

Bachelor Project



**Czech
Technical
University
in Prague**

F3

**Faculty of Electrical Engineering
Department of Computer Science**

AI voice-guided app for runners

Kryštof Gärtner

Supervisor: Ing. Karel Frajták, Ph.D.

Field of study: Open Informatics

Subfield: Software

May 2024

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Gärtner** Jméno: **Kryštof** Osobní číslo: **507396**
Fakulta/ústav: **Fakulta elektrotechnická**
Zadávající katedra/ústav: **Katedra počítačů**
Studijní program: **Otevřená informatika**
Specializace: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Běžecká aplikace s AI hlasovým průvodcem

Název bakalářské práce anglicky:

AI voice-guided app for runners

Pokyny pro vypracování:

Bakalářská práce se zaměřuje na vývoj prototypu mobilní aplikace určené pro běžce. Hlavním cílem práce je navrhnout a vyvinout aplikaci, která během výkonu poskytuje kontextově relevantní a motivující hlasové pokyny generované umělou inteligencí. Tyto pokyny budou adaptabilní a schopné reagovat na reálný výkon uživatele, jako je tempo, vzdálenost a čas. Součástí bude také důkladný výzkum existujících běžeckých aplikací, stejně jako analýza potřeb cílové skupiny uživatelů. Tyto informace budou sloužit jako základ pro design a funkčnost aplikace.

Metodologicky bude práce zahrnovat vývoj softwaru pomocí vhodných programovacích nástrojů a jazyků pro mobilní platformy, návrh intuitivního uživatelského rozhraní, implementaci backendové systému schopného obousměrné komunikace, integraci TST technologie a NLP frameworku nebo napojení na API textového AI modelu, a provedení uživatelských testů pro sběr zpětné vazby a iterativní vylepšení aplikace.

Očekávaným výsledkem je funkční prototyp mobilní aplikace s adaptivním AI hlasovým průvodcem, doprovázený podrobnou dokumentací procesu vývoje. Tato dokumentace bude zahrnovat popis designových rozhodnutí, překonaných výzev a řešení problémů. Dále bude obsahovat analýzu uživatelského testování a doporučení pro budoucí vývoj aplikace.

Seznam doporučené literatury:

JANSSEN, Mark, et al. Who uses running apps and sports watches? Determinants and consumer profiles of event runners' usage of running-related smartphone applications and sports watches. PloS one, 2017, 12.7: e0181167.
VICKEY, Ted; BRESLIN, John; WILLIAMS, Antonio. Fitness--There's an App for That: Review of Mobile Fitness Apps. International Journal of Sport & Society, 2013, 3.4.
SHEI, Ren-Jay. Competitive influences of running applications on training habits. The Physician and sportsmedicine, 2018, 46.4: 414-415.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Karel Frajták, Ph.D. laboratoř inteligentního testování systémů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **01.02.2024**

Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

Ing. Karel Frajták, Ph.D.
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Mgr. Petr Páta, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání

Podpis studenta

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Ing. Karel Frajták, Ph.D., for his invaluable guidance and insightful suggestions throughout this project. I am also profoundly thankful to my family for their unwavering support and to my girlfriend for her dedication during our thorough study sessions. I also extend my appreciation to all the individuals who participated in testing my application. Lastly, I am grateful to the maintainers of the open-source libraries and tools that were essential to my work.

Declaration

I hereby declare that I have elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis.

In Prague on 23 May 2024

Kryštof Gärtner

Abstract

This bachelor's thesis focuses on the development of a mobile application designed for runners, featuring a coach that provides contextually relevant and motivating voice narration during runs. These coaching entrances are generated by artificial intelligence and can adapt to the user's real-time performance. The application allows users to set various parameters and save runs through a user account.

The text focuses on exploring market alternatives and a comprehensive analysis of application requirements, comparing technological options, designing the software system, and detailing the implementation itself. A significant portion of the text is devoted to the integration of large language models and the associated challenges. A chapter is also dedicated to the results of testing the application with real users.

The implementation uses TypeScript and React Native, Expo, NativeWind, Node.js, tRPC, and Drizzle. The application is connected to external services from OpenAI, OpenRouter, Clerk, and AWS.

Keywords: Mobile Application Development, AI Coaching, Running App, Adaptive Technology, Large Language Model, Text-to-speech narration, Software project, API Integration

Supervisor: Ing. Karel Frajták, Ph.D.

Abstrakt

Tato bakalářská práce se zaměřuje na vývoj mobilní aplikace určené pro běžce, která disponuje koučem poskytujícím kontextově relevantní a motivující hlasové pokyny během běhu. Tento doprovod je generován umělou inteligencí a dokáže se adaptovat živě na reálný výkon uživatele. Aplikace poskytuje možnost nastavovat různé parametry a běhy ukládat přes daný uživatelský účet.

Text se zaměřuje na průzkum alternativ na trhu a celkovou analýzu požadavků aplikace, porovnání technologických možností, návrhu softwarového systému a detailně samotné implementaci. Větší část textu je věnována integraci velkých jazykových modelů a úskalím s tím spojenými. Kapitola je též věnována výstupům testování aplikace na reálných uživateli.

Implementace využívá jazyku TypeScript a knihoven React Native, Expo, NativeWind, Node.js, tRPC a Drizzle. Aplikace je napojena na externí služby od OpenAI, OpenRouter, Clerk a AWS.

Klíčová slova: Vývoj mobilních aplikací, AI koučování, Aplikace pro běh, Adaptivní technologie, Velký jazykový model, Text-to-speech doprovod, Softwarový projekt, Integrace API

Překlad názvu: Běžecká aplikace s AI hlasovým průvodcem

Contents

| | | | |
|------------------------------------|-----------|-------------------------------------|-----------|
| 1 Introduction | 1 | 5.8.2 Hosting services | 37 |
| 2 Running applications | 3 | 5.8.3 Scaling | 38 |
| 2.1 Existing alternatives | 3 | 6 Testing | 39 |
| 2.1.1 Nike Run Club | 4 | 6.1 Methodology | 39 |
| 2.1.2 Peloton | 5 | 6.2 Test group | 40 |
| 2.1.3 Zombies, Run! | 6 | 6.3 User testing results | 40 |
| 2.1.4 Comparative Analysis Results | 7 | 6.4 Prompt testing | 42 |
| 2.2 User requirements | 7 | 7 Conclusion | 43 |
| 2.2.1 Functional requirements | 7 | 7.1 Evaluation | 43 |
| 2.2.2 Non-Functional Requirements | 8 | 7.2 Potential enhancements | 44 |
| 3 Technology analysis | 9 | 7.3 Future plans | 45 |
| 3.1 Mobile application | 9 | Bibliography | 47 |
| 3.2 Server and communication | 11 | A List of abbreviations | 49 |
| 3.3 Large Language Model | 11 | B Application demonstration | 51 |
| 3.3.1 State of the art | 13 | C LLM dialog transcript | 53 |
| 3.3.2 Privacy concerns | 14 | D Software usage declaration | 57 |
| 3.3.3 Open-source models | 15 | | |
| 3.3.4 OpenRouter | 16 | | |
| 3.3.5 Conclusion | 16 | | |
| 4 Design | 17 | | |
| 4.1 User interface | 17 | | |
| 4.1.1 Low-fidelity prototyping | 17 | | |
| 4.1.2 UI Components | 17 | | |
| 4.1.3 Color scheme | 19 | | |
| 4.2 Software architecture | 19 | | |
| 5 Implementation | 23 | | |
| 5.1 Dependencies | 23 | | |
| 5.1.1 Client-side dependencies | 23 | | |
| 5.1.2 Server-side dependencies | 24 | | |
| 5.2 Project structure | 24 | | |
| 5.3 Client-Server communication | 25 | | |
| 5.4 Authentication | 26 | | |
| 5.5 Persistence | 27 | | |
| 5.5.1 Client-side state | 27 | | |
| 5.5.2 Redis | 27 | | |
| 5.5.3 Relational database | 28 | | |
| 5.6 Location tracking | 29 | | |
| 5.7 LLM Integration | 30 | | |
| 5.7.1 Application logic | 31 | | |
| 5.7.2 Settings | 31 | | |
| 5.7.3 Prompt Engineering | 32 | | |
| 5.7.4 Potential issues | 34 | | |
| 5.7.5 Measurements | 35 | | |
| 5.8 Deployment | 36 | | |
| 5.8.1 Mobile application | 36 | | |

Figures

| | |
|---|----|
| 2.1 Nike Run Club screenshot of the running screen | 4 |
| 2.2 Nike Run Club screenshot of the run starting screen | 5 |
| 2.3 Screenshots from the Zombies, Run! application | 6 |
| 3.1 Overview of the React Native architecture | 10 |
| 3.2 LLM Quality comparison | 12 |
| 3.3 LLM Speed comparison | 13 |
| 3.4 LLM Price comparison | 13 |
| 4.1 Example of the low-fidelity prototype | 17 |
| 4.2 React Native Reusables demo .. | 18 |
| 4.3 Button component with applied color scheme | 19 |
| 4.4 Component diagram of the system architecture | 20 |
| 4.5 Activity diagram of completing a run | 21 |
| 5.1 Project's directory structure ... | 25 |
| 5.2 Example of a Redis entry from the Redis Insight monitoring tool | 28 |
| 5.3 Deployment diagram | 36 |
| 5.4 Output of Expo start command | 37 |
| 5.5 Redis monitor dashboard on Upstash | 38 |
| B.1 Application screenshots of running screens | 51 |
| B.2 Application screenshots of running history and run detail | 52 |
| B.3 Application screenshots of settings and authentication | 52 |

Tables

| | |
|--|----|
| 5.1 Performance, token usage and costs measurements for different LLMs . | 35 |
| 6.1 Test participants | 40 |
| 6.2 Uncovered issues or bugs found throughout user testing | 40 |
| 6.3 Suggestions, proposed improvements and new features from the testers | 41 |



Chapter 1

Introduction

Mobile fitness applications offer a significant opportunity to promote exercise among the general population. Although not everyone has the means to attend coaching sessions or join a sports club, almost everyone has access to a mobile phone, and downloading an application is straightforward. Fitness apps offer significant psychological and behavioral benefits. They help users establish and maintain healthy habits through regular reminders and progress tracking, making it easier to stay committed to their fitness goals.

Running applications, in particular, have seen significant growth in the past decade, with various features that positively enhance the intrinsic motivation of runners[1]. One particularly interesting feature is the use of audio coaches, often pre-recorded by real athletes, which guide users during their runs. Building on this concept, there is potential to integrate artificial intelligence in the coaching that can adapt to a user's performance, thereby increasing engagement levels and providing further incentives for runners.

Having personally used numerous running applications, the Nike Run Club has been the most impactful for me. It introduced me to the world of running and taught me many valuable lessons, particularly through its guided audio runs, which helped me enjoy running in new ways. These guided runs taught me how to avoid exhaustion and maintain a consistent running habit. However, the pre-recorded nature of these audio guides means they are static and always deliver the same messages at predefined points. This lack of customization based on user needs and preferences (e.g., frequency and intensity of guidance) and their inability to react to significant changes in pace or elevation conditions limit their effectiveness.

Recognizing the potential for more dynamic coaching, I quickly found that there were no existing applications with this capability. Furthermore, there is currently limited research on integrating AI in fitness applications[2], particularly in this manner. The primary goal of this project is to analyze existing technology and develop a fully functional prototype of an application with this feature. The result will be a comprehensive software project in the form of a mobile application targeted at runners of various experience levels.

In this paper, I will perform an analysis of existing running applications and user needs. Based on this analysis, I will make decisions on the appropriate technology and create designs for the application's functionality. Then, a

detailed description of the final implementation is provided and user testing is carried out to evaluate the quality of the implementation. Given the crucial role of artificial intelligence in this project, I will also explore various options in this space, address the privacy of user data related to AI providers, and discuss techniques and pitfalls of integrating this technology.

This project aims not only to create a software application but also to map the current landscape of artificial intelligence for software integrations and to uncover the potential of using such technology in the fitness context. A successful implementation may be able to transform the fitness industry by making personalized coaching more accessible and effective. For users, personalized real-time feedback and adaptive challenges offered by AI coaches can enhance motivation, prevent injuries, and promote general wellness. Ultimately, this project can change the way people engage in exercise by fostering a more informed approach to health and exercise.

Chapter 2

Running applications

According to a study analyzing users of running apps and sport watches, approximately 55% of runners reported using a dedicated running app.[3] For casual runners who do not participate in many events, are not in any clubs, and do not consider running their primary sport, maintaining motivation and consistency can be challenging. Running apps are designed to address this issue, helping casual runners stay engaged and avoid quitting. Compared to sport watches, apps are a more economical option, although they lack built-in heart rate measurement. The main advantages of using these apps are the instant feedback and insights they provide, such as data on distance, pace, and duration. Running technology can also educate users and reduce the likelihood of injury.

To further motivate runners, some applications include coaching audio tracks that play during the run. These coaches not only provide updates on the run's status (e.g., remaining time), but also offer tips on maintaining proper running form and mental focus. They serve as positive distractions, making the run more engaging and helping runners to reach the finish line.

2.1 Existing alternatives

The most popular running applications include *Runkeeper*, *Strava*, *Adidas Runtastic*, and *Nike Run Club*. These apps track users' locations with GPS, save detailed run history logs, and provide maps of completed routes. In addition to the general features they share, the specific functionalities and user interface designs of these apps have inspired this project's application.

Since most existing apps do not offer guided runs, this section will focus on those that do to some extent. The rest only provide short, template audio messages generated with live run data. Although dynamic, these messages are hard coded (e.g. "*The current pace is <x> min/km.*") and primarily serve to eliminate the need for runners to look at their watch or phone while exercising.

2.1.1 Nike Run Club

Nike Run Club (NRC)¹ is one of the most widely used running applications and served as a primary model for this thesis. The application is completely free with no in-app purchases. It features a polished, user-friendly interface that changes accent color based on the overall run distance. The application focuses on motivating runners through social participation, challenges, and training plans primarily aimed at casual runners. It also allows users to attach useful information to each run, such as custom notes, the shoes worn, the terrain surface, and the effort level.

