College of Engineering
COMP 491 – Computer Engineering Design
Project Proposal

RunWithMe

Fall 2025

Participant Information:

Name	ID	Email	Phone
1 Ege Erdem Özlü	80481	eozlu21@ku.edu.tr	+90 532 582 28 21
2 İdil İşsever	80447	issever21@ku.edu.tr	+90 546 536 26 16
3 Tuna Çimen	80418	tcimen21@ku.edu.tr	+90 530 010 71 11
4 Mina Durhasan	79776	mdurhasan21@ku.edu.tr	+90 536 560 56 35

Project Advisor: Öznur Özkasap

Abstract

Running is one of the most preferred exercises among adults because all it requires is comfortable clothes and sports shoes. Independent of the time of day and without a budget, running is easily accessible. Since it doesn't require professional help or personal trainers, most beginner and mid-level runners try to improve themselves for a healthier lifestyle, or just for fun. Professionals and trainers also make use of tracking tools, many times paired with training/tracking hardware.

Despite the popularity of running, the lack of a dedicated social platform hinders community building, motivation, and safety for solo runners. RunWithMe addresses this by creating a unified application for socializing, direct communication, and building community.

RunWithMe utilizes Flutter (frontend) and SpringBoot with Kotlin (backend). Its stack includes Postgres, PostGIS, MongoDB, and Redis caching. It implements KNN route matching with FastDTW, utilizing open-source routing APIs. The app also features an AI assistant, supported by automatic OpenAI documentation for MCP.

RunWithMe will be a cross-platform mobile app that will keep a detailed track of runners' progress and performance while connecting people who enjoy running by enabling them to discover, share, and match with others on nearby routes.

Core features will include route discovery, live run tracking, an AI chatbot that the user can give commands to, and a matchmaking system that will pair runners based on route overlap, pace, and time availability. Privacy controls, token-based security, and battery-efficient background tracking will ensure a safe and reliable user experience.

We want to make running easier and more social. Our goal is a simple, private app that tracks progress, matches runners, and helps people stay motivated and safe.

TABLE OF CONTENTS

Section 1 Introduction

1.1 Concept

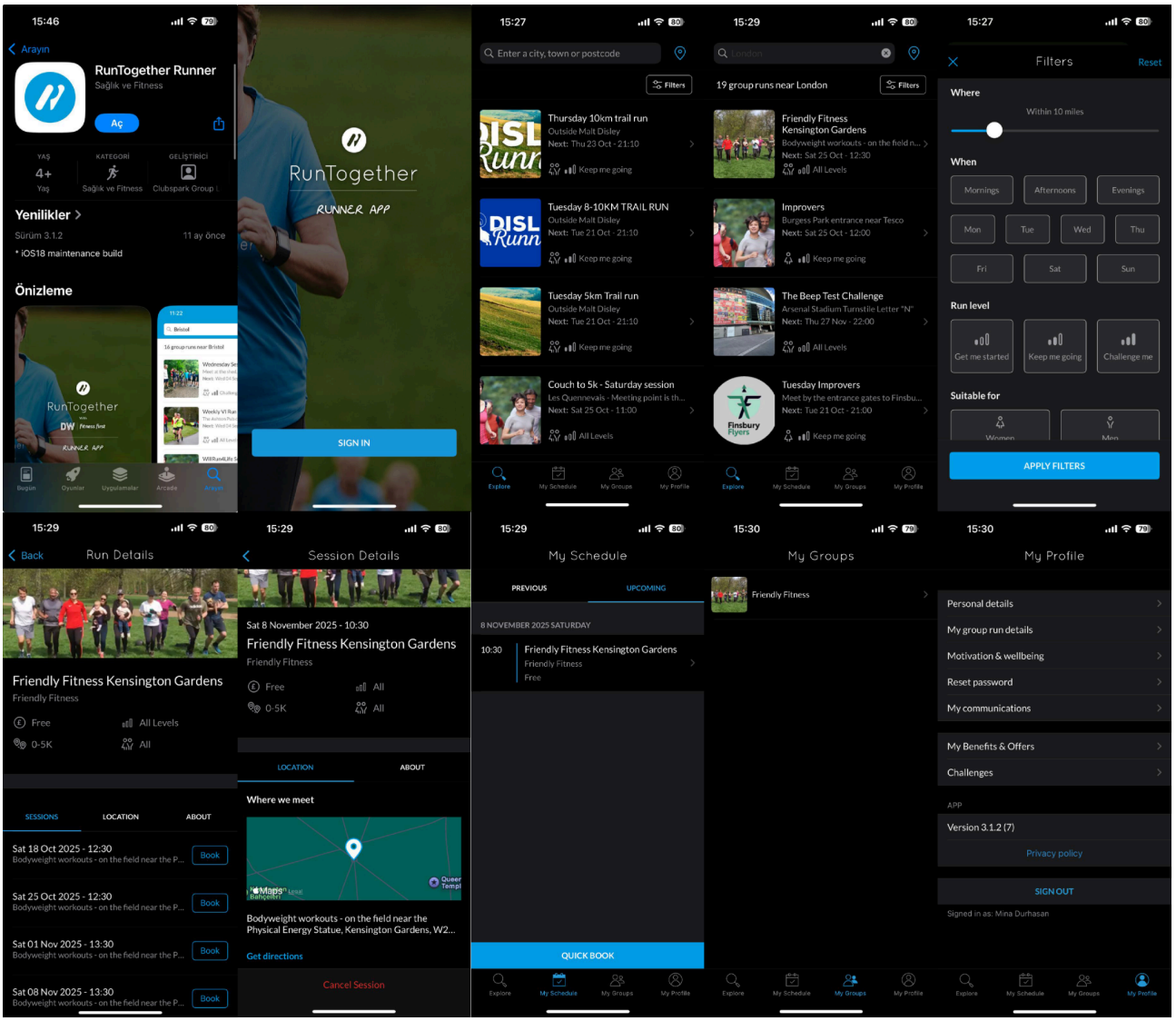
RunWithMe is a simple mobile app that helps people run more and run together. It lets users discover nearby routes, track runs with battery-efficient logging, message one-on-one, and match with compatible runners by route overlap, pace, and time. The app is built with Flutter; a Kotlin/Spring Boot REST API handles all data and security using JWT and role-based access. PostgreSQL with PostGIS stores geospatial and user data; MongoDB is used for direct messages. MapLibreGL renders maps with offline caching, and OpenRouteService provides routing. Real-time chat uses STOMP over WebSocket, with push notifications via FCM and APN. A separate Python MCP service suggests routes, summarizes progress, and answers questions. Everything runs in Docker with Compose for bypassing the need for Kubernetes; repositories are split for clean development and CI/CD. The outcome is a fast, reliable, and secure app that unifies tracking, discovery, social matching, and AI help.

