PPIT Project – Documentation

The Training Plan

Developers: Oskar Grzenda (G00373428) & Kacper Grzenda (G00373427)

Supervisor: Martin Hynes

Website: https://ppit-25a16.web.app

GitHub Repository: https://github.com/kacpergrzenda/PPIT_PROJECT

GitHub Commits: https://github.com/kacpergrzenda/PPIT_PROJECT/commits/main

Project Screencast: https://github.com/kacpergrzenda/PPIT_PROJECT/blob/main/README.md

Contents

Introduction

This project is designed to help a fitness instructor get a better communication with their client. It is designed for clients to post their workout's so that the trainer can keep track of who is completing their daily fitness task's. The user can post a picture of them performing the exercises or send a message onto a chat feed talking about an exercise assigned to them by the fitness instructor.

The fitness instructor has an administrative role where he can access the video page and has permissions to add or delete a video. This part of the website is used so the instructor can post different videos that his clients can follow to help them perform the exercises they were assigned. The page is based on 3 different types of sports such as running, swimming, and weightlifting. The users can change what type of content they can see by using the selection menu on the right-hand side of the video page.

System Requirements

The system is required to work on any device that has access to an internet connection. The system is to be accessed by applications such as (Google, Safari, Firefox).

A user of the website should be able to Sign up for an account by inputting an email address, username, password, and an image of themselves. The email address should be unique, the password and username should have more than 7 characters.

The user should be able to Sign into their own account if they have one created. On Sign in the user should be asked for an existing email and their password. If the user forgets their password, they should be able to change their password by receiving a link to their email asking for change of password.

Users should be able to communicate with other members and Administrator of the website through a chat feed. The chat feed should be able to display strings and images sent by the users under their usernames. A user should also be able to delete a feed message they no longer wish to display by clicking a delete button.

A user should have their own profile page where they input their private details (height, weight, gender). The details can be changed by the user at any time. The details should be private between the user and the administrator of the website.

A user can watch exercise videos on a video page. The user should be able to pick what genre of exercises they wish to watch e.g. (swimming, weight, running).

The administrator should have special access to the videos and at any point of time they should be able to delete or add videos to the database through the website with the click of a button.

