Department of Computing

Bachelor of Science (Hons) in Software Development

PaceRunner – Adaptive Runner's Training App

Functional Specification 2023 – 2024

Student name: Sebastian Firsaev

Student Number: C00263348

Supervisor: Dr Chris Meudec

Date: 30/10/2023

Table of Contents

Introduction

PaceRunner an adaptive Runner's Training App is a fitness application designed to provide customized running training programs that adapt dynamically to each runner's unique profile and training updates. This functional specification document outlines the core principles, objectives, and features of this innovative web app.

Project Overview

Running marathons and running in general is a popular fitness goal, but traditional training programs are often static and generic. This app aims to bridge this gap by leveraging personalization and data from popular training platform "Strava". Strava takes many of the interaction-fostering features found in social media platforms such as Instagram, and pairs it with activity tracking technology, such as those offered by MapMyRun [4].

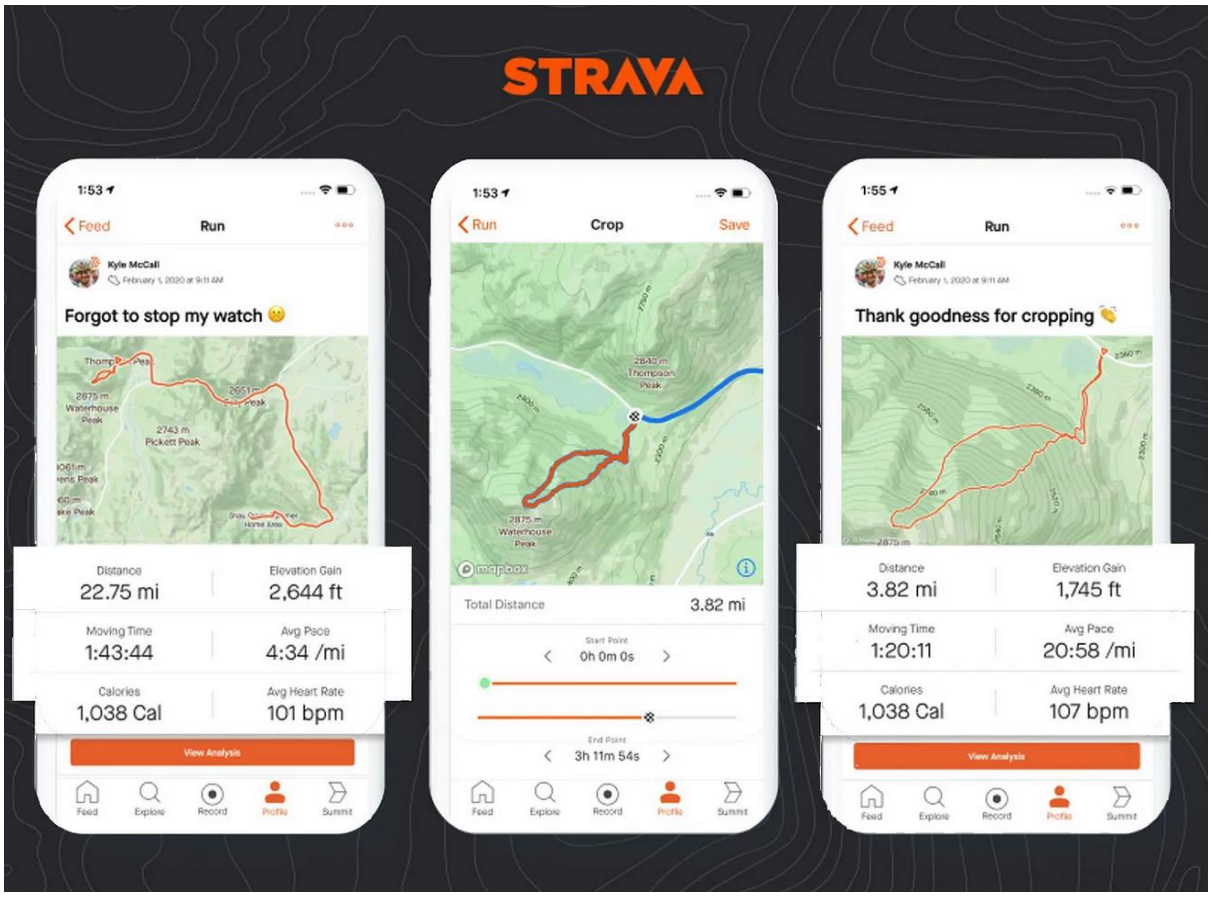

Figure 1, Strava,Source: Strava.com

PaceRunner will offer runners a cost-effective alternative to hiring expensive personal trainers while delivering tailored training programs that evolve as the runner progresses.