1.2 Objectives

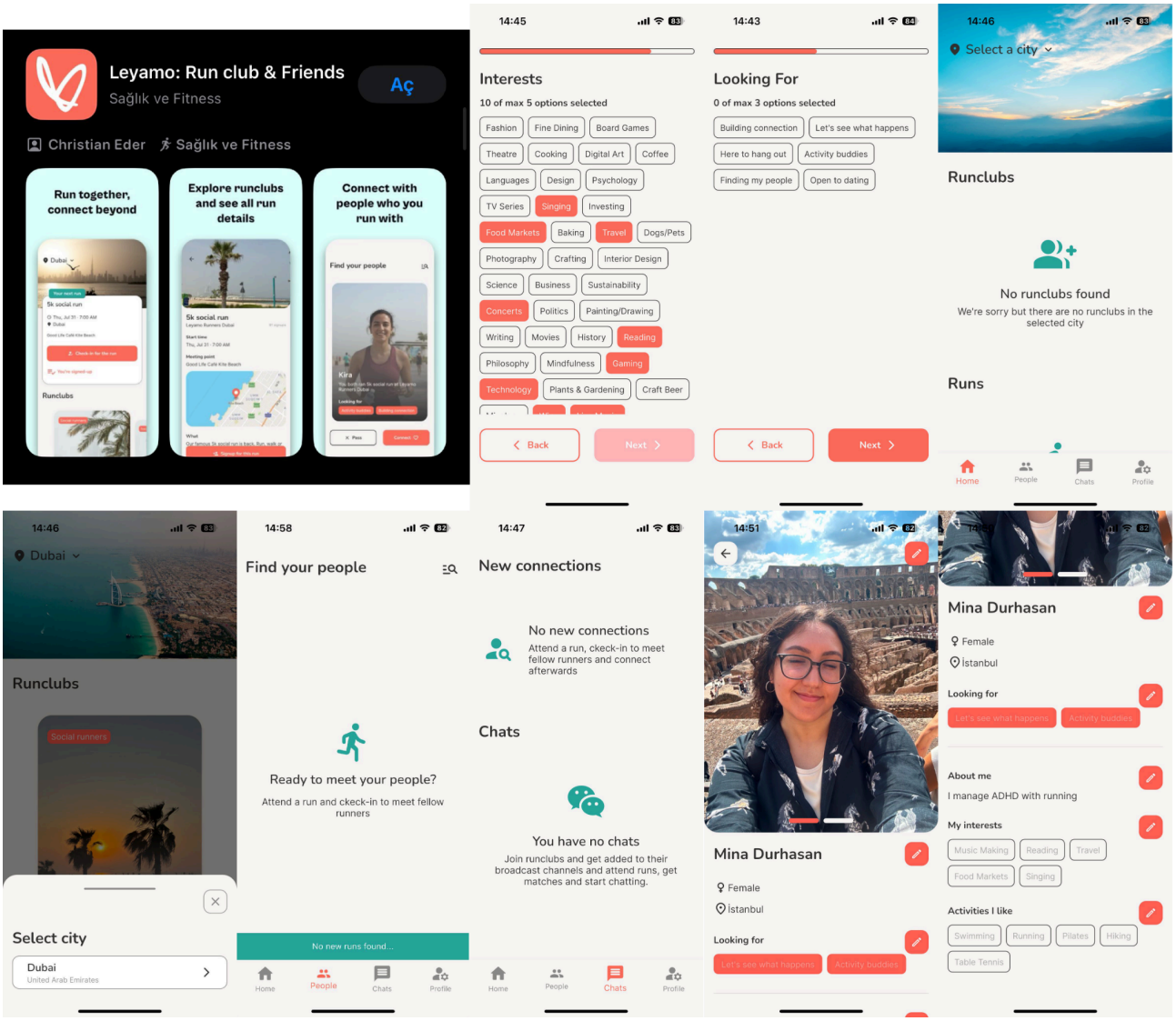
This app is aimed to be a go-to app for runners who seek to record their exercise and match with other runners who run similar routes or who have similar goals to them. The app is open to both personal use for performance tracking and social use to match and run together with others who have similar motivations and are running from/to nearby locations.

