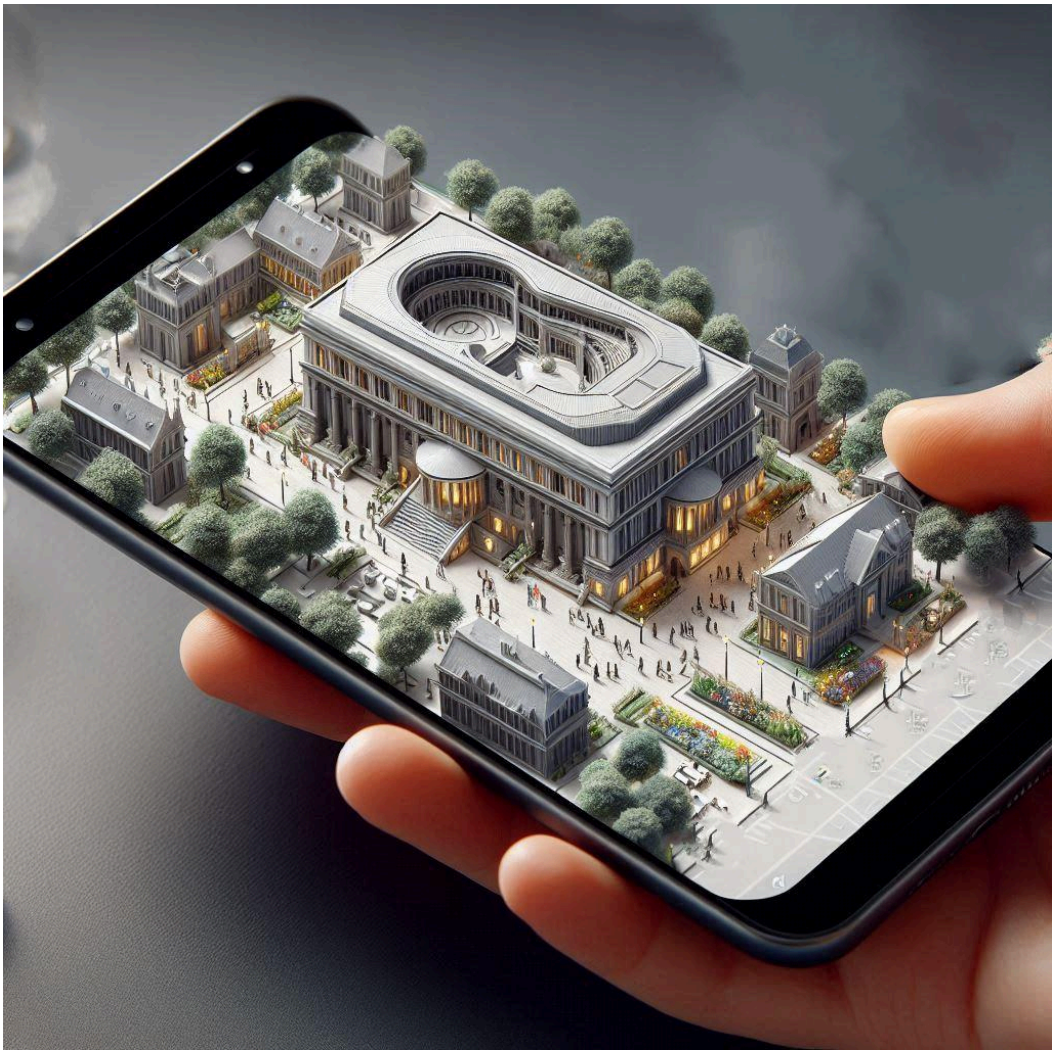
NavarLost



Afeka Final Project

Guidance: Revital Marom
Approved By: Revital Marom

Author: Yuval Cohen
Email: cohenyuval315@gmail.com
Github: https://github.com/cohenyuval315
Linkedin: https://www.linkedin.com/in/yc315/

Start Date: 1.10.2023
End Date: 8.8.2024

Revital Marom Elgrabli (ComCour Systems)
to me

Wed, 7 Aug, 20:49 (13 days ago)

היי,

מאושר להגשה

בברכה,
רביטל

מאת: יובל כהן <cohenyuval315@gmail.com>
נשלח: Wednesday, August 7, 2024 2:53:36 AM
אל: Revital Marom Elgrabli (ComCour Systems) <Revitalme@comcour.co.il>; יובל כהן <cohenyuval315@gmail.com>
נושא: Fwd: Indoor Navigation Yuval Cohen , Final Draft

•••

Table of Contents:

Assets Table of Content

1.Compliance

1.1.Updates - changes and fixes:

7/8/2024:

Changes	System Component	Change Reason	Initiator	Approval
Renaming Navigation Engine to Map Engine	System architecture	Navigation isn't accurate as the operation is called localization in that context	Yuval Cohen	Revital marom
Change WIFI to be optional, depending on the performance of the INS Change Magnetic profiling to be optional	Engine architecture	In case INS will be enough	Yuval Cohen	Revital marom
Splitting our Team into 2 teams As each one , went a different path in the project.	Team	Different projects implementations	Yuval Cohen	Revital marom
Added photoshop	Tools and services	More accurate	Yuval Cohen	Revital marom
Project Accuracy Objective from 95% success to 1-2 meter range	Project goals	Measurements in position accuracy measured in meters	Yuval Cohen	Revital marom
Adjusted Use cases to be more precises	Use Cases		Yuval Cohen	Revital marom

Added high level scope and out of scope	Introduction	Emphasizing the purpose of the system	Yuval Cohen	Revital marom
Added more extensive market survey Added PathGuide, Mapsted Added more features and functionality compares into the comparison table Pass All Text into the main comparison table	Market Survey	More extensive market survey after researching , and keying the features of each competitor.	Yuval Cohen	Revital marom
Added more User Stories and Personas	User Stories	More examples to demonstrate the most of the users profiles	Yuval Cohen	Revital marom
Changes and modifications to the architecture to be more precise	System Architecture		Yuval Cohen	Revital marom
Added Trello to Methodology	Methodology	Started using trello	Yuval Cohen	Revital marom
Added map heading map scale, map height scale.	Building Map Configuration	Configurable must be, key features,	Yuval Cohen	Revital marom
Added must have additional features	Wifi database	For orchestration those additional features are required	Yuval Cohen	Revital marom

Added must have additional features	Magnetic database	For orchestration those additional features are required	Yuval Cohen	Revital marom
Complete rewrite, to adjust to the new information and experience	Wifi fingerprint Matching	After extensive learning , adjust this to be general enough for multiple implementation types of wifi	Yuval Cohen	Revital marom
Added Euler angles and Gait Cycle explanation / images Removed initial heading after experience. Added Heading Realigning	INS	Some knowledge required in order to understand the INS	Yuval Cohen	Revital marom
Added machine learning section	Sensors Overview	Explaining why I'm not using machine learning enough though it is a really strong use case for machine learning.	Yuval Cohen	Revital marom
Rewrite algorithm	Transition area management.	Realizing is offset is needed after seeing the accuracy of GPS is based of number of meters.	Yuval Cohen	Revital marom
Added Geolocation to the integration of all the results.	Map Engine Integration Algorithm	Geolocation has variables that are relevant to whenever geolocation or not,therefore	Yuval Cohen	Revital marom

		required in the integration as well.		
Changed to vector tiles map	Visual Map	Realized i need Dynamic rendering	Yuval Cohen	Revital marom
changed microservices architecture to be a very valid option with explanation.	Network architecture	After understanding i can decouple the system for multiple usable services i understood microservices is very strong architecture	Yuval Cohen	Revital marom

[Table 1.1.Changes]

1.2.Current Project Goals Objective Progress:

7/8/2024:

Main Project Objective	**Achievement**	**Criterion**
Accurately determine a user's position	In Progress: No successful implementation of Wifi fingerprinting No implementation of magnetic profiling (require more manual labor) No solid implementation of the INS	Visual representation of a path, and accuracy per meter.
Display maps of indoor spaces	Achieved successfully	The building maps are displayed,

Navigation and accessibility features	Achieved successfully , accessibility not implemented in UI yet.	The shortest path does not include the accessibility constraint

[Table 1.2.Objectives Progress]

1.3.Issues and challenges

7/8/2024:

Issues	**Hindrance Details**	**Impact on Progress**	**Solution**
Startup Role Priorities	Started working in a startup environment focused on developing new ideas and products, which required me to fulfill multiple roles simultaneously. As a result, I had limited time available to dedicate specifically to this project, impacting its progress and development pace.	Hindered project advancement due to competing responsibilities and reduced dedicated time, affecting milestone achievements and delivery timelines.	
Complexity of Accurate Positioning	Indoor navigation with Wi-Fi and sensors is inherently challenging, requiring significant time for accurate measurements. The project demands a deep understanding of complex mathematical models (space models,	Slowed down progress due to the high complexity of achieving accurate positioning, with no clear solution available and each potential solution requiring significant adjustments.	

	observation models) and extensive literature review. Even Google hosts an annual challenge (Google Smartphone Decimeter Challenge 2023-2024 on Kaggle) to address similar issues.		
Algorithm Integration Difficulties	Various algorithms needed for the project often do not fit together seamlessly and require extensive modifications. The goal was to develop a modular, scalable application that could easily switch algorithms, but this added to the complexity.	Increased development time and complexity, as integrating different algorithms required careful adjustments to ensure compatibility.	
Map Implementation Challenges (and lack of custom maps libraries)	Implementing zoom and panning features in the map required a lot of linear algebra and posed significant difficulties.	Delayed progress in the map functionality due to the mathematical complexity involved.	
Academic Commitments	The project timeline coincided with semester tests, adding to the workload and time management challenges.	Reduced available time for project work, leading to delays in meeting project milestones.	
Solo Project Effort	Working alone on this project increased the workload and slowed	Limited collaboration opportunities, resulting in slower	

	down the overall progress.	problem-solving and development speed.	
Data Collection and Hardware Limitations	Collected data using my phone and computer, creating a host server on my phone to connect to my computer due to dynamic IP on Afeka Wi-Fi. Manual measurements using a tape measure were labor-intensive. My weak hardware caused constant crashes and data loss. My phone struggled with collecting large JSONs, leading to difficulties with sampling rates under 100 ms and Android's constraint of one Wi-Fi scan per 2 minutes, making data collection slow and cumbersome.	Resulted in inefficient and labor-intensive data collection, frequent crashes, and significant delays in obtaining necessary data for accurate positioning.	

