Department of Computing

Bachelor of Science (Hons) in Software Development

PaceRunner – Adaptive Runner's Training App

Final Project Report 2023 – 2024

Student name: Sebastian Firsaev

Student Number: C00263348

Supervisor: Dr Chris Meudec

Date: 19/04/2024

Abstract

The PaceRunner web application aims to create an easy to use and cost-effective platform for runners to enhance their marathon training by focusing on the runner's pace during training and providing motivational messages simulating a digital coach.

Table of Contents

Introduction

This document serves as a comprehensive overview of the development journey of the PaceRunner web application. Aiming to showcase the project's objectives, the technology utilized, and the challenges encountered. It also reflects on the significant learning outcomes, both technical and personal, achieved during the project.

The report is structured to provide clarity on the initial goals, the technological framework employed, and how the project adhered to its specified designs and specifications. Additionally, it offers insights into the achievements, unmet goals, and unexpected accomplishments. Through this report, I aim to share the developmental milestones and the lessons learned, offering a reflective view on the entire process of bringing the PaceRunner application to life.

Description of project

PaceRunner is an adaptive marathon training application, integrated with Strava, a popular fitness tracking website used by runners. Strava takes many of the interaction-fostering features found in social media platforms, and pairs it with activity tracking technology [1]. By acting as a companion app and using Strava's activity tracking, PaceRunner allows runners to enhance their training by focusing on running pace and giving runners feedback on their training. As a web application, PaceRunner can be accessed from any device with internet access and is designed for optimal mobile use.