A standout feature of NRC are its already mentioned excellent guided runs by renowned Nike coaches. These audio recordings are well-thought-out, follow an overall theme, and have specific duration or distance goals. There are even runs specifically designed for beginners. The main types of runs include recovery runs, long runs, fartlek², and interval training. Coaches provide short monologues at predetermined points, often leaving the runner with something to think about or a specific aspect of the run to focus on. The audio transitions are smooth and not overly distracting.

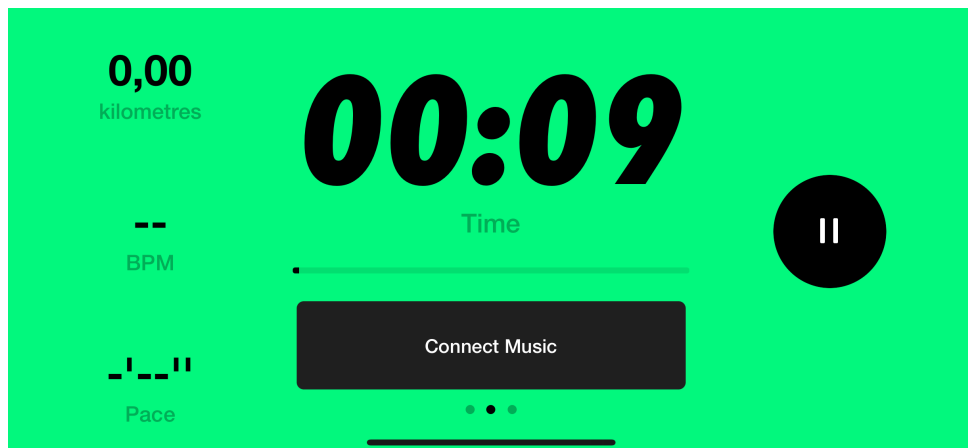


Figure 2.1: Nike Run Club screenshot of the running screen

¹<https://www.nike.com/cz/en/nrc-app>

²Fartlek is an interval training method that alternates between fast and slow running.

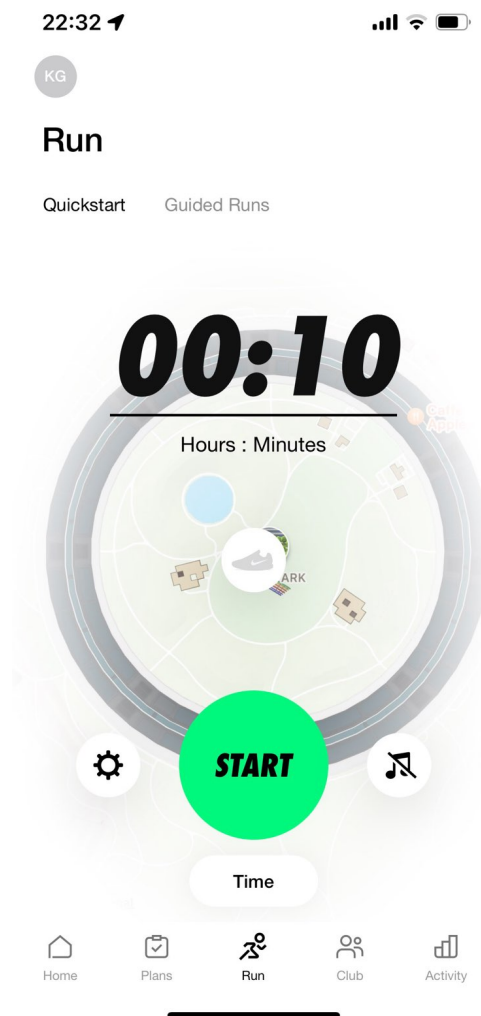


Figure 2.2: Nike Run Club screenshot of the run starting screen

2.1.2 Peloton

Peloton³ focuses more on indoor exercises such as cycling, yoga, rowing, treadmill workouts, and stretching. It offers dedicated audio and video classes by expert instructors for all activity categories, which can be combined to create comprehensive training sessions. No Peloton equipment is required for these classes, but a monthly subscription is necessary.

Recently, Peloton introduced outdoor walks and runs⁴ similar to the guided NRC runs. These classes are generally shorter than an hour and serve as warm-ups or intense performance sessions. As Peloton is not solely a running app, it lacks some dedicated features, and the overall experience may seem incomplete (e.g., inaccurate distance tracking, high battery consumption).

³<https://www.onepeloton.com/app>

⁴<https://www.onepeloton.com/classes/outdoor-workout>

2.1.3 Zombies, Run!

Zombies, Run!⁵ takes a unique approach to increasing the user's motivation by gamifying the activity. Set in a post-apocalyptic world, the app narrates runs as though the user were a survivor. Runners complete various tasks while listening to voice actors playing other survivors describe scenes and play out dialogues, almost as an audiobook.

The app involves rapid speed shifts during runs, with higher speeds leading to the collection of items useful for building a survivor base and unlocking new episodes. Occasionally, zombies "chase" the runner, and a slow pace results in the loss of collected items. This clever mechanism immerses runners in the fictional world even more, while it motivates them uniquely. Although the voice-acted parts are static, this functionality is the closest instance of application reacting to the runner's performance that I found throughout this research.

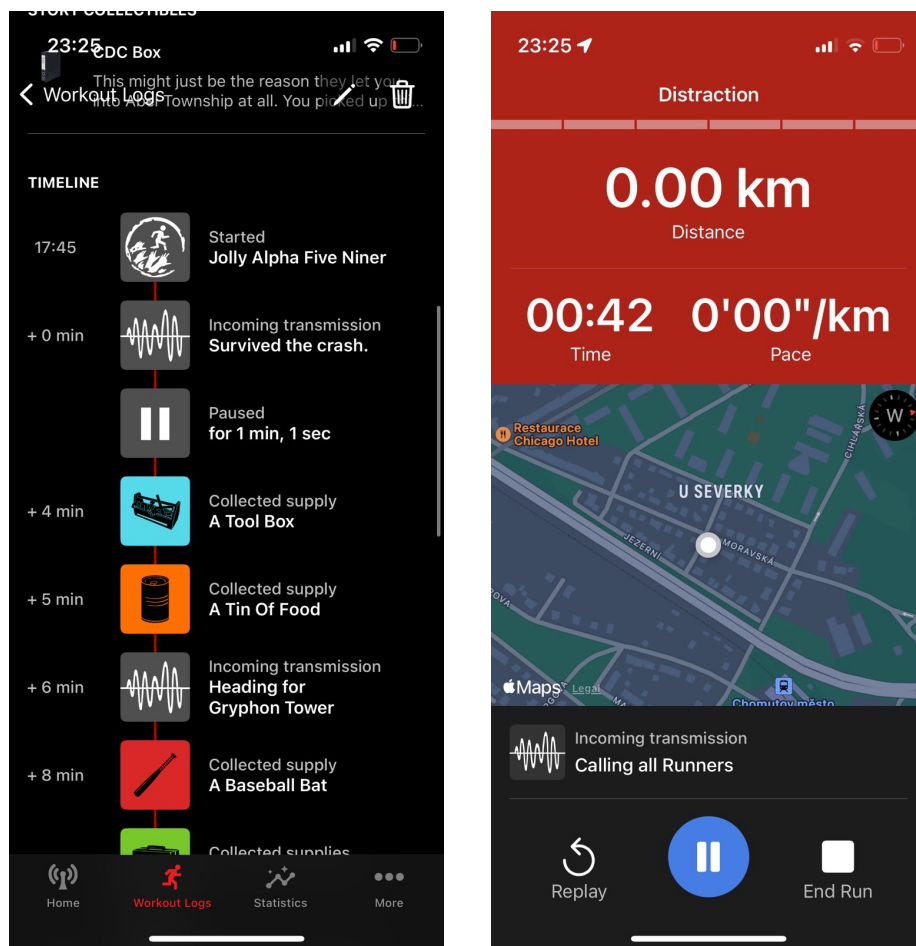


Figure 2.3: Screenshots from the Zombies, Run! application

⁵<https://zrx.app/>

■ 2.1.4 Comparative Analysis Results

There are many mobile applications for both iOS and Android that offer a great experience for runners. Although more apps are beginning to incorporate pre-recorded audio coaching, none yet dynamically generates audio content in real time based on the user's effort and specific settings. I believe that this project is the first application with such features and should be considered completely novel.

■ 2.2 User requirements

The potential user base of the application is quite diverse. There is no restriction on age, gender, or athletic form, as the settings and conditions can be adjusted freely. More experienced and professional athletes will probably see less value in both the accompanying background audio and motivational aspects. Hence, the primary focus should first be on the wider group of beginner to moderate runners.

The lists below model specific base requirements that the users will expect the app to satisfy. Additional extra features will be discussed in later parts. There is no point in including a use-case diagram, as there is only one user role, and it would merely mirror the listed requirements.

■ 2.2.1 Functional requirements

1. **User Registration and Authentication:** Users must be able to register and log in to the application. The authentication part must also allow for the reset of forgotten password, the change of password, and the deletion of the account.
2. **Performance Tracking:** Precisely capture and display statistics such as pace, distance, and time, while the run is ongoing.
3. **User Interface:** An intuitive and easy-to-navigate user interface will be provided. The simple controls and clarity must be especially high on the screen displayed during runs.
4. **Voice Guidance:** The application must provide contextually relevant voice guidance in real time during runs.
5. **Adaptive Feedback:** Voice guidance must adapt based on user performance metrics such as speed, distance covered, and running duration. The system should adjust motivational messages based on the user's historical performance and goals.
6. **Customizability:** Settings will offer a variety of options for customizing the audio guide, including the frequency of the voice entrances and the topic of the run.

7. **Saving the data:** Store historical performance data for user review. Include all the important information and the map of the completed route.
8. **Transferability:** Running data should not only be stored locally, but also transferable to other devices with the user account.

2.2.2 Non-Functional Requirements

1. **Performance:** The app should have minimal latency and be responsive to user actions. Quick loading times for user interfaces and during the interaction must be ensured.
2. **Usability:** The application must be user-friendly and suitable for users of various running experiences.
3. **Reliability:** The application must be reliable, with minimal downtime. Voice guidance and tracking features should function under various environmental conditions.
4. **Scalability:** The infrastructure should support an increasing number of users and simultaneous connections.
5. **Security:** User data must be sufficiently protected by taking security measures to ensure privacy. Compliance with data protection regulations should not be neglected. The minimum amount of user data should be transferred to the software external service.
6. **Portability:** The application should be compatible across different mobile platforms (iOS, Android) and with the major and future versions.

Chapter 3

Technology analysis

The crucial components of the proposed software system are a mobile application serving as a presentation layer, a server handling persistence and manipulation with external APIs, and a Large Language Model (LLM) for generating the narration content. In this part, I will describe the main technological choices to building such system and justify which ones are the most appropriate. All the remaining individual libraries and other minor frameworks will be heavily influenced by the decision of those three technologies.

3.1 Mobile application

The substantial part of the application logic will be solely on the client-side. User interfaces for all the different screens and the navigation between them must be created. It is also necessary to access the mobile's local storage to persist some application data without the need to fetch them from the server. Many of the mobile's APIs will be accessed, certainly the GPS location sensor, the instantaneous speed sensor, and background audio playback.

When there is sufficient time and developer capacity, building dedicated native apps for both major mobile operating systems should result in a superior product. With removed abstractions, there is more room for optimization, and the UI can have a unified platform-specific feel. The leading programming languages for native development on Android and iOS are Kotlin and Swift, respectively. To cover the majority of users, it does not suffice to build only for one of the operating systems. Furthermore, as a single developer and with regard to the scope of this thesis, developing separate applications for both platforms is neither suitable nor feasible.

The logical conclusion is to choose a cross-platform framework for building applications. The required workload can be significantly reduced by sharing a single codebase for multiple target environments. The most widely used and stable solutions currently are React Native¹ with 116k stars on GitHub and Flutter² with 162k stars.

¹<https://react.dev/>

²<https://flutter.dev/>

React Native was released by Facebook in 2015. It is essentially a mobile port of the React framework that is used to build reactive web applications in JavaScript. It slightly modifies the original framework functionality and creates a binding to the native rendering API, so the final application has a native look. Due to its popularity, a large ecosystem of useful libraries has grown around it. Additional development boost can be provided by using Expo³, which is another platform built on top of React Native. It substantially simplifies the development and testing process with dedicated developer tools. Additionally, Expo offers an extensive suite of libraries in its SDK⁴ to make working with mobile subsystems easier.

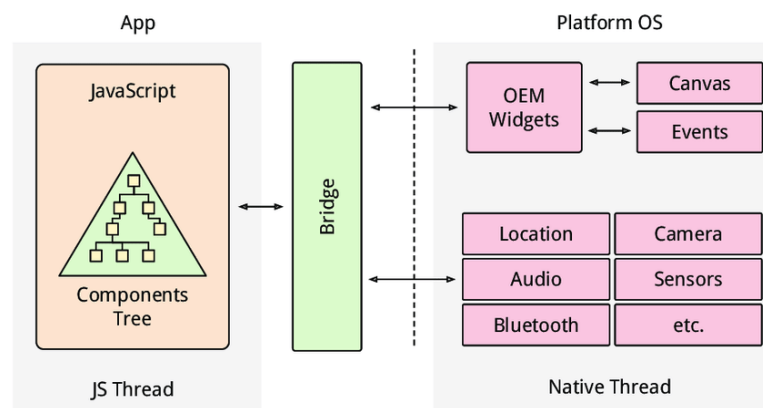


Figure 3.1: Overview of the React Native architecture[4]

Flutter is a UI framework from Google released in 2018. It uses the Dart language and takes a different approach to multi-platform. The code is compiled ahead of time and later rendered on the device by an engine also provided by the framework. Because of that, the application looks the same in every environment. In this way, it is easier to optimize for performance and eliminate bugs than with React Native. Thus, Flutter may perform better with respect to CPU and memory usage. [5]

I decided to use React Native for this project, mainly because of the Expo framework. It reduced the complexity of deploying a mobile application while providing an extremely useful set of libraries. Moreover, as opposed to Flutter, the newly learned skills can be directly translated to the React web development environment.