1.3 Background

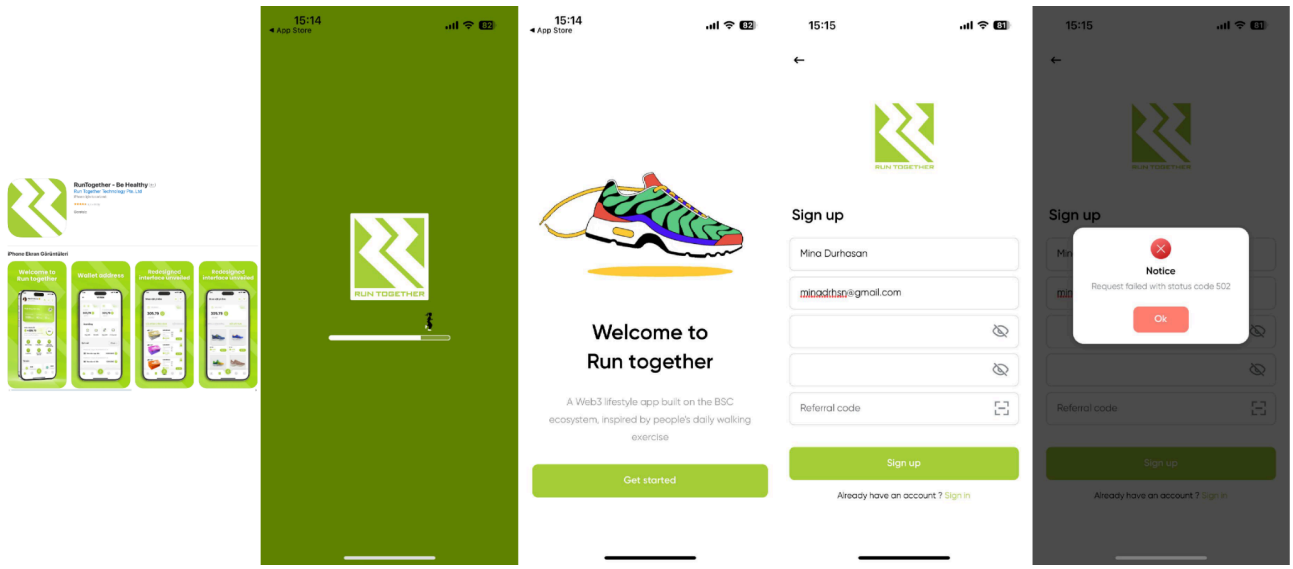
- **RunTogether Runner:** This app is the mobile application for the RunTogether community. Sign-up is only allowed through the website, and after signing up, the user can log in to the mobile app. The app has a handy interface and a wide range of search options. Since RunTogether is a UK-based organization, there aren't many running groups outside of the UK. The running group meeting location and start time details are well-documented and easy to understand. By just tapping "Book", users can notify the group leader that they'll join. The app is well organized in general, but lacks social profiles, non-organizational matching, globality, and personal tracking.



- **Leyamo:** This app has the elements of a dating app. There are many interests to choose from, but for some reason, users can only choose up to 5 interests. There are profile pictures, location, gender, and about me sections on the profile panel. The only city option for searching running clubs is Dubai, and there are no results for that. Connections with other users can only be made after attending a run. Since there are no runclubs listed, the app is basically useless.



- **Runtogether Be Healthy:** From the storefront, it is seen that this app is focused on selling products that are used by runners. The sign-up function basically didn't work, so we couldn't observe the app from the inside.



Section 2 S/T methodology and associated work plan

2.1 Methodology

The Approach

The overall approach can best be described by going through the tech stack that will be employed in-depth. Since there are many connected parts to this project, the following is a best-effort approach to explain the various software and tools that are planned to be used.

For persistence, we will be using a combination of SQL and NoSQL databases. For all but direct messaging (DM) functionality, the choice will be to use PostgreSQL alongside the PostGIS extension to efficiently manage geographical data, and properly store other information, such as run logs, posts, and follower/friend information of a user in normalized tables. For DMs, we will be using a NoSQL database, namely MongoDB, instead, as the consensus favors document-based storage of text messages for efficiency.