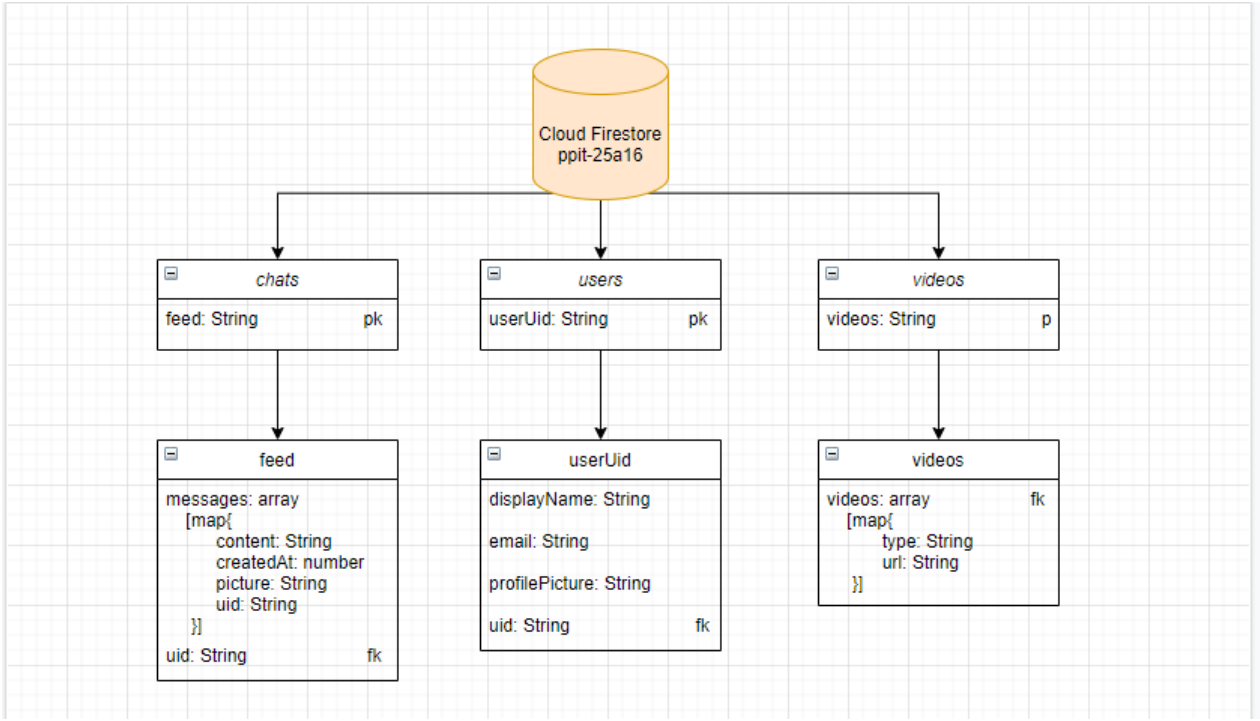
Technology Used and Why

- Angular 11.1.3:
    - After doing some research we found that Angular is a good framework to use for making single page client applications.
    - It also provided us with some nice, modern styling for our website as we had access to the angular material library which helped us with constructing consistent web pages.
- NodeJS 14.15.4: NodeJS
- Firebase:
    - We decided to use firebase as our front-end & back-end database. We went with firebase because it is a cloud-based database, so it was easier to work with. We knew we were going to be using user authentication and firebase decreases the complexity of that so that is why we went with firebase over MYSQL.
- Typescript:
    - We used typescript as the angular already implements typescript into its framework. Typescript allowed us to validate our code that it is working correctly.
- HTML:
    - We also used HTML to help with the structure of our website.
- CSS:
    - We used CSS to help with the styling of our HTML code.
- RxJS:
    - We used RxJS to give the user more of a reactive experience. RxJS provided us with the ability to produce the concept of reactive programming on the web. RxJS is an extension that deals with asynchronous data call. It is also designed to work perfectly with firebase calls which give us an advantage with firebase.