As JavaScript is a dynamically typed language, where runtime errors due to incorrect typing are a real danger, I determined that it would be beneficial to use TypeScript⁵. It is a superset of JavaScript that adds a type system layer. A new compiling build step is added, but in return, static analysis is enabled and the chance of bugs is lowered.

³<https://expo.dev/>

⁴Battery, Map, Audio-Video, Location, Haptics, File System and many others.

⁵<https://www.typescriptlang.org/>

3.2 Server and communication

The selection of the appropriate back-end technology for this project was relatively straightforward. I decided to use Node.js⁶ because of its efficient performance and the ability to seamlessly integrate in various environments. It can also be easily deployed across multiple hosting services, as it is a popular choice in the back-end development space. My primary motivation for choosing Node.js is the ability to maintain a consistent full-stack codebase throughout the project by utilizing a single language, TypeScript. This continuity ensures that both the client and the server side of the application are developed using the same programming language, which simplifies development and maintenance.

To facilitate communication between the client and the server, adopting tRPC⁷ can be highly beneficial. tRPC allows for type-safe API calls, which means that the types defined in TypeScript are preserved end-to-end from the client to the server without requiring separate validation or data transformation layers. This type safety significantly reduced the risk of runtime type errors and enhanced the overall reliability of the application. tRPC borrows some concepts from both REST and GraphQL and tRPC integrates naturally with Node.js and TypeScript environments. It provides a streamlined development process and efficient data handling capabilities.

3.3 Large Language Model

The foundation of the application is the large language model responsible for generating content for the audio coach. As this is a rapidly evolving field, staying current with technology trends is crucial. The performance and quality of responses from different models, both proprietary and open source, can vary significantly. The required LLM does not need advanced capabilities such as document querying or external function calls, which allows us to choose from a wide range of models. However, several key factors must be taken into account before making a selection.

The following is a detailed list of the primary parameters and requirements for the LLM used in this application.

- **General quality of responses:** Essential for the appeal of the application, the quality of LLM responses must engage the user effectively. While not every response needs to be perfect, they should generally be coherent and engaging. Occasional minor inaccuracies or variability should not negatively impact the user experience or the functionality of the application.
- **Throughput and latency:** Immediate feedback is pivotal in a running context, meaning response times should be short to prevent conveying

⁶<https://nodejs.org/en>

⁷<https://trpc.io/>

outdated information during use. Although high-quality responses are desired, the system should deliver them in a reasonable timeframe, ideally under 30 seconds.

- **Cost:** Cost-efficiency is important, typically calculated by the volume of data in tokens processed and outputted by the LLM. Although higher costs often correlate with better performance, finding a balanced solution is vital. The costs should be taken into account with respect to the potential future business model and scalability of the application.
- **Privacy:** The responsibility of handling sensitive user data, such as geolocation, is paramount. The LLM solution chosen must ensure robust data privacy measures to minimize risks.
- **Ease of integration:** The project prioritizes rapid development. Solutions that are simpler to integrate are favored, thus accelerating the overall product development cycle.
- **Reliability:** Depending on third-party services introduces potential risks, such as service outages or policy changes, which could disrupt the functionality of the application.

The primary analysis of model performance and quality metrics is derived from the Artificial Analysis[6], which benchmarks the connectivity and performance of models via API endpoints from various providers. This simulates realistic end-to-end performance, although results may differ slightly in a self-hosting setup. The benchmarks incorporate scores from Chatbot Arena Elo, Massive Multitask Language Understanding (MMLU), and MT Bench. However, these measurements are not flawless; factors such as server location discrepancies between the US and EU and the models' settings (e.g., temperature) can affect results. Despite these limitations, the data provide a valuable general overview, allowing for deeper analyses as needed.

Examinations can be conducted by comparing the most relevant LLM models:

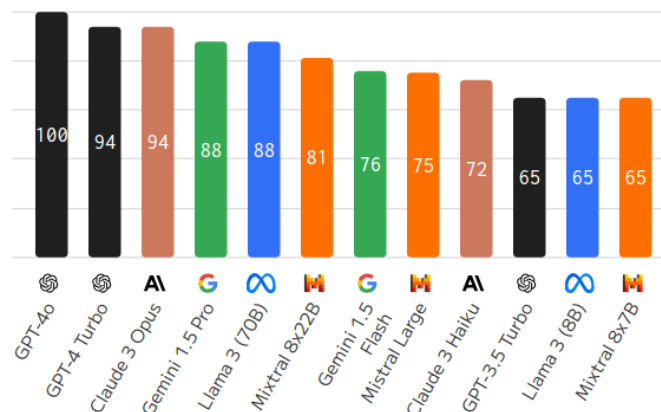


Figure 3.2: LLM Quality comparison[6]

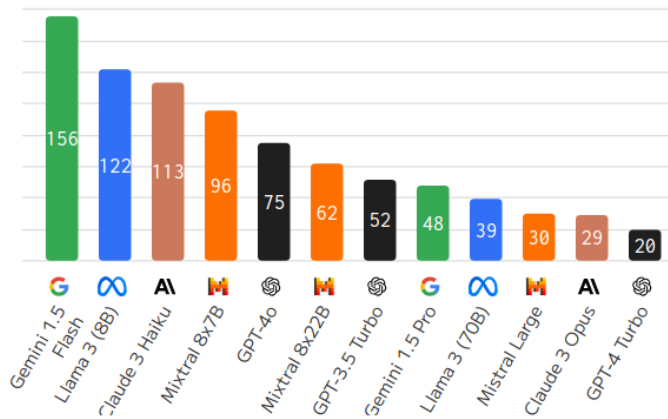


Figure 3.3: LLM Speed comparison (throughput in tokens per second)[6]

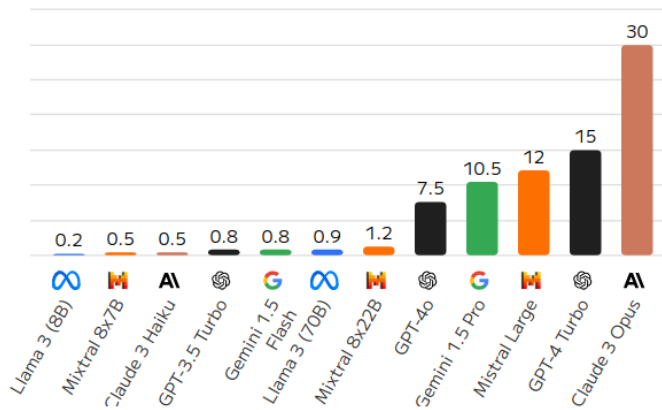


Figure 3.4: LLM Price comparison (USD per 1M tokens)[6]

3.3.1 State of the art

Leading models in the current landscape as of year 2024 in terms of output quality include Claude Opus by Anthropic⁸ and GPT-4o by OpenAI⁹. Both of these models are closed-source and available only through their respective APIs. They are known for their extensive analytical capabilities and the ability to solve a wide range of complex tasks in various domains.

Claude Opus stands out with its large input context window, which enhances its understanding of lengthy texts or detailed conversation histories. Anthropic also provides two smaller models, Haiku and Sonnet. Despite its advanced reasoning abilities, Claude Opus is the most expensive option on the market, and the additional cost may not be justifiable for this application’s needs compared to GPT-4o.

⁸<https://www.anthropic.com/claude>

⁹<https://openai.com/index/hello-gpt-4o/>

GPT-4o is a notable improvement over GPT-4 and GPT-4 Turbo in terms of performance. It offers the highest-quality responses available at present and does so at half the cost of its predecessor. OpenAI's reputation in software development and application integrations makes it a reliable long-term choice. Although GPT-4o is marketed as a multimodal model, it performs exceptionally well in text-based scenarios, similar to the previous versions.

Although these models offer exceptional quality, they are accompanied by higher costs and potential privacy concerns due to their proprietary nature and single-company development.

■ 3.3.2 Privacy concerns

In discussing OpenAI specifically, and applicable to other commercial providers of LLM services, privacy issues are prominent. OpenAI's privacy policy¹⁰ acknowledges that it automatically collects linked account information, other usage data, logs, and analytics, which may even be shared with third-party entities. This policy makes the service problematic for use in environments that handle sensitive data. The management and disposal of collected data by OpenAI are opaque, with compliance with privacy and security laws such as GDPR in the European Union unverifiable. OpenAI essentially operates as a "black box," offering minimal transparency into its data handling processes.[7]

A significant concern with GPT models, including OpenAI, is their use of reinforcement learning from user prompts, which has demonstrated privacy risks. The risks of sensitive data being retrieved later with the involvement of unreliable third parties or stolen from potentially insecure data centers are present. Furthermore, malicious actors already managed to extract private user data from the model by careful prompting[7].

One mitigation strategy is the use of a secured endpoint provided by Microsoft¹¹, which claims that no inputs or outputs are shared with OpenAI, external parties, or used to fine-tune the models, claiming stateless operation of hosted GPTs. Although the data remain within the same Azure infrastructure and regional confines, complete transparency on back-end operations is still lacking, but it represents a safer option compared to others.

In our specific use case, the risk of misuse of running data is relatively low, provided that the application does not include sensitive information, such as usernames, in the prompts. Regardless, it is crucial for the app to implement robust precautions when interacting with closed-model APIs, ensuring minimal personal data exposure. Alternatively, the application could consider using more transparent open-model options that typically offer greater control over data privacy.

¹⁰<https://openai.com/policies/privacy-policy/>

¹¹<https://learn.microsoft.com/en-us/legal/cognitive-services/openai/data-privacy>

3.3.3 Open-source models

Open-source models present an alternative, with the main advantage being the public availability of model weights for fine-tuning, though commercial use often requires a license. These models generally fall short of proprietary models in quality, but have been catching up recently.

One approach to optimizing these models is to replace a large general-purpose model with a smaller task-specific model that still delivers comparable quality. For instance, a model could be developed specifically for engaging in conversations during physical activities, trained with the help of professional coaches, objectives of motivation, and in-depth knowledge of running, nutrition, and biomechanics. However, this approach would require a deep dive into the nuances of artificial intelligence, focusing more on the model's capabilities rather than simply delivering a complete software solution.

Closely related option is fine-tuning a selected model with a custom dataset, which can specialize and enhance the model for specific applications. In return, the responses are quicker and more relevant. However, the costs associated with training and the need for a meticulously crafted dataset are significant.

Open-source models offer the possibility of local hosting, which guarantees full privacy of user data but requires substantial investment in infrastructure and maintenance. The simplest tool for interacting with locally hosted LLMs is Ollama¹² and even for smaller models, hardware capable of handling large amounts of compute is necessary. Despite the appeal, the high initial costs and lack of access to adequate computing resources exclude this option for the project. Not viable at the moment, but in the future a capable model could run offline directly on the device^[8], which would achieve complete privacy of user data.

An alternative is to utilize a hosted LLM service via an API provided by a service provider. This approach simplifies the integration process, and the costs associated with these services are generally lower than those for closed-source models, as the competitive market allows multiple providers to host similar models, driving down prices. The privacy issues also remain the same as for the proprietary models.

Among the most notable and widely used models are Llama¹³ by Meta and Mistral/Mixtral^{14 15}. The latest release, Llama 3, is available in two configurations: 8b and 70b parameters. These models provide quality nearly matching that of the proprietary models from companies like OpenAI, but at a significantly lower cost. The main limitation is their smaller context window of 8k tokens, which should not be adequate for the needs of this application. This makes Llama 3 one of the most suitable options.

¹²<https://ollama.com/>

¹³<https://llama.meta.com/>

¹⁴<https://mistral.ai/>

¹⁵Mixtral is a Mixture of expert version of Mistral.

■ 3.3.4 OpenRouter

OpenRouter¹⁶ offers a robust solution by providing a single API interface that enables connectivity to most LLMs currently available. It functions as a seamless drop-in replacement for interfaces like those provided by OpenAI. With a single API key, it allows changing a parameter in the API call to access various models across different cloud providers, adjusting for price, latency, and throughput. OpenRouter also supports dynamic routing based on current model performance, costs, or availability, and includes automatic fallback routing when services are down.

Given the rapid advancements in LLM technology, it is likely that models will continue to improve in capability and affordability. OpenRouter facilitates easy adaptation to these advancements without requiring changes to the application's core logic or platform, enhancing the application's flexibility. This flexibility ensures that the application is not overly dependent on any single service provider, reducing risk in the event of downtime.

OpenRouter allows users to opt out of logging AI completions, and since each request is routed to potentially different providers, individual providers cannot easily compile comprehensive user data.

■ 3.3.5 Conclusion

Considering these factors, I have chosen to integrate GPT-4o due to the quality of the responses and it being an established standard within the industry. Additionally, OpenRouter will be employed to maintain operational flexibility, enabling the use of open source models. This setup allows for quick adaptation to the best available models as they evolve.

¹⁶<https://openrouter.ai/>

Chapter 4

Design

4.1 User interface

4.1.1 Low-fidelity prototyping

The initial stage in designing the user interface involved the creation of low-fidelity wireframes, which illustrate the basic layout and functionality of the application. The wireframes were developed using the 'Basic UX/wireframing elements' library on Excalidraw¹. I adhered to *Nielsen's Ten Usability Heuristics*[9] and emphasized simplicity to maintain the straightforward nature of the application's functionality. These wireframes served as foundational templates for subsequent development stages.