The database layer will never directly communicate with the frontend for several valid reasons, such as security, data integrity, maintainability, scalability, and access control. Instead, a REST API will provide a secure interface for the frontend. The programming language of choice for the backend is Kotlin due to its strong type system, concise syntax, and full interoperability with Java, which together reduce runtime errors and improve developer productivity. Combined with the Spring Boot framework, Kotlin enables rapid development of robust, maintainable REST APIs with built-in dependency injection, structured configuration, and seamless integration with databases and authentication layers.

Per expected standards, the entities in the PostgreSQL database will be represented as Kotlin classes through an ORM layer, specifically Spring Data JPA. Any document-based entities in MongoDB will also be represented in a similar fashion. A data transfer object (DTO) layer will also be utilized for further decoupling of the data models from the domain.

The backend will manage user authentication and access control to ensure secure interactions. All users will authenticate through token-based mechanisms, JWT to be exact, to maintain stateless sessions and protect API endpoints. Authorization will follow a role-based structure to ensure that users can access and modify only their own data, while higher-privilege actions such as editing shared routes or viewing private activities will be restricted to verified and permitted users.

The backend APIs will be documented using the OpenAPI (Swagger) specification via the springdoc-openapi library. Documentation will be generated automatically from controller annotations and served locally during development, allowing developers to view and test endpoints through a lightweight web interface. This ensures up-to-date API references without needing external hosting or manual documentation maintenance. This is especially necessary in the environment that we will be working in, where at least two of the students are focused on the frontend, and mostly only one other student will be developing the backend, so any potential usage of the API calls on the frontend will hopefully necessitate significantly less communication from both parties, allowing asynchronous development.

The frontend development framework of choice for this multiplatform application will be Flutter. Flutter is written in the Dart programming language. Dart's strong, sound typing will enable safer parallel work, clearer contracts, and easier verification of REST API responses and their downstream processing. UI implementation will leverage open-source Flutter examples and component libraries to accelerate a polished, consistent interface.

Map rendering and routing will be handled using MapLibreGL, an open-source, vector-based map framework that provides smooth zooming, fluid movement, and efficient parallel rendering through vector tile processing. Its support for offline caching and full map style customization makes it ideal for an application that depends heavily on geospatial interaction. To implement routing and navigation, OpenRouteService will be integrated with MapLibre, allowing the display of both existing routes from external APIs and user-generated routes within the app.

For real-time communication, STOMP over WebSocket will be employed to support one-on-one messaging between users. Silent backend checks will trigger push notifications through Firebase Cloud Messaging (FCM) for Android and Apple Push Notification (APN) for iOS, ensuring reliable user updates without active polling.

A Python-based Model Context Protocol (MCP) assistant will be developed to provide intelligent, context-aware functionality within the app. It will handle tasks such as recommending running routes, summarizing performance data, and responding to user queries about progress or goals. The assistant will operate as an independent microservice, containerized alongside the Spring Boot backend, and will communicate through REST or WebSocket APIs. This modular structure will simplify integration, enable scalability, and maintain a clean separation between the AI component and the core application logic.

Docker and Docker Compose will be used to ensure consistent development and deployment environments across the team. All core services (PostgreSQL/PostGIS, the Spring Boot backend, and the Python MCP assistant) will be containerized for easy setup, reproducibility, and reliable integration. The application will run locally using Docker Compose, and for mobile testing, it can be securely exposed through tools like ngrok. We will run a VM on AWS, integrated with CI/CD pipelines and deployed with Docker Compose.

The project will be structured into four separate repositories for clarity and modularity: a Flutter frontend (runwithme-app), a Spring Boot backend (runwithme-api), a Python MCP assistant (runwithme-mcp), and an infrastructure repository (runwithme-infra). This separation allows independent development, cleaner dependency management, and isolated CI/CD pipelines while keeping Docker Compose and environment configurations centralized under the infra repository for a clean build on AWS.