[Table 1.3.Challenges]

2.Executive Summary

Indoor navigation can often be a challenging and frustrating experience, particularly for those unfamiliar with a building's layout, and complex large buildings areas. Some people avoid asking for directions due to shyness or other personal reasons. Using a camera for navigation can also be uncomfortable. A user-friendly 2D interface, like Waze, could offer a familiar and comfortable solution.

Our new indoor navigation app aims to revolutionize indoor navigation by providing users with an easy-to-use, reliable, and cost-efficient solution. We prioritize user experience and accessibility by offering customizable options, and we utilize advanced technologies, including WiFi, Magnetic Fields, sensors, and satellites to ensure as low as possible range of error indoor navigation.

We will create detailed floor maps of buildings, including points of interest (POI), to provide users with a comprehensive understanding of the indoor space. These maps are integrated into our app, allowing users to navigate with confidence.Based on the map WiFi signals will be used to provide accurate indoor positioning and navigation guidance. By scanning available WiFi networks and measuring signal strength, we estimate the user's location within the building, ensuring precise navigation assistance.

To enhance positioning accuracy and provide a seamless navigation experience, our app incorporates global gps/gnss location, as well as smartphone sensors such as gyroscopes, accelerometers, and magnetometers. These sensors contribute to precise movement tracking and orientation detection, improving the overall indoor navigation experience.

Moreover, we prioritize user preferences and requirements by offering customizable features. Users can adjust font sizes and colors for better readability. Voice guidance and audio descriptions are available for visually impaired users, enabling them to navigate independently. Additionally, we provide features like wheelchair-accessible routes to cater to individuals with mobility challenges.

The market for indoor navigation apps is growing rapidly as businesses, institutions, and individuals seek efficient ways to navigate complex indoor environments. Our app targets a diverse range of users, including tourists, shoppers, employees, and students who require seamless indoor navigation at an affordable price point.

Our app's competitive advantage lies in its affordability, WiFi-based positioning, sensor integration, and focus on user experience and accessibility. By utilizing existing smartphone capabilities, we offer a cost-effective indoor navigation solution. The integration of WiFi-based positioning and sensor technologies ensures dependable navigation guidance, enhancing the user's overall experience.

To implement our app, we will collaborate with building owners and managers to create detailed floor maps that include points of interest. Rigorous testing will be conducted to validate the app's functionality and performance. Our marketing efforts will focus on

businesses, institutions, and individuals seeking an efficient and user-friendly indoor navigation solution. The app's customizable options will cater to users with different preferences and requirements, including adjustable font sizes and colors, voice guidance. We will also incorporate features to accommodate individuals with disabilities, such as wheelchair-accessible routes

3.Background

The existing indoor navigation apps suffer from several significant problems, including inaccurate directions, limited coverage, poor user experience, limited accessibility features, and a lack of customization options. These issues can lead to frustration and inconvenience for users, particularly those with disabilities or special assistance requirements. Therefore, there is a pressing need for new indoor navigation apps that address these challenges and provide a more user-friendly and accessible navigation experience.

Additionally, there is a notable scarcity of indoor navigation apps in large, complex areas such as hospitals and malls. In these environments, people often become confused and rely on asking others for directions. Even after multiple visits, finding specific locations like shops can remain challenging. Furthermore, not everyone is comfortable interacting with others, which may hinder their navigation experience, possibly due to shyness or other reasons.

In response to these challenges, our goal is to create an app that is as affordable as possible while delivering reliable and efficient navigation capabilities.

To ensure the effectiveness of our app, we will be building a prototype specifically tailored to the premises of Afeka College. By utilizing advanced mapping technologies and prioritizing user experience and accessibility, we aim to revolutionize indoor navigation within the college and make it a more convenient and inclusive experience for all users.

While our app has a broader scope and potential application in different environments, our focus on affordability will enable us to provide a cost-effective solution for users. By offering a general app, we aim to cater to a wide range of individuals, including tourists, shoppers, employees, and students who require seamless indoor navigation.

Through the development of a prototype at Afeka College, we will gather valuable user feedback and refine our app's features and performance. This iterative process will enable us to fine-tune the app and ensure its compatibility with different environments and user requirements.

By developing this innovative indoor navigation app, we are committed to addressing the issues faced by existing solutions and providing an affordable and user-friendly navigation experience for all users. Our emphasis on advanced mapping technologies, coupled with our dedication to user experience and accessibility, will pave the way for a convenient and inclusive indoor navigation solution.

4.Project's objectives

Develop an indoor navigation app that provides accurate and efficient navigation guidance to users within indoor spaces.

Objective	Measurement
Accurately determine a user's position	The system should be able to determine a user's location within the indoor space within an accuracy **range under 2 meters.**
Display maps of indoor spaces	The system should display a detailed map of the indoor space, including **accurate** and **up-to-date information on points of interest**, and the **building's layout**.
Offer accessibility features	The app should provide audio-based directions for visually impaired users, using a clear and natural-sounding voice. The app should offer routing options based on user preferences (e.g., avoiding stairs, prioritizing elevators)
Navigation	The system should navigate users to their destinations in real time, without logic errors.

[Table 3.2.Objectives]

5.Scope

*for both product and prototype

- **User Preferences and Accessibility Settings:**
    - Users can configure their preferences and accessibility needs.
    - The System supports redirection to selected buildings using smartphone map providers.
- **Building Selection:**
    - The System displays a comprehensive map of all supported buildings.
    - Users can select buildings directly from the map interface.
    - Building layouts are accessible remotely for pre-visit planning.
- **Building Area Navigation:**
    - The System provides detailed maps of building areas and points of interest.
    - Users can manipulate maps for enhanced exploration.
    - The System supports the selection of specific floors within buildings.
    - Points of interest can be selected directly on the map or through search.
    - Navigation to selected points of interest is facilitated based on contextual information.
    - Real-time location detection and positioning are integrated.
    - Users can lock onto their current position for enhanced spatial awareness.
- **Pre-Navigation Planning:**
    - The System generates and displays optimized routes on building maps, accommodating multi-floor paths.
    - Route continuity across multiple floors is supported.
    - Notifications alert users to inaccessible routes due to specific accessibility constraints.
    - Distance and estimated travel times are provided for route planning.
- **Navigation:**
    - Real-time navigation to selected destinations is facilitated.
    - The System displays the user’s live location on building maps.
    - The next direction is displayed during navigation.
    - The System can reroute users in case of deviation from the planned path.
    - Manual floor changes during navigation are restricted for route consistency.
    - Relevant floor layouts are dynamically displayed for spatial orientation.
    - Real-time navigation metrics are provided.
    - Users are notified upon reaching their destination, concluding the navigation session.

6.Out Of Scope:

*for both product and prototype

- **Technical Limitations:**
    - No support for QR code functionality or virtual reality integration.
    - Advanced geospatial analysis beyond basic route planning is not included.
    - Detailed user behavior analysis or personalized recommendations based on historical data are excluded.
    - Non-navigation features such as social networking or gamification elements are not included.
- **Environmental Considerations:**
    - Highly specialized environments (e.g., underground facilities, secure areas) are outside the project scope.
    - Integration with external systems or databases not directly related to indoor navigation is excluded.

7.Requirements:

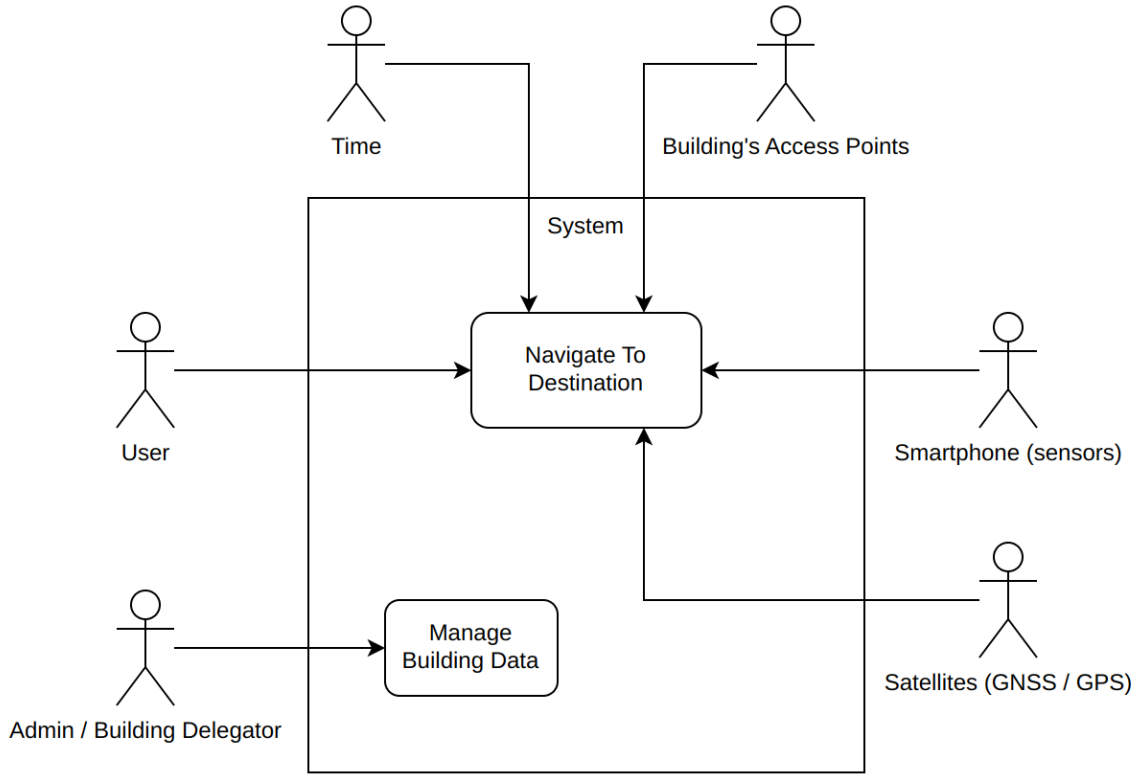
7.1.Functional Requirements:

- **Settings:**
  - Users can set system settings, preferences, and accessibility configurations on a dedicated settings screen.
  - Accessibility settings include stairs, elevators, and escalators.
  - Voice guidance configuration is available.
- **Searching:**
  - Building search functionality is available through a dedicated search bar.
  - Users can enter search queries to find specific buildings or points of interest.
  - Search results display relevant details such as name, address, and open status.
- **Maps Display:**
  - **Outside Map:**
    - Users can view an outside map displaying the surroundings and nearby locations/buildings.
    - The system can detect the user's location and offer redirection to the building map if the user is in the vicinity.
  - **Building Area Map:**
    - The map displayed should be larger than the actual map by half its side to center each point.
    - Important features like entrances, exits, and points of interest are marked with different icons.
    - The system accurately determines and displays the user's location with an arrow-like icon indicating the heading.
    - Users can manipulate maps (pan, zoom, rotate, center to location, lock to location).
    - The map can be rotated by 90 degrees for orientation changes.
- **Navigation:**
  - Users can confirm or cancel routes based on provided information.
  - Navigation displays speed, relative north direction, height, time left, next direction, and distance left.
  - The user's location updates in real-time during navigation.
  - Voice guidance provides turn-by-turn directions.
  - The current route is displayed on the map.

7.2.Non-Functional Requirements:

- **Usability:**
  - The system displays the current route on the building map.
  - The system is designed to work exclusively in Israel.
  - Compatibility with different screen sizes.
  - Availability of routers within the building.
  - Multi-language support starting with English.
- **Privacy:**
  - All data is encrypted.
  - Only relevant route information is provided without unnecessary details.
  - Compliance with Israel's privacy laws.
  - Permission from building owners is required for usage.
- **Accessibility:**
  - Clear and accurate voice audio instructions to minimize errors.
  - Specific colors and design cater to users with impaired vision, targeting an average user satisfaction of 4/5.
- **Reliability:**
  - Uses encryption protocol for messaging data.
  - Users stay connected using tokens.
  - In the event of a crash, the system should become usable again within 5 minutes.
  - Points of Interest (POIs) are updated on demand.
  - The system is available at all times.
- **Performance:**
  - Pathfinding takes no more than 3 seconds.
  - Voice directions have a maximum delay of 0.1 seconds.
  - Capable of handling multiple users simultaneously.
  - Accuracy range is under 2 meters.
- **Supportability:**
  - Provides log files.
  - Integrates physical routers.
  - Application size does not exceed 500 MB.
- **Integration:**
  - Integrates with at least one building.
- **Availability:**
  - The system is a mobile application supporting at least Android.
- **Cost Efficiency:**
  - Strives to be the most cost-effective option among competitors for fulfilling navigation requirements.

8.Use cases:



[Diagram 3.6.Use Case Diagram]

8.1.User Navigates to Destination

Primary Actor: User,Sensors,Satellites, Smartphone,Time

Preconditions:

- The user has the app installed on their smartphone.
- The user is within a building supported by The System.
- The user has launched the app and has set their preferences.

Trigger:

- The user wants to navigate to a specific destination within the building.

Main Success Scenario:

1. The user opens the app and selects the building map.
2. The user searches for or selects the desired point of interest (POI).
3. The System generates an optimized route from the user's current location to the destination.
4. The user follows the on-screen and voice guidance directions.
5. The System updates the user's position in real-time as they move.
6. The System provides turn-by-turn instructions and real-time navigation metrics.
7. Upon reaching the destination, The System notifies the user and concludes the navigation session.

Alternate Flows:

Alternative Flow 1: User Closes Navigation

1. At any point during navigation, the user selects the option to close the navigation.
2. The System stops providing navigation instructions and returns to the building map view.

Alternative Flow 2: User Goes Different Path (Reroute)

1. The user deviates from the suggested route.
2. The System detects the deviation and recalculates the route from the user's current location to the destination.
3. The System provides updated directions to the user.

Alternative Flow 3: User Outside Building

1. The user attempts to navigate while being outside the supported building.
2. The System detects the user's location outside the building.
3. The System prompts the user to enter the building and may offer directions to the nearest entrance.

Postconditions:

- The user reaches the desired destination or closes the navigation.

8.2.Manage Building Data through API

Primary Actor: Admin / Building Delegator (permitted user)

Preconditions:

- The admin has valid credentials and appropriate permissions.
- The admin has access to the API.

Trigger:

- The admin needs to update, modify, or delete data related to the building data, map, graph, POIs.

Main Success Scenario:

1. The admin authenticates and gains access to the API.
2. The admin sends a request to the endpoint with the appropriate action (create,update/modify/delete) and building map data.
3. The System validates the request and the admin's permissions.
4. The System processes the request and updates, modifies, or deletes the building map data in the database.
5. The System returns a success response to the admin.

Alternate Flows:

Alternative Flow 1: Invalid Credentials

1. The admin sends a request with invalid credentials.
2. The System rejects the request and returns an authentication error.

Alternative Flow 2: Insufficient Permissions

1. The admin sends a request without the necessary permissions.
2. The System checks the admin’s permissions and finds them insufficient.
3. The System rejects the request and returns a permission error.

Alternative Flow 3: Invalid Data

1. The admin sends a request with invalid or incomplete building data.
2. The System validates the request and identifies the invalid data.
3. The System returns a data validation error and prompts the admin to correct the data.

Postconditions:

- The building map data is successfully created,updated, modified, or deleted in the database.
- The System logs the admin action for auditing purposes.

9.User Stories and Personas

9.1.User Story 1: Navigating to a Destination

Persona: Emily, a First-Time Visitor

Description: Emily is visiting a large office building for a job interview. She needs to navigate the complex layout of the building to find the HR department where her interview is scheduled.

Steps:

1. Emily opens the app and selects the building map.
2. She searches for the HR department and selects it as her destination.
3. The app utilizes Wi-Fi signals and sensor data to pinpoint Emily's current location.
4. Emily receives turn-by-turn directions, ensuring she reaches the HR department on time.
5. If Emily deviates from the suggested path, the app recalculates the route and provides updated directions.
6. Upon reaching the HR department, the app notifies Emily and concludes the navigation session.

9.2.User Story 2: Admin Updates Building Map Data

Persona: David, a Building Manager

Description: David is responsible for maintaining the digital maps of a large shopping mall. He needs to update the map data regularly to reflect changes such as new store locations or updated points of interest (POIs).

Steps:

1. David logs into the admin API with valid credentials and permissions.
2. He accesses the endpoint.
3. David sends a request to update, modify, or delete building map data, such as adding a new store or removing an old one.
4. The system validates David's request and permissions.
5. The system processes the request and updates the building map data in the database.

6. The system returns a successful response to David, confirming that the changes have been made.
7. The updated map data is reflected in the user app, ensuring all users have the most current information.

9.3.User Story 3: Navigating in a Hospital

Persona: Sophia, a Patient's Visitor

Description: Sophia is visiting a hospital to see her friend who is recovering from surgery. She needs to navigate through the hospital's corridors and floors to find the patient's room.

Steps:

1. Sophia opens the hospital's indoor navigation app on her smartphone.
2. She enters the room number of her friend as the destination.
3. The app uses Wi-Fi signals and sensor data to determine Sophia's current location within the hospital.
4. Sophia receives turn-by-turn directions on the app to navigate to the patient's room, including guidance on which floor and ward to visit.
5. If Sophia encounters confusing signs or takes a wrong turn, the app recalculates the route and provides updated directions.
6. Upon reaching her friend's room, the app notifies Sophia and concludes the navigation session.

9.4.User Story 4: Navigating in a Hospital

Persona: Sophia, a Patient's Visitor

Description: Sophia is visiting a hospital to see her friend who is recovering from surgery. She needs to navigate through the hospital's corridors and floors to find the patient's room.

Steps:

1. Sophia opens the hospital's indoor navigation app on her smartphone.
2. She enters the room number of her friend as the destination.
3. The app uses Wi-Fi signals and sensor data to determine Sophia's current location within the hospital.
4. Sophia receives turn-by-turn directions on the app to navigate to the patient's room, including guidance on which floor and ward to visit.
5. If Sophia encounters confusing signs or takes a wrong turn, the app recalculates the route and provides updated directions.
6. Upon reaching her friend's room, the app notifies Sophia and concludes the navigation session.

9.5.User Story 5: Navigating in a Large Shopping Mall

Persona: Michael, a Shopper in a Large Mall

Description: Michael is visiting a large shopping mall to buy a birthday gift for his wife. He needs to find the jewelry store where he intends to make the purchase.