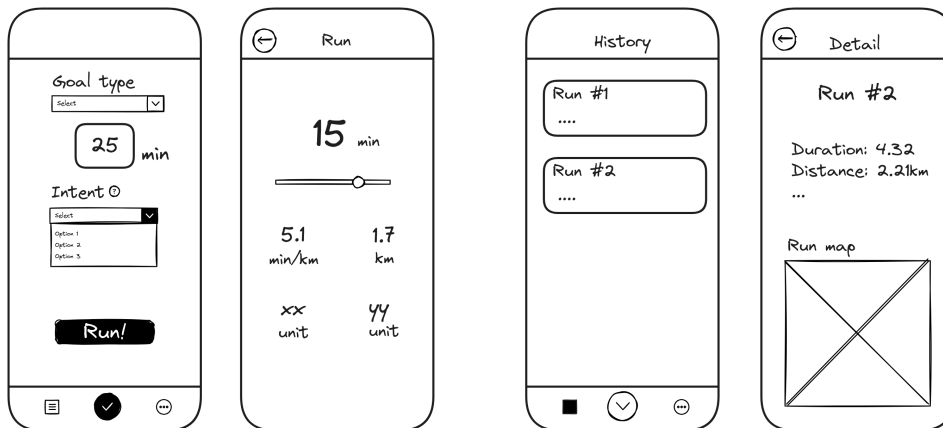


Figure 4.1: Example of the low-fidelity prototype

4.1.2 UI Components

In modern development practices, efficiency is often prioritized by utilizing pre-built component libraries rather than creating new components from scratch. However, limitations in customization can arise when relying solely on module

¹<https://excalidraw.com/>

imports and parameter adjustments. To overcome this, a different approach is often adopted, in which developers manually copy and paste component files into their projects, thus retaining full control over customization. A notable example of this approach is demonstrated by `shadcn/ui`², built on top of `Tailwind`³ and compatible with various web frameworks, including React.

For mobile-specific development, I selected `React Native Reusables`⁴, a robust library succeeding `shadcn/ui`, geared toward native environments. The implemented components include buttons, inputs, labels, switches, and navigation elements, along with the capability to centrally configure the application theme. This choice was influenced by my previous experience working with `Tailwind`, which should provide a smoother UI development process compared to using plain CSS or style sheets.

Select framework

More important than your skills.

Required

If you were an email, which one would you be?

napoleon@dynamite.com

Hint: it is not the one you use.

Enable notifications

We will send you spam.

Date of birth

Pick a date

Required

Accept terms & conditions

Submit

Figure 4.2: Screenshot from React Native Reusables demo

²<https://ui.shadcn.com/>

³<https://tailwindcss.com/>

⁴<https://rnr-docs.vercel.app/getting-started/introduction/>

4.1.3 Color scheme

The application color scheme features a dynamic primary yellow paired with a secondary blue⁵. This distinctive palette is not only unique from the application's competitors, but also aims to energize and motivate users with the vibrant visual appeal.

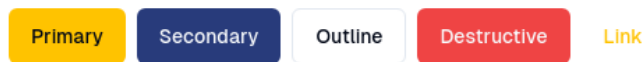


Figure 4.3: Button component with applied color scheme

4.2 Software architecture

The software architecture of this project is based mainly on the client-server⁶ model, which is a common structure for web applications. This architecture is particularly suitable given the tools and technologies employed, which are traditionally optimized for such an environment. Adopting this model facilitates a more seamless user experience and simplifies the implementation of security measures, notably authorization, since the server operates on an external machine, separate from the user's device.

The use of tRPC within the application introduces aspects of the Call and Return architecture, which enhances the interaction between the client and the server through defined procedural calls. This is evident in the server's structured approach to handling requests, which are segmented by distinct responsibilities.

The architecture also exhibits characteristics of a layered design:

- **The Presentation Layer** consists of the JSX code that structures the HTML within each TypeScript file in the client application.
- **The Business Logic Layer** is distributed across both the client and the server. On the client-side, business logic is handled within React hooks, which manage state, data preprocessing, and validation. Offloading some computations to the client-side is a deliberate choice to optimize performance and minimize network traffic.
- **The Data Access Layer** is exclusively managed by the server, ideally utilizing an Object Relational Mapping (ORM) library to facilitate database interactions.
- **The Database Layer** is implemented using either a self-hosted or cloud-based relational database.

⁵Colors were selected and tested using <https://ui.jln.dev/>

⁶Here, the term "client" refers to the mobile application developed using React Native.

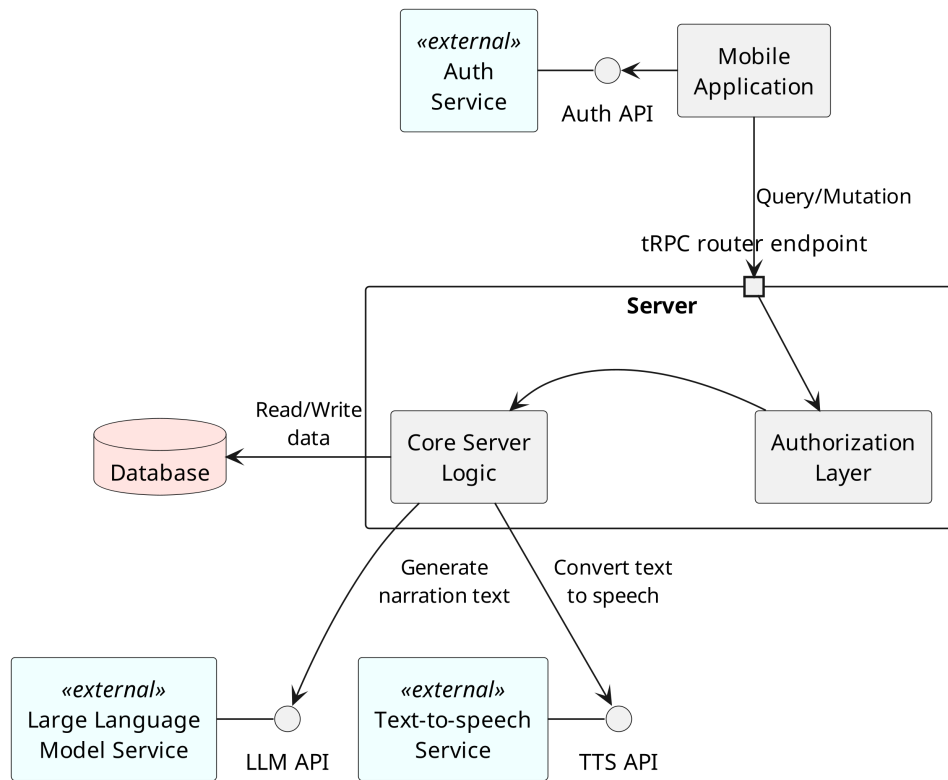


Figure 4.4: Component diagram of the system architecture

Additionally, the system has the potential to evolve into a Microservices Architecture. The clear division of responsibilities among the server's routes or procedures makes it feasible to decompose them into smaller, more manageable services. This modular approach could be advantageous for scalability, especially in alignment with the use of external services like third-party authentication systems. Transitioning to a more modular structure may offer long-term benefits as the user base grows.

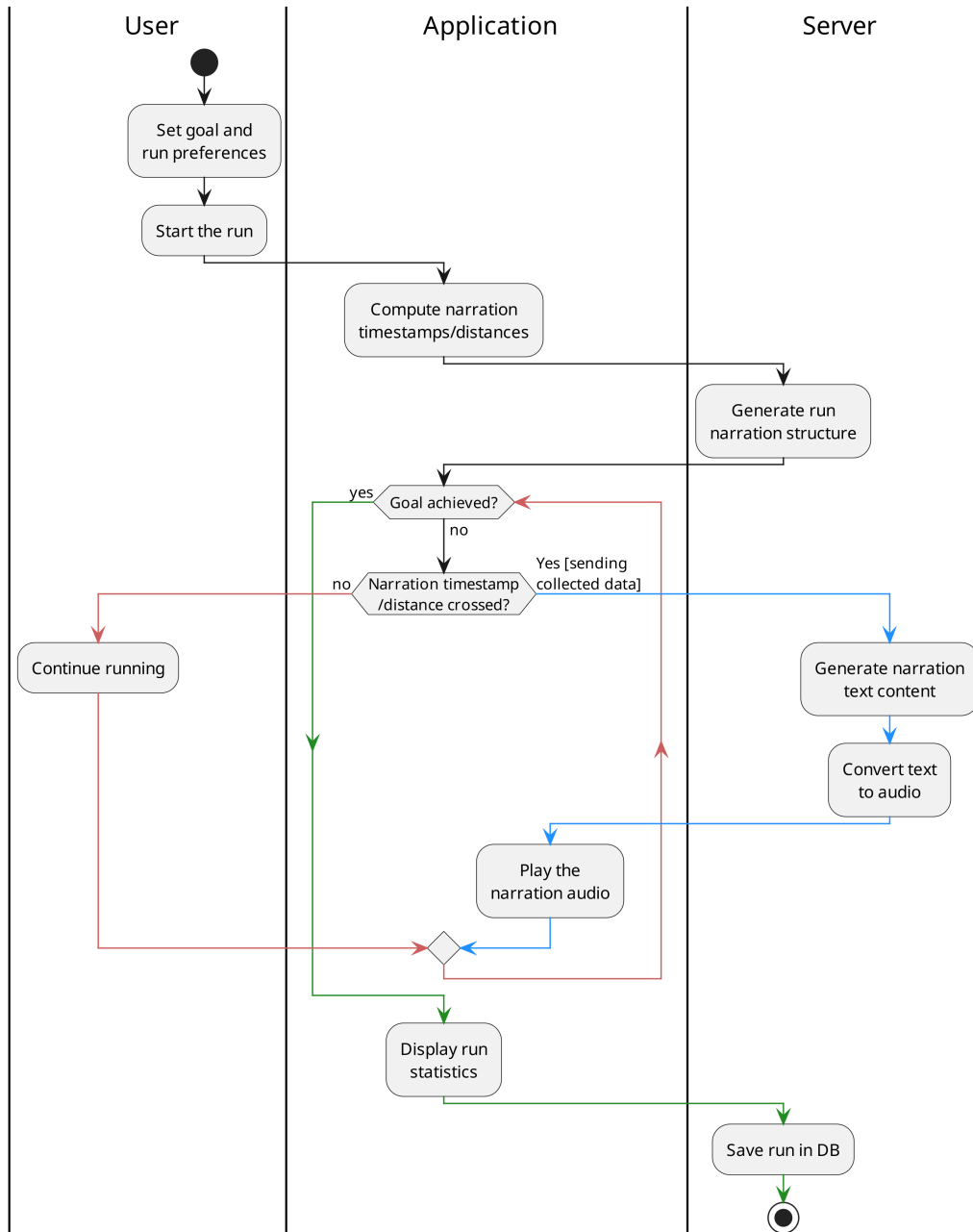


Figure 4.5: Activity diagram of completing a run

Chapter 5

Implementation

5.1 Dependencies

This project leverages a variety of dependencies to streamline development, avoiding the need to reinvent the solutions already provided by existing packages. This approach not only saves time, but also enhances reliability, as public packages typically undergo rigorous testing to mitigate potential bugs and security vulnerabilities. However, it is still essential to be careful, as excessive reliance on packages can inflate application bundle size and introduce conflicts during version upgrades. An overview of key packages with their specific integrations for this application is provided next.

Both the front-end and back-end are relying on packages provided by tRPC. Maintaining consistency in versions across client and server for those is crucial, because it ensures seamless functionality and eliminates unexpected behavior.

5.1.1 Client-side dependencies

- **Base React Native Packages:** Essential tools for animations, safe area handling, maps, and media management.
- **Expo Ecosystem Packages:** Using Expo's robust ecosystem for location services, secure storage, status bar management, multimedia support, and deep linking. Expo's CLI streamlines package management and ensures version compatibility.
- **Styling Tools:** Utilizing *NativeWind* for styling, along with a tailwind class merging package for seamless integration with external UI components. In addition, the *lucide* library facilitates icon integration.
- **State Management:** Employing the minimalistic *Zustand* library for efficient global state management.
- **Data Fetching:** Using *TanStack Query* ¹ for simplified fetch requests to the server, integrated with tRPC.

¹Formerly React Query

- **Authentication:** Integrating with external authentication services via the Clerk library for Expo.

■ 5.1.2 Server-side dependencies

- **Express:** Serving as the foundation for creating the REST server, integrated with tRPC.
- **Zod for request validation:** Zod facilitates request validation, while automatically generating schemas and types for client-side arguments.
- **External APIs:** Using OpenAI and AWS Polly libraries to communicate with external services.
- **Database Management:** Employing Drizzle ORM for relational database operations. IORedis for efficient handling of a Redis key-value database.

The selection of these dependencies followed a deep evaluation. I prioritized stability, maintenance, and suitability for future scalability. Later in this chapter, we will delve into specific use cases of some of these packages, providing comprehensive insights into their contributions to the project.

■ 5.2 Project structure

The project adopts a monorepo structure, having both client and server code within the same directory and Git repository. This approach enhances maintainability and also aligns with the requirements of the tRPC package. The client imports the type definitions of the tRPC router from the server's codebase, so there is seamless type-checking between client and server queries.

The client and server base directories contain a `package.json` file to manage dependencies and NPM configurations. In addition, various configuration files for TypeScript, different frameworks, and libraries are present.

In the client root directory (`client/app/`), React screens, layout files, and most of the logic are saved. Expo Router then uses file-based routing, so directory structure and filenames dictate navigation within the application. Within the server root directory (`server/src/`), the `index.ts` file initiates the Express server setup, and neighboring files construct the main tRPC router. The rest of the server-side code is divided into directories described below.