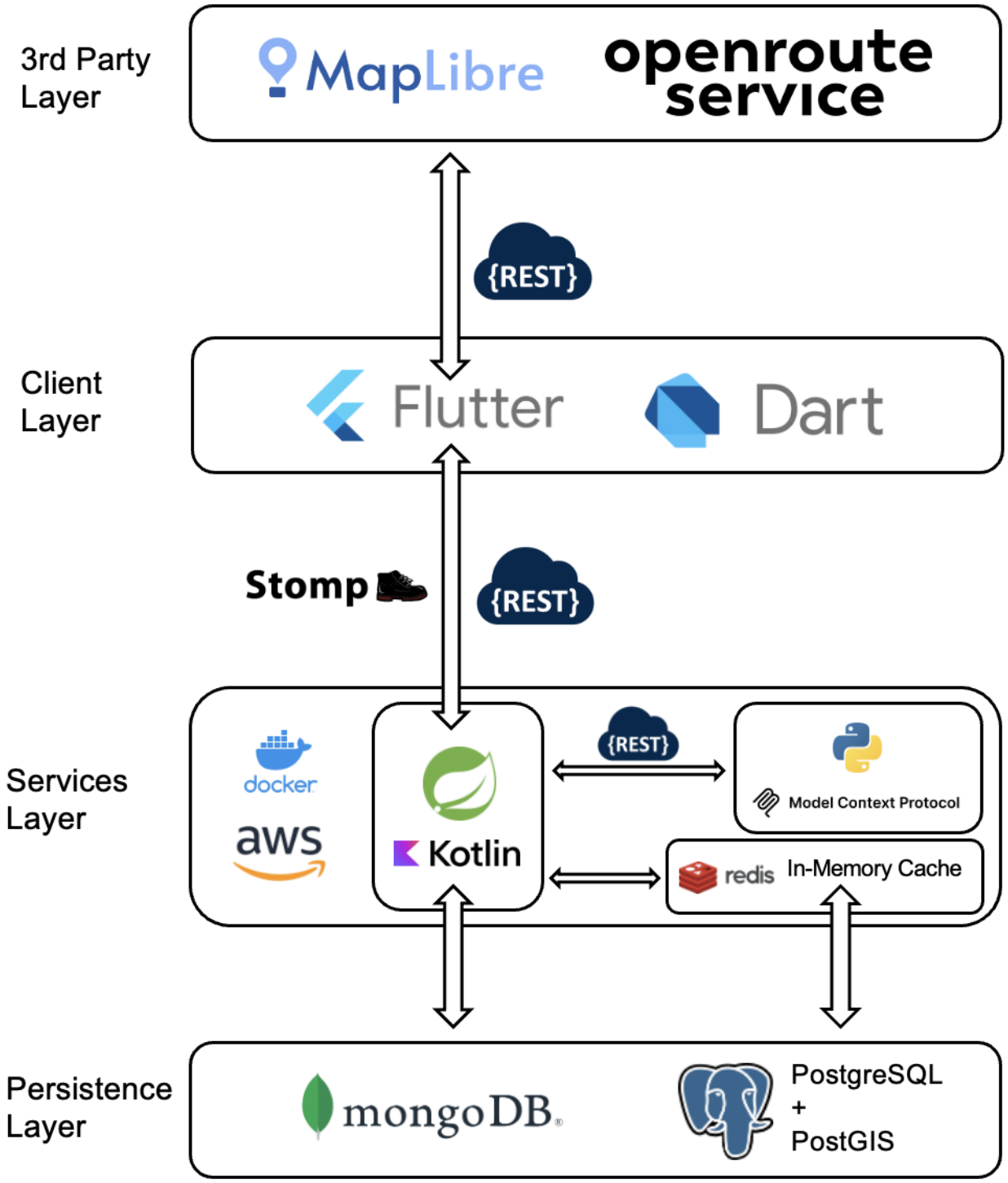
Together, this methodology ensures a well-structured, maintainable, and extensible system. Each component (database, backend, frontend, mapping, messaging, and AI) has a clearly defined role and communication boundary. The result is an architecture that balances complexity with modularity, enabling parallel development across team members while maintaining consistency, reliability, and long-term scalability.

Relevance and Effectiveness

The relevance and effectiveness of this approach are the primary reasons behind the chosen technology stack. Since the project’s objective is to meet well-defined, domain-driven goals rather than pursue a research-oriented task, these qualities can be evaluated in two main dimensions: development efficiency and runtime performance. The clear separation between the frontend, backend, and MCP components ensures independent, parallel development and simplified maintenance, directly improving productivity. Similarly, the use of optimized mapping libraries and a NoSQL database for direct messaging enhances runtime performance and scalability on mobile devices. Finally, adopting an MCP-based architecture for agentic AI aligns with current industry practices, reinforcing both the practical relevance and technical effectiveness of the proposed methodology.

Diagram

Below is the diagram for the tech stack mentioned above.



2.2 Work Package Descriptions

Work package number	1	Start date or starting event:		Week 1	
Work package title	Set Up				
Participant number	1	2	3	4	5
Participant name	Ege	Tuna			
Weeks per participant	1 week	1 week			

Objectives

- Have all the repositories ready and initialized for development.
- Have the services up as their barebones versions, so further developments can be easily tested between the front-end and the back-end.

Description of work

T1.1 (w1) Set up repositories.

Create necessary git repositories for the application

T1.2 (w1) Setup the infra repository.

The infra repository will enable easy connection for the frontend whenever a local backend and/or MCP connection is needed.

T1.3 (w1) Setup the databases.

We will ideally host them non-locally for the entirety of the project.

T1.4 (w1) Verify the infrastructure.

Test out a working connection between the frontend and backend, while using the Docker Compose approach.

Deliverables

D1.1.**1**(w1) 4 repositories defined in the methodology.

D1.2.1 (w1) Docker compose capable of running the application with placeholders for future MCP and backend.

Work package number	**2**	Start date or starting event:		**Week 2**	
Work package title	**Frontend**				
Participant number	1	2	3	4	5
Participant name	İdil	Tuna	Ege	Mina	
Weeks per participant	8 weeks	8 weeks	1 week	1 week	

Objectives

- Create the user interface from scratch and connect it to the backend.