The Business Model Canvas is provided from an initial vision document which was created to plan the project:

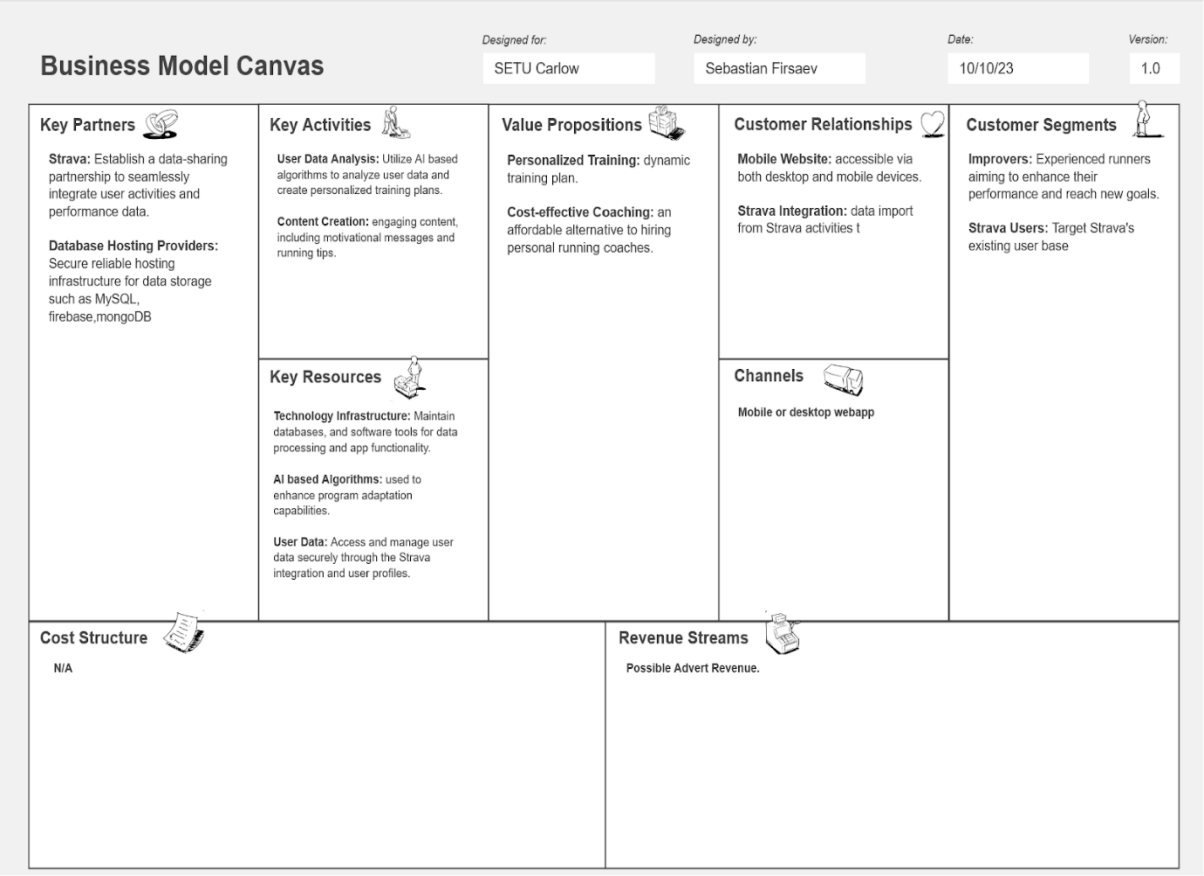

Figure 2, Business Model Canvas

Objectives

The primary objectives of this project are as follows:

- Develop a mobile focused web app that adapts training programs based on runner performance using AI-based algorithms.
- Utilize runner data such as heart rate (HR), pace, and distance from Strava to personalize training plans. And analyse whether or not they are adhering to the plan.