Steps:

1. Michael opens the mall's indoor navigation app on his smartphone.
2. He searches for the jewelry store by name within the mall.
3. The app displays the location of the jewelry store on the mall map and guides Michael to the store's exact location.
4. Michael uses features like zoom, pan, and rotate to explore the mall map and find the shortest route to the jewelry store.
5. As Michael approaches the jewelry store, the app updates him on his progress and notifies him of any special offers or discounts.
6. After purchasing the gift, Michael can navigate to other stores using the app's waypoint feature for efficient shopping.

9.6.User Story 6: Wheelchair User Navigation

Persona: Alex, a Wheelchair User

Description: Alex is a wheelchair user who frequently visits a large shopping mall to shop and socialize. He relies on the mall's indoor navigation system to navigate the complex layout and access facilities that accommodate his mobility needs.

Steps:

1. Alex opens the mall's indoor navigation app on his smartphone.
2. He selects accessibility options tailored for wheelchair users within the app settings.
3. Alex searches for a specific store or category of stores, such as clothing or electronics.
4. The app displays the locations of accessible elevators, ramps, and barrier-free routes on the mall map.
5. Alex navigates to the nearest accessible entrance using the app's turn-by-turn directions.
6. Throughout his journey, the app provides real-time updates on accessible paths and points out potential barriers.
7. Alex uses the app to locate accessible amenities, such as restrooms and seating areas, as he navigates the mall.

8. Upon reaching his destination, Alex receives notifications for nearby accessible facilities and services, enhancing his shopping experience.

10.Methodology:

I have adopted a Scrum framework for our project, organizing into sprints lasting one to two weeks. Each sprint focuses on specific goals, promoting efficiency and order. By breaking down the project into manageable increments, I aim to deliver valuable results continuously, adapting to feedback and evolving needs.

My goal is project success, optimizing productivity and workflow. Regular evaluation, and process adjustments ensure I meet the system's objectives.

For task tracking and project management, I use Trello. It helps me visualize tasks, track progress, and manage workflows effectively. Each sprint has its dedicated Trello board, organized by task status (To Do, In Progress, Done), ensuring clear visibility and streamlined development management.

11.Market Survey:

	Google Indoor Maps	Apple Indoor Maps	IndoorAtlas	Situm	Mapsted	PathGuide (Microsoft)	NavarLost
Infrastructure Technology used	GNSS/GPS, Sensors	Wi-Fi, GNSS/GPS, Sensors	Wi-Fi, BLEs, Eddystones, GPS/GNSS,Sensors	BLE,GPS/GNSS,Sensors	Sensors, GPS/GNSS	GPS/GNSS Sensors	Sensors, Wi-Fi, GNSS/GPS
Indoor positioning technology	Wi-Fi, BLEs, GPS	Wi-Fi, GPS, ARKit	Magnetic Fields, BLE/WIFI Signals GPS Signals	Machine learning algorithms	Machine learning algorithms	Leader-based sensory data and walking patterns without infrastructure.	Relies mainly on phone's internal sensors
Require Map	Yes	Yes	Yes	Yes	Yes	No	Yes
Map Customization options	Limited	Limited	Yes	Yes	Yes	No	High
Supported platforms	Android and iOS, Web	iOS only	Android and iOS,Web	Android and iOS,Web	Android and iOS,Web	Web	Android
AR	No	No	Yes	Yes	Yes	No	No
Wayfinding	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Deployment Service	No	No	No	Yes	Yes	No	No
Accuracy	Within 3-5 meters	Within a 3-5 meters	Within 1-2 meters	Within 1-3 meters	Within a 1-3 meters	Within 1 - 3 meters	Within 2 meters
Content Management System	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Map Editor	No	No	Yes	Yes	No	No	No
Required hardware	Wi-Fi and GPS-enabled device	BLE-enabled device	Magnetic sensor-enabled device	Sensors found on most modern smartphones	sensors found on most modern smartphones	Sensors	Sensors WI-FIi

Setup Complexity	Very Hard	Very Hard	Hard	Hard	Hard	Very Hard	Hard
Maintenance Complexity	Very Hard	Very Hard	Low	Low	Low	Very Hard	Low
Infrastructure cost	Free	Free	Require BLEs	Require BLEs	free	free	free
Cost	Free	Free	high-cost	high-cost	high-cost	Free	Free
Availability	Worldwide	Limited to select locations and buildings	Worldwide	Worldwide	Worldwide	Worldwide	Israel Tel Aviv Afeka Building
Has API	Yes	Yes	Yes	No	No	Yes	No
Has SDK	Yes	Yes	Yes	No	No	No	No

[Table 11.Market Survey]

Analysis:

Based on the table presented, we can see that indoor navigation solutions vary significantly in terms of technology, setup complexity, maintenance complexity, and costs.

For instance, Google Indoor Maps and Apple Indoor Maps, while offering high accuracy and extensive features like wayfinding and content management systems, come with very hard setup and maintenance complexities. This is primarily due to the rigorous validation processes for map data, ensuring high data integrity. Additionally, as free services with worldwide availability, they handle millions of requests, making updates a tedious task.

On the other hand, solutions like IndoorAtlas and Situm utilize a combination of Wi-Fi, BLE, and GPS technologies, providing more customizable maps and offering deployment services. These services, while high-cost, present lower maintenance complexity and better map customization options. Similarly, Mapsted and PathGuide offer innovative technologies like magnetic fields and machine learning algorithms to enhance indoor positioning accuracy, but they also come with higher costs and specific infrastructure requirements.

PathGuide, a unique solution by Microsoft, stands out for its reliance on the phone's internal sensors and leader-based sensory data, eliminating the need for additional infrastructure. This makes it a cost-effective solution, albeit with higher setup and maintenance complexities due to the need for precise calibration and data integrity.

Lastly, our system, although currently limited to Israel's Tel Aviv Afeka Building, offers a simpler setup and maintenance process compared to the other solutions. It utilizes standard sensors found on most modern smartphones, ensuring wide compatibility without the need for additional hardware.

In summary, choosing the right indoor navigation solution depends on balancing factors like technology used, infrastructure cost, setup and maintenance complexity, and specific deployment needs. Each solution offers unique strengths and potential challenges, making it essential to align the choice with the specific requirements of the intended use case.

12.System Architecture and Design:

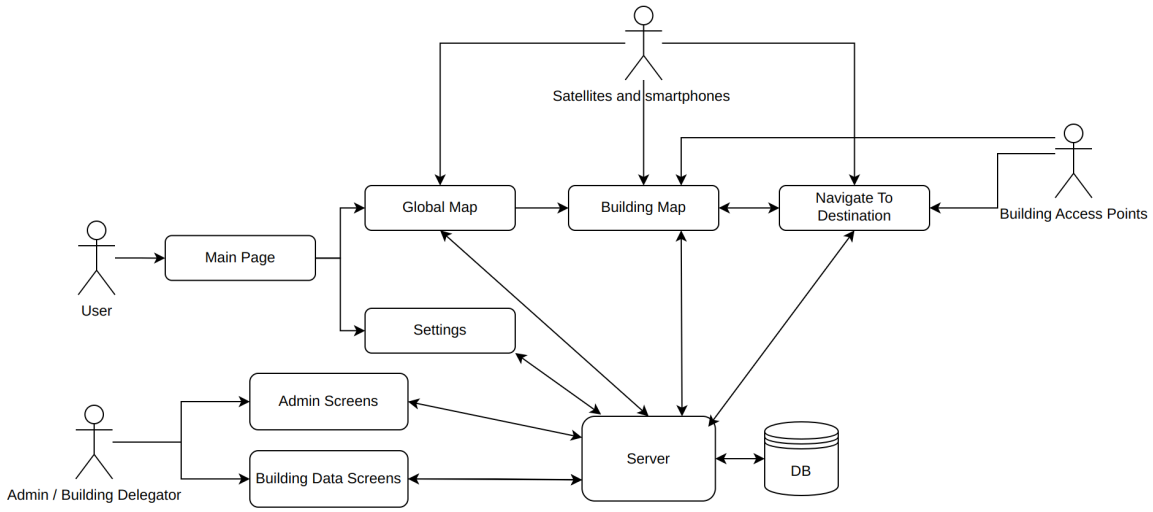
Description

The system prototype we are building is designed to be adaptable for various building environments that will be executed at Afeka College. The prototype consists of two distinct applications:a frontend app and a backend app.

The frontend app, developed using React Native and TypeScript, focuses on delivering a user-friendly interface (UI) and seamless user experience (UX), as well as UI for admin data collection..

The backend app is responsible for handling the system's core functionalities. It includes handling connections, authentication, analysis of sensor data, managing user preferences, handling map data and navigation to destination.

The following diagram showcases how the actor interacts with our system, then we will go over the high level choices and more into the lower levels implementations and considerations.



[Diagram 12.Actor System Diagram]

12.1.Network Architecture:

Component	**selected**	alternative	alternative
Network Architecture	Client Server	Peer to Peer	Microservices

[Table 12.1.Network Architecture]

In our thorough research, we encountered the peer-to-peer architecture utilized in an indoor navigation system, which capitalizes on users' movements for navigation. It's worth noting that peer-to-peer architecture shows promising results alongside the client-server model. However, we opted for the client-server architecture due to its simplicity and the abundance of research papers available in this domain. Unlike the peer-to-peer architecture, which has limited studies to draw upon, the client-server model offered a more robust foundation for our project.