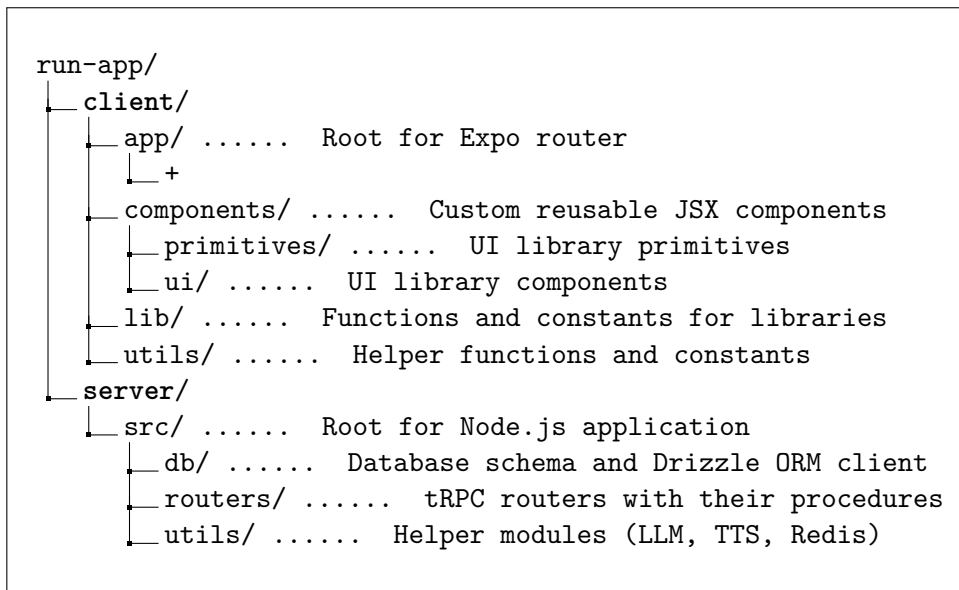


Figure 5.1: Project's directory structure

5.3 Client-Server communication

As previously mentioned, communication between the client and the server is facilitated by tRPC². It establishes a Typescript binding between both parts of the codebase and, in turn, ensures type checking consistency.

The foundation of the tRPC API is to define routers, each containing procedures that act as individual endpoints. Procedures can either be queries, returning responses, or mutations, performing side effects on the server such as updating, creating, or deleting data.

To organize procedures logically, I created three distinct routers, subsequently merging them into a single main router. The client then imports the type definition of this final router because it holds all the definitions of the procedure, input, and output type definitions.

```

export const appRouter = createTRPCRouter({
  narration: narrationRouter,
  tracking: trackingRouter,
  db: dbRouter
})
export type AppRouter = typeof appRouter

```

Procedures within these routers can be public or private and may include inputs and schema declarations. Input validation is facilitated using Zod, as demonstrated in the example below, where the procedure `getRunPositions` is defined to accept a single argument representing the ID of a database object and return the corresponding object:

²Typescript Remote Procedure Call

```

getRunPositions: protectedProcedure
  .input(z.number())
  .query(async ({ input: id }) => {
    // Retrieve object from DB
    return await ...
  })

```

Although Express tRPC creates standard endpoints similar to REST, making calls from the client via simple fetch requests is not ideal. The integration of tRPC with TanStack Query³ enhances the developer experience by eliminating a considerable amount of boilerplate code involved in manual fetching, refetching, and error handling. The front-end query is thus streamlined, as demonstrated here:

```

const { data, error, status } = trpc.db
  .getRunPositions
  .useQuery(id ? id : skipToken, {
    // ... settings
  })

```

Moreover, queries are automatically cached and re-tried by TanStack Query. This caching mechanism optimizes server calls by remembering previous queries and retrieving cached results when applicable. Mutations can trigger query invalidation and discard cached results, rerunning queries as needed. I also leverage the prefetch function to load data in advance, which results in instant screen loading when data would have to be fetched later.

5.4 Authentication

To use the application, users must create an account. This precautionary step is necessary due to operational costs associated with running the application, particularly the expenses related to the LLM service. Implementing user accounts allows the server to enforce access limits, restrict unauthorized access attempts, block potential attackers, and prevent abuse of the LLM service.

The authentication process is handled by Clerk⁴, a platform that offers APIs for user management and session handling. Clerk provides a dedicated Expo library that significantly simplifies authentication processes. The application interacts with this library through normal function calls with actions such as Log in, Sign in, or Log out. The underlying complexities are managed automatically. Upon invocation, the connection to the Clerk service is established seamlessly, with responses providing either rejection or successful user data. Sessions are managed in the background, with JWT⁵ access and refresh tokens securely stored in local storage. Token expiration is handled

³<https://tanstack.com/query/latest>

⁴<https://clerk.com/>

⁵<https://jwt.io/>

transparently and tokens are automatically refreshed in the background, without the need for additional logic.

The authorization of requests on the server is managed through access tokens attached to HTTP headers by the client. tRPC features an authorization middleware that intercepts requests before execution of the procedure. The middleware first verifies the validity of the token with a dedicated cryptographic key. Successfully verified tokens grant access to procedure execution and provide access to relevant user context data. Decoded JWT tokens can encapsulate user information such as ID, email, or current session ID.

Overall, the implemented authentication system ensures robust security measures, adequately meeting the requirements of this application's context. Clerk offers various authentication options, including integration with different OAuth providers, which can be added incrementally as needed. However, the convenience of using an external service comes with costs. While the current prototype operates on a free plan, scaling to more than 10,000 active monthly users would necessitate upgrading to a Pro plan. Nevertheless, should future requirements exceed Clerk's capabilities, alternative authentication services, including self-implemented solutions, can act as drop-in replacements. The primary additional task in such a scenario would involve migrating user data to the new service.

■ 5.5 Persistence

■ 5.5.1 Client-side state

The state of the application in the context of front-end frameworks like React resides on the client-side and is manually synchronized with the backend. In this setup, the application utilizes React Native's default state management⁶, combined with the Zustand library to handle global state. Zustand keeps state management from being too complicated, offering stores⁷ with defined properties and functions that act as global variables accessible from any part of the application.

By default, the state is reset upon restarting the application. However, middleware can be attached to stores to persist state in local storage⁸. For example, application settings are stored and hydrated using AsyncStorage. This ensures that the settings are loaded upon the start of the app and instantly saved upon any settings changes.

■ 5.5.2 Redis

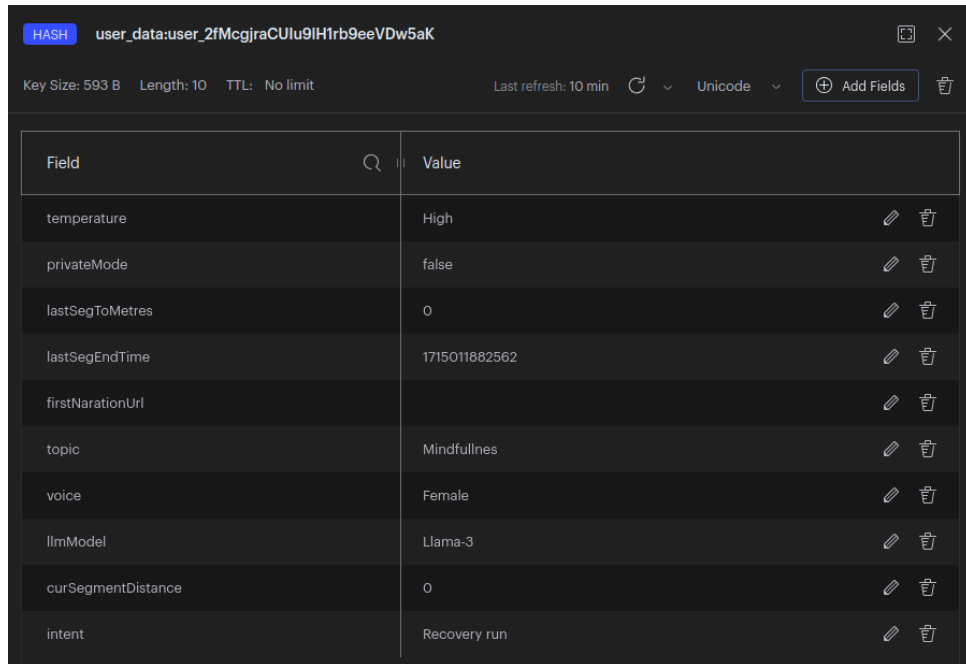
To effectively handle concurrent user sessions, the server uses the Redis in-memory database. Redis is favored for its extremely quick access times and easy integration. Most saved values are stored in a key-value format as

⁶Via the `useState` hook

⁷In this project saved in `client/utils/stores`

⁸<https://docs.pmnd.rs/zustand/integrations/persisting-store-data>

two strings. In this application, Redis⁹ stores temporary data while users are active, such as run data and LLM message history. This minimizes access to the slower main database, improving performance. I also defined an abstraction layer above the Redis object to safely query values and mitigate risks associated with querying non-existent keys or data.



| Field | Value |
|--------------------|---------------|
| temperature | High |
| privateMode | false |
| lastSegToMetres | 0 |
| lastSegEndTime | 1715011882562 |
| firstNarrationUrl | |
| topic | Mindfulness |
| voice | Female |
| llmModel | Llama-3 |
| curSegmentDistance | 0 |
| intent | Recovery run |

Figure 5.2: Example of a Redis entry from the Redis Insight monitoring tool

5.5.3 Relational database

For larger data requiring remote storage and scalability, the application utilizes a PostgreSQL relational database accessed by the persistence layer on the server. Drizzle ORM¹⁰ facilitates database operations by offering object relational mapping customized for different database types. This ensures syntax consistency across databases, with SQL-like queries mirrored by Drizzle functions. TypeScript's integration with Drizzle also enables type safety all the way from the database to the client, which has a potential of largely reducing developer errors.

The crucial feature of Drizzle ORM is the schema definition in a special TypeScript file¹¹, from which the type definitions are derived. A CLI tool provided by Drizzle can then generate migration SQL files from the schema and push them to the database. With minimal setup, Drizzle handles all database management tasks, requiring only the database URL.

⁹Specifically the *ioredis* library for Node.js

¹⁰<https://orm.drizzle.team/>

¹¹`server/src/db/schema.ts`

: Schema definition for the Positions table

```
export const positions = mySchema.table('positions', {
  id: serial('id').primaryKey(),
  runId: serial('run_id') // foreign key
    .references(() => runs.id,
      { onDelete: 'cascade' }),
  lat: real('lat').notNull(),
  long: real('long').notNull(),
  alt: real('alt').notNull(),
  instantSpeed: real('instant_speed').notNull(),
  timestamp: bigint('timestamp', { mode: 'number' }),
  accuracy: real('accuracy')
})
```

: Querying for users last run

```
await db.select()
  .from(runs)
  .where(eq(runs.userId, userId))
  .orderBy(desc(runs.serial))
  .limit(1)
```

Currently, the database consists of two tables: *runs* and *positions* corresponding to each run¹². The database primarily serves to store run data upon completion and retrieve data, such as running history, as needed. Nonetheless, the infrastructure is prepared to accommodate new migrations and the addition of new tables.

5.6 Location tracking

To accurately monitor the user's activity, the application must track the user's location, enabling it to compute the covered distance, the pace of the runner and later reconstruct the running route. This is achieved by subscribing to the device's location service, which provides GPS coordinates at regular intervals. The distance is then calculated by adding the distances between the individual points¹³.

However, GPS locations provided by the device are often inaccurate. Even with the highest accuracy settings, GPS can still fluctuate by up to 10 meters around the runner. These fluctuations result in a squiggly route on the final map and, more importantly, an overestimated distance, as the points differ more significantly. This issue is inherent in GPS itself and GPS chips in mobile devices[10]. Other running applications face the same flaw, each employing different algorithms to correct GPS data, leading to slight variations in measured values.

Initially, I tested the app without any tracking adjustments. Using the Android emulator, I uploaded noise-free GPX data and simulated the run

¹²Given the database's simple structure, a class diagram is unnecessary.

¹³The Haversine formula is used due to the Earth's curvature.

in this way, which yielded correct and precise results. However, real-world testing while running outside with a physical device resulted in distances approximately 15% larger than expected¹⁴.

The most common approach to address this issue is to smooth the location data, typically using a *Kalman filter*. This method returns a smoothed estimate that is more accurate than the raw GPS points. An adaptive Kalman algorithm is known to provide great results with minimal performance impact[11]. I implemented a simplified version of this filter based on an article focused on tracking with Expo¹⁵. This approach utilizes the GPS location, the accuracy of each location, and the timestamp to compute the speed and statistical variance. Despite fine-tuning the constants and adjusting the algorithm, I was unable to smooth the data for any real runs. A more complex Kalman filter, integrating signals from multiple sources, such as the device's gyroscope and acceleration sensor, might be necessary for this scenario.

Another issue is the location data subscription through the Expo location library, which, according to its documentation, is not guaranteed to work in all scenarios and is expected to be improved in the future. They recommend using *react-native-background-geolocation*¹⁶, the most advanced React Native library for tracking, which has been already developed and researched for many years. Although it is open-source, using this library in production requires a license, which is currently not financially viable.

As an interim solution, I have continued using the Expo tracking library and scaled down the measured distance with a specific coefficient. Although this approach is not perfect, it yields values closer to the real distances. After the run is finished, all running data and the route map¹⁷ can be inspected in detail. The squiggly lines on the route are only noticeable when zoomed in significantly and are not distracting.

5.7 LLM Integration

The integration of a Large Language Model (LLM) is a key feature of this application, ensuring that the coaching instructions are current and motivational. This section outlines the integration process and the challenges encountered. Both OpenAI models and open-source ones via OpenRouter are utilized for this task. An example transcript of a single run is included in the appendix C.

¹⁴I tracked the run simultaneously with other applications and compared the results.

¹⁵<https://blog.expo.dev/a-complete-guide-to-displaying%2Dand-normalizing-location-data-in-react-native-7e448c760fc2>

¹⁶<https://github.com/transistorsoft/react-native-background-geolocation>

¹⁷Each device uses its native mapping option for displaying the map, with Google Maps for Android and Apple Maps for iOS.

■ 5.7.1 Application logic

The goal is for the coach to provide input multiple times during the run, reflecting on live data, and engaging the user. At the start of the run, approximate timestamps or distances (based on the user's goal) are computed to determine when and how many times the LLM will be called. This allows for planning ahead, which ensures that the coaching messages are cohesive and flow well without repetitive phrases or ideas.

Thus, first the LLM generates an outline for all the monologues beforehand. This structure is maintained in each subsequent LLM call for the given run. All coaching entrances should start with an introduction, continue with a main part that incorporates current data and contributes to the run's overall theme, and end with a teaser for the next part.

When the specified time during the run arrives, the LLM is called from the server with instructions and context. The resulting text is returned and immediately processed by the Text-to-Speech (TTS) model. AWS Polly's¹⁸ neural voices are used here for their quality and cost-effectiveness, as generating voice audio natively on the mobile device often results in low-quality robotic sounds. The TTS model processes text with SSML tags, including breaks and volume adjustments. Then, only the URL link to the audio is returned to the mobile device, which then plays it in the background.