- Provide motivational messages and tips to keep runners engaged and motivated.

Application precedent

While there are existing running and fitness apps, they often both ask for and provide runners with too much data. This creates too many distractions and takes away the runners focus from the core run. PaceRunner differentiates itself by only displaying and using necessary data such as heart rate, pace, and distance to keep the runner informed, allowing them to be laser-focused on the run. PaceRunner also offers cross-platform (Mobile & desktop) AI-driven training programs, seamless integration with Strava, and emphasis on personalised, adaptable training plans. This combination of features sets it apart from traditional static training apps.

Target Audience

- Improvers - People who are already established runners that want to “up their game” and reach the next level.
- Fitness enthusiasts looking for an affordable alternative to running coaches.
- Runners who use Strava and are looking to enhance their training experience.

Target Platforms

- PaceRunner will target mobile platforms and will have a UI designed to work best with mobile.
- However, as a PWA (Progressive Web App) [2], PaceRunner will run in browser on most mainstream operating systems such as IOS, Android, Windows, Mac OS or Linux. Allowing for greater accessibility to runners as they can use Runmate from basically any device at any time.

Obstacles and Risks

Possible Risks

- **Risk:** Facing technical difficulties or limitations in implementing AI-based features, or other core functionalities.

- **Risk:** Handling runner data may prove difficult as it may be noisy and data quantity too large. It may also pose privacy and security risks, such as data breaches and private user information protection.

- **Risk:** Strava integration may face technical issues or changes in their API, affecting data import and analysis.

- **Risk:** Runners may lose interest or motivation over time, leading to abandonment of the app.

- **Risk:** General limitation on time and resources

- **Risk:** future scalability If the app gains unexpected popularity, it may face scalability issues, affecting performance.

- **Risk:** GDPR implications for user data.

Context Diagram

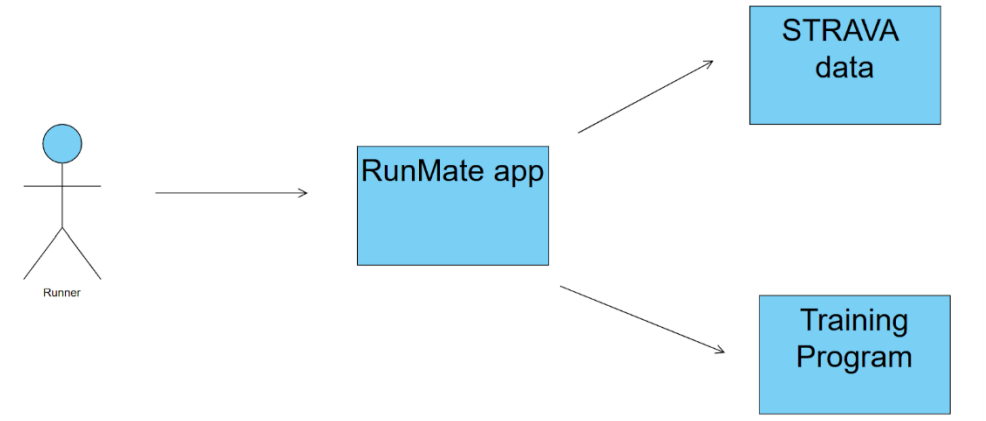

Figure 2, context Diagram.

Use case Diagram

Figure 3, Use case Diagram.

Brief Use Cases

Use Case Name: Register/ Link Strava account
Actor(s): Runner, Strava API
Description: This use case is initiated when the runner first opens the application and intends to create a new account. The runner provides their details, including a username and password. The system then captures this information and successfully registers them for the application. **Alternative:** Runner can select to register with Strava. As a result their account is created using their linked Strava account details.

Use Case Name: Login
Actor(s): Runner
Description: This use case commences when the runner launches the application and wants to log into their account. The runner inputs their username and password. The system verifies the runner's credentials and authorizes their access to the application.

