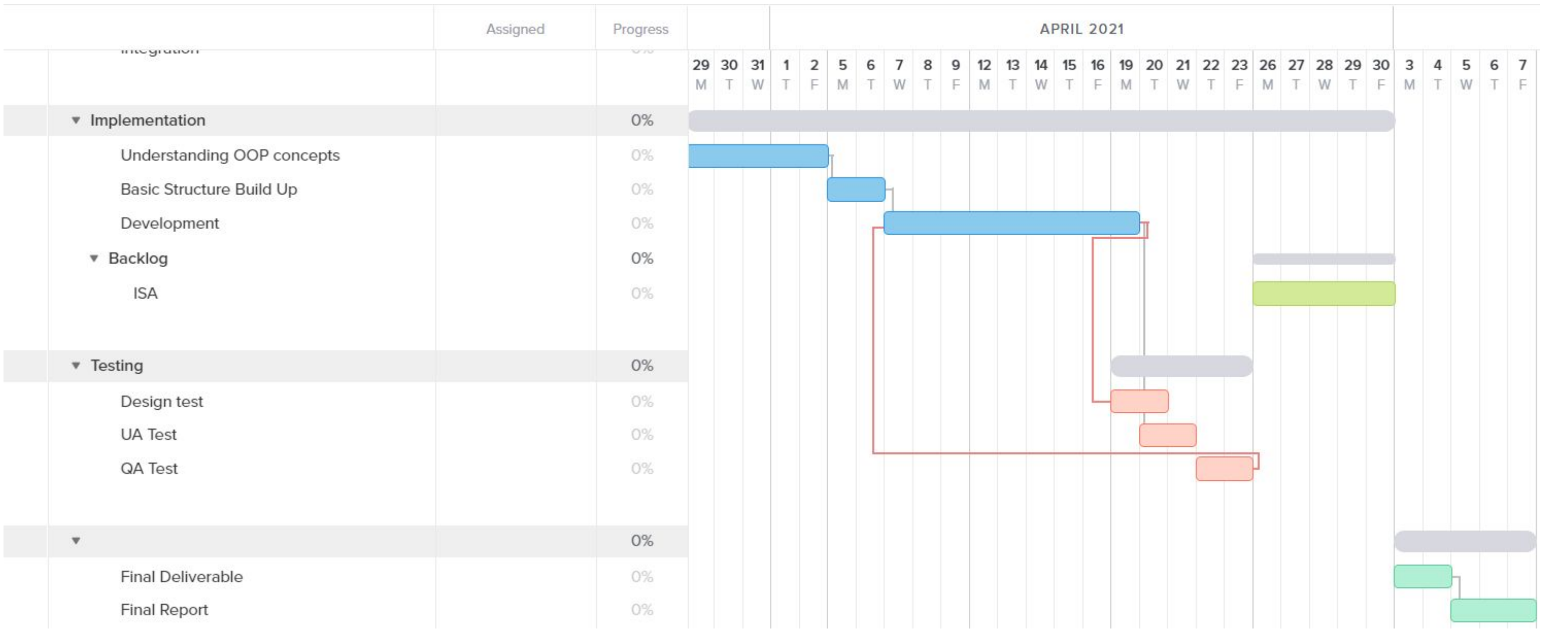


	Assigned	Progress
Photo Editor		33%
▾ Discovery		100%
Team Selection		100%
Product Discovery and Analysis		100%
▾ Requirement Analysis		100%
Use Case Diagram		100%
SRS Design		100%
▾ Project Planning		100%
Estimate Tools and Resources		100%
Plan WBS		100%
Gantt Chart creation		100%
▾ Problem Sets		0%
Set-1		0%
Set-2		0%



	Assigned	Progress
▾ Project Design		0%
Activity Diagram		0%
Sequence Diagram		0%
System Architecture		0%
Design Phase-2		
State Diagram		0%
Class Diagram		0%
Component Diagram		0%
Design Phase 2 end		
▾ Backlog		0%
ISA		0%
▾ Test Cases		0%
Test Case Creation		0%
Case Analysis		0%
Integration		0%

(Continued on the next page…)



	Assigned	Progress
Integration		0%
▼ Implementation		0%
Understanding OOP concepts		0%
Basic Structure Build Up		0%
Development		0%
▼ Backlog		0%
ISA		0%
▼ Testing		0%
Design test		0%
UA Test		0%
QA Test		0%
▼		0%
Final Deliverable		0%
Final Report		0%