Microservices were initially considered unsuitable for indoor navigation due to the system's centralized nature. However, upon further evaluation, it is clear that microservices can offer significant benefits. By differentiating the services, we can separate the map service, which handles map data and updates, from the actual navigation service, which deals with route calculation and real-time directions. This separation allows for more modular development, easier maintenance, and scalability. For instance, the map service can continue to function independently, providing essential map data even if the navigation service is temporarily offline. Additionally, another good use case for microservices in this context is the integration of user management and authentication as a separate service, enhancing security and user experience.

Furthermore, microservices can support peer-to-peer communication, enabling services to directly communicate with each other, increasing flexibility and resilience. The main downside is the potential increase in cost due to the complexity and resources required to manage multiple independent services.

However, despite the potential benefits, we have decided not to implement microservices initially as it would be an overkill for the start. The added complexity and cost are not justified at this early stage. We will start with a more centralized approach and consider transitioning to a microservices architecture depending on how the product evolves and scales.

Therefore, microservices architecture can be a valid and advantageous choice for our indoor navigation system.

12.2.Client Server:

12.2.1.High Level Client Side:

- User interface with screens for settings, searches, map displays ,navigation, and data mining.
- Managing user workflow.
- Display and voice guides for turn-by-turn directions and notifications.
- Sensor Integration: Utilizes smartphone sensors (gyroscopes, accelerometers, magnetometers) for movement tracking and orientation detection (not during navigation).
- Reactive streaming of the sensors data, wifi data and GNSS/GPS data.
- Integrating with the smartphone capabilities in order to extract sensors's data, wifi data and GNSS/GPS data.

12.2.2.High Level Server Side:

- RESTful API for client communication.
- WebSockets for real time communication during navigation.
- Search: Processes search queries and retrieves relevant results.
- GPS/GNSS, WiFi data, sensors data: Utilizes data to determine user location.
- GPS/GNSS, WiFi data, sensors data Processing: Processes data for positioning accuracy.
- Generates optimal navigation route based on preferences, using graph operations.
- Handles authentication, profiles, and access control.
- Handles CRUD operations on the different building's components.

12.2.3.High Level Five layer architecture:

Presentation Layer (PL): The presentation layer, also known as the user interface (UI) layer, is responsible for handling the interactions between users and the software application. It focuses on the visual presentation and user experience of the application. This layer handles user interactions and is responsible for presenting information to users. It includes components such as UI/UX design, user input handling, and displaying data to the user. In our application this can consist of presenting the map and the route path of the navigation in the map.

Application Layer (AL): This layer is all the logic related in our client which does not correspond to the UI or backend. In our system , this layer is responsible for retrieving data from sensors , wifi ,GPS/GNSS and integrating it into the application. It deals with gathering data from various sensors. This layer is also responsible for interfacing t o the backend.

Business Logic Layer(BLL): The business layer represents the core logic and rules of an application that define how the system operates based on the specific requirements and guidelines of our application .The business layerI receive requests from the client . encapsulates the business logic and processes, focusing on the application's domain-specific operations and workflows. The business layer defines how data is processed, transformed,

and validated to achieve the desired outcomes. In our application this can be rerouting and validating the request and providing the services , such as getting a new navigation route, getting buildings list.

Persistence Layer (PL): Also called Data Access Layer(DAL) or Data Services Layer (DSL). The data access or persistence layer is responsible for managing the interactions between the application and the underlying data storage system, such as a database. It provides an abstraction that allows the application to perform operations like storing, retrieving, updating, and deleting data without directly dealing with the complexities of the database being used. this layer can only be accessed by the services layer

Database Layer (DBL): The database layer is responsible for storing, managing, and retrieving data. It provides a persistent storage solution for the application and ensures data integrity, security, and scalability. The DBL interacts with the Data Access Layer (DAL) in the server-side to perform operations such as storing, retrieving, updating, and deleting data.

Initial intuition choice and possible alternatives for implementing the system

Component	Initial	Alternative	Alternative
Database	MongoDB	FireBase	neo4j
User positioning	Wi-Fi + Sensors	Bluetooth	LiDAR
Platform Front (Framework)	Cross-platform (React-native)	IOS (Swift)	Android (Java)
Backend Framework	Node.js	Spring Boot	

[Table 12.2.Initial Intuition System Architecture]

12.3.Database Architecture:

Component	**second option**	**selected option**	alternative	alternative	alternative
Database Architecture	SQL	No SQL Documents	No SQL Column-Family	No SQL key-value	No SQL Graph

[Table 12.2.Database Architecture]

The database design is outlined below in the database schema section. Our database primarily consists of informational data, with operations predominantly revolving around

read operations. Write operations occur mainly during the build phase of the application or for managing user profiles, representing a small portion of the overall database during production. Additionally, our database includes a graph data structure that maps the building routing, organized as nodes and edges for routing algorithms, albeit not extensive in size. Given these constraints, we delve into the available options to determine the most suitable database architecture

Comparison details:

SQL:

SQL databases offer a viable option for our indoor navigation system,as well as their acceptable compatibility with graph structures, making them a strong contender for our database. While they provide robust transactional support and can manage relational data effectively.SQL databases still present a feasible solution for our needs. In contrast to NoSQL alternatives, SQL databases may require more intricate schema designs, potentially translating to additional development effort and maintenance. Despite this, they remain a solid choice, especially if our system evolves to include more structured data management requirements.

NoSQL Column-Family:

like Apache Cassandra, were considered due to their capability to manage large data volumes efficiently and facilitate rapid read and write operations. However, considering the document-based structure of our data and the straightforwardness of our schema, NoSQL column-family databases were perceived as less fitting for our indoor navigation system's demands, especially considering our team's limited experience in this area. While they could potentially function well, their suitability might increase if we decide to incorporate more extensive statistical features or fully manage sensors in the backend, yet this does not align with our hybrid model approach.

NoSQL Documents:

NoSQL document databases, like MongoDB, emerge as an attractive option for our indoor navigation system, primarily due to their simplicity and scalability. With predominantly read-intensive operations and minimal write operations, except during initial setup and user profile management, NoSQL documents fit well with our data access patterns. Moreover, the flexible schema design of NoSQL documents reduces the complexity of data management compared to SQL databases. However, it's worth noting that while MongoDB offers flexibility, implementing graph algorithms may incur a hit as they would need to be executed in the backend rather than within the database itself. Despite this challenge, opting for NoSQL documents means significantly less work, allowing us to focus more on implementing the complex logic of our system rather than managing database intricacies. This streamlined approach aligns well with our development goals and ensures we can efficiently implement the sophisticated features of our system.

NoSQL Key-Value:

Stores, such as Redis or Amazon DynamoDB, are known for their proficiency in straightforward read and write operations. However, they lack the document-oriented querying capabilities essential for our indoor navigation system's data retrieval needs. Additionally, it's worth noting that we have limited experience with this type of database solution. Moreover, the structure of our data doesn't align well with the notion of keys holding extensive JSON documents, which further reduces the suitability of Key-Value Stores for our system architecture.

NoSQL Graph:

Graph databases, such as Neo4j, are adept at managing interconnected data and executing complex graph algorithms efficiently. While graph databases were considered for routing algorithms, the majority of our data does not exhibit strong graph characteristics, making document-based NoSQL databases a more practical choice overall. Integrating a graph database solely for routing would introduce unnecessary overhead, given its limited role in the overall system architecture.

12.3.1.Database Technology:

Database Technology	**selected**	Alternative	Alternative
No SQL	MongoDB	DocumentDB	-
SQL(alternative)	MySQL	PostgresQL	SQL Server

[Table 12.3.Database Technology]

Since we've opted for a NoSQL database, SQL databases won't be in consideration. It's worth noting that the disparity between them isn't significant. Therefore, selecting a SQL option will primarily hinge on preference and team familiarity. Our primary focus will be on NoSQL, particularly document-based, as it serves as our main database choice.

Among the document-based databases we explored, most offered the core functionality we required but lacked familiarity. MongoDB and DocumentDB emerged as standout options. We opted for MongoDB due to its familiarity, extensive team experience, and proven track record in handling large reads efficiently. While DocumentDB is a compelling choice, we prioritize MongoDB due to our team's expertise. Moreover, DocumentDB's compatibility with MongoDB ensures a smooth transition should we decide to migrate, requiring minimal effort.

Possible alternative:

Database Solution	MongoDB	Neo4j	Firebase
Team Experience (%15)	High (15/15)	First time (0/15)	Not all team familiar (7/15)
Developer Community (%10)	High (10/10)	Moderate (7/10)	High (9/10)
Ease of Use (%15)	Relatively easy to use with prior JavaScript and NoSQL knowledge. (12/15)	Requires understanding of graph databases and Cypher query language. (8/15)	Straightforward interface, ideal for rapid development. (15/15)
React Native Integration (%25)	Comprehensive (25/25)	Available libraries to simplify integration. (20/25)	Comprehensive (25/25)
Performance (%35)	Good performance, but may be impacted with complex queries or write-intensive operations. (30/35)	Excellent performance for graph-based queries and traversals, but may be less performant for read-heavy workloads or non-graph-based data. (35/35)	Good performance for real-time synchronization and small to medium-sized applications, may have limitations for complex data needs. (25/35)
Grade	92	70	81

[Table 12.3.Database Alternatives]

12.4.Technology Architecture:

12.4.1.Frontend Technology:

Component	selected	Alternative	Alternative
Frontend Framework	React-native (Cross-platform) Typescript	IOS (Swift)	Android (Java)

[Table 12.4.1.Frontend Technology]

Our system focuses on indoor navigation solely for mobile devices. To reach the largest user base, we've opted for React Native as it's cross-platform. Using TypeScript helps us overcome JavaScript's complexities, providing a solid object-oriented programming framework. React Native is quick to develop with and boasts extensive library support. With our team's expertise in React Native, along with iOS and Android experience, we're well-prepared to deliver efficiently.