Use Case Name: Reset Password
Actor(s): Runner
Description: This use case is triggered when the runner needs to reset their password. The runner provides their email address and specifies the new password they want to use. The system then proceeds to modify the password accordingly.

Use Case Name: CRUD Training Plan
Actor(s): Runner, Strava API
Description: This use case is initiated when a runner wants to create, read, update, or delete (CRUD) training plans for their individualised training journey. The Strava API provides relevant data for the Runner's plans. **Create Training Plan (C):** The Runner begins by selecting “start training plan”. They can design a personalised training plan by selecting parameters like training objectives, duration, frequency, The app will leverage data from the Strava API to make data-driven decisions for the plan. **Read Training Plan (R):** The Runner can access and review their existing training plans. This allows them to track their progress, and compliance with the plan. **Update Training Plan (U**): When required, the Runner can modify an existing training plan. This may involve changing training parameters, updating goals and forcing breaks. The updated plan is saved and linked to the Runner's profile.

Delete Training Plan (D): The runner will have the option to delete any plan. This action removes the plan from their profile and the application's database.

Use Case Name: View Progress
Actor(s): Runner
Description: This use case is triggered when the runner wants to view their overall progress from the start of their plan to what they are at currently. The Runner selects view progress and the information is displayed.

Use Case Name: Leave Feedback
Actor(s): Runner
Description: This use case is triggered after a runner was finished a run; they are asked to give feedback on how they are feeling. They select an option from the following: exhausted, Fatigued, Moderately Fatigued, Energetic.

Use Case Name: Logout
Actor(s): Runner
Description: This use case is initiated when the runner desires to exit the application. The runner selects the logout option. The application effectively logs out the user.

Detailed Use Case

Use Case Name: CRUD Training Plan

Actors:

1. **Runner:** interacts with the application to manage their training plans.
2. **Strava API:** Provides relevant data for the Runner's training plans.

Description: This use case facilitates the creation, retrieval, modification, and deletion (CRUD) of training plans for individualized training programs of runners. The Strava API serves as a source of data to inform and adjust the Runner's plans.

Basic Flow:

Create Training Plan (C):

- **Runner** initiates the creation of a new training plan by selecting "start training plan" within the application.
- They select parameters such as training objectives, duration, and frequency.
- The application integrates with the **Strava API** to leverage historical and real-time data, allowing for data-driven decisions in adjusting the training plan.
- Based on the input and data received, the application generates a training plan for the Runner that will be adjustable based on their performance.

Read Training Plan (R):

- **Runner** accesses the application to review and access their existing training plan.
- **Runner** selects view plan, the application displays detailed information about their plan, including objectives, scheduled activities, progress tracking, and adherence to the plan.

Update Training Plan (U):

- If necessary, the **Runner** can modify an existing training plan manually.
- They have the option to adjust training parameters, update goals, or incorporate necessary changes such as forcing a break day.
- **Runner** selects force break; the plan adjusts to the schedule.
- Upon completion of modifications, the updated plan is saved and associated with the Runner's profile.

Delete Training Plan (D):

- The **Runner** has the option to delete any training plan that is no longer relevant or needed.
- **Runner** selects view plan and presses delete plan.
- The plan is removed from the user's account.

Alternative Flow:

Invalid Data/Parameters:

If the Runner inputs invalid or conflicting data while creating or updating a plan, the application will prompt for correction or adjustment to ensure the plan's accurate data.

Core functional requirements

AI-Based Training Plans: The application utilises AI based algorithms to generate personalised training programs. These programs adapt dynamically based on runner profiles, performance data, and daily activity updates.

Runner Registration and Profile Creation: Runners can create profiles by providing essential information such as age, sex, weight, fitness goals, and current running ability through their mobile devices.

Additional non-core functional requirements

Motivational Messages and Tips: Runners receive daily motivational messages and running tips to keep them engaged and motivated throughout their training journey.

Non-Functional Requirements

The non-Functional Requirements for PaceRunner are explained via FURPS+, The acronym FURPS is Functionality, Usability, Reliability, Performance, and Supportability [1].