Description of work

T2.1 (w2) UI design

Decide on the looks of the main UI screens and pop-ups.

T2.2 (w4-w5) Implement API client

API client for backend and data classes for API response handling.

T2.3 (w2-w8) Implementation of the UI

Implement Basic Screens (Home, Profile, Chat, Feed) using Flutter components.

T2.4 (w4) Implement login/registration

Integrate backend with client side for handling JWT tokens

T2.5(w2-w6) Map and Routing

Create a screen for map and communication with the Routing Api, route creation, and showing predetermined routes.

Deliverables

D2.1.**1**(w1) Tentative UI designs for the Home, Login, Feed, Route Creation, Chat, and Profile screens
D2.2.1 (w5)API client on Flutter and classes for API handling, converting JSON responses into classes.
D2.3.1(w6**)** Working but visually not polished frontend, screens capable of saving and changing state, and logically sound.
D2.3.2(w8**)** Visually polished UI
D2.4.1(w4) Login/SignUp front end connected to backend and can verify users.
D2.5.1 (w6) Map that can utilize MapLibreGL, create or show premade routes by other people. Saving routes, etc.
.

Work package number	**3**	Start date or starting event:		**Week 2**	
Work package title	**Core Backend**				
Participant number	1	2	3	4	5
Participant name	Ege	Mina	Tuna	İdil	
Weeks per participant	6 weeks	6 weeks	1 week	1 week	

Objectives

- Get a functional backend running as fast as possible.
- The faster this is achieved, the fewer synchronous developmental blocks will occur on the frontend side.

Description of work

T3.1 (w2) Spring Boot Initialization

T3.2 (w2-w3) Auth with JWT

Implement authentication using JWT

T3.3 (w3-w5) eg. Users, routes, activities, RestAPI, and CRUD

T3.4 (w6) OpenAPI via springdoc

T3.5 (w2) AWS cloud service and CI/CD pipeline

Automatic API documentation

Deliverables

D3.1.**1**(w2)Project Bootstrap, Spring Data JPA, Flyaway
D3.2.1 (w3)Auth with JWT, privacy flags.
D3.3.1(w5**)** CRUD for the database and the Rest API.
D3.3.2 (w5)Migration scripts

D3.4.1 (w6) Automatic API documentation Demo
D3.5.1 (w3) CI/CD demo to cloud service

Work package number	**4**	Start date or starting event:		**Week 3**	
Work package title	**GeoSpatial Data**				
Participant number	1	2	3	4	5
Participant name	Tuna	İdil	Mina	Ege	
Weeks per participant	3 weeks	3 weeks	1 week	1 week	

Objectives

- Learn how to work with and implement GeoSpatial data in the project. This WP will concern both the frontend and the backend.

Description of work

T4.1 (w3) PostGIS schema for routes, segments, nodes.

T4.2 (w4) GPS track, elevation, and stats pipeline.
T4.3 (w5) KNN overlap scoring for recommendation of similar routes using FastDTW.

Deliverables

D4.1.1(w3)SQL schema and indexes
D4.2.1 (w3)Getting GPS data following the path, and stats of the run
D4.3.1(w4**)** Working KNN scoring to assess how similar 2 paths are
.

Work package number	**5**	Start date or starting event:		**Week 6**	
Work package title	**Messaging and Networking**				
Participant number	1	2	3	4	5
Participant name	Tuna	Ege	Mina	İdil	
Weeks per participant	3 week	1 week	1 week	1 week	

Objectives

- An extension of the core backend WP. Composed of all the advanced backend-frontend functionality related to networking and direct messaging, which will be implemented here.

Description of work

T5.1 (w6) Stomp/WebSocket Backend EndPoint

T5.2 (w6) NoSQL Message and Feed Storage
MongoDB for chat and feed storage, this will most likely be ready beforehand, having been set up alongside PostgreSQL.
T5.3 (w7) Integrate front-end and backend for 1v1 messaging

T5.4 (w8) Feed screen

A screen in which users can see routes run by others or join others for future runs

T5.5 (w8) FCM and APN wiring, live notifications

Notification system, silent checks from the backend. Live map updates on new suitable routes.

Deliverables

D5.1.1(w6) WebSocket and STOMP handlers and backend service
D5.2.1 (w7) NoSQL setup with MongoDB, if it had not already been.
D5.3.1(w7**)** Working 1v1 chat in the application with chat history
D5.4.1 (w8) Feed screen in which users can reuse routes published by others.
D5.5.1 (w8) Notification system.

Work package number	**6**	Start date or starting event:		**Week 7**	
Work package title	**MCP**				
Participant number	1	2	3	4	5
Participant name	Mina	Ege			
Weeks per participant	4 week	1 week			

Objectives

- This WP is about creating the Model Context Protocol in Python, and enabling the usage of an agentic AI assistant within the application.