Possible alternatives:

Front Platform (Framework)	TypeScript with React Native	iOS development with Swift	Android development with Java
Team Experience (%35)	Moderate (30/45)	High (30/35)	Moderate (25/35)
Cross-Platform (%20)	Yes (20/20)	No (0/20)	No (0/20)
Work well with Wi-Fi (%15)	Moderate (10/15)	High (15/15)	High (15/15)
Work well with Sensors (%15)	Moderate (10/15)	High (15/15)	High (15/15)
Ease of Use (%15)	Easy to use, flexible with React Native's component-based architecture and TypeScript's static typing. (15/15)	Moderate learning curve, requires understanding of Swift and iOS development practices. (10/15)	Moderate learning curve, requires understanding of Java and Android development practices. (10/15)

Grade	85	70	65

[Table 12.4.1.Frontend Alternatives]

12.4.2.Backend Technology:

Component	selected	Alternative	Alternative	Alternative	Alternative
Backend Framework	Express.js + Node.js Typescript	Flask Python	Spring Boot Java	Django Python	FastAPI Python

[Table 12.4.2.Backend Technology]

Our backend presents several frameworks options. Our team boasts extensive frameworks experience in JavaScript (specifically express.js), and predominantly Python, with a strong familiarity with Python frameworks.

Initially, Spring Boot emerged as an option. While it is a robust choice for backend development, offering comprehensive features and a strong ecosystem,its tendency to generate excessive boilerplate code and our team's relative unfamiliarity with it may pose challenges, diverting focus from the core complexities of implementing our localization and navigation engine.

Although we are proficient in Python, we have chosen Express.js TypeScript for our backend, aligning it with our frontend's TypeScript structure. This decision arises from a significant uncertainty: the performance division of our localization and navigation engine between the frontend and backend, which delivers live user location and navigation. Employing the same programming language for both backend and frontend provides flexibility, enabling potential module relocation based on real-world development experiences.

Possible alternatives:

Feature	Node.js	Spring Boot
Team Experience (%35)	Moderate (30/35)	Moderate (30/35)
Ecosystem (%10)	Vast and Growing (8/10)	Mature and Extensive (10/10)

Tooling & Libraries (%10)	Rich set (10/15)	Comprehensive (15/15)
React Native Integration (%35)	Excellent (35/35)	Moderate (30/35)
Time to build (%10)	Fast (10/10)	Good (6/10)
Grade	93	91

[Table 12.4.2.Backend Alternatives]

12.5.Hardware Technology:

Component	**selected**	alternative(Emulator)
Core Functionality Hardware Technology	Inertial Device Sensors + Optional WIFI Fingerprints + Optional Magnetic Profiles + GPS/GNSS	+ Bluetooth and/or UWB
Maps Generation	Blueprint from buildings and manual measuring	LiDAR

[Table 12.5.Hardware Technology]

After extensive research into indoor navigation technology, we gained deep insights into our system's requirements. We swiftly eliminated options that didn't align with our project goals, considering our budget constraints and the ambition to create a fully functional, real-world project, not limited to emulator usage. Consequently, technologies like UWB and Bluetooth were ruled out due to their high costs. We explored cost-effective alternatives such as leveraging existing WiFi infrastructure, magnetic field-free environments, and internal sensors found in smartphones, notably the gyroscope and accelerometer. These sensors are widely available and offer significant potential, unlike less common ones such as the barometer and pedometer, which, while useful, are less prevalent.

We want to emphasize that the utilization of WiFi and magnetic fields is incremental, based on the availability and behavior of the interior sensors. Our approach prioritizes the use of internal sensors first. If these sensors prove to be sufficient for our needs, that's ideal. If not, we then consider integrating WiFi data into our navigation system. Lastly, if neither the internal sensors nor WiFi data provide satisfactory results, we explore incorporating magnetic field information into our solution. This incremental approach ensures that we optimize the functionality of our indoor navigation system while considering various levels of sensor availability and accuracy.

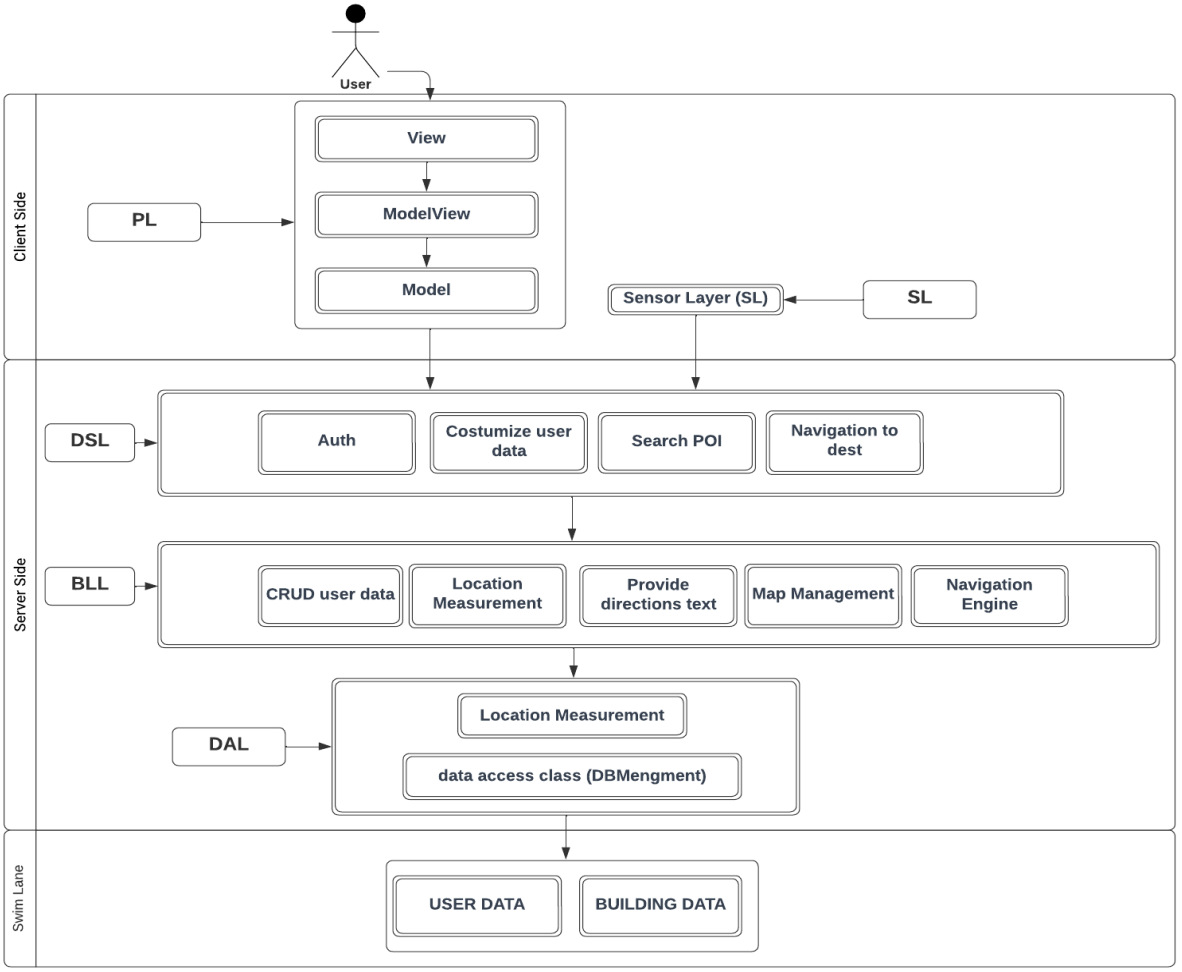
While LiDAR emerged as a solid choice for generating maps, its expense remains prohibitive. Therefore, we opted to develop our mapping solution from blueprints and manual measuring. While this decision is pragmatic, it is unfortunate that we couldn't leverage the capabilities of LiDAR for our indoor navigation project.

Possible Alternatives:

User Positioning Method	Wi-Fi	Bluetooth	LiDAR
Cost (%30)	Relatively low cost, leveraging existing Wi-Fi infrastructure. (30/30)	Moderate cost, requires investment in Bluetooth beacons or devices. (20/30)	High cost, requiring specialized LiDAR hardware and expertise. (15/30)
Developer Community (%10)	Strong (10/10)	Moderate (7/10)	Limited (5/10)
React Native Integration (%10)	Yes (10/10)	Yes (10/10)	No (0/10)
Ease of Use (%15)	Easy to implement and utilize, well-established technology. (15/15)	Moderate learning curve, requires understanding of Bluetooth devices and protocols. (10/15)	Complex implementation, specialized expertise and hardware required. (7/15)
Performance (%35)	Moderate performance, accuracy affected by environmental factors. (30/35)	Moderate performance, limited accuracy for short-range positioning. (30/35)	Excellent performance, highly accurate and precise positioning. (35/35)
Grade	90	72	62

[Table 12.5.Hardware Alternatives]

System Architecture Design Diagram



[Table 12.5.System Architecture Design]

12.6.Team Delegations:

Everything will be conducted and implemented by the author Yuval Cohen.

12.7.Engineering Knowledge acquired:

During the engineering report design phase, we immersed ourselves in newly acquired references, deepening our comprehension of indoor navigation at a profound level. Our investigation spanned the intricate mechanics of indoor navigation systems, elucidating various components like the Inertial Navigation System (INS) and its Attitude and Heading Reference System (AHRS). We examined the nuanced design considerations for individuals utilizing Pedestrian Dead Reckoning (PDR), encompassing both strap-down and free-moving systems.