Screenshots of Web Application

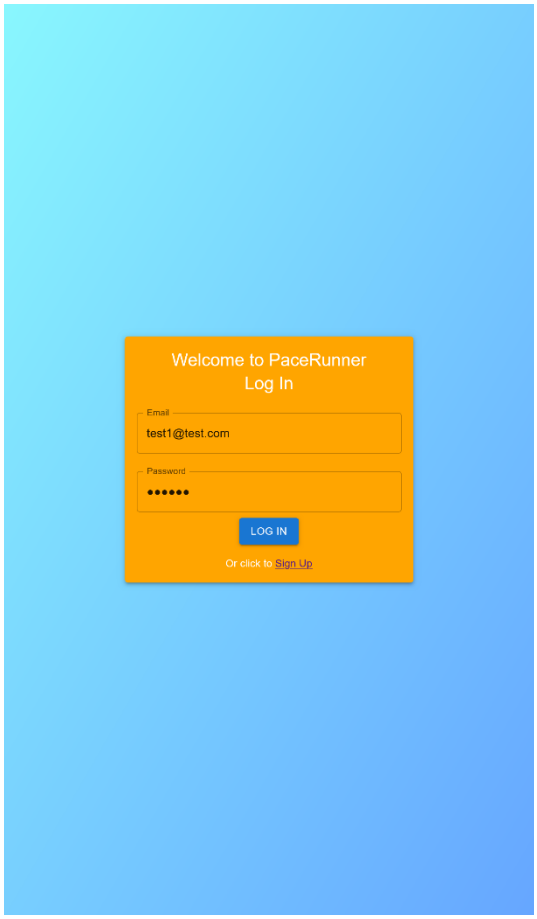


Figure 01. Login Page

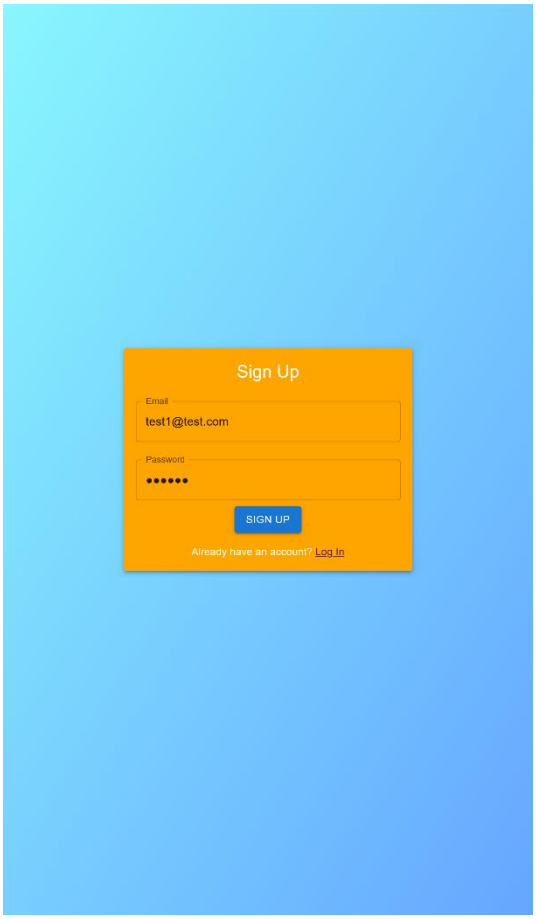


Figure 02. Sign Up Page

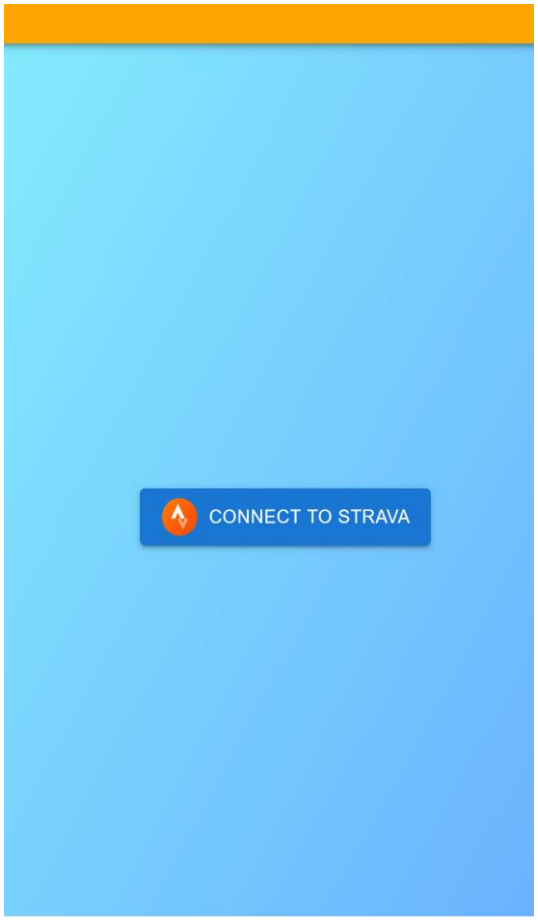


Figure 03. Connect to Strava button

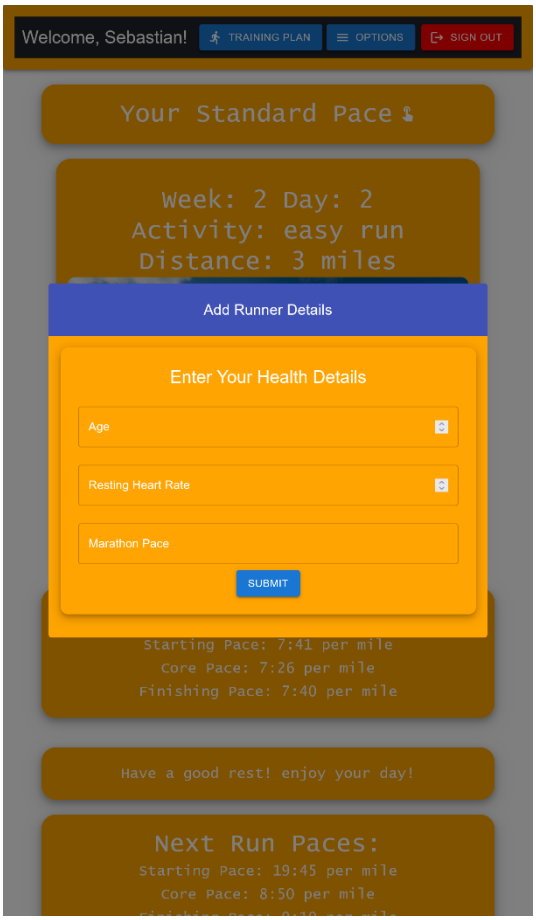


Figure 04. Runner Details Form

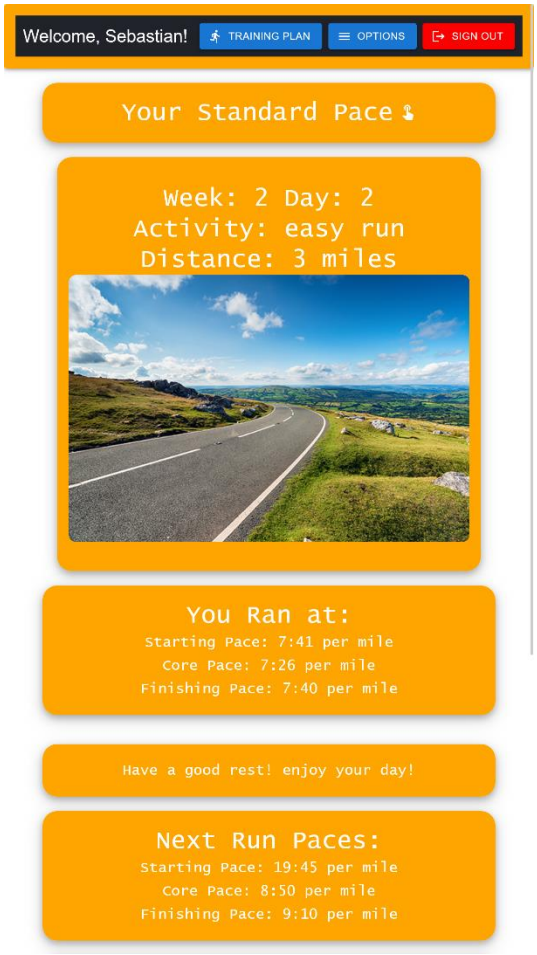


Figure 05. main page

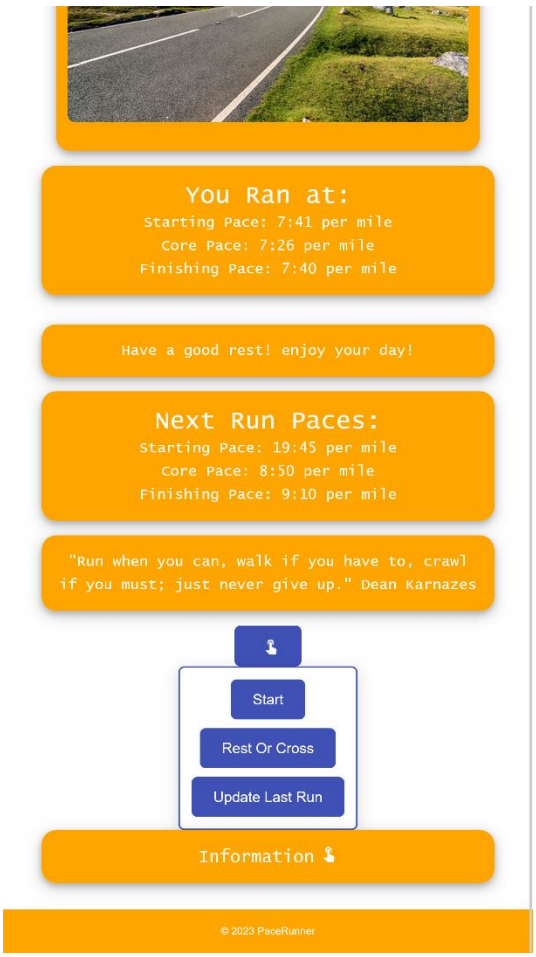


Figure 06. Bottom of Main Page

Figure 07. Training Plan View

Figure 08. Training plan View In Detail

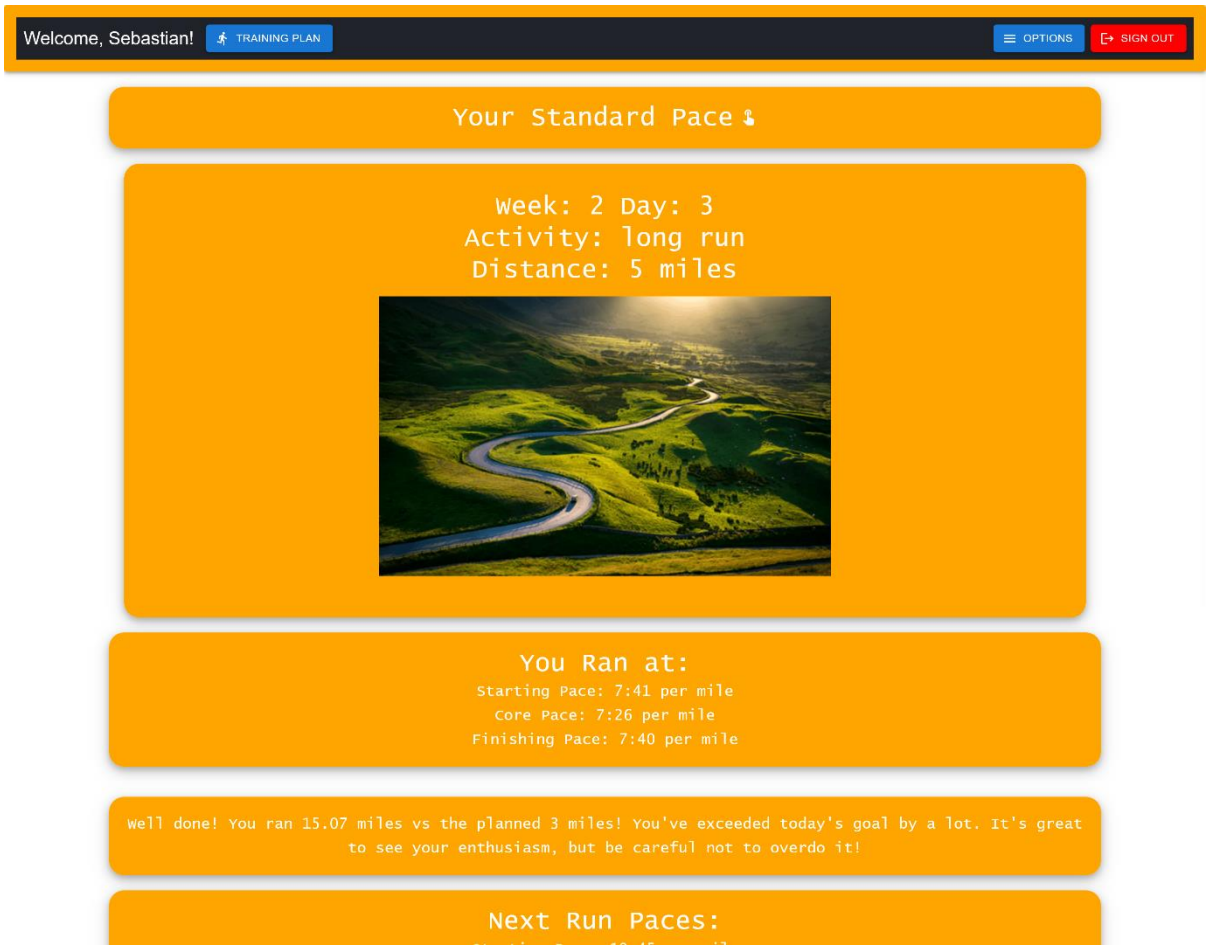


