Software Project Management 2021/2022

Flashee

D1.2.1 - Software Development Plan

Authors

- Ângelo Paiva.
- Burak Tinman.
- Jan Frank.
- Álvar San Martin.
- Pedro Henriques.

Reviewer

- ∉ 0.92, Ângelo Paiva.
- ∉ 0.92, Burak Tinman.
- ∉ 0.92, Jan Frank.
- ∉ 0.92, Álvar San Martin.
- ∉ 0.92, Pedro Henriques.
- ∉ 0.92, José Almeida
- ∉ 0.92, Rúben Lousada
- ∉ 0.92, Burak Tinman

Approver

- ∉ 1.0 Burak T

Table of Contents

1. Introduction
2. Project Organization
   - 2.1. Project Lifecycle
   - 2.2. Deliverables
   - 2.3. Supporting Documents and Records
   - 2.4. Project Management

3. Baseline Plan and Control
   3.1. Estimate
   3.2. Schedule
   3.3. Tracking and Control
4. Technical Process
   4.1. System architecture
   4.2. Methods tools and techniques

1. Introduction

The purpose of this project is to create a user-friendly web-based application with the goal of increasing the amount of people with the motivation to learn, be it words in different languages or information about a topic of choice. In the future, we hope to have achieved our goal of creating a friendly community who willingly share their flash cards with everyone.

As we know, as human beings, we crave knowledge but there are many things in the way, be it laziness, lack of time, or just a matter of having low motivation. All these reasons make it awfully hard for people to learn anything. With this in mind, we thought of a simple flash card app to help people (including us) learn about anything.
The application's key features are simple, just like a normal flash card you will have a **question** (or word) and an **answer** (or translation). It is as simple as it can get, but besides that we wanted to make it even easier for the user to use our app by adding many other features. With these features, our main objective is to make people's lives easier by providing them with a fast and effortless way of learning.

2. Project Organization

2.1. Project Lifecycle

The Project will have a life cycle based on the waterfall model because the requirements and scope are well defined, and the project will not be subject to big changes during the development.

2.1. **Software Requirement Specification**, the functional and nonfunctional requirements get defined based on the negotiation with the client.
- 2.1.1. **Software Requirement Specification,** the team creates the first document with the needs of the client.
- 2.1.2. **Risk Plan**, the team must identify the risks that could have a negative impact in the project and will estimate the impact on the development.
- 2.1.3. **Acceptance Test Plan**, during this phase test are designed based on the requirements and standard designed for the project.
- 2.1.4. **Milestone Report,** the team reports how the phase was performed.

2.2. **Software development,** the software gets designed and developed during this phase.
- 2.2.1. **Software Architecture and Design,** the team creates diagrams and fills the document with the detailed architecture of the software.
- 2.2.2. **Source Code,** the main part of the software gets developed, this phase is divided in multiple parts due its complexity.
  - 2.2.2.1. **Development of components**, after every member of the team is assigned with the development of some number of components these are developed individually (when it is possible) to ensure work distribution and efficient development.

2.2.2.2. **Testing of components,** after a component is finished, it must be tested to ensure quality, will be done with Unit Testing.

2.2.2.3. **Integration of components**, after a considerable number of components are finished (developed and tested), we will proceed to integrate them into the software and test it afterwards.

2.2.3. **Milestone report.**

2.3. **Software acceptance,** tests are performed, and conclusions will be obtained during the phase, will be decided if the software meets the requirements.

2.3.1. **Acceptance Test Report,** test cases are created to test the performance and reliability of the features.

2.3.1.1. **Alpha testing,** the team tests the project to find bugs and other more technical errors.