The first audio plays at the start of the run, with subsequent narrations distributed regularly, and the last one during the final minute. The conversation history is saved and always inputted into the LLM, as LLMs lack memory. To minimize the delay between crossing a timestamp and playing the audio, several optimizations were implemented for a seamless user experience.

From a coding perspective, OpenAI provides a dedicated TypeScript library for their API, which also works with OpenRouter. The code for this and for all interactions with the LLMs is located in `/server/src/utils/llm.ts`.

■ 5.7.2 Settings

To ensure that coaching monologues are varied and personalized, multiple settings are introduced. They allow users to gain some control and promote long-term use of the app. These settings include:

- **Username** (string): A user-defined name that the LLM may use to talk directly to the user.
- **LLM model** (GPT-4o/GPT-3.5/Llama 3 70B/ Mixtral 8x7B): Users can choose their preferred model, which affects flavor, quality, and response delay.
- **Voice** (male/female): Users can select the coaching voice. The most enthusiastic ones were selected from AWS Polly.

¹⁸<https://aws.amazon.com/polly/>

- **Temperature** (low/normal/high): This adjusts the randomness of the LLM responses, with values between 0 and 2¹⁹. Higher values may provide more creative output, while lower values result in more precise and factual texts. The available settings are 0.5, 1.0, and 1.3.
- **Frequency** (low/medium/high): Determines how often and how many times the LLM will be called during the run. The formula scales, so the intervals between the narrations stay approximately the same even when the goals are very different.
- **Private mode** (on/off): This setting ensures maximum privacy when on. In this mode, the data sent to the LLM only includes the run-segment data, goals, and themes. Essentially, the LLM provider only knows that someone is using the app to run and their performance, which does not significantly affect user privacy. When more runners are using the app simultaneously, it becomes nearly impossible to derive useful information. When the setting is turned off, the LLM can achieve better adaptability and overall response quality because of the larger context provided. In this mode, the LLM will also use the runner's name, start date/time, previous efforts, starting location's name²⁰, and current weather²¹.

There are also settings for each run separately:

- **Goal** (value): Distance in km or duration in minutes, with a minimum of 5 minutes or 1 km.
- **Intent** (base run/recovery run/long run/tempo run/race): Provides context on running motivation and can change the overall feel of the coaching.
- **Topic**: The main theme of the run, with around 50 predefined topics, such as "Passion," "Technique," and "Running History." In each monologue some information related to this topic will be included.

■ 5.7.3 Prompt Engineering

Prompt engineering is the process of refining the input prompts for a language model (LLM) to achieve optimal output. The general approach begins with a one-shot method, which provides only a single instruction. If this does not yield satisfactory results, a few-shot method is used, which includes instructions along with examples. The final step, fine-tuning the model, requires substantial computing power and extensive training data, such as coaching transcripts, which are difficult to obtain online. However, the one-shot approach gave very good results in this project, leading me to use it with adjustments to the prompts to achieve the best results. Most prompting

¹⁹ChatGPT uses the temperature set at 0.7

²⁰The display name of the location is reverse geocoded from GPS coordinates, using the OpenStreetMap API.

²¹Received from OpenMeteo API. Includes temperature, wind speed, and other indicators.

techniques revolve around deep reasoning, such as Chain of Thought, but they are not necessary here. All prompting text and logic are located in `/server/src/utils/prompts.ts`.

The basis for the prompts follows the *RISEN* framework[12], which stands for Role, Instruction, Style, Examples, and Name. In our case, the prompt construction includes the first three elements. Examples are usually used with the few-shot approach, and the Name is relevant when multiple models are used in the system. There are three types of messages saved in the LLM dialog history: system (defining the LLM's behavior), user (direct input/instructions), and assistant (LLM's responses). The crafted prompts always include the following components:

- **Role definition:** Informs the LLM about its role as an assistant audio coach, detailing what its speech should include, and specifying the desired attitude and emotion.
- **Style:** Describes that the output should be directly readable by a TTS model, avoiding emojis, and specifies which SSML tags can be used.
- **Instruction:** Simple sentences instructing the LLM to generate the corresponding structure or narration.
- **Context:** Provides relevant contextual information to include in the speech, such as
 - Run related information: goal, intent, topic
 - Personal information: username, starting location, date/time, ...
 - Last segments data: processed running data into readable segments over hundreds of meters, including pace and distance, to identify performance trends and current status.

Several prompting techniques and tactics can improve the quality of responses, as recommended in the OpenAI documentation²² ²³ and other research studies[12]. These were taken advantage of:

- **Clear messages:** Prompts are simple and direct, with no ambiguities, asking precisely what is needed.
- **Splitting into simple subtasks** (prompt chaining): First generating an outline, then the corresponding narrations.
- **Avoiding negation:** Prompts state what to do or avoid, rather than what not to do (e.g., avoid using emojis).
- **Clear separation:** Context is separated from instructions with a clear separator ("###").

²²<https://platform.openai.com/docs/guides/prompt-engineering>

²³<https://help.openai.com/en/articles/6654000%2Dbest%2Dpractices%2Dfor%2Dprompt%2Dengineering%2Dwith%2Dthe%2Dopenai%2Dapi>

- **Consistent input:** Prompts maintain a consistent format across different inputs.

An interesting idea to enhance response quality without extensive fine-tuning is *Priming*, which involves ending the prompt with the beginning of the desired output. In a study focused on the use of LLMs for activity coaching[2], they used a randomly selected coaching introduction as the priming factor and observed improved quality results.

■ 5.7.4 Potential issues

There are certain pitfalls that can arise when dealing with LLMs.

■ Hallucinations

These occur when the model generates factually incorrect or non-sensical information. Although they can be mitigated, hallucinations are inherent in the way LLMs are constructed[13]. In the scenario of this project, hallucinations do not pose substantial danger. In the worst case, users might receive factually incorrect narration or inconsistent text, leading to confusion but not serious issues. Higher temperature values increase the risk of hallucinations by causing the model to generate random token sequences and delaying stopping. To avoid this, I lowered the temperatures in the settings, though there is a slight chance hallucination could still occur, especially with smaller and older models.

■ SSML tags validation

The output must conform to the SSML tag structure used for adjusting TTS on AWS Polly. Messages must be enclosed in the <speack> tag, and other format rules apply to additional tags. To address this, the program uses a library to check if the output is valid SSML. If not, it attempts to repair it, and if that fails, it removes the SSML tags, sending plain text to AWS Polly. This ensures that incorrect LLM output format does not break the application while maintaining SSML format requirements when possible.

■ Prompt injections

To mitigate the risk of prompt injections, which involve overriding the application's prompt with user-provided instructions, conscious decisions were made during development to exclude long text areas allowing user input. Although custom user-defined backgrounds and expectations could enhance the application experience, they also introduce risks. Prompt injection could be exploited by attackers in various ways[14]. This is typically mitigated by a filtering layer that sanitizes user input. In the final application, the only user text input is the username, limited to 15 characters without spaces, which is restrictive enough to prevent LLM model jailbreaks.

■ 5.7.5 Measurements

I measured the performance and token usage for a single run to demonstrate concrete numbers for using the app. I evaluated three LLM models and compared their metrics. The measuring conditions for each model were consistent: a single 15-minute session with a specified intention and topic, private mode off, medium creativity, and medium frequency (four coaching monologues per session). For each API call to the LLM, the time from sending the request to receiving the response was measured, as well as the tokens used for both prompts and responses. Note that the token count does not directly correlate with the message length, as each model has a different tokenizer that segments the messages into varying token counts. Furthermore, external factors such as network conditions could have influenced the measurements.

The pricing for each LLM varies and is calculated based on the number of tokens (in millions) consumed and generated. For *GPT-4o*, the costs are \$5 for the input tokens and \$15 for output tokens. The costs of open-source models differ by provider, but *Llama 3 Instruct (70B)* in the nitro variant (faster throughput) costs \$0.90, and *Mixtral 8x7B Instruct* typically ranges from \$0.25 to \$0.50.

| LLM | Time | Prompt tokens | Result tokens | Prompt tokens costs | Result tokens costs | Single run costs |
|---------|------|---------------|---------------|---------------------|---------------------|------------------|
| GPT-4o | 4.77 | 5127 | 1031 | 0.0256 | 0.0154 | 0.0411 |
| Llama 3 | 2.12 | 5810 | 1256 | 0.0052 | 0.0011 | 0.0063 |
| Mixtral | 6.20 | 5646 | 1561 | 0.0013 | 0.0003 | 0.0017 |

Table 5.1: Performance, token usage and costs measurements for different LLMs (the average time in seconds and the costs in dollars)

Both GPT-4o and Llama 3 deliver high-quality responses, with GPT-4o having a slight edge. However, Llama 3 is significantly more cost-effective and faster. Mixtral’s performance is less consistent, probably due to its smaller model size, which struggles with larger contexts and produces variable output. In general, these results illustrate that LLM costs are not negligible and should be carefully considered when developing the business model for the application.

5.8 Deployment

This section outlines the deployment strategies for the primary components of the application, focusing on those directly managed and set up by me, rather than external services that oversee their own infrastructure.

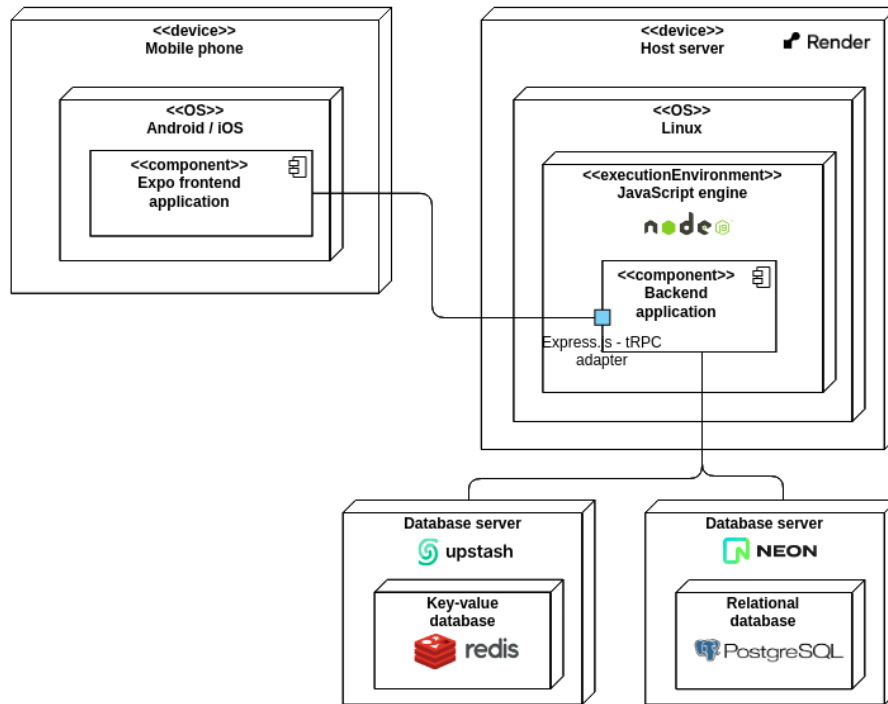


Figure 5.3: Deployment diagram

5.8.1 Mobile application

The development of the mobile application was facilitated using the Expo ecosystem, which provides a streamlined method for testing directly on devices. A single command within the codebase generates a QR code. Scanning this code compiles, bundles, and transfers the application to the device. This process requires the installation of the Expo Go app²⁴, which acts as a sandbox for the bundled JavaScript, allowing for immediate preview and testing. However, this set-up should not be considered sufficient for production-level testing.

As the project approaches full release, the deployment can be managed manually or through Expo Application Services (EAS)²⁵. EAS simplifies the build and submission process to app stores via command-line instructions, which, for example, obviates the need for a Mac to compile iOS versions. The

²⁴<https://expo.dev/go>

²⁵<https://expo.dev/eas>



```

) npm run start
> run-app@1.0.0 start
> expo start --port 19000

Starting project at /home/lmkn/repos/run-app/client
env: load .env
env: export EXPO_PUBLIC_TRPC_URL EXPO_PUBLIC_CLERK_PUBLISHABLE_KEY
Starting Metro Bundler

[QR Code]

> Metro waiting on exp://172.20.10.7:19000
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

```

Figure 5.4: Output of Expo start command

demo versions of the apps can be distributed for internal testing through Google Beta and iOS TestFlight. Paid developer accounts for both OS's app stores are required, but Expo handles the whole application submission process. Similarly, EAS allows for app updates and provides analytics for further insight.

■ 5.8.2 Hosting services

The server-side components, including the Node server, Redis instance, and PostgreSQL database, are hosted on platforms that were chosen for their reliability and ease of integration. For instance, both database providers are relied upon by various Vercel services.

The Node server is deployed on Render²⁶, which supports both plain codebases and containerized applications. The deployment process is automated through continuous delivery that triggers a redeployment on each commit to the main branch. The deployment is then managed by the following simple commands from `package.json`:

- **build:** `npm install && npx tsc` (compile TypeScript to JavaScript)
- **start:** `node dist/index.js` (start the node express server)

For databases, PostgreSQL is managed by Neon²⁷ and Redis by Upstash²⁸. Both platforms specialize in their respective technologies and offer fully managed services. They provide interfaces for easy instance creation and include comprehensive monitoring dashboards that aid in traffic and performance management.