Figure 09. Desktop View of Main Page

Screenshot of Data Structure

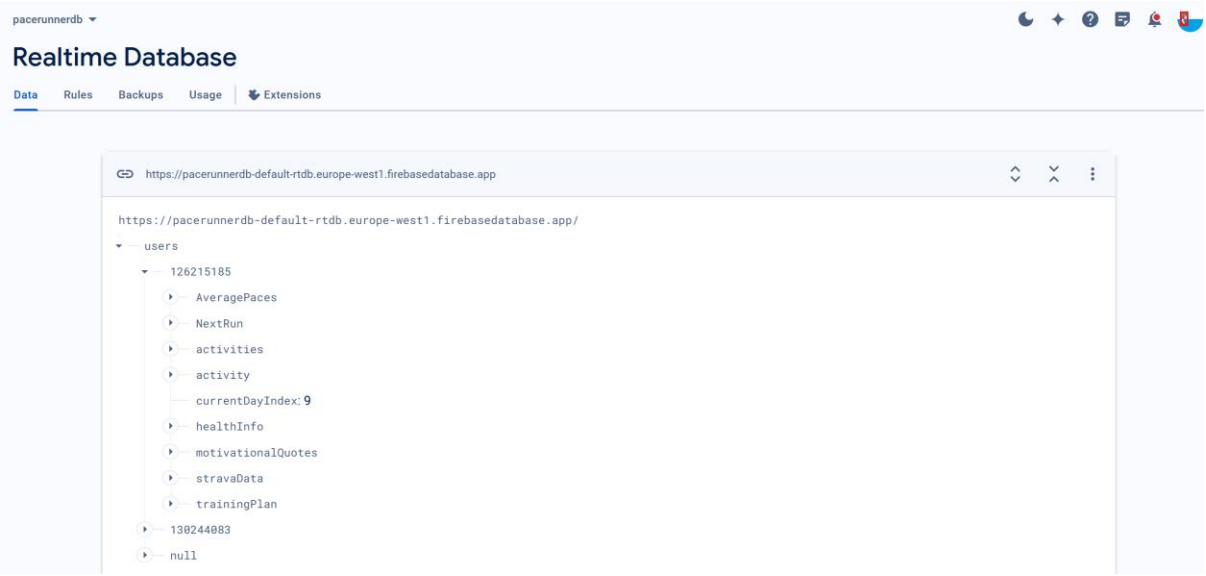


Figure 10. Firebase Data Structure of Set Up Profile

Technologies

Frontend

PaceRunner's frontend is built using React components, an open-source JavaScript library, crafted with by Facebook [2]. React is used in creating reusable HTML code for PaceRunner user interface.

Backend

PaceRunner's backend infrastructure is constructed utilizing Firebase, a robust Backend-as-a-Service (BaaS) solution engineered by Google. This comprehensive platform extends multiple functionalities, encompassing user authentication processes, data storage solutions, and access to real-time databases, thereby facilitating seamless app development and management. Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes [3]. Thereby enhancing the app's performance and scalability, allowing for efficient data synchronization. This choice of technology significantly streamlined the development process, enabling me to focus on delivering a user-friendly experience without the complexities of managing server-side operations.