Usability

- PaceRunner must have an intuitive, runner-friendly, and mobile focused interface that makes it easy for runners to access and navigate its features. The UI must be clear any easy to navigate on mobile devices. Runners must be able to access and see their training plan within 2 seconds of opening the app.
- PaceRunner must be accessible to runners with varying levels of technological expertise, ensuring inclusivity.
- The onboarding process should be straightforward, guiding new runners in setting up their profiles and understanding how the app works. Runners should be able to set up the app within 10 seconds.
- PaceRunner should include mechanisms for runners to provide feedback, helping to improve the runner experience over time.

Reliability

- PaceRunner must provide accurate and reliable data, especially in the adaptation of training programs.
- The app should be available for use consistently and should be available to run 99% of the time, minimising downtime.
- Runner data and progress should be regularly backed up to prevent data loss in case of technical issues.

Performance

- PaceRunner should be fast and responsive, with average response times being less than 2 seconds, ensuring a seamless runner experience.
- PaceRunner should be able to handle increased runner load and data processing without significant performance degradation.

Supportability

- The app must be maintainable and updatable to fix issues, add features, and adapt to changes in runner needs.
- Comprehensive documentation must be available for runners and developers, providing guidance on app usage and integration.

Security

- Runner data must be handled with the utmost care, and GDPR and privacy standards must be maintained.
- login and runner authorization mechanisms such as password hashing should be in place to protect runner accounts.
- HTTPS protocol will be used for all information sent through the web browser.

Plus

- As a PWA, PaceRunner should allow runners to access certain features or content when they're not connected to the internet.

Metrices

- The success of this application will be determined by the fact of it having met the following criteria.
- Functioning base app structure including runner account creation, logging in and accessing training plan
- Strava integration allows the app to access runner's training data such as pace and heart rate.
- Training programme being able to adjust itself based on runners past performance.

Iteration backlog

Iteration 1:

Use Cases to Tackle:

1. **Register/Link Strava Account**
2. **Login**

Non-Functional Requirements to Address:

- **Usability:** Ensure an intuitive interface for account creation and login within the app.
- **Reliability:** Establish a secure and reliable password reset mechanism.
- **Supportability:** Provide initial design documentation for functionalities.

Risks to Consider:

- Technical difficulties in account creation and login functionality.
- Potential issues with Strava integration during account linking.
- Issues with strava data retrieval

Iteration 2:

Use Cases to Tackle:

CRUD Training Plan

Create, Read, Update, and Delete training plans.

Non-Functional Requirements to Address:

- **Usability:** Ensure a user-friendly interface for creating, reading, updating, and deleting training plans.
- **Reliability:** Guarantee consistent data availability and accuracy for training plan modifications.
- **Supportability:** Offer more comprehensive documentation and support for training plan management.

Risks to Consider:

- Handling invalid or conflicting data during training plan creation or modification.
- Issues with seamless integration with the Strava API for accessing relevant training data.
- User data storage and security

Iteration 3:

Use Cases to Tackle:

1. **View Progress**
2. **Leave Feedback**

Non-Functional Requirements to Address:

- **Usability:** Ensure easy access to progress tracking and feedback mechanisms within the app.
- **Reliability:** Consistency in displaying accurate progress information.
- **Supportability:** Offer support for logging out and providing feedback features.

Risks to Consider:

- Issues in displaying progress data accurately and in real-time.
- Challenges in implementing a feedback system that engages users effectively.

Additional Considerations for All Iterations:

- **Security:** Ensure data protection and compliance with GDPR and privacy standards throughout each iteration.
- **Performance:** Monitor and optimize app responsiveness and speed to meet the specified response time criteria.
- **Supportability:** Provide continuous documentation and support for each added functionality.

References

[1] Business Analyst Training in Hyderabad. (2014). What is FURPS+? Available at: https://businessanalysttraininghyderabad.wordpress.com/2014/08/05/what-is-furps/

[2] MDN Web Docs - Mozilla. (2023). Progressive web apps. Available at: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

[3] Pfitzinger, P., Douglas, S., & Huddle, M. (2019). *Advanced Marathoning*. Human Kinetics. Available at: https://books.google.ie/books?id=rT6lDwAAQBAJ

[4] Meschke, J. (2021). What Is Strava? Runner's World. Available at: https://www.runnersworld.com/beginner/g25619156/what-is-strava/