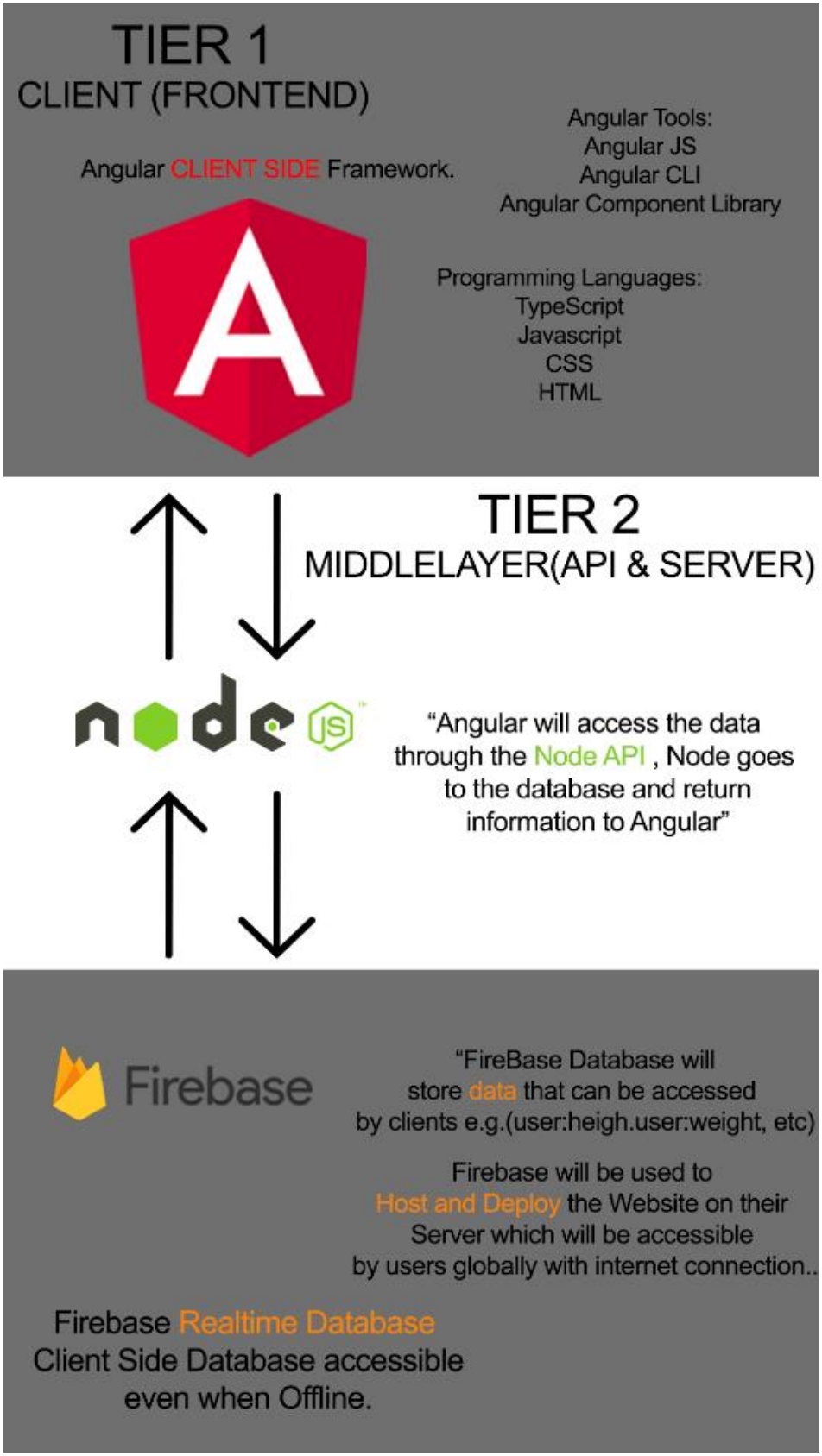
Database Design

The project uses Firebase as the database as it functions well with Angular, and it is a secure way to store user data. Firebase also services encrypt data in transit using HTTPS and logically isolate user data, also each Firestore data object is encrypted under the 256-bit Advanced Encryption Standard and each key is itself encrypted.

The image below showcases the database. The database consists of three collections with each collection having a set of documentation which in some cases holds user data or website data.