²⁶<https://render.com/>

²⁷<https://neon.tech/>

²⁸<https://upstash.com/>

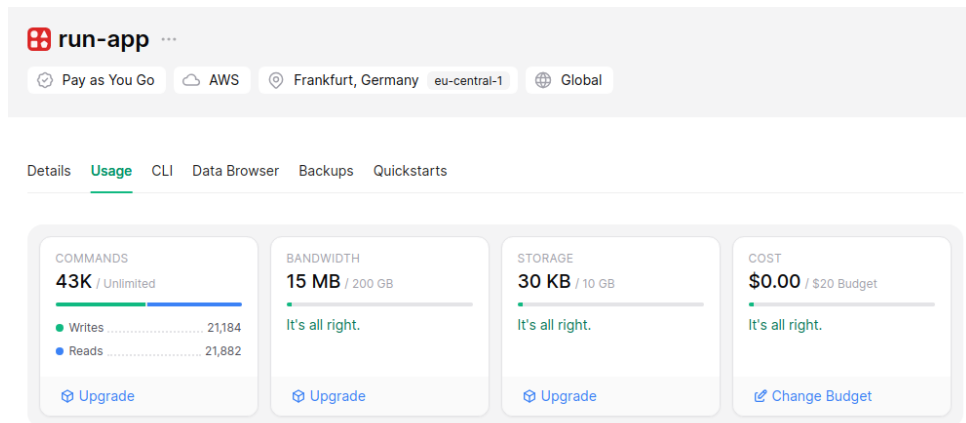


Figure 5.5: Redis monitor dashboard on Upstash

These server-side components are hosted on serverless platforms that are activated only upon requests, thereby significantly reducing operational costs due to their *pay-for-what-you-use* pricing models. Before the official launch, it may be beneficial to consolidate these services under a single provider like AWS or Render, to further reduce costs and simplify system management as the application scale. The host migrations in the code are extremely simple by just changing URLs in the `.env` file, without any intervention in the code logic.

5.8.3 Scaling

As the user base and the number of simultaneous connections increase, scaling the back-end system becomes critical to prevent overload and potential service interruptions. Render offers automatic scaling²⁹ options by incorporating a load balancer that distributes incoming requests across multiple server instances. While our current external APIs, such as those for authentication and OpenAI, have rate limits, they are generally capable of handling high traffic volumes, so there should be no problem regarding those.

Over time, transitioning out of serverless setups to containerized systems might be advantageous due to the latency associated with the startup times of serverless services. This setup allows for running on any high-compute server, either through a trusted provider or self-hosting, which can reduce operational costs in the long term, although it requires a significant initial investment. For highly successful applications, more complex orchestration techniques, such as Kubernetes and deploying services on the edge (i.e. closer to the user), can enhance performance significantly.

²⁹<https://docs.render.com/scaling>

Chapter 6

Testing

To verify the quality and usability of the application, several user tests were conducted. The primary focus was on usability testing, in which regular users tested the application as they would in normal usage scenarios. This testing process was summative and aimed at evaluating the prototype in its final state. While it is also possible to conduct tests during the development phase to influence the process more directly, the main goal here was to assess whether the final implementation was well-made. The objective was to measure the overall user response, evaluate whether ergonomics and UI were simple and intuitive, and determine if users felt that any features were missing.

6.1 Methodology

The testing methodology used was the observation method, in which users tried the application with minimal assistance. This process involved concurrent thought listing, where potential users performed tasks and were encouraged to verbally express their thoughts freely. The entire testing session was audio recorded to allow for thorough analysis of user feedback and identification of any issues.

Each test followed a similar procedure. I personally met with the participants, provided a brief overview of the application, explained the tasks they should perform, and encouraged them to express their thoughts openly. The participants then carried out the tasks, voiced their ideas and received help when needed. After completion of the tasks, we discussed possible improvements and their overall experience with testing.

The test case tasks included:

1. Creating an account and logging into the application
2. Exploring the user interface
3. Adjusting settings on the settings screen
4. Setting parameters for the upcoming run
5. Completing the run while listening to the coaching audio
6. Reviewing the running history and run details screens

6.2 Test group

A small sample size of three testers was used, each bringing a slightly different perspective to the use of the application. The only requirements were some running experience and the ability to understand English.

| | Gender | Age | Running experience | Additional information |
|---|--------|-----|--------------------|---------------------------------|
| 1 | Male | 29 | Beginner | Previously used Strava |
| 2 | Male | 21 | Intermediate | Knowledgeable in SW development |
| 3 | Female | 41 | Beginner | |

Table 6.1: Test participants

It is important to note that the testing group was very small and lacked variety. To draw scientifically based conclusions about the quality of the application, more extensive testing with a diverse user base would be necessary. Subject bias may also have influenced the results, as the testers had a relationship with me, which could have affected the candor of their critiques. Conducting larger-scale long-term testing with a more diverse group of users before the application's release would provide a more accurate overview of its effectiveness.

6.3 User testing results

Overall, the tests yielded positive results. There were no performance issues or crashes during testing. Testing on their devices revealed some new problems, particularly with Android devices, which behave slightly differently from the iOS and Android emulators used in initial tests. I compiled a list of encountered issues and proposed improvements, sorted by severity / priority, including the expected difficulty of implementing them.

| Severity | Difficulty | Issue |
|----------|------------|--|
| High | Medium | Unlistenably quiet coaching audio on Android |
| Medium | Low | Minor UI glitches (eg., input flickering) |
| Medium | Low | Lack of confirmation when terminating the run |
| Low | High | LLM not providing factually correct data (tester disagreed with a running tip provided by the LLM) |

Table 6.2: Uncovered issues or bugs found throughout user testing

| Priority | Difficulty | Suggestion |
|----------|------------|--|
| High | Medium | Support for application running in background and in locked mode |
| High | Medium | Add more personalization |
| High | Medium | Make the LLM setting options more understandable for regular users (or remove them) and refine the setting hints |
| High | Medium | Add more authentication options (tester expected Google sign-up option) |
| High | Low | Fine-tuned support for headphones |
| Medium | High | Add a way to interact with the coach (eg., speech to text) |
| Medium | High | Create multiple coach personalities with different name, characters, and tuned LLMs |
| Medium | High | More seamless integration with music players (could support dynamic playlists picked by AI) |
| Medium | Medium | Add more animations throughout the application |
| Low | High | Czech language support, including for coaching |
| Low | High | Coach could assist with real-life navigation |
| Low | Medium | Dark mode support |
| Low | Medium | Integration with Google fit (save data to the app) |
| Low | Low | Add a setting for weight/height to compute burned calories more accurately |

Table 6.3: Suggestions, proposed improvements and new features from the testers

The aspect in which the application truly excelled was in maintaining high levels of engagement among users. There was a noticeable "wow" effect when test subjects listened to the coaching messages. They often found the coaching humorous and even laughed at the jokes provided. The participants especially appreciated the personalization aspect, where the coach addressed them by name and mentioned relevant information, such as the current outside temperature. This personalized approach significantly boosted their motivation, indicating that this feature should be expanded in future iterations of the app. For instance, one user suggested including favorite keywords or topics for each user, which the coach could then incorporate into the messages.

Users reported that the app was quite simple to use and did not identify significant issues with regard to its design and core functionality. From their point of view, everything worked well and they struggled to find any notable shortcomings. They also praised the clean look of the interface and found it very understandable, with the hints in the settings being particularly helpful.

Overall, the testing indicates that the features included in the prototype were well-designed and effectively implemented. Feedback was overwhelmingly positive, with suggestions for new features far outweighing any issues with the current state of the application. Most of the suggested improvements were seen as enhancements rather than essential fixes, indicating that the current core functionality of the app is complete enough.

6.4 Prompt testing

Prompt testing is important for effectively integrating LLMs. It enables the creation of accurate and effective coaching instructions and improves the reliability of LLM responses. Using the evaluation methods, can overcome the limitations of subjective quality assessments. Aside from the general prompt engineering principles, various strategies can be employed to refine the prompts and ensure optimal performance and user experience in this project.

For example, *regression testing*[15] ensures that new changes do not disrupt existing features. *Contextual coherence testing* maintains logical flow throughout a run, ensuring that the prompts reference previous narrations accurately. *Stress and edge case testing* evaluates system performance under high-usage scenarios and rare conditions, identifying potential bottlenecks and ensuring robustness. *A/B testing* compares different prompt versions to optimize user engagement and satisfaction.

Measuring the quality of responses is challenging due to its subjective nature. While other LLM models could be used for evaluations, a high-quality data set would be required. User testing is another option, but requires consistent surveying, time, and human resources.

Frameworks such as OpenAI Evals¹ can be used to validate and benchmark the results, potentially further improving the responses. Another related and valuable tool is Promptfoo². Users can define metrics to automatically score the output, ensuring objective assessments. It enhances efficiency by accelerating evaluations with caching, concurrency, and live reloading. Although not essential for the program's core functionality, these frameworks offer additional layers of quality assurance.

The prompt testing methodologies and tools illustrate how prompts can be rigorously tested and refined, leading to reliable high-quality coaching messages. Combining automated and manual testing techniques optimizes the integration of LLM for performance, user experience, and cost-effectiveness. Employing these strategies would certainly be the next step in the development process of this application.

¹<https://github.com/openai/evals>

²<https://github.com/promptfoo/promptfoo>

Chapter 7

Conclusion

7.1 Evaluation

Throughout this project, I navigated all stages of software development, from analysis and design to implementation and testing. This was the most complex project I have ever worked on. It involved precise coordination of multiple components without causing any user and performance issues. I started by creating a basic user interface to track location and to experiment with fetching data from external APIs. Expo proved invaluable for quickly starting the project and testing in an emulator. I then integrated the narration functionality for a seamless run experience, gradually adding features like settings and running history.

In future projects, I would rely less on external libraries as I spent considerable time resolving version conflicts. A forced upgrade to the Expo SDK once broke many libraries, requiring waiting for fixes outside my control. tRPC also presented problems, requiring version synchronization between client and server and sometimes producing unclear errors linked to the protocol itself or hosting issues, where the initial provider randomly blocked requests from my IP address. One more challenge was that the server initially supported only one connection at a time, which required the addition of an authentication flow, server authorization, and Redis database for session management. Despite these challenges, I stabilized the app to work relatively reliably. Using TypeScript across the codebase also saved significant debugging time and preempted many issues that I probably would not have noticed without it.

During development, some advances in the field of AI were happening, notably the releases of Llama 3 and GPT-4o. These models changed the initial assumptions I had about integrating LLMs. They helped significantly reduce the cost and the coaching audio delay, with only minor server code adjustments.

The finished application is more than a complete prototype. After sufficient testing and verification that it works stably, it could even be released as a working application as is. I managed to create a unified UI look for the application that is consistent and intuitive to use. The user running data are persistent in a database and tied to the user account in case of changing devices. Reliable technology was chosen so that the app would be easily

- Pairing the app with a heartbeat monitor could enable it to detect potential dangers, such as sudden stops that indicate injury, and automatically call for help or prompt user action.
- Enhancing the user interface with gamification and social interaction options could further engage users.

7.3 Future plans

I believe that this application could be successful due to its novel concept of integrating dynamic coaching narration and the quality of its execution. However, releasing the app is a complex process. It requires applying to Google Play and the iOS App Store with a developer account and passing their acceptance processes. Before release, a business model should be established to cover hosting and LLM service costs. This could involve offering limited free usage, such as one run every two days, with unlimited access behind a paywall.

The current LLM technology is very powerful, but has not yet been widely integrated into many potential applications. This project can serve as a pioneer in the integration of LLM technology into the fitness coaching sector. The potential is clear: LLMs can provide a much more personalized user experience that would be nearly impossible to achieve with traditional programming. As AI technology continues to develop rapidly, the possibilities for expanding this app are only increasing. With further advancements, this application can grow and incorporate more exciting features to attract even more users, solidifying its place in the market.



Bibliography

- [1] Dorothea Gute, Stephan Schlögl, and Aleksander Groth. “Keep on Running! An Analysis of Running Tracking Application Features and Their Potential Impact on Recreational Runner’s Intrinsic Motivation”. In: *Human-Computer Interaction. User Experience and Behavior*. Ed. by Masaaki Kurosu. Cham: Springer International Publishing, 2022, pp. 359–373. ISBN: 978-3-031-05412-9.
- [2] Narayan Hegde et al. “Infusing behavior science into large language models for activity coaching”. In: *PLOS Digital Health* 3.4 (Apr. 2024), pp. 1–15. DOI: [10.1371/journal.pdig.0000431](https://doi.org/10.1371/journal.pdig.0000431). URL: <https://doi.org/10.1371/journal.pdig.0000431>.
- [3] M. Janssen et al. “Who uses running apps and sports watches? Determinants and consumer profiles of event runners’ usage of running-related smartphone applications and sports watches”. In: *PLoS ONE* 12 (2017), e0181167.
- [4] Paolo Barsocchi. *Detecting Proximity with Bluetooth Low Energy Beacons for Cultural Heritage*. 2023. URL: https://www.researchgate.net/figure/Overview-of-the-React-Native-architecture_fig8_355670705 (visited on 05/03/2024).
- [5] Gustav Tollin and Lidekrans Marcus. “React Native vs. Flutter: A Performance Comparison Between Cross-Platform Mobile Application Development Frameworks”. Bachelor’s thesis. Linköping University, Department of Computer and Information Science, 2023, p. 10.
- [6] *Artificial Analysis*. URL: <https://artificialanalysis.ai/> (visited on 05/16/2024).
- [7] Xiaodong Wu, Ran Duan, and Jianbing Ni. “Unveiling security, privacy, and ethical concerns of ChatGPT”. In: *Journal of Information and Intelligence* 2.2 (2024), pp. 102–115. ISSN: 2949-7159. DOI: <https://doi.org/10.1016/j.jiixd.2023.10.007>. URL: <https://www.sciencedirect.com/science/article/pii/S2949715923000707>.
- [8] Sachin Mehta et al. *OpenELM: An Efficient Language Model Family with Open Training and Inference Framework*. 2024. arXiv: 2404.14619 [cs.CL].

- [9] Jakob Nielsen. *10 Usability Heuristics for User Interface Design*. Updated January 30, 2024. 1994. URL: <https://www.nngroup.com/articles/ten-usability-heuristics/> (visited on 05/04/2024).
- [10] Christine Bauer. “On the (In-)Accuracy of GPS Measures of Smartphones: A Study of Running Tracking Applications”. In: Dec. 2013. DOI: 10.1145/2536853.2536893.
- [11] Christopher Hide, Terry Moore, and Martin Smith. “Adaptive Kalman filtering for low-cost INS/GPS”. In: *The Journal of Navigation* 56 (Jan. 2003), pp. 143–152. DOI: 10.1017/S0373463302002151.
- [12] Ayush Thakur. *The Art of Prompting: Unleashing the Power of Large Language Models*. Mar. 2024. DOI: 10.13140/RG.2.2.18470.54089.
- [13] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. *Hallucination is Inevitable: An Innate Limitation of Large Language Models*. 2024. arXiv: 2401.11817 [cs.CL].
- [14] Kai Greshake et al. *Not what you’ve signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection*. 2023. arXiv: 2302.12173 [cs.CR].
- [15] Wanqin Ma, Chenyang Yang, and Christian Kästner. *(Why) Is My Prompt Getting Worse? Rethinking Regression Testing for Evolving LLM APIs*. 2024. arXiv: 2311.11123 [cs.SE].



