Project Plan

Vibe Check

Date	**:**	**01/10/2024**
Version	**:**	**0.2**
State	**:**	**Finished**
Author	**:**	**Nuno Dias**

Version history

Version	Date	Author(s)	Changes	State
0.1	04/09/2024	Nuno Dias	First Draft	Finished
0.2	01/10/2024	Nuno Dias	Based on Sprint 1: - Changed Version Numbers - Added Research Questions - Added Testing Strategy - Added Risk Matrix - Removed 1st person - Better algorithm and scope explanation	Finished

Contents

1. Project assignment 4
   1.1 Context 4
   1.2 Goal of the project 4
   1.3 Scope and preconditions 4
   1.4 Strategy 5
   1.5 Research questions and methodology 5
   1.6 End products 6
2. Project organisation 6
   2.1 Communication 8
3. Activities and time plan 9
   3.1 Phases of the project 9
   3.2 Time plan and milestones 9
4. Testing strategy and configuration management 10
   4.1 Testing strategy 10
   4.2 Test environment and required resources 10
   4.3 Configuration management 11
5. Finances and risk 12
   5.1 Risk and mitigation 12

1. Project assignment

1.1 Context

The project consists of an app similar to X (previously known as twitter). The project will encompass a wide range of topics from Web Apps at scale, web sockets, media attachments, content moderation, user moderation, storing large amounts of small data (posts, likes, etc) and a lot more, which is perfect for a learning project.

As previously mentioned this app will be similar to Twitter and open to adding modules, as many as possible with the time available, learning a lot of different technologies in the process. For example, with enough time, some simple games in a style similar to the way Facebook. This would be entirely optional and isolated in its own subsection.

1.2 Goal of the project

The goal of this project is to provide learning opportunities, learn new concepts, technologies and ways to implement certain features. It will also require learning a bit about APIs and isolating project layers (frontend & backend).

Ideally, by the end the project will result in a rudimentary twitter- like website with posts that you can add media to (images and music samples). As a user you will be able to post messages, like posts, save posts and reply to posts. All only while being logged into your account. Optionally it will include some customization options as well as general settings.

Finally, if time permits, a subsection that will host a couple simple games (similar to Facebook games). As well as deploying the project as a docker file on a server.

The advantages of this project are that it covers a lot of different technologies. From data storage of complex objects with tree like structures (posts with various replies & replies to replies), to a user friendly UI, to auto-censoring posts and customizability (settings). It also has a lot of opportunities for exploring new concepts that aren't mandatory and so will not affect the project if implementation fails. These include P2P encryption, OAuth, linking & personalizing profiles, games and more.

1.3 Scope and preconditions

Inside scope:	Outside scope:
1 Website	1 Maintenance
2 Posts, Likes and Replies	2 API Keys for Music Databases
3 Accounts	3
4 Feed Algorithm	4
5 Basic Auto-Post Moderation	5
6 Adding Song samples to posts	6
7 Notifications	7

The Backend must be written in Java with the Spring boot framework.
Front end should be in HTML, CSS and React but the JS framework can be swapped out after deliberation.
Any CSS framework (or none) can be used.

1.4 Strategy

The first approach will be to use the Agile SCRUM methodology and work in sprints. This should allow flexibility and error correction as the project develops.
For the frontend we will stick to JS React, HTML and Tailwind CSS to speed up the page styling. The backend will be done in Java Spring boot.

Although there is a lot to do, there are also learning goals to check off for this semester so the first focus will be on getting the basic "twitter" functionality up and running. This includes accounts, posts, likes, replies and saves. Then implementing some algorithms to censor posts, as well as 1 or 2 feeds, a global one and a personalized one.
Once checked with the "client" that all the requirements are met, Settings, Customization and upgrading the Interface are next. As well as trying to set 1+ APIs for encoding music samples when posting something.
The final step will be exploring the rest of the "Wishlist" by interest since at this point all tasks will be mainly done by self-interest. These include security, encryption, simple browser game hosting (similar to flash).

For planning and documentation Jira will be used to keep track of the backlog, sprints and goals, as well as word and google docs for documentation (Project Plan, URS, Testing, etc). For presentations at the end of sprints PowerPoint if necessary, but the main focus will be on practical demonstrations of the progress and improvements relative to the last sprint.

1.5 Research questions and methodology

1. What is the use of Web sockets in a Social Media app?

- Which parts can be asynchronous?
- How often\When should the content update?
- Best\Popular ways to use\optimize Web sockets?

Importance: By researching and adding as many asynchronous features as possible the app will run a lot more smoothly, possibly faster and more secure. As this is a new topic, researching best practices, methods and where to use these methods is very important.

Research methods: (DOT Framework) Library & Lab – This question will mainly be solved through internet research and testing. What can be done? Can we do it? Etc. When in doubt there is also the possibility of asking the tutors for advice.

2. What makes an interface user friendly?

- Popular designs
- Accessibility options & designs
- What is intuitive to a proficient user? What about a technologically illiterate user? Young users? Old users?
- What is the target demographic?

Importance: As full-stack developers sometimes frontend can be unintuitive or too practical. This research is important to understand what the *users* need and want, not us. Without a friendly interface users won't interact with the backend, even if it provides an amazing service.

Research methods: (DOT Framework) Library & Field & Showroom – This very important subject is solved by exploring what others have done, being inspired, noticing the subtleties and a lot of testing. Show your design and ask how you can make it better. Will real users use it. Finally ask experts for an experts touch. They know tips and tricks that users appreciate but don't notice consciously.

3. API security considerations?

- Where should API keys be stored? And how?
- Should API keys be encrypted?
- How often should they be updated?
- What should be done if they are compromised?

Importance: For developers who have not worked with APIs and API keys before this is a potential security issue and if the system relies on the API for a key feature this could cripple the system. It could also lead to wrongful traffic and potentially higher bills. APIs are sensitive features and should be treated with caution so it is important to know *how* to do so.

Research methods: (DOT Framework) Library & Field & Showroom – Just like any research that starts from 0 the bases will be investigated on the internet. It is important to understand the topic well and look at examples given by others to understand how the system works. Next experimentation is essential to apply those findings to a specific case, this project. Finally, with such a potentially dangerous topic it is better to consult experienced teachers for some feedback, making sure the chosen method is appropriate.

1.6 End products

A. WebApp that includes a website for the frontend, and a backend to run all the logic and processing

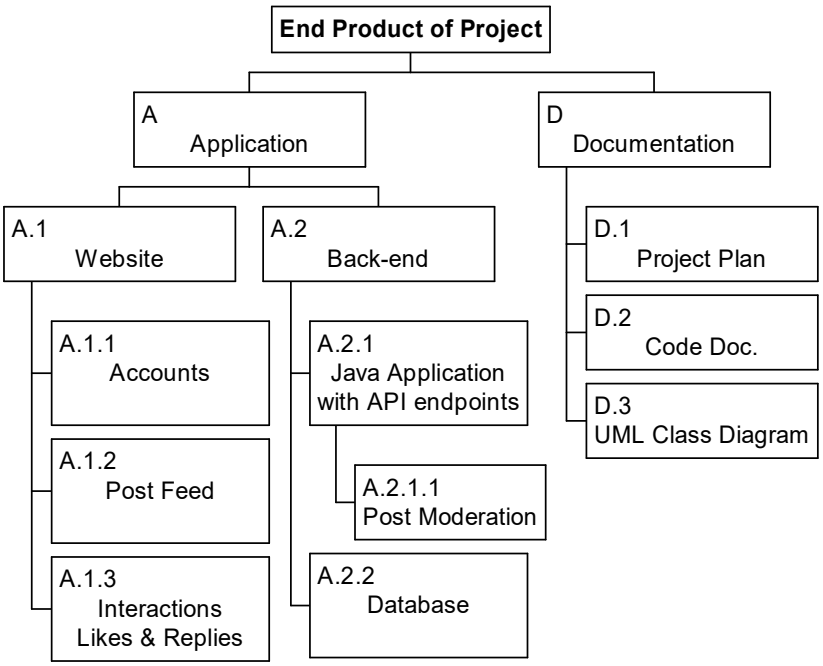
A.1. Front-end for WebApp that is user friendly. Written in HTML, CSS and JS React.
A.1.1. Accounts for user verification, moderation and personalization.
A.1.2. A feed for Users to browse through of recommended posts.
A.1.3. Interaction Actions for Users including liking, replying and saving posts

A.2. Back-end for WebApp that handles logic, the algorithm, data storage and more. Written in Java Spring boot.
A.2.1. Java application that handles all the logic of the program from user actions to data management to moderation algorithms.

A.2.1.1. A moderation algorithm that checks the posts text and possibly images for banned content
A.2.2. Database with all the information the App needs to run. Provides data permanence.

D. Documentation on project, the development process and how it works. Helpful for expansion and maintenance.
D.1. This Document. An overview of the development plan.
D.2. Documentation on how the code works. Includes Code comments and a possible helper document.
D.3. An UML Class Diagram that represents how the Classes, Packages and project works in general from a technical but high point of view. Useful for developers.

Project Breakdown Structure (PBS)



2. Project organisation

2.1 Communication

Ideally there will be 2 meetings a week for feedback on progress and alignment of objectives, so the project doesn't derail. However these will be quick meetings and only brief overviews of progress.
So on top of these there will be end-of-sprint meetings every 3 weeks. These will be longer format, more organized and formal. They will cover all the progress of the past sprint and will be more effective at seeing if the direction the project is going is the right one.

Finally, these will be followed by an internal team meeting that will aim to process and apply the feedback received in the end-of-sprint meeting.

3. Activities and time plan

3.1 Phases of the project

1 Initial Planning

Reading of any documents provided by client and composition of first Project Plan draft. Interview of client asking questions created the in previous steps. Project Plan second draft as well as other documents (Requirements, MOSCOW, other) and more concrete planning for first sprint. Finalize plan by checking with client.

2 Sprints (3 weeks)

Every sprint will start with a short retrospective of the previous sprint followed by planning the next sprint and adding the corresponding tasks to the sprint backlog. Then tasks are mostly determined for the next 3 weeks but they can still be slightly changed depending on feedback.

3 End of Sprint

The last week is dedicated to finishing ongoing tasks, polishing the code a little for presentation, optionally creating some extra material for presentation such as a PowerPoint and a practice run of the demo to avoid crashes while presenting.

4 Final Presentation

Similar to the End of Sprint phase but more expansive. The last sprint will be dedicated to finishing all lose ends, testing and bug fixing to that the end product is stable as well as preparing a presentation going over all deliverables and functions in the end product. This presentation will also require more practice and thought put into it.

5 Wrap Up

After the final presentation all documentation will be updated and products exported into a final package to be delivered to the client.

3.2 Time plan and milestones

Phasing	Start date	Finish date
1 Sprint A	01-09-2024	11-10-2024
2 Sprint B	11-10-2024	08-11-2024
3 Sprint C	08-11-2024	29-11-2024
4 Sprint D	29-11-2024	20-12-2024
5 Sprint E	20-12-2024	17-01-2025

4. Testing strategy and configuration management

4.1 Testing strategy

Testing will happen at all scopes. Unit tests for individual components, Integration tests for full components and finally end-to-end tests for full user features.

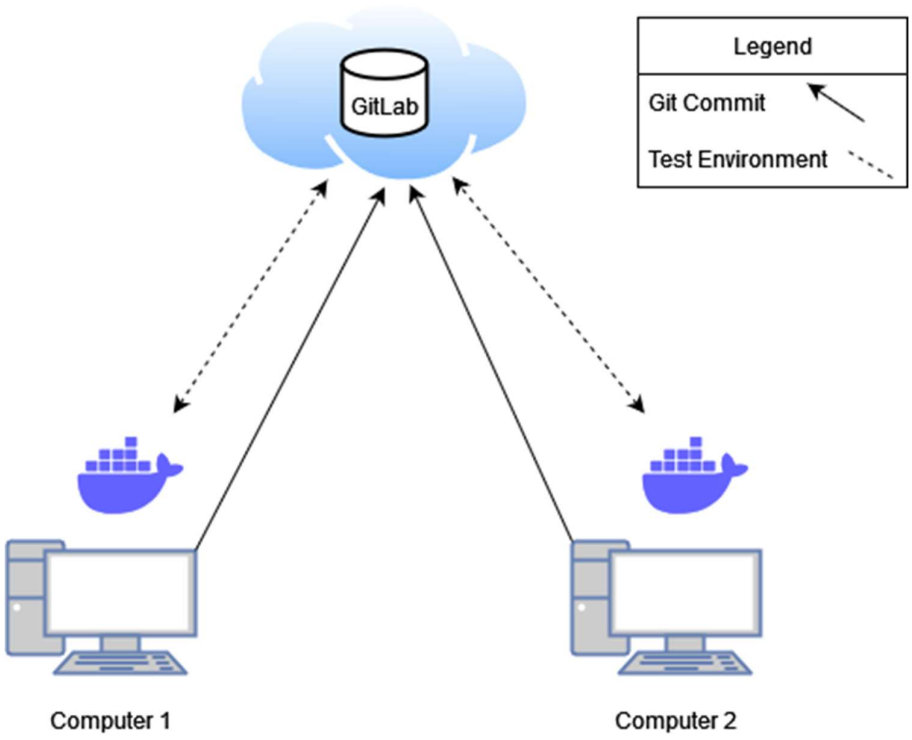
The more specific tests will focus on the Back-end, more specifically the Logic layers (Service, Domain, etc.) and the test with wider scopes will encompass the rest of the layers and systems. Tests should cover all edge cases (null, empty object, maximums & minimums, etc.), positive and negative outcomes.

If the more specific tests reach at least 95% code coverage in the Logic parts of the Application and there are enough end-to-end tests continuous delivery will also be implemented to distribute risk.

4.2 Test environment and required resources

The test environment consists of a CI/CD GitLab pipeline with some GitLab runners running on two different machines. The environment however will be docker containers so they should be clean and identical. The specific docker image for the Gradle build and test steps is the `gradle:8.10.1-jdk17`.

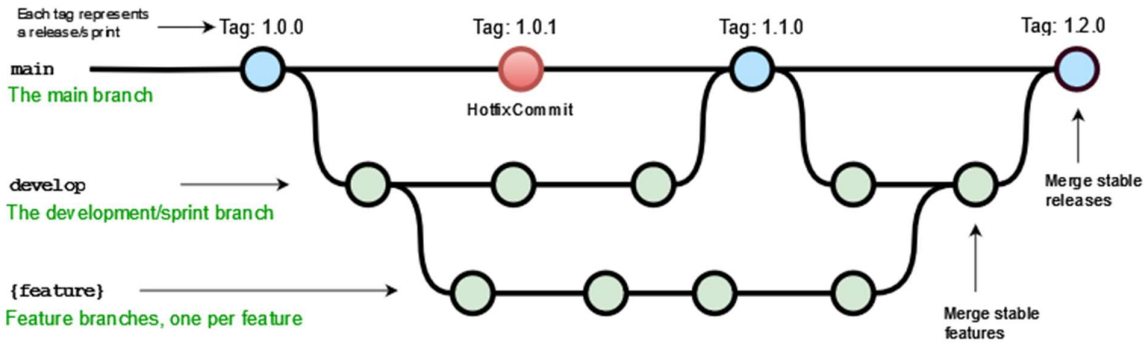
This does introduce the need for one of the two personal computers to be running at test time but the cloud alternative involves additional costs. Since the runners work on docker, they could also be deployed somewhere else at any point rather easily.



4.3 Configuration management

The Git repository will include two branches: development and main (releases). Between sprints most commits will be towards the development branch. However, if a commit happens mid-feature implementation that could break the app, stopping the building or execution stages, it will instead be committed to another branch named after the feature.

Git workflow:



5. Finances and risk

5.1 Risk and mitigation

Risk	Prevention activities	Mitigation activities
1 Loss of contact with Teacher	Frequent updates so time between loss of contact is minimal. Discuss short but also long-term plans with Teacher	Updates over email and with an alternative guide (teacher)
2 Sickness	Add an extra week of work for delays	Online work and redistribution of tasks in planning
3 Data Loss	Version Management with Git and cloud backup with Gitlab on Fontys Servers	Checking for latest code version on other computers and re-merging
4 Expectations different from result	Frequent updates and sprint-end meetings to align/re-align goals and expectations with client	Reorganizing priorities and planning
5 Scope Creep	Frequent internal meetings (retrospective on sprints) to assess the feasibility of the project, timelines and goals MOSCOW & Phasing that prioritizes critical parts of project first	Reducing the scope on less critical parts of the project
6 Time Loss	Phasing that prioritizes critical parts of project first Thorough planning with some breathing room for bug fixing, testing and other problems that can be relied on if needed	Reorganizing priorities and planning

Risk	Likelihood	Damage	Total
1 Loss of contact with Teacher	10	10	100
2 Sickness	10	15	150
3 Data Loss	5	30	150
4 Expectations different from result	10	15	150
5 Scope Creep	15	10	150
6 Time Loss	10	15	150