Conformance to specification and design

Concept Design vs Final Product

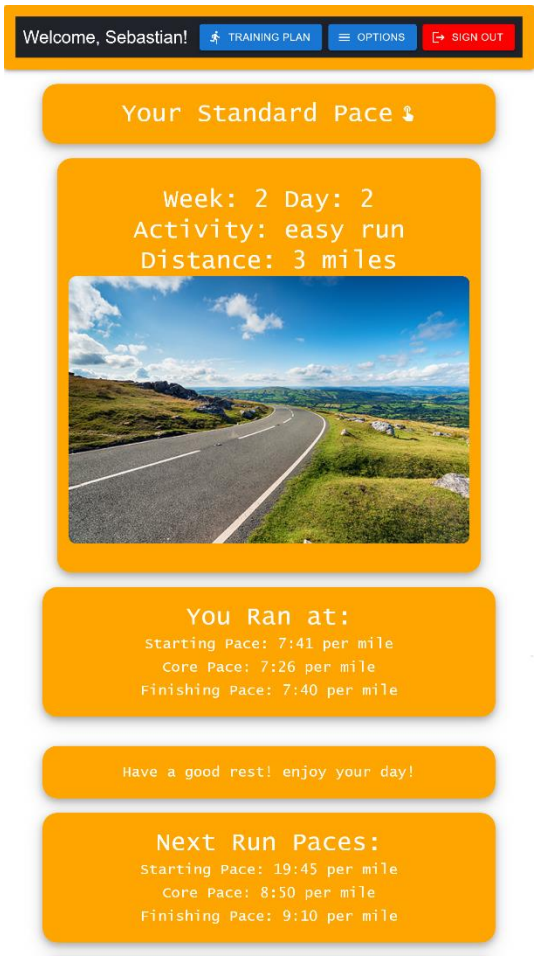


Figure 12. Final Main Page

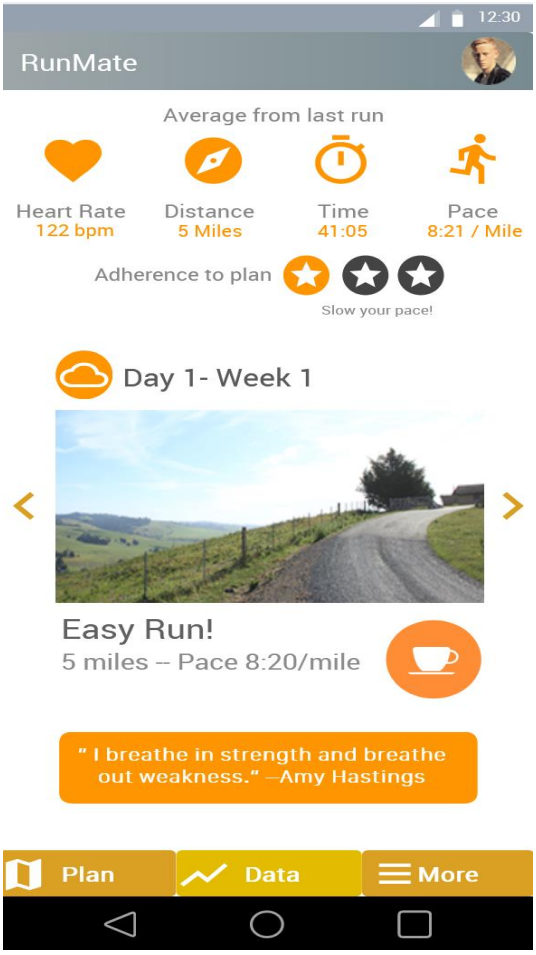


Figure 13. Concept Main Page

The main evolution in the user interface involved prioritizing a clean and accessible user experience while maintaining the app's core functionalities. This involved removing some clutter such as an overuse of icons, favouring a cleaner look. Some components such as the direct access to various metrics at the top of the screen, which were present in the app until iteration 2 were removed following feedback from my supervisor. The logic was that as a companion app to Strava, all this information would already be easily accessible, therefore it was unnecessary to duplicate it. After simulating the webapp on a phone sized touch screen, the final product also embraced a more vibrant, orange-themed interface with larger boxes and more readable information, compared to the more utilitarian layout of the original concept. Making it easier to use on mobile devices.

Personal Reflection

While satisfied overall, I can't say I'm one hundred percent happy with the final design. I found myself relying too much on plain text boxes to display information. This led to a larger amount of scrolling than I originally wanted, however the boxes are still able to convey the information clearly. With extra time, I would have investigated the possibility of making the boxes more moveable, so the user could have some control over the layout.

Objectives Achieved

The primary objectives as outlined in the functional spec were to create mobile a focused web application that adapts training programs based on runner performance and utilizes runner's data such as heart rate and pace.

- This was achieved as PaceRunner successfully uses and analyses a runner's past performance to recommend a new pace for their next training day. The application fetches comprehensive break-downs of runs by lap and compares this to the set standard pace that the runner should run, as well as considering other factors such as heart rate data to set a threshold. PaceRunner also uses an attractive and easy to use on touch-screen UI, allowing for mobile focused use while still being available on desktops.

Another achieved objective was to create motivational messages to motivate runners.

- PaceRunner incorporates several running related motivational messages that refresh randomly, providing motivation to runners.

Core Functional Requirements Achieved

Two core functional requirements were set in the original functional specification: AI-Based Training Plans, and Runner Registration and Profile Creation. Both were also achieved.

- Runners can access a training plan that dynamically changes based on their previous performance. As the pace goals for each training day change based on previous run data, the plan can conform to and grow with the runner as they progress, while also being able to slow down if the runner is under-achieving.
- Runners can create functional user profiles that save and store their progress, run data, age, and pace goals. Runner profiles are secured with login details and are connected to their Strava account.

Not Achieved

Web Application Deployment.

PaceRunner is fully functional in localhost or local python servers, however I was unable to deploy the web application online. I discuss this in detail in the challenges section.

Enhanced AI-based features

I was unable to implement a fully "smart" AI-based pace recommendation and feedback feature. While being able to analyse past run activities and make pace adjustments, PaceRunner is not fully smart.

Comprehensive feedback system

While PaceRunner does have basic feedback. I was unable to incorporate a comprehensive feedback system that analysed a runner's full performance. This would have provided better understanding of progress to runners and would have aided in PaceRunner's value proposition as a cost-effective alternative to a running coach.

Adherence to Metrics

PaceRunner fully adheres to the success metrices set out in the functional specification. Fully meeting the following criteria:

- Functioning base app structure including runner account creation, logging in and accessing training plan
- Functioning Strava integration that allows the app to access runner's training data such as pace and heart rate.
- Training programme being able to adjust itself based on runner's past performance.

Usability Metrics:

"Runners must be able to access and see their training plan within 2 seconds of opening the app."

- If the runner is logged in, they are able to access a training plan within 2 seconds by clicking the training plan button on the header.

"Runners should be able to set up the app within 10 seconds."

- Runners can set up the app within 10 seconds by clicking start followed by setup.

Performance Metrics:

“PaceRunner should be fast and responsive, with average response times being less than 2 seconds,”

- According to Google Chrome lighthouse (webpage performance measure tool), PaceRunner has a Speed Index of 1.6s.

Learning process and outcomes

Technical

Over the course of developing this project, I gained better understanding and proficiency in several technologies and programming practices. With no in depth prior experience in some areas, the development process required not only grasping new concepts, but also applying them practically to meet the project’s objectives. This process was both challenging and rewarding, providing a hands-on experience that significantly enhanced my technical skills and problem-solving capabilities.

STRAVA API

The use of the STRAVA API was a cornerstone of the project. This was my first attempt integrating with extensive API services. I had some issues early on in development with failed attempts to extract the exact data I needed. At the time I was only able to extract basic averages, however, to achieve its full functionality PaceRunner needed more compressive activity data. This initial set back caused me to thoroughly study the API documentation and experiment with various API calls. As a result, I gradually became proficient in leveraging the Strava API to successfully fetch and utilize athletic data for PaceRunner’s functions. This experience showcased the importance of clear documentation and the practical aspects of applying theoretical knowledge to solve real-world application problems.