Description of work

T6.1 (w5) Containerization with Docker and initialization

T6.2 (w6) Bridge to the backend API
T6.3 (w7-w8) Implement functions for tasks

T6.4 (w7-w8) Integrate route recommendation heuristic

T6.5 (w9) Connect MCP to front-end

Deliverables

D6.1.1(w7)MCP service image
D6.2.1 (w7) API connection of the MCP
D6.3.1(w8**)** Commands Demo
D6.4.1 (w8)Route recommendation Demo according to prompt
D6.5.1 (w9)Fully working MCP via front-end

Work package number	**7**	Start date or starting event:		**Week 1**	
Work package title	**Performance, QA, Demo, Poster, and Final Report**				
Participant number	1	2	3	4	5
Participant name	Tuna	İdil	Ege	Mina	
Weeks per participant	10 week	10 week	10 week	10 week	

Objectives

- Stabilization of the application and delivery of the artifacts.

Description of work

T7.1 (w8-w11) Final Report

T7.2 (w10-w11) Poster
T7.3 (w1-w11) QA

Regular Quality Assurance Tests for the features.

Deliverables **D7.1.1**(w11)Final Report **D7.2.1** (w11)Poster **D7.3.1(**w11**)** Fully Polished, Bug-Free App for IOS and Android.
Milestones **M7.3** (w3,w5,…,w11) Components implemented are tested

2.3 Demonstration

The final demonstration of RunWithMe will showcase a fully functional cross-platform mobile application fully supported by a live backend server. The users of the application will be able to register, create, or select running routes, match with nearby runners to join their workouts based on route similarity, pace, and time availability. In the demonstration, we will emphasize the interaction between Flutter frontend and Kotlin Spring Boot backend, as well as real-time updates on MapLibreGL's interactive map.

Performance evolution will focus on several key metrics, including the accuracy of GPS-based tracking, the responsiveness of the UI, and the efficiency of backend communication through REST and WebSocket connections. The stability and scalability of the application will also be observed under multi-user scenarios, with attention to the backend latency, database query times, and performance of Docker services. Additional testing may be done to measure the energy consumption of background tracking to ensure battery-efficient operation. Overall, the application will be tested for user experience, smoothness, performance, and efficiency.

Quantifiable performance metrics will be as follows:

- GPS tracking accuracy: Average position error $\leq$ 10 meters in outdoor conditions
- Backend response speed: REST API responses $\leq$ 300 ms under normal load
- WebSocket message delivery: Delay $\leq$ 100 ms in 1v1 messaging
- System stability: $\geq$ 95% uptime during the multi-user demo session
- Load handling: Consistent performance with at least 10 concurrent users
- Battery efficiency: maximum 5% battery drain per 30 minutes of live tracking
- Matchmaking functionality: correct and deterministic runner matches with $\geq$ 90% route overlap accuracy (KNN-based)

The success of the demonstration will be determined by the performance evolution metrics mentioned. Mainly, the system should be able to perform route creation, live tracking, and messaging functions reliably. A positive user experience, seamless synchronization across devices, and successful operation of the AI-powered assistant will indicate that the project objectives have been achieved.

2.4 Impact

RunWithMe is designed to create a meaningful impact across social, technological, health, and environmental dimensions. The project goes beyond a simple fitness tracker by combining community engagement, innovation in mobile software, and long-term contributions to well-being and sustainability.

Social Impact

In a time of digital connections replacing the real-world interactions, RunWithMe aims to restore the human element of community through shared physical activity. By allowing users to discover, match, and run together with people from their neighbourhoods, the app encourages in-person socialization and friendship

building. This not only motivates people to exercise regularly but also helps reduce social isolation that many people experience in today's digital world. The platform brings people with shared interests-such as running, hiking, or walking- together in a safe and inclusive space where they can organize activities, set goals, and celebrate achievements together. Ultimately, RunWithMe promotes a sense of belonging and teamwork that contributes to both mental and emotional well-being.

Technological Impact

From an engineering perspective, RunWithMe is an application that integrates advanced and modern technologies in a single product. The combination of a Flutter frontend, Kotlin Spring Boot backend, and PostgreSQL/PostGIS database illustrates the application of cross-platform, real-time, and geospatial data management. Furthermore, the use of open-source frameworks such as MapLibreGL and OpenRouteService highlights a commitment to accessibility and transparency in software development, which we as a team value highly. The inclusion of an AI-based MCP assistant that can analyze user performance and suggest routes exemplifies how an AI agent can enhance everyday user experience, which is a recent challenge in today's technology. Collectively, the project will demonstrate our team's ability to integrate different technologies to create a fully functional, modular, and scalable mobile application.

Health and Well-Being Impact

The project's primary aim is to improve individual and community health through accessible physical activity. RunWithMe motivates its users for a consistent exercise routine, such as walking, hiking, or running, which are activities that are both simple and effective in enhancing cardiovascular health and mental balance. The social aspect of the application also boosts motivation and accountability, while making users adhere to regular activity schedules. The app can therefore serve as a preventive health tool, indirectly reducing lifestyle-related illnesses. Its progress tracking and supportive community engagement can inspire people to maintain long-term fitness habits, even resulting in behaviour changes in its users.