Appendix A

List of abbreviations

LLM

Large Language Model

AI

Artificial Intelligence

API

Application Programming Interface

TTS

Text-to-speech

tRPC

Typescript Remote Procedure Call

NRC

Nike Run Club

SW

Software

SSML

Speech Synthesis Markup Language

URL

Uniform Resource Locator

UI

User Interface

ORM

Object Relational Mapping

Appendix B

Application demonstration

GitHub repository: <https://github.com/lmnek/run-app>

Video demo: <https://youtu.be/RuJs6ZCeMoo>

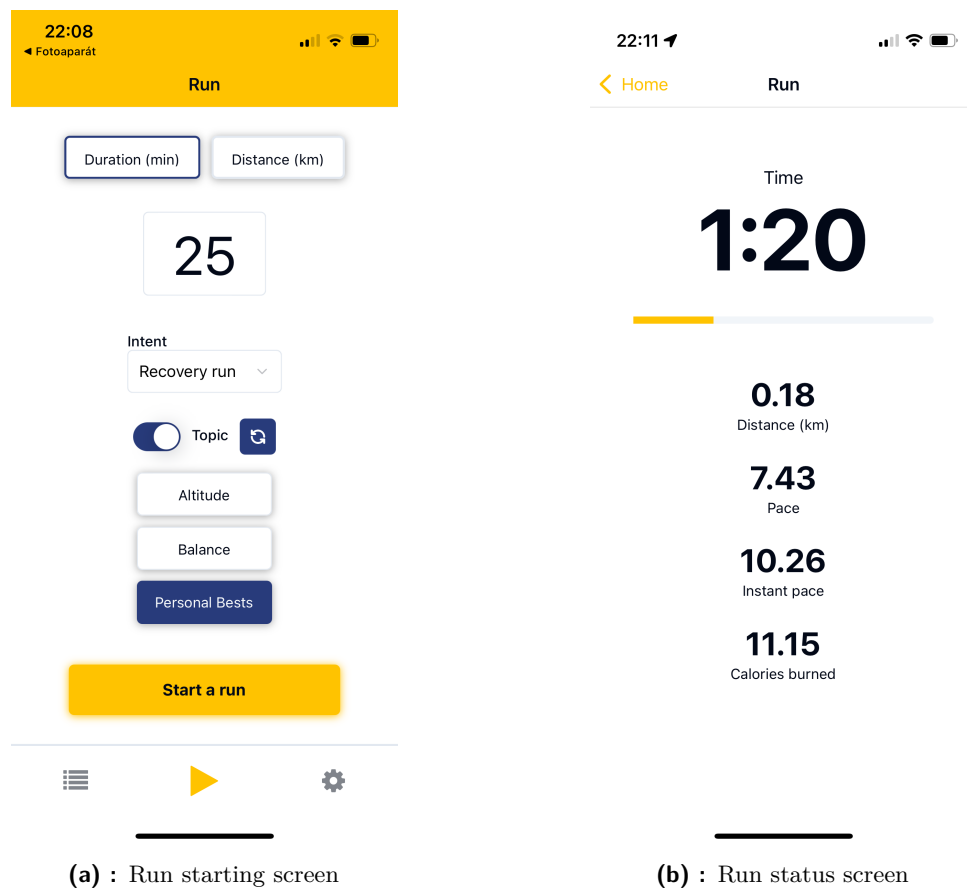
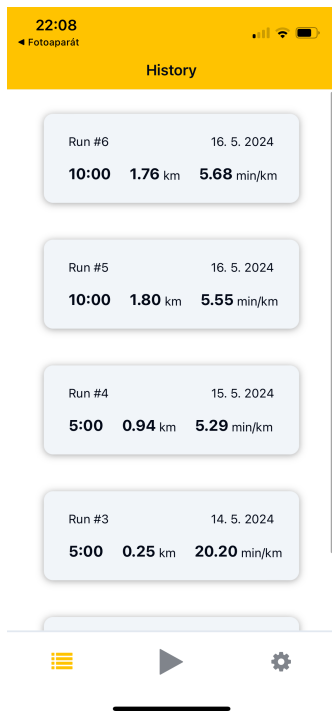
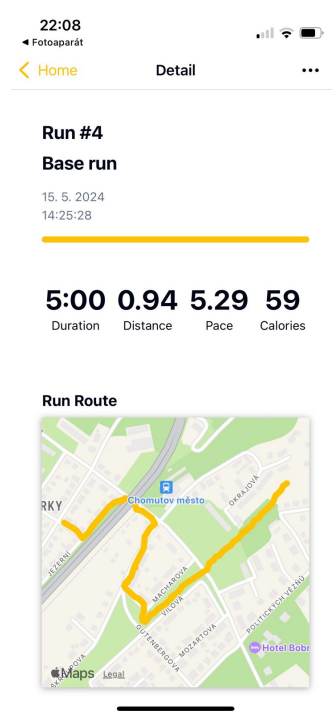


Figure B.1: Application screenshots of running screens

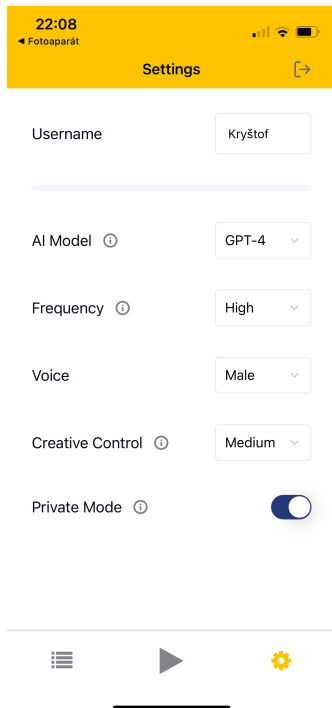


(a) : List of users previous runs

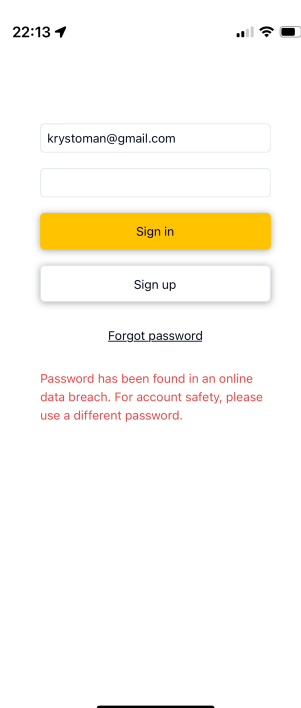


(b) : Specific run's details

Figure B.2: Application screenshots of running history and run detail



(a) : Application settings



(b) : Login/Sign up auth form

Figure B.3: Application screenshots of settings and authentication

Appendix C

LLM dialog transcript

Example of application's LLM message history for a single run, using GPT-4o, high frequency, medium creativity setting, and private mode turned off:

System:

```
Role: You are an assistant audio coach accompanying a runner.
      You dynamically adapt to the specific users available data.
      During the run, you'll join in many times and reflect on
      live data, like pace and covered distance. Inform about
      important milestones and different stages of the run.
      Discuss segments data as approximate values and trends.
      Motivate the runner, provide tips, and offer encouragement.
      Be kind, excited, and occasionally funny. Integrate as much
      user data as possible, especially the not yet mentioned data
      . Ensure your responses smoothly transition from one to the
      next, with an intro, main message (the longest), and a
      teaser for the next part.

###
Context:
- Coach entrance count: 5
- Goal: 10 min (mention it!)
- Intent:Base run
- The main topic: Famous Runners
-> center all your monologues around this!
- Runners name: Christoph
- Last run: { 5 run, on 5/16/2024, 5:13:51 PM,1.80 km, 5.55 min/
  km, 10.0 min, intent: Recovery run, topic: Running Technique
  }
- Start date/time: 5/16/2024, 5:47:46 PM
- Weather: [temperature_2m: 16.4 C ; relative_humidity_2m: 59%;
  apparent_temperature: 13.4 C ; is_day: 1; precipitation: 0mm
  ; rain: 0mm; showers: 0mm; snowfall: 0cm; weather_code: 3wmo
  code; cloud_cover: 100%; wind_speed_10m: 17.5km/h; ]
- Starting location is Chomutov I, Chomutov, okres Chomutov,
  steck kraj, 430 04, Czechia (type residential)
Crude outline for your entrances: {
' ##### Entrance 1
- **Initial Encouragement**
- Weather mention: 16.4 C , perfect running condition!
- Last run progress
- Training goals: 10 min mark
```

```
#### Entrance 2
- Famous Runner: Usain Bolt
- Incredible sprint speeds
- Bolt's training secrets
- Quick highlight: world's fastest man

<Delete the rest of the outline to make the appendix shorter>
...

}
Style: Write text directly processable by AWS Polly neural text-
to-speech model. Avoid emojis. Use SSML tags for better
emotion expression. Allowed SSML tags: <break>, <p>, <s>, <w
>, <prosody> (only volume and rate). <speak> tag is
prohibited.
```

User:

```
Create the 1. audio entrance, runner is starting.
```

Assistant:

```
<break time="1s"/>
Hi Christoph!
<p>Perfect day for a run, isn't it? The temperature is just
right at 16.4 degrees Celsius, and you've got a nice breeze
going.</p>
Last time you did a great job on your recovery run.
<s>Today, let's aim for that 10-minute mark with a solid base
run.</s>
Let's get going and enjoy the journey!
<p>Stay tuned, because I've got some cool stories about famous
runners coming up!</p>
```

User:

```
Create the 2. audio entrance.

Already run duration: 2:08
Last segments: [
- From 0 to 201 metres; Pace 7.69 min/km; 92.8 secs
- From 201 to 270 metres; Pace 8.70 min/km; 36.0 secs
]
```

Assistant:

```
<break time="2s"/>
Hey Christoph! You're two minutes in.
<s>You're doing great!</s>
<p>You've covered over 270 meters so far.</p>
Pace has varied a bit, but that's normal as you settle into your
run.</p>
Let's talk about Usain Bolt, the world's fastest man! He's known
```

for his incredible sprint speeds, clocking in at 9.58 seconds for the 100 meters.
<p>Bolt's secret? Intense training, consistent routine, and a positive mindset.</p>
Keep that in mind as you keep pushing forward. You're on your way!
<p>In the next stretch, I'll tell you about a truly inspiring marathon runner.</p>

User:

Create the 3. audio entrance.

Already run duration: 4:15
Last segments: [
- From 270 to 467 metres; Pace 8.59 min/km; 101.5 secs
- From 467 to 502 metres; Pace 12.27 min/km; 25.8 secs
]

Assistant:

<break time="2s"/>
You're doing fantastic, Christoph!
<s>Four minutes in and over 500 meters covered!</s>
<p>Your pace has had some fluctuations, but remember, it's all about maintaining that steady rhythm in a base run.</p>
Now, let's talk about Paula Radcliffe, a marathon legend.
<s>She set a world record in 2003 with a time of 2:15:25.</s>
How did she achieve such greatness?
<p>Her training was rigorous, focusing on building endurance and overcoming challenges, including injuries and setbacks.</p>
Keep her determination in mind and channel that spirit into your run.
<p>Next up, I'll share insights about another legendary marathoner.</p>
Keep going strong!

User:

Create the 4. audio entrance.

Already run duration: 6:23
Last segments: [
- From 502 to 685 metres; Pace 6.29 min/km; 69.0 secs
- From 685 to 841 metres; Pace 6.26 min/km; 58.6 secs
]

Assistant:

<break time="2s"/>
All right Christoph, you're past the six-minute mark and cruising!
<s>You've covered more than 840 meters now.</s>

```
Your pace in the last segments was fantastic, right around 6.3
minutes per kilometer.
<p>Nice and steady!</p>
Let's talk about Eliud Kipchoge, the man who broke the two-hour
marathon barrier. His pacing strategy is legendary,
maintaining a consistent speed throughout the run.
<p>His focus and discipline are truly inspiring.</p>
So, channel some of that Kipchoge magic as you keep pushing
forward.
<p>You're doing great, and the finish line is getting closer!</p>
>
Get ready for that final motivational push in a bit!
```

User:

```
Create the 5. audio entrance.

Already run duration: 8:30
Last segments: [
- From 841 to 1047 metres; Pace 4.83 min/km; 59.7 secs
- From 1047 to 1247 metres; Pace 5.14 min/km; 61.7 secs
]
```

Assistant:

```
<break time="2s"/>
You're almost there, Christoph!
<s>You've just hit the 8:30 mark.</s>
<p>Your pace has been stellar recently, around 5 minutes per
kilometer for the last segments.</p>
Way to pick up the speed in the final stretch!
<p>You've been maintaining a great rhythm just like last time.</p>
p>
Keep your eyes on that 10-minute goal, it's right around the
corner.
<p>Stay strong and finish with the same determination and
consistency you've shown throughout.</p>
Let's bring it home!
<s>You're doing an amazing job.</s>
```




Appendix D

Software usage declaration

According to *Methodological guideline No. 5/2023*¹, the following software was used in the development of this thesis:

- ChatGPT (OpenAI)²: for paraphrasing suggestions
- Writefull³: for grammar and spell checking
- LTEX LS (L^AT_EX language server)⁴: for grammar and spell checking

¹<https://www.cvut.cz/sites/default/files/content/d1dc93cd-5894-4521-b799-c7e715d3c59e/en/20231003-methodological-guideline-no-52023.pdf>

²<https://chatgpt.com/>

³<https://www.writefull.com/>

⁴<https://github.com/valentjn/ltex-ls>