JavaScript & React

My personal knowledge of JavaScript was greatly enhanced over the course of development. As the main programming language used, JavaScript played a crucial role, especially in leveraging React's capabilities to manage state and lifecycle features effectively. PaceRunner includes state management, where I used React hooks such as useState and useEffect to handle user data and UI state across the application. The useState hook was pivotal for tracking values like user paces, training data, and dialog visibility states, enabling a dynamic and responsive user experience.

JavaScript's asynchronous capabilities were also extensively utilized, particularly through async/await syntax in functions like fetchStravaInfo and loadNextRunPaces, ensuring that PaceRunner handled network requests efficiently.

Firebase

Before starting this project, I had no experience with Firebase or any Backend-as-a-Service platforms. The decision to use Firebase was driven by its scalability, real-time data handling, and authentication features. Learning to navigate its documentation and integrate its services into PaceRunner was initially challenging. However, it became one of the most valuable skills I acquired during the project. I learned not only about data storage and retrieval, but also about implementing secure user authentication and authorization from the ground up.

Git

While using git frequently was a major challenge during the development process. Developing Git skills was another critical learning outcome of this project. Prior to this, version control was a concept I knew of in theory but had little full project development experience with, as I mostly used git for small class assignments. Working on this project, I learned to appreciate the value of Git in tracking changes, collaborating with my supervisor, and managing code across different stages of development. Mastering Git commands and resolving merge conflicts became an easier task as the project went on.

Personal learning

This project has been a journey of personal growth and learning, extending beyond the technical skills into time management, organization, cooperation, and self-reflection. One of the most significant personal achievements was learning to navigate the complexities of project development while balancing academic responsibilities. This experience taught me the value of resilience, the importance of continuous learning, and most importantly the need to remain adaptable and open to new ideas and methods.

Time Management

Time management emerged as one of my most challenging areas, particularly in the initial stages of the project. The underestimation of the workload and a late start in programming led to a significant catch-up effort, especially during holiday periods. This experience was a wake-up call, highlighting the importance of effective planning and realistic time allocation for tasks. Over time, I developed strategies such as breaking down tasks into smaller, manageable goals and using tools such as google calendar to set development reminders and track progress, which gradually improved my ability to manage time efficiently.

Organisation

The project also served as an invaluable lesson in organization. Initially, managing code and development files was overwhelming, given the project's scale and my propensity

to work in a “no structure” manner. However, the necessity to keep track of multiple versions of the project and collaborate effectively forced me to adopt a more organized approach. I learned to structure my workspace, categorize files logically, and document using a diary, which not only made the development process smoother but also enhanced the quality of the final output.

Development Process

The development process was a journey of learning to balance ambition with practicality. Early on, I faced hurdles in gauging the scope of work, which impacted project timelines. However, these challenges were invaluable learning opportunities that taught me about the importance of iterative development, early testing, and incorporating feedback. Adjusting my approach to plan more effectively, set achievable milestones, and adapt to feedback became crucial steps that significantly improved the project's development process.

Cooperation

Cooperation with my supervisor presented a unique set of challenges. Accustomed to working independently, I found it difficult to use collaborative technologies such as Google Docs, accept some guidance, and incorporate some advice, occasionally viewing it more as a critique than a catalyst for improvement. This experience taught me the importance of effective communication and the value of external perspectives in refining ideas and overcoming blind spots. As the advice turned out to be invaluable and critical to the success of the project. Learning to view cooperation as an opportunity for growth rather than a constraint was a pivotal shift in my approach to collaboration.

Review of project

Difficulties and Challenges

Major Challenge: Deployment of the Website

A major challenge that I was unable to overcome is the deployment of the website. A major reason for not being able to overcome this issue was time, as I left deployment to the very last stage. Even though this contradicts agile development practices, the idea behind this was to build the project fully while hosting on a local server and deploy it once fully finished. This was my plan because I didn’t want to create extra issues by deploying immediately and then dealing incompatibly errors or breaking things as I added new features. I wanted to deploy a complete product. I had full confidence in this process as all tutorials and information I researched beforehand re-assured me that deploying a React app was a relatively simple process and there was a large amount of

free hosting services available. By the end, I came to realise that this was not the case. The main issue across all deployment attempts is the build is created and the site deployed but nothing is rendered.

The following are my deployment attempts:

Vercel was first choice for deployment. Vercel makes it as easy as possible to deploy and use your frontend [4]. A cloud platform that enables developers to host and deploy their websites and web applications with zero configuration, automatic scaling, and high performance. With Vercel I attempted both automatic building and CLI building.

The errors encountered were:

MIME type ("text/plain") mismatch.