Environmental Sustainability Impact

Beyond its personal and social benefits, RunWithMe contributes to environmental sustainability. Encouraging users to run, walk, or hike for short-distance travel can lead to reduced dependence on motor vehicles, thereby lowering carbon emissions in everyday life. As people adopt walking or running not only for sport but also as a means of short-range mobility, small yet cumulative environmental benefits emerge-less fuel consumption, cleaner air, and quieter, safer neighbourhoods. By inspiring sustainable urban movement habits, the app indirectly supports global goals related to reducing carbon footprints and promoting greener cities. In this way, RunWithMe aligns individual health improvements with collective environmental responsibility.

2.5 Risk analysis

The development and use of RunWithMe involve certain risks related to privacy, security, and system reliability. The primary concern is the handling of sensitive location data. Users may unintentionally share real-time location information with others, creating privacy vulnerabilities. To mitigate this risk, all location-based features will be opt-in, and users will have full control over visibility settings.

Another risk arises from the security of user data and authentication tokens. Communication between clients and servers will follow standard security practices, ensuring the confidentiality of user data and authentication tokens. Additionally, ethical misuse of the app, such as unwanted contact or harassment, will be addressed through clear community guidelines and built-in reporting and blocking mechanisms.

Technical risks include potential backend downtime, API failures, or compatibility issues between Flutter, Kotlin, and the database systems. These will be minimized through continuous integration, Docker-based deployment for reproducibility, and automated testing pipelines. Another potential issue might be battery drain during continuous GPS usage, which could be solved by adaptive tracking intervals and efficient background processing. All these measures together aim to ensure both a safe and reliable experience for all

users. All risk items are listed in the following table.

Risk Item	Description	Likelihood	Impact	Mitigation Strategy
Location Privacy Exposure	Users may unintentionally share their real-time location publicly.	Medium	High	All location features require explicit opt-in, with private mode and visibility controls.
Unauthorized Data Access	Authentication tokens or user accounts may be compromised.	Low	High	Token-based authentication, secure storage, and standard backend security practices.
Misuse of Unwanted Contact	Users could interact inappropriately or harass others.	Medium	Medium	Report/block functionality, clear community guidelines, and moderation capabilities.
Backend Downtime	Server or API failures disrupt the application experience.	Medium	High	Docker-based deployment, CI/CD integration, and monitoring during demo.
Performance Bottlenecks	High latency or slow operations under concurrent usage.	Medium	Medium	Performance testing, database indexing, and local caching.
Battery Drain from GPS Tracking	Continuous geolocation updates may consume excessive battery.	High	Medium	Adaptive tracking intervals and background processing optimization.
Technology Integration Issues	Flutter, Spring Boot, and database services may conflict during development.	Medium	Medium	Frequent integration testing and modular architecture.
Data Loss or Corruption	A malfunction in the backend or database may cause information loss.	Low	High	Regular backups and fail-safe data handling strategies.

2.6 Gantt Chart

Section 3 Economical and Ethical Issues

Economical Dimension

The development of RunWithMe will be done with free and open-source technologies such as Flutter, Spring Boot, PostgreSQL, and MapLibreGL, which significantly reduce the overall cost of implementation. The project will be developed and hosted using local infrastructure and containerized environments, eliminating the need for paid cloud services during development. The scalable architecture allows for future deployment on affordable platforms if the project continues beyond academic use.

In the long term, from an economic perspective, the system created can be scaled up and support premium features that could generate revenue, such as personalized AI coaching, exclusive event planning, or fitness challenges. There are also other potential areas to expand, such as partnerships with gyms, sports brands, or local running communities.

Ethical Considerations

In accordance with engineering ethics, RunWithMe is designed with an emphasis on user safety, data protection, and social benefit. We, as engineers, have a responsibility to develop technologies that respect human rights, promote welfare, and minimize harm. Therefore, privacy and informed consent are central

principles guiding this project. Users will be clearly informed of when and how their data is collected, stored, and shared, and they will have the ability to withdraw consent at any time.

The project will not tolerate any misuse of user information or discriminatory behavior within the community. Data handling will be done under the principles of the General Data Protection Regulation (GDPR), promoting transparency, data minimization, and secure deletion. Overall, RunWithMe aligns with the ethical responsibility of engineers to develop systems that serve society's well-being, protect users' autonomy, and foster trust in technology.

Section 4 References

1. https://www.postgresql.org/docs/
2.FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. Stan Salvador & Philip Chan. KDD Workshop on Mining Temporal and Sequential Data, pp. 70-80, 2004.
3. flutter.dev
4. spring.io/projects/spring-boot
5. kotlinlang.org
6. postgresql.org/docs/
7.postgis.net/documentation/
8. mongodb.com/docs/
9. redis.io/docs/
10. https://platform.openai.com/docs/overview
11. https://openrouteservice.org/
12. https://maplibre.org/
13. https://www.rfc-editor.org/rfc/rfc7519
14. https://stomp.github.io/