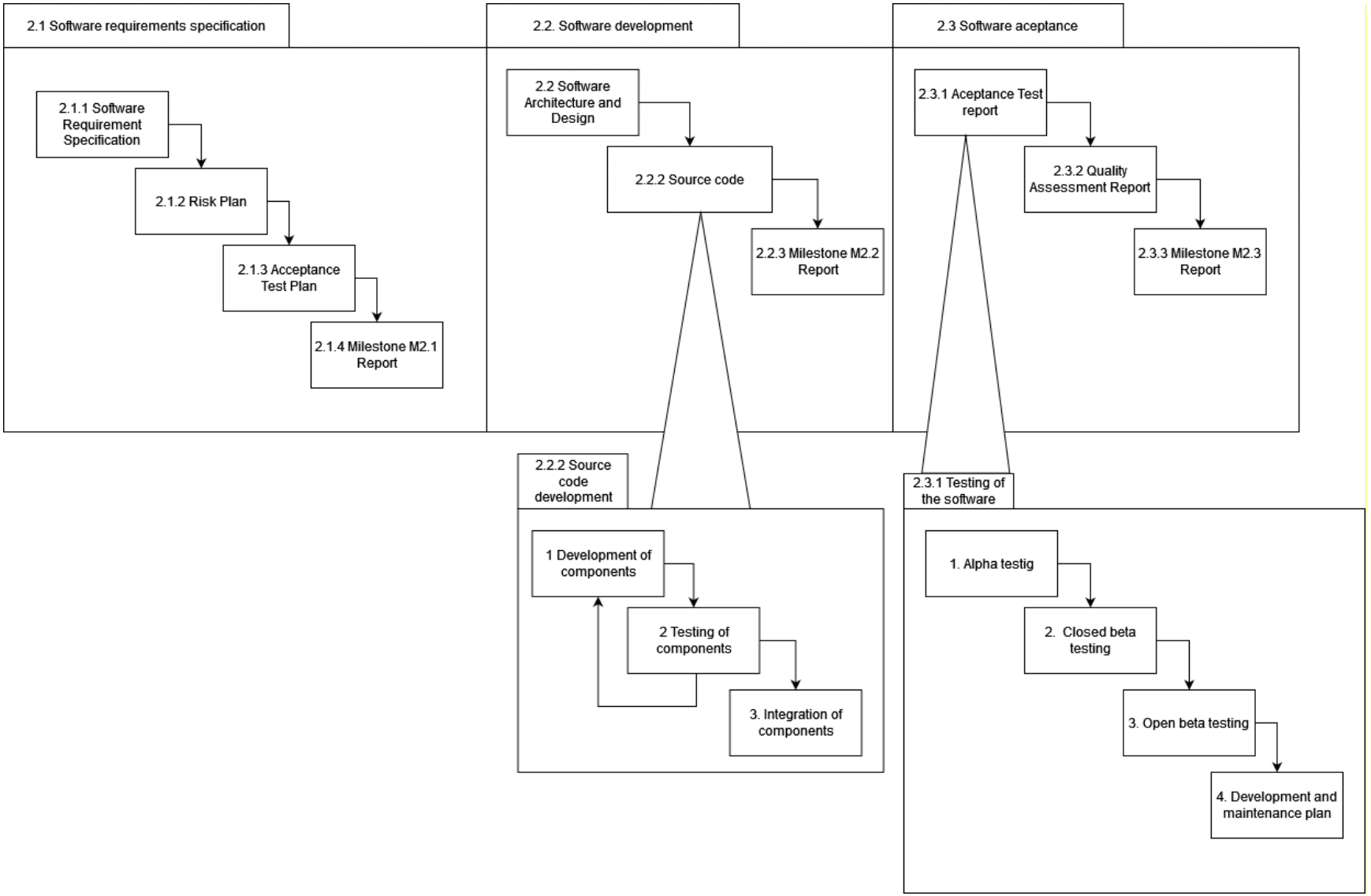
2.3.1.2. **Closed beta testing,** some costumers and stakeholders try the software as users and report errors or suggestions, mainly about the user experience. If the client or stakeholders don't like the product or make reasonable suggestion, the project goes back, totally, or partially to the design phase, depending on the impact of the changes.

2.3.1.3. **Open beta testing,** the software is delivered to the public and are the costumers who report problems in the software. To reach the *open* phase, the software must be approved by the closed beta testers.

2.3.1.4. **Development and maintenance plan,** a plan with the main tasks to keep the project working is created if the software needs modifications of patches in the future.

2.3.2. **Quality Assessment Report,** the team reviews the development of the project to measure, based on the Quality Assurance Plan.

2.3.3. **Milestone Report.**


2.1 Software requirements specification
2.1.1 Software Requirement Specification
2.1.2 Risk Plan
2.1.3 Acceptance Test Plan
2.1.4 Milestone M2.1 Report
2.2. Software development
2.2 Software Architecture and Design
2.2.2 Source code
2.2.3 Milestone M2.2 Report
2.3 Software aceptance
2.3.1 Aceptance Test report
2.3.2 Quality Assessment Report
2.3.3 Milestone M2.3 Report
2.2.2 Source code development
1 Development of components
2 Testing of components
3. Integration of components
2.3.1 Testing of the software
1. Alpha testig
2. Closed beta testing
3. Open beta testing
4. Development and maintenance plan

2.2. Deliverables

2.1.1. Software Requirement Specification ~ 21 Dec
2.1.2. Risk Plan ~ 21 Dec
2.1.3. Acceptance Test Plan ~ 21 Dec
2.1.4. Milestone M2.1 Report ~ 21 Dec

2.2.1. Software Architecture and Design ~ 11 Jan
2.2.2. Source code ~ 11 Jan
2.2.3. Milestone M2.2 Report ~ 11 Jan

2.3.1. Acceptance Test Report ~ 18 Jan
2.3.2. Quality Assessment Report ~ 18 Jan
2.3.3. Milestone M2.3 Report ~ 18 Jan

2.3. Supporting Documents and Records

- User documentation: Basic manual to teach a user how to use the app
- Deliverables listed above
- Support documents found here (team log and baseline plan)
- Meeting minutes found here
- Angular documentation found here:
    - Best practices
    - Coding conventions
- Source code

2.4. Project Management

- Roles:
    - Project manager: Jan Frank.
    - Quality manager: Pedro Henriques.
    - Risk manager: Ângelo Paiva.
    - Technical manager: Álvar San Martin.
    - Client manager: the entire team.
    - Test manager: Burak Tinman.
- Groups:
    - Software design group: responsible for software architecture and design
    - Software production group: responsible for code, reuse component selection and unit test
    - Software requirements group: responsible for the software requirement database

3. Baseline Plan and Control

3.1. Estimate

The team decided to use planning poker to get the estimations. The issues on the planning poker were based on the project lifecycle.
In the end, this project estimation report was filled out with the estimates the team came up with during the planning poker.
The project estimation report consists of the stages, phases and tasks with its subtasks which were estimated accordingly.

3.2. Schedule

The top-level schedule we are following can be found here. Summarizing:

- Requirement's specifications: 2 weeks, 4 hours per member => 40 hours total (30 Nov. – 14 Dec.)
  - 2.1.1 Software Requirement Specification: 17 hours
  - 2.1.2 Risk Plan: 8 hours
  - 2.1.3 Acceptance Test Plan: 11 hours
  - 2.1.4 Milestone M2.1 Report: 4 hours
- Software development: 4 weeks, 4 hours per member => 80 hours total (14 Dec. – 11 Jan.)
  - 2.2.1 Software Architecture & Design: 19 hours
  - 2.2.2 Source Code: 46 hours
  - 2.2.3 User Documentation: 10 hours
  - 2.2.4 Milestone M2.2 Report: 5 hours
- Software acceptance: 1 week, 4 hours per member => 20 hours total (11 Jan. – 18 Jan.)
  - 2.3.1 Acceptance Test Report: 8 hours
  - 2.3.2 Quality Assessment Report: 8 hours
  - 2.3.3 Milestone M2.3 Report: 4 hours

The work breakdown structure (WBS) we are following can be found here.

3.3. Tracking and Control

In the team log (which can be found here) we will keep track of the cost of the project.

4. Technical Process

4.1. System architecture

The system architecture of our software project will apply the ***Model-View-Control*** architecture pattern.
The Model-View-Control architecture pattern divides the code into three logical parts:

Model:

The model is responsible for maintaining and handling the data. It deals with database operations by responding to the controllers' requests, in our case only with the local storage.

- Example: Flashcard model class containing the required properties such as question, answer, tags, etc.

View:

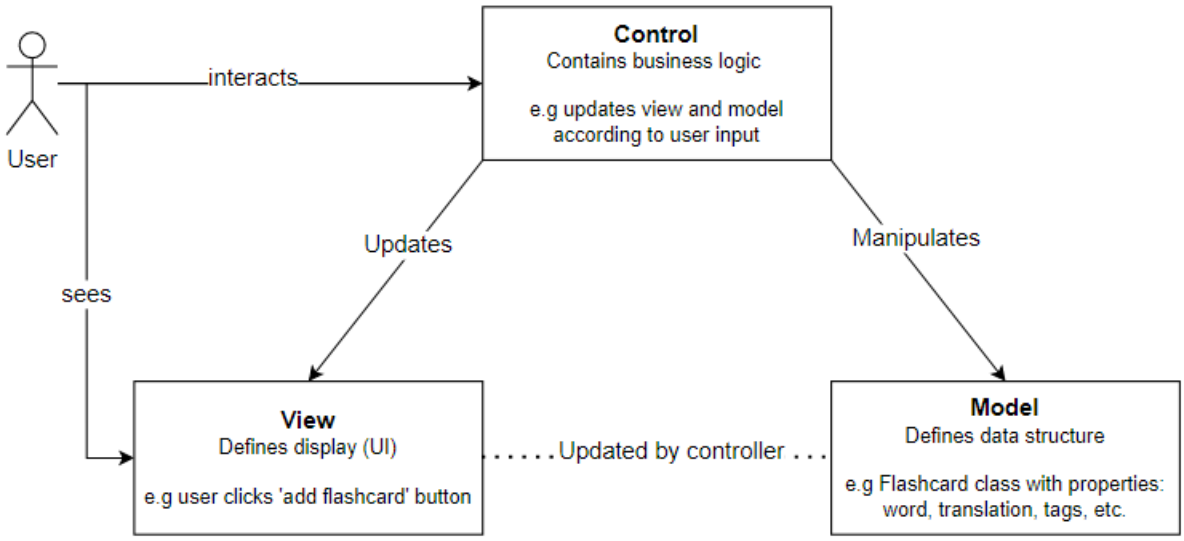
The view is responsible for the representation of the data by providing a user interface for the user. Data is collected by the model component. However, it is not talking to the model directly but through the controller.

- Example: View, which represents all flashcards in a user deck

Controller:

The controller is responsible for the business logic of the application and acts as an intermediary by connecting the model with the view. Model and view cannot talk directly but only through the controller.

- Example: Controller, which is responsible for creating a specific flashcard and adding it to a user deck



4.2. Methods tools and techniques

- Programming language
  - JavaScript with Angular framework (Frontend and backend, version 11)
  - Node.js for package managing
- Tools
  - Visual Studio Code (IDE, version 1.61)
  - JIRA (Kanban board with backlog)
  - GitHub (Version control)
  - NPM to manage the packages of the project

- Methods / Techniques:
    - Continuous Integration with GitHub Actions (Configurable testing of our code)
    - Quality Assurance by doing pull requests and reviews on branches before merging it to main branch
    - Planning poker (Estimation of task and requirements)