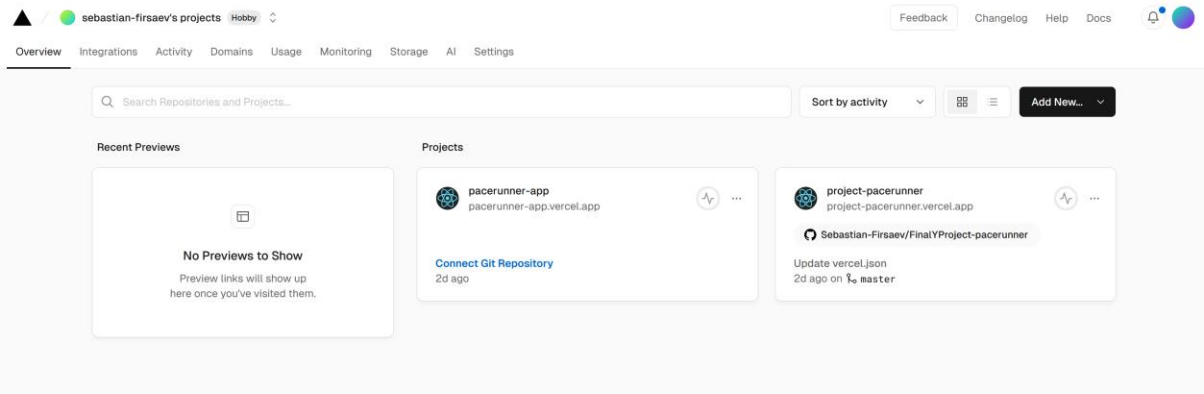
This was fixed by creating a vercel.json file that defined the MIME type for specific file extensions.

React dependency incompatibilities.

This was fixed by editing the package.json file to create direct dependency management and explicitly define react versions to avoid conflict.

Loading failed for the <script> with source "static/js/main.9877c8c0.js".

The last error was common across all deployment services and to be impossible to solve. It seems to be unable to local varies script files, however these files and file paths were confirmed to exist both on github and locally. I used both relative and absolute file paths to try to resolve this error but nothing worked. This issue was also not present when I deployed to a local python server.

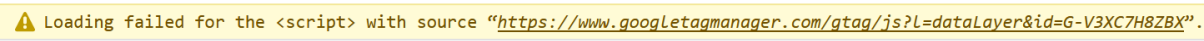


Github Pages was a logical choice as the entire project was already stored there. GitHub Pages is a site hosting service that takes HTML, CSS, and JavaScript files straight from a repository on GitHub, runs the files through a build process, and publishes a website [5]. In Github pages I ran into the same error as vercel:

Loading failed for the <script> with source "static/js/main.9877c8c0.js".

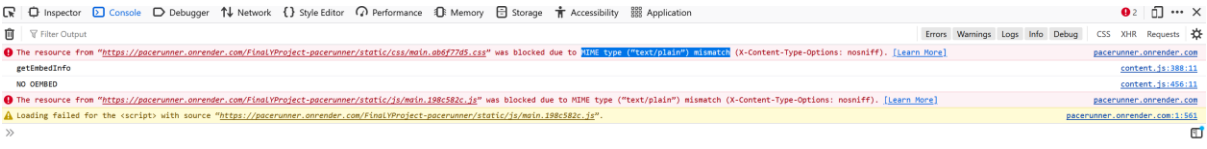
This time I was able to revolve the path issues because I had direct access to the build's index.html were I could manipulate the paths directly. But after fixing this, I ran into the next error:

Loading failed for the <script> with source "".



I was unable to resolve this issue as I have no access to Google Tag Manager. This issue persisted on all browsers used such as Google Chrome, Firefox, Microsoft Edge.

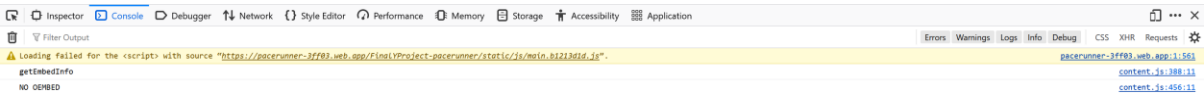
Render is another popular deployment service, it allows users to easily deploy code from existing GitHub repositories [6]. Here I was confronted with both MIME type and Loading fail errors:



As I have no access to the build files on Render, I was unable to resolve the MIME issues directly and still had no solution for the path issues.

On Netlify, the build failed due to dependency errors.

Lastly, I attempted to deploy with Firebase itself. Firebase Hosting is a fully-managed hosting service for static and dynamic content, as well as microservices [7]. The same error persisted as before, and I had no access to build files to edit file paths.



Difficulty: Getting started

Despite best practices of agile development, I started programming late into iteration 1. The main cause of this was spending more time that I should have setting up the connection to firebase/ Strava and getting started with React's structure. The late start put extra pressure on development closer to deadlines.

Difficulty: Infinite Loop Error

During Iteration 2 I ran into an infinite loop that was triggered by a navigation side effect within the React component lifecycle. This unexpected behaviour resulted in the application continuously reloading the home route ('/') and crashing. This was caused by an oversite while coding. I had coded multiple new functions while the web app was running in the background, I did check if the new functions caused any errors, but I didn't note any state changes. The error only became known after some time when I had to restart and log back into the application. As soon as the app landed on the Plan.js page it crashed.

The core of the problem was located within a useEffect hook in one of my plan.js main components. This hook was intended to redirect users to the home page if no user ID was found in the local storage, indicating that the user was not logged in or the session had expired. T logic was flawed because navigating to '/' did not change the state or outcome of the user ID check. Therefore, every time the component mounted, it found no user ID and triggered another navigation to '/', creating an infinite loop.

It took days to first find and the resolve the issue through trial and error. Eventually, to break the cycle, I implemented a guard clause that checks if the current route already is the home route before performing the navigation.

Reflecting on this, I realised that this issue could have avoided or been resolved a lot easier if a followed agile best practices more strictly. The situation was made worse because I neglected git commits and created multiple new functions and files without pushing. As a result, I was unable to go back a stable version without losing a lot of work.