Our exploration extended to the complexities of information fusion, particularly focusing on sensor fusion and calibration techniques. Throughout our research, we encountered inherent challenges in developing dynamic systems and explored potential solutions. Furthermore, we explored a spectrum of design alternatives, acknowledging the significant influence of requirements on system architecture.

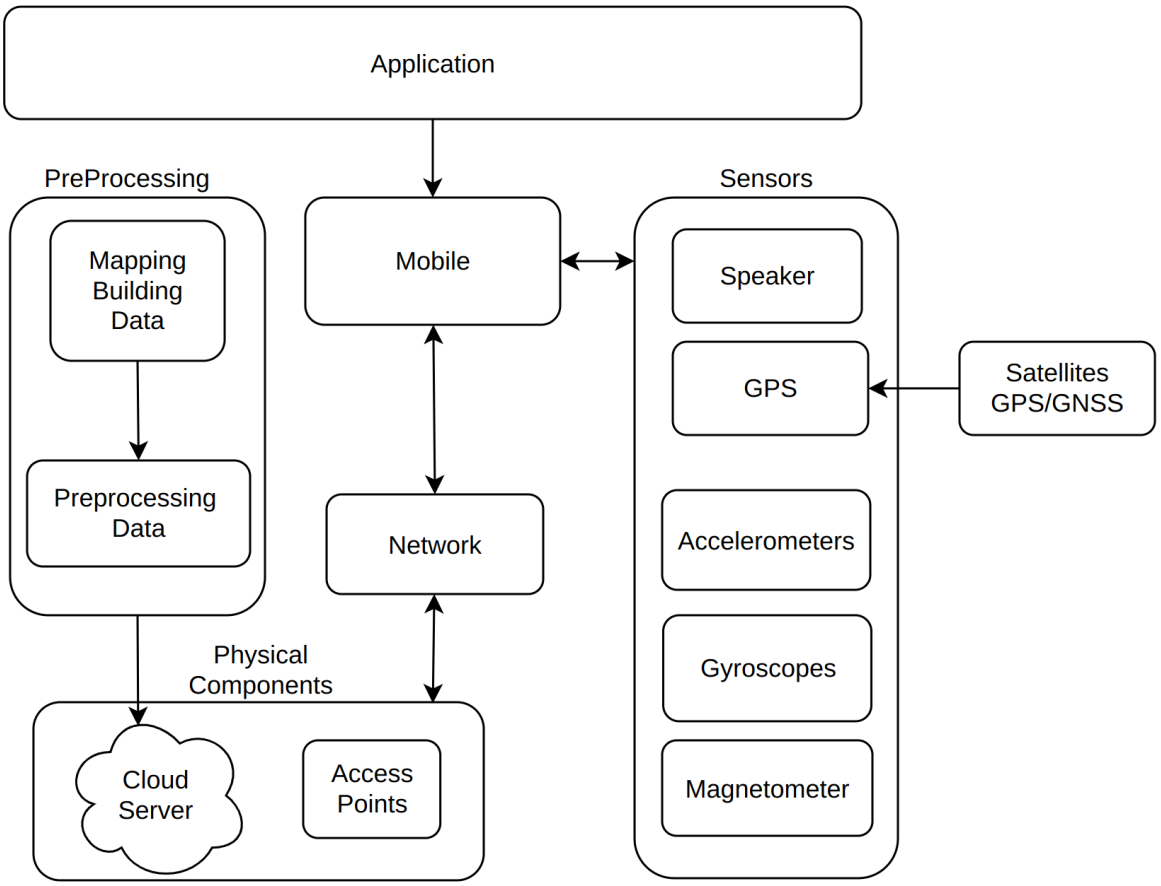
As we delved into numerous academic articles about implementation, we encountered many new terms specific to indoor navigation systems and related fields. This required thorough checking and understanding to effectively apply these concepts to our project.

At the conclusion of our exploration, we briefly explored the documentation of Google Maps, gaining initial insights into its features and functionalities. Additionally, we investigated various libraries available for both React Native, Android, and iOS platforms, along with existing sensor APIs. These inquiries expanded our awareness of the tools and resources essential for implementing indoor navigation solutions.

In addition to these explorations, I had to learn and relearn a substantial amount of mathematical concepts, including quaternions, information fusions, space models, and observation models. This project also required me to delve into animation techniques, Photoshop, and various React Native libraries. I had to master reactive streaming in the UI, adding another layer of complexity to the project. The combination of these new skills and technologies was essential for developing our indoor navigation solution.

12.8.Core System Components:

12.8.1.Hardware Architecture Overview:

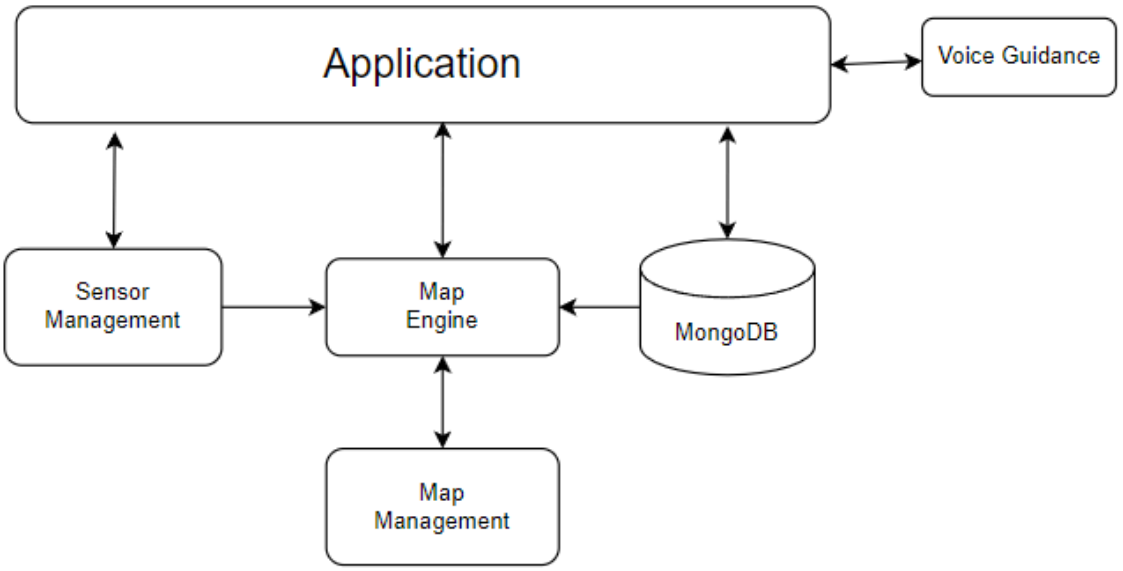


[Diagram 12.8.1.Hardware Architecture Overview]

The mobile device is at the center of the physical hardware. It requests GPS data in specific situations and collects readings from various sensors at configurable intervals. Additionally, the device captures RSS (Received Signal Strength) from Wi-Fi networks, facilitating location-based services.

Moreover, the mobile device sends and receives data across the network to a cloud server housing our application server. Additionally, a preprocessing module sends data directly to the cloud server for further processing. During Navigation the speakers are utilized for voicing the next route direction.

12.8.2.High Level Logic Overview:



[Diagram 12.8.2.High Level Logic Overview]

The application stores the user's physical device data state according to our system design, including their current position and window-based hardware information.It employs a sensors management module to oversee sensor logic, covering calibrations, fusion, and filtering processes. Using a socket connection, the application continuously sends its state to our backend, which utilizes our core engine for managing the state and providing localization and navigation services. Subsequently, the application handles the outcomes as per the system design.

The Map management module aids in organizing extensive building data.

User data and configurations are saved and served in the MongoDB database by our application.

During route navigation, a voice guidance module at the frontend controls the logic for voice directions.

12.8.3.General description of the main components:

Component	Role	Layer	Tier
Floor diagrams	Floor diagrams are used to **map out the layout** of a building and provide reference points for location tracking.	**Presentation layer**	**Front-End**
Wi-Fi Service	Scan the nearing access points, Using the scan output , to calculate user position , using various methods.		
GPS/GNSS Service	When available, GPS can be used to provide additional **location data** for more accurate tracking	**Application Layer**	**Physical**
Gyroscope Service	Used to determine the **angular velocity** of the device		
Accelerometer Service	Used to determine the **acceleration** of the device		
Magnetometer Service	Used to determine the **magnetic fields** and **heading** of the device		
Live Location Measurement	This component **processes** the **sensor data** to determine the location of the user's device.	**Business Logic Layer**	**Back-End**
Map Engine	Once the user's location is determined, this component calculates and **provides directions** to the user based on their **destination**.		

CRUD user data	The CRUD user data operations involve creating, reading, updating, and deleting user records in the system.		
Building Data Management	**Managing** and **storing** the map data from the **Floor diagrams**.		
Global Map	**Displays** the indoor **map**s to the user, and their **current location** and provides a visual representation of the **building layout**.	**Presentation Layer**	**Front-End**
Building Area Map	**Displays** the indoor **map**s to the user, and their **current location** and provides a visual representation of the **building layout**.		
MongoDB	noSQL **database** used for storing and managing data.	**Database Layer**	**Data**

[Table 12.8.3.Main Components]

12.9.Tools & Services:

12.9.1.System services:

- Destination Navigation
- User Real Time Live Location.
- Accessible Building Information
- Accessible Building Facilities Details
- User Profile Configuration
- Accessibility Features for Special Needs
- Voice-guided Directions
- Building Map Layout with Visualized Navigation Routes
- Language Support for Multilingual Users

12.9.2.System files:

- The system comprises SVG images representing maps for each building.
- Each building contains its own set of images depicting its physical location, along with various images highlighting points of interest areas within its premises.
- Additionally, it encompasses images specifically depicting entrance points within each building's context.