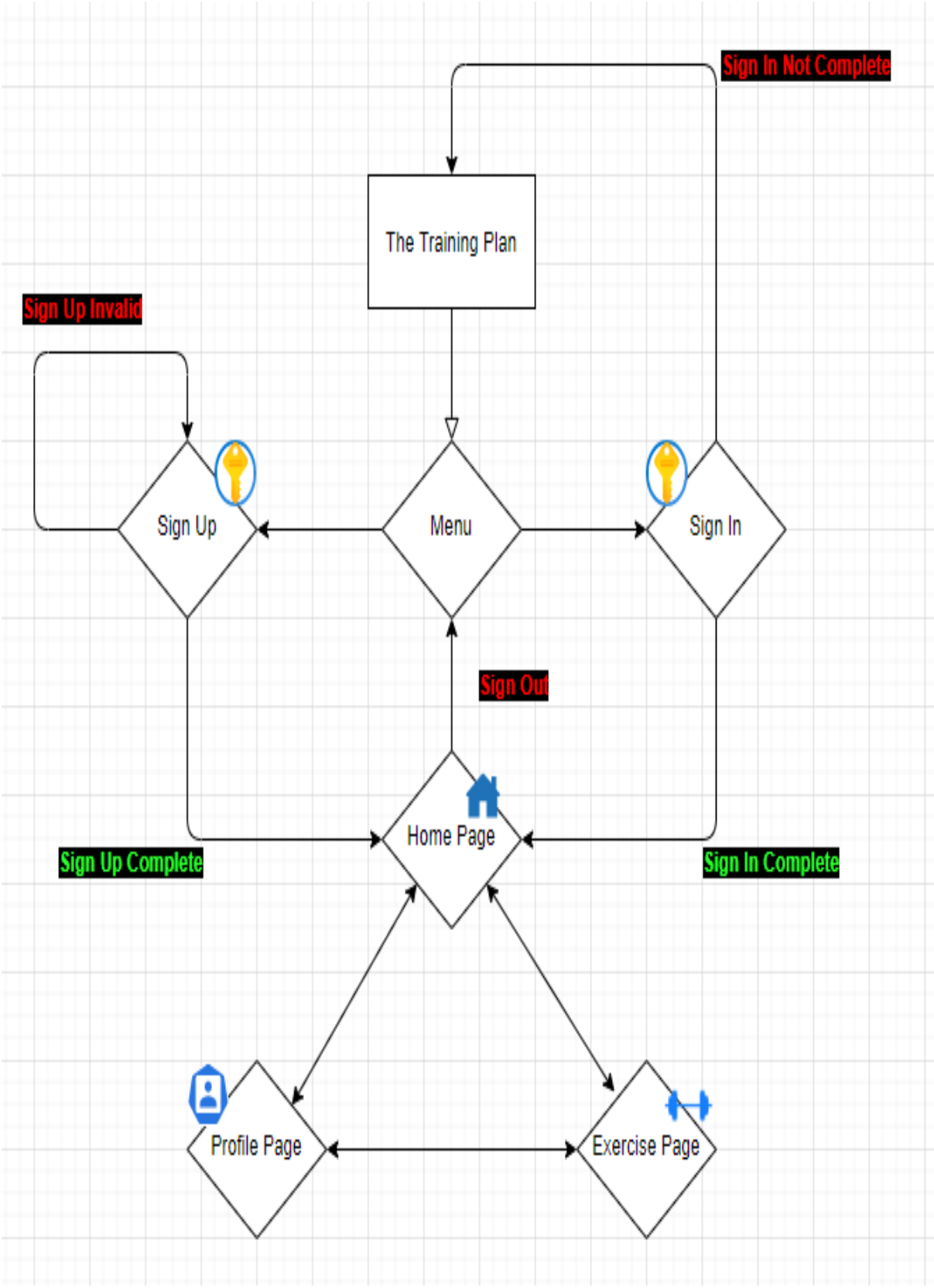


Architecture of the Solution



Web Layout Design

The following images showcases the website map, how users will be able to navigate through out the website.



Design Methodology Applied

- Discovery:
    - When we started this project, we began by researching different ways we could create a website and what would be the best framework to use after some research we decided to go with Angular as it is one of the most popular frameworks for web development. We also have had some experience of using Angular before in 2nd year of our course.
- Planning:
    - Once we figured out our framework, we began planning what we want this website to be and what other technology we would need to develop this website. We knew we would need some sort of database, so we went with firebase. At first, we were going to use MYSQL as our database, but we decided to use firebase as it is a cloud-based database, and it works well with angular to decrease the complexity of user authentication and that was going to be a big part of our website. And after further research we could also use Firebase to Host our website.
- Creative:
    - We would have weekly meetings where we would discuss our plan for the week and what we were planning on doing, the meetings were great as it helped us have a clearer vision of what the project would look like and as a team, we could discuss things and catch up on what everyone has done. The meetings also allowed us to give each other constructive criticism on things that we should maybe do or things that might not be worth doing. This would help a lot with the creative side of the project as everyone on the team could give their own ideas and we would come to a conclusion of what we should do with the project.
- Application:
    - Once we had a plan of what we are going to work on each week we then assigned different roles to each other of who was going to do what task. Must of the time one person worked one day on the project and the other day it was someone else. We used a scrum board on GitHub to track some of the things that we needed to get done.

Software Development Life Cycle

- Planning:
    - We used our weekly meetings to do most of the planning for our project. In the meetings we could discuss what we are planning on doing each week. We also used the Scrum board and Wiki on GitHub for better tracking of the project.
- Requirements:
    - Some main requirements for the project were that users could create their own account and successfully log in and have access to their own profile.
    - User can engage in conversation with other members of the fitness app through a chat.
    - Also, that the fitness instructor would have access to administrator roles and could add or delete videos on the video page.

- Design:
    - The main structure was that the left side would be the navigation bar between pages, the middle column would be the user's feed posts and on the right-hand side we would have specific features relating to the specific page the user was on.
    - At the beginning before we began coding the website, we created a project map so we would have a better understanding of all the technology we are using and how it will all link with each other.
- Build:
    - As we had made a project map, we knew what we were using for the project so when we started coding it was easy to follow what tools and languages, we needed to build the project.
    - The Development of the app started by coding the skeleton of the app so the backend could talk to the frontend and then we began coding the user functionality from there we worked on the design for the website with CSS, Html, and angular material.
- Test:
    - To test the project, we used Selenium's chrome extension. We decided to use this as we had some experience with it in one of our modules in 3rd year of College.
- Deployment / Maintenance:
    - After the product was tested and we found there were no major bugs that would disrupt the user experience we used firebase to host our project on the web.

Features of the Implementation

What we did to keep the project on time and make sure there is consistent work done on the project, we held weekly meetings. This way it made everyone on the team consistently do some work on the project so when we would meet next week there would be some new work done to show to the team. We also used the issues page on GitHub which helped us keep track on what must be done.

We also used a wiki on GitHub to keep track of everything that was done in the week and what the plan was for next week this way we remembered what we had planned to do and it kept everyone on the team accountable to do the work as they said that's what they are going to do and if they didn't do it everyone at the meeting next week would know as there would be no progress on the project from last week.

RxJS Extension was used to keep the chat feed user friendly and reactive so as soon as the subscribed to database was updated the chat feed would be too with the new data. This allows the user to be update date and would have no need to reload the website on every update it would be automatically updated.

Lazy-Loading feature, by default NgModules are eagerly loaded which means that as soon as the app loads, so do all the NgModules, whether they are immediately necessary. Lazy-Loading is a design pattern that loads NgModules as needed. Lazy loading helps keep initial bundle sizes smaller, which in turn helps decrease load times and improve user experience.

Angular Guards allowed us to protect the website by only allowing authenticated users to interact with certain pages and if a not authenticated user tried to navigate to a page on the website, they would be sent to the menu page and asked to login or signup.

YouTube-Player angular API extension allowed us to load YouTube videos and display them quickly for the user to watch this also solved a problem we were having with the "<iframe>" element.

Firebase provided us with our database and website deployment and hosting. After careful consideration we decided to use Firestore (firebase database) over MySQL server-side database because of the helpful feature Firestore provides us with and the compatibility with angular. This provides us with strict user Authentication and user security for the website and user data.

Firebase also give us the advantage by allowing us to deploy and host our website on their servers by linking our angular project to our firebase account through the CLI and by connecting the Firebase API unique key for our project to our Angular Environment. We first had to build the project on our machine, and we deployed it to our firebase account and then hosted it.

Limitations & Known Bugs

- There is no one on one private chat between administrator and user.
- You cannot view other people's profiles.
- Users cannot interact with other user's posts.
- User posts are posted to the bottom of the feed instead of top.
- User can't change username.

Testing Plans

For the project we done unit testing, we used selenium's chrome extension to test different parts of the project as we went along. We made sure to test the log in page so the user can create an account and make sure he can then actually log in once the account is created, and his information is stored on the firebase database.

We also tested the feed section of our project. We tested if the user could post information on the feed without any bugs.

Recommendations for Future Development

Some weeks we planned to much for the week and we did not get it done. I would say just plan less for the week but if you get it done just plan something else you can do that week. Because some weeks we planned to much and we had to move it into next week and then we had other plans already for that week, so a lot of things were just getting on top of each other at some stages throughout the project.

Do more frequent testing as we did not perform a test every week. If we done more testing, we could have potentially found the bugs faster and got them fixed. As we found some small bugs at the end that could have been fixed earlier on if we had tested the website.

Some more frequent meetings for example if you have a meeting on Wednesday and get stuck on a problem on Thursday and you need people on the team to help you try to get in contact before the meeting next week to solve the problem you encountered.

Conclusions

What we learned from this project is how important communication is. If we had not communicated as a team, we could have planned to do the same stuff each week and that would slow down the development of the project. So, when we communicated it helped us properly plan what we will do and what task each person has for the week.

We also learned it is important to properly research what tools/technology you will be using for the project and if they will allow you to perform the things you will need for the project.

We found that it is important to understand and read the documentation for different tools that you use, once you understand the documentation the coding will be made easier.

It is also very important to stay consistent with your project and keep on top off all the work you plan on doing each week. Some weeks you will not get everything done you planned on doing but make sure you do not decide to take a few days off because you done a lot in one week as that could put you off track and you might start to fall behind on your project.

Overall the plan we had for the project from the beginning stayed pretty much the same, as we went along there were some minor changes and ideas that we implemented into the project but the idea of making a training fitness web page stayed the same.