Advice for Student Developers

To fellow student developers attempting similar projects:

- The earlier, the better - Ideally it would be best to have an idea for a project before formally starting the process, allow your passion to drive the project forward.

- Have a plan - Begin your development process as soon as possible to allow ample time for learning, experimentation, and revision.
- Embrace feedback - Openly seek and incorporate feedback from supervisors and peers; their insights can significantly refine your project and learning experience.
- Be realistic - invest time in planning your project, breaking down tasks into manageable pieces, and setting achievable deadlines for yourself.
- Procrastination is your enemy - make sure to set appropriate time aside to work on the project regularly and stick to it.

What if I Could Start Again

Reflecting on the development process, if I had the opportunity to start the project again, several aspects would be approached differently.

- Firstly, I would take my own advice, and start the development process earlier. I spent a large amount of time at the start researching and planning rather than creating code. I had very little code written by Iteration 1, which made it more difficult to get a good start as I was under pressure from other modules at the end of the semester. Starting early would have given me more room to manoeuvre and would have relieved a lot of pressure closer to deadlines.

- Secondly, I would follow the principles of agile development more closely. Particularly, having a “potentially Shippable product” at the end of every iteration from the start. This would have mitigated a lot of difficulties faced later in the project as they would have been dealt with at an early stage.

- I would incorporate more regular check-ins with my supervisor to better align with the project's vision and requirements. Especially at the start I neglected a lot of feedback from my supervisor, this was a mistake. Embracing feedback more openly and viewing it as a valuable part of the learning process would have allowed me to see flaws in the project more clearly.

- Finally. I would host the web app much earlier in the development process. As described in the “difficulties faced” section. Hosting and deploying turned out to be a much more difficult task than I anticipated, leaving me with little time to correct the errors.

If I Had an Extra Month

With an additional month of development time, I would make further attempts to publish the app. I would look into rebuilding the application with React Native and attempt to deploy it as a mobile application. I briefly looked into this while trying to deploy and it seems very possible to convert from React to React Native with extra time. I would also work on expanding the capabilities of the pace recommendation system to make it “smarter”.

Review of Technologies

The selection of technologies for this project, including Firebase, the Strava API, JavaScript, and React, played a pivotal role in its development and learning outcomes. Firebase offered a robust and flexible backend solution, though its steep learning curve was initially challenging. The STRAVA API, while complex, was integral in providing real-time data and enhancing the application's functionality. JavaScript's versatility facilitated dynamic content creation. React transformed the frontend development experience, offering a modular and efficient approach to UI design, though it demanded a considerable investment of time to learn effectively. Overall, I am happy with my chosen technologies, as each one contributed uniquely to the project's success and my personal growth as a developer.

Conclusion

In conclusion, working on the PaceRunner app was a rewarding experience, despite some challenges along the way. While the final product didn’t include all the features I initially envisioned, the parts that were developed worked well, providing a solid foundation in both front-end and back-end technologies. As the primary goal was to create an adaptive training app for runners that integrated with Strava. I'm satisfied to conclude that the core functionalities of PaceRunner were implemented. This project was not only about coding but also about solving problems as they emerged, which was a valuable part of the learning process.

Reflecting on the entire process, I appreciate the successes and the obstacles alike, as each was a learning opportunity that contributed to my growth as a developer. By learning from my mistakes, this project has prepared me to tackle future challenges with confidence and gave me a better understanding of effective software development practices.

Acknowledgements

I would like to extend my heartfelt thanks to Dr. Chris Meudec for his exceptional guidance throughout my project. His support was instrumental in getting me through the challenging periods of the past two semesters. His availability and flexibility in scheduling meetings greatly alleviated the stress and allowed me to better manage my workload. Chris consistently provided constructive feedback and practical advice that was crucial for keeping the project on track.

I also owe a big thank you to all my lecturers at SETU for providing me with the knowledge and skills needed to tackle this project.

Declaration

- I declare that all material in this submission, e.g. thesis/essay/project/assignment, is entirely my own work except where duly acknowledged.
- I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams, or other material, including software and other electronic media in which intellectual property rights may reside.
- I have provided a complete bibliography of all works and sources used in the preparation of this submission.
- I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: Sebastian Firsaev

Student Number: C00263348

Student Signature: S Firsaev

References

[1] Meschke, J., 2021. How to Start Using Strava. Runner's World. [online] Available at: https://www.runnersworld.com/beginner/g25619156/what-is-strava.

[2] Herbert, D. (2023). React.js: What It Is and How It Works. HubSpot Blog. Available at: https://blog.hubspot.com/website/react-js.

[3] Google. (2023). Firebase Realtime Database Documentation. [Online]. Available at: https://firebase.google.com/docs/database.

[4] Gage, J., (2023). What does Vercel do? Technically. [online] Vercel. Available at: https://vercel.com/blog/what-is-vercel.

[5] GitHub, (2024). About GitHub Pages. [online] Available at: https://docs.github.com/en/pages/getting-started-with-github-pages/about-github-pages.

[6] The Full-Stack Blog, 2024. Render Deployment Guide. [online] Available at: https://coding-boot-camp.github.io/full-stack/render/render-deployment-guide

[7] Firebase, 2024. What can you do with Firebase Hosting? [online] Available at: https://firebase.google.com/docs/hosting/use-cases.