12.9.3.Third-party services:

We aspire to utilize most of these services within the given time constraints, although we cannot make a definite commitment at this point.

- Global Maps (Google Map): Google Maps serves as a global mapping service that enhances our application's indoor navigation functionality. While our focus remains primarily on indoor navigation, Google Maps offers valuable outdoor navigation support, allowing users to locate nearby buildings before transitioning indoors. Alternatives: Leaflets, OpenStreetMap, OpenLayers.
- Google OAuth: Google OAuth is employed for user authentication within the application. It allows users to securely log in using their Google credentials, enhancing the authentication and authorization process.
- AWS S3 (Simple Storage Service): AWS S3 is utilized for storing building-related files crucial for indoor navigation. These files may include floor plans, room layouts, and other navigational data required for effective indoor navigation within the building premises.
- AWS EC2 (Elastic Compute Cloud): AWS EC2 instances are utilized for deployment purposes. They host the application's backend and facilitate its scalability and reliability.
- AWS DocumentDB with MongoDB Compatibility: Amazon DocumentDB, compatible with MongoDB, serves as the cloud-hosted database solution for storing and managing application data. Its compatibility with MongoDB allows seamless integration with existing MongoDB applications, offering scalability, reliability, and managed services.
- Online Photoshop Tools: Photopea for maps.

12.9.4.Tools Development:

- Raw Data Collector: A React Native tool designed to gather data points and track sensor information, including WIFI fingerprints, magnetic profiles, smartphone orientation, and GPS coordinates.
- Map blueprint tool**:** An HTML and JavaScript tool designed to create an SVG visual map based on a building's measurements, including nodes positioning with edges and route mappings to edges for comprehensive representation.

12.10.Challenges

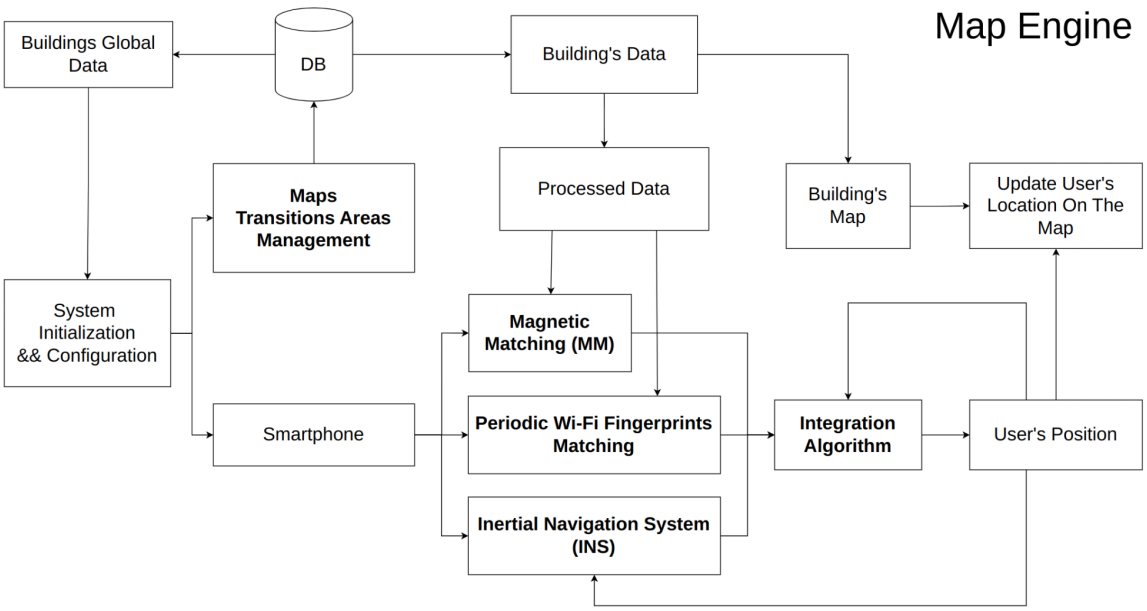
- Wi-Fi Signal Strength: One challenge could be ensuring consistent and reliable Wi-Fi signal strength throughout the building. Signal interference or weak signal areas might affect the accuracy of location tracking.
- Map Accuracy: The accuracy of map measurement and layout, can cause a lot of unwanted errors.
- Sensor Accuracy: The accuracy of sensors such as GPS, gyroscope, accelerometer, and magnetometer can vary based on environmental factors or device limitations. Ensuring the sensors provide accurate and reliable data for location tracking can be a challenge.
- Data Processing: Processing and analyzing the sensor data to determine the location of the user's device can be computationally intensive. Handling large volumes of data in real-time and performing triangulation calculations efficiently can pose challenges.
- Integration and Compatibility: Integrating the different components, layers, and tiers of the system can be complex. Ensuring compatibility between hardware, software, and databases, as well as seamless communication and data exchange between components, can be a challenge.
- Floor Diagram Accuracy: Keeping the floor diagrams up to date and accurate is crucial for precise location tracking. Any changes to the building layout or infrastructure need to be reflected in the floor diagrams, which can be time-consuming and require regular maintenance.
- User Interface and Experience: Designing an intuitive and user-friendly interface for users to interact with the system, such as displaying indoor maps, providing directions, and utilizing search functionalities, can be a challenge.
- Scalability: As the system grows and accommodates more users and locations, scaling the infrastructure, databases, and processing capabilities to handle increased data volumes and user demands can be challenging.
- Data Management: Efficiently managing and storing the map data, sensor data, and user data in a structured and accessible manner can be a challenge. Choosing the appropriate database system and implementing effective data management strategies is crucial.
- Maintenance and Support: Ensuring the system remains operational, performing regular maintenance, and providing technical support to address any issues or user inquiries can be challenging, especially in a real-time location tracking system.

13. Localization And Navigation - Algorithms Overview

Our map engine is our central component that drives and coordinates various functions and algorithms within our indoor navigation system. Our engine is responsible for performing complex calculations, processing data, or coordinating modules to achieve our system purpose.

This Map Engine is the core component of our system that handles the user's localization and orientation in the building's space. Which will be our foundation for tracking during the navigation algorithms.

13.1.High Level Overview:



[Table 13.1.High Level Map Engine Overview]

Our Indoor navigation engine is built upon an innovative, cost-effective smartphone model that combines Inertial Navigation System (INS) with Optional Wi-Fi fingerprint matching and Optional Magnetic Matching.

Numerous structures and techniques are at our disposal, which is why we approach the flow at a high level initially. As we delve deeper, we will explore the various options that align with our engine requirements, including our initial choices and any adjustments made during the development process.

13.2.High-level Engine System Requirements:

13.2.1.Hardware Requirements:

Our primary objective revolves around creating a cost-effective navigation system, and as such, the Inertial Navigation System (INS) stands as an indispensable component that cannot be omitted. We are keen on harnessing the sensors readily available within the device infrastructure, which forms the core of our localization system. We have consciously excluded hardware that is not commonly found, such as UWB (Ultra-Wideband) and Bluetooth beacons.

The prevalence of Wi-Fi networks, coupled with their cost-effectiveness and wide availability, makes them a critical infrastructure we cannot afford to overlook. We are actively exploring their potential to enhance and expand our localization solution. For instance, Wi-Fi can play a pivotal role in determining a user's location, particularly when the system initializes with the user already inside a building. This approach not only offers periodic error reduction but also aligns with our cost-effective approach.

Furthermore, magnetic fields are universally present and can be utilized as an additional method to enhance the positioning accuracy.

The architectural layout of buildings offers opportunities for leveraging two distinct common localization techniques: Wi-Fi fingerprints using the Wi-Fi network and magnetic field matching. Both of these methods contribute to reducing positional errors. However, it's important to note that they necessitate regular surveying and maintenance due to changes in the infrastructure. Additionally, they are susceptible to radio frequency interference and magnetic disturbances that commonly occur indoors due to environmental fluctuations.

GPS can be utilized for transitioning between global and site coordinate systems, but its indoor utility is limited. The weak GPS signal indoors is prone to errors and is not a dependable source for accurate positioning.

13.2.2.Logical Requirements:

The orientation of the smartphone within physical space plays a pivotal role in the seamless operation of our system. Successful localization is heavily reliant on the device's precise orientation for accurate navigation. However, the device's orientation may exhibit variations due to the user's freedom to handle it in various ways, such as placing it in a pocket, swinging it, or holding it by hand.

Managing this diversity is a critical concern that our Inertial Navigation System (INS) addresses. Although holding the smartphone by hand might seem like a more straightforward solution, determining its practicality can be challenging. Factors like the user's ability to sustain a specific posture for an extended period, attention span, and the potential interruption of phone calls need to be considered.

Our primary objective is to ensure that the INS consistently furnishes the user with an accurate heading direction (azimuth) relative to the site map, regardless of the smartphone state.

Our primary objective is to ensure that the INS consistently provides the user with an accurate heading direction (azimuth) relative to the site map, regardless of the smartphone's state. The diversity of smartphone devices further complicates this matter. Calibration and fusion methods are likely necessary to mitigate variations among different devices and reduce discrepancies to some extent. However, achieving identical outputs from different sensors is often unfeasible. To address this, techniques like Wi-Fi fingerprinting can be employed to periodically realign the positional error drift stemming from different devices, enhancing the overall accuracy and reliability of our system.

13.3.Localization And Navigation Content Overview:

- Engine preprocessing:
    - Preprocessing Building Layout: - Found in the engine preprocessing section.
- "Building Map Acquisition" section: Obtaining the building map is the foundational step for our engine. This map is vital for directions, