Just Your KidieZ

Project Plan

Márcio Paixão Dantas
Felipe Ebert

Date	**:**	**6/18/2023**
Version	**:**	**2**
State	**:**	**Released**
Author	**:**	**Aya Shikh Suliman**

Version history

Version	Date	Author(s)	Changes	State
1	3/2/2023	Aya Shikh Suliman	Start up	Unreleased
2	6/18/2023	Aya Shikh Suliman	Scope Tasks Testing strategy Risks/migration	

Distribution

Version	Date	Receivers
1	03/03/2023	Márcio Paixão Dantas Felipe Ebert
2	6/18/2023	Márcio Paixão Dantas Felipe Ebert

Contents

1. Project assignment

1.1 Context

The company KidieZ is a company that has schools for children between 4 until 14 in several counties in the world. These schools are not the classic schools people know; they are special schools that teach children in a creative way using activities so that it is not boring for them. The owners of the company, Márcio Paixão Dantas, and Felipe Ebert wants also to target babies from 0 until 3.

1.2 Goal of the project

The aim is to help parents raise up their children professionally. For example, with giving them creative and educative game ideas so that their babies do not stay the whole day watching TV. The owners want to do that online so that it is easier for parents to access it. Besides, the Center for Disease Control and Prevention defines the preschool age range as being between three and five years old. So, it is not recommended to put babies between 0 and 3 in preschool but instead they should stay beside their parents and feel their warmth and care. And since the company doesn't use the classic way of teaching in their schools, having it online will also not be the classic preschool way for babies.
After taking all the owners needs into consideration, the ICT team suggests a website as a solution for this problem. Since it easy to access from a phone, laptop, iPad, etc.

1.3 Scope and preconditions

Inside scope:	**Outside scope:**
1 JavaScript and React Frontend	1 Desktop application
2 Java and Spring MVC Backend	2 Physical lessons
3 MySQL Database	

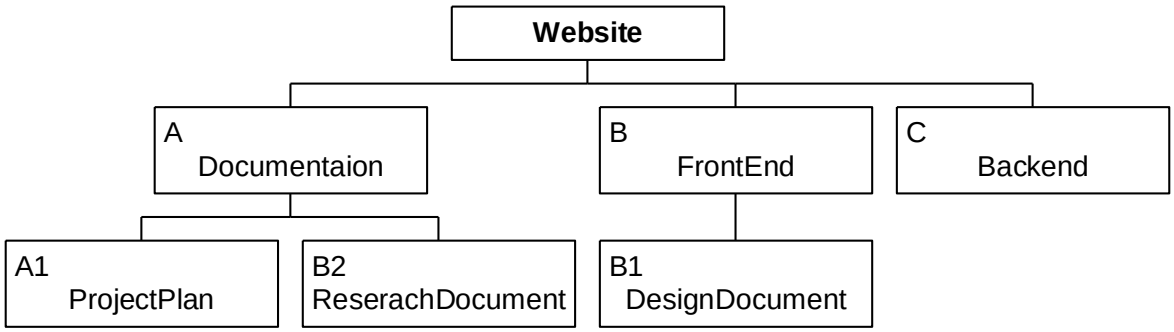
The owners want an online platform, that is why a website is the most suitable solutions for their needs.

1.4 Strategy

For this project, agile is going to be used as a strategy, or more specifically the scrum approach. Other approaches like waterfall do not fit this project because the requirements are not concrete yet. So, it is not possible to decide on all the functionalities and then start implementing everything because with time, and with seeing the work, more requirements are going to be added. For now, the only thing that is known is that it is a website with creative, educative activities/courses for babies between 0 and 3. But for example, what kind of courses and how many courses are there going to be is not known yet.

1.5 End products

Project Breakdown Structure (PBS)



2. Project organisation

2.1 Stakeholders and team members

<<Indicate all stakeholders and team members for your project. For each stakeholder indicate the role for your project. Note that the role that a person has for your project is different from the function the person has. E.g., someone with the function "department manager of department X" can have the role of product owner for your project.

Name	Abbreviation	Role and functions	Availability
Aya Shikh Suliman	*Aya*	*Developer*	*On school days*
Felipe Ebert	*Felipe*	*Company owner*	*On school days*
Marcio Paixao Dantas	*Marcio*	*Companer owner*	*On school days*

2.2 Communication

Every week, 2 small meetings/discussions take place between the developer and the company owners. The goal of these meetings is to talk about what the developer did so far and what he/she is going to do. It is also to confirm some decisions so that the developer can go on.

3. Activities and time plan

3.1 Time plan and milestones

<< For a waterfall project you can indicate the phases and milestones below (can be adapted as required).

For an agile project, describe how the artefacts are planned. E.g., length of sprint (with justification), organization of stand up, demo, retrospective.
>>

Phasing	Effort	Start date	Finish date	Submission
1 Sprint 1	5	6/2/2023	3/3/2023	Projectplan Initial product backlog Backend: first setup CI/CD environment initialization
2 Sprint 2	9	4/3/2023	24/03/2023	Design document version 1 Backend Initial Frontend setup Initial Applied research document
3 Sprint 3	17	25/03/2023	14/04/2023	Design document version 2 Initial Backend to Database setup SonarQube Research document
4 Sprint 4	33	15/04/2023	12/05/2023	Design document version 3 Authentication and authorization implementation Continuous Integration and Sonarqube
5 Sprint 5	65	13/05/2023	2/06/2023	Final design document Websockets feature Minimum viable product (MVP) features implemented Continuous Integration and Sonarqube
6 Sprint 6	129	3/6/2023	21/06/2023	Final UX feedback report Final individual track product with minimum viable product (MVP) features implemented Continuous Integration and Sonarqube Continuous Delivery A web performance review document, Reading guide: what is what in your zip (max 1 A4) All feedback from individual track and group project (as PDF -> download feedpulse) All peer reviews, reflections and other individual things related to the group project All work from your individual project All work that is from your hand and you think is important from the group project

4. Testing strategy and configuration management

4.1 Testing strategy

In programming, testing strategies refer to the approaches and techniques used to verify the correctness and quality of software applications. There are several common testing strategies employed in software development, including:

1. ***Unit Testing:***

- Justification: Unit testing is essential to verify the individual units of code in isolation and ensure their correctness. It helps catch bugs early in the development process and provides a foundation for other testing levels.

- Goals: Aim for a high percentage of code coverage, such as 80% or above, to ensure comprehensive testing of the codebase.

- Automation: Unit tests are highly suitable for automation using unit testing frameworks like Mockito.

2. ***Component Testing:***

- Justification: Component testing focuses on testing individual software components or modules in combination. It ensures that each component functions correctly and integrates properly with others.

- Goals: Ensure that each component is tested thoroughly, covering different input combinations and edge cases.

- Automation: Component tests can be automated using testing frameworks specific to the programming language or framework being used.

3. ***Integration Testing:***

- Justification: Integration testing verifies the interactions and data flow between different components or modules. It helps identify issues related to integration, such as data inconsistencies or communication problems.

- Goals: Ensure that the integrated components work together as expected, validating their collaboration and proper data exchange.

- Automation: Integration tests can be automated using frameworks like Selenium or REST-assured, which simulate interactions between components.

4. ***System Testing:***

- Justification: System testing evaluates the behavior and functionality of the entire system. It ensures that all integrated components function correctly and meet the specified requirements.

- Goals: Validate the system against functional and non-functional requirements, including user interactions, performance, and security aspects.

- Automation: While some aspects of system testing can be automated, such as performance testing using tools like Apache JMeter, certain aspects like user interactions or usability testing may require manual testing.

In terms of quality testing setups, ***SonarQube*** *is a valuable tool that can be used for static code analysis, code quality metrics, and identifying code smells or potential issues in the codebase. It can be integrated into the development pipeline to enforce code quality standards and best practices.*

4.2 Test environment and required resources

In the envisioned test environment, I will consider a typical DTAP (Development, Testing, Acceptance, Production) setup to ensure a structured and controlled software development and testing process. This approach allows for smooth progression of the software application from development through various testing stages before reaching the production environment.

1. ***Development Environment:***
- Purpose: This environment is dedicated to individual developers and small-scale development activities.
- Resources: Developer machines or workstations with development IDEs, version control systems (e.g., Git), and local development databases or servers.

2. ***Testing Environment:***
- Purpose: This environment is dedicated to various testing activities, including unit testing, component testing, integration testing, and system testing.
- Resources: Dedicated servers, virtual machines, or cloud-based testing environments that mirror the production environment as closely as possible. The resources should include hardware, software, and network configurations like the production environment. Testing frameworks and tools (e.g., Mockito) are required for automated testing.

*3. **Acceptance Environment:***
- Purpose: This environment is used for acceptance testing, where the software is tested by end-users or stakeholders to ensure it meets their requirements and expectations.
- Resources: Like the testing environment, this environment should closely resemble the production environment in terms of hardware, software, and configurations. It should have representative test data and collaboration tools for gathering feedback.

*4. **Production Environment:***
- Purpose: This environment is the live production environment where the software application is deployed for actual usage.
- Resources: Production-grade servers, cloud infrastructure (if applicable), databases, load balancers, and necessary security measures (e.g., firewalls and encryption).

To facilitate a smooth and efficient development and testing process, a CI/CD (Continuous Integration/Continuous Deployment) environment can be utilized. This environment automates the build, testing, and deployment processes, enabling frequent integration of code changes, automated testing, and seamless deployment to various environments.

CI/CD environment resources include:
- Version control systems (e.g., Git) for source code management.
- Build tools like Jenkins, GitLab CI/CD for continuous integration and automated builds.
- Testing frameworks and tools (e.g., unit testing frameworks) for automated testing.

4.3 Configuration management

The project approach for version management typically revolves around utilizing a GIT repository as the primary version control system. Here are the key aspects to consider for version management:

*1. **Tooling:***
- Utilize a GIT repository management tool such as GitHub, GitLab to facilitate collaborative development and version control.
- Choose a GIT client (e.g., Git command-line) that best suits the team's preferences and workflows.

*2. **Branching Strategy:***
- Adopt a branching strategy like GitHub.

*3. **Promotion Strategy:***
- Establish a promotion strategy to move code changes from one environment to another (e.g., from development to testing to acceptance and production environments).
- Automated build and deployment processes through a CI/CD pipeline can be employed to ensure smooth promotion of code changes.

*4. **Release Strategy:***
- Define a release strategy for determining when and how software releases are created and deployed.
- This may involve versioning schemes and a release schedule or cadence.

*5. **Baseline Strategy:***
- Establish baselines for specific milestones or stable versions of the software.
- Baselines help capture the state of the codebase at significant points in the project and serve as reference points for future development or troubleshooting.

*6. **Change Request and Problem Report Management:***
- Implement a mechanism for managing change requests and problem reports, such as using an issue tracking system like Jira.
- Establish processes for triaging, prioritizing, and addressing change requests and problem reports through collaboration between developers, testers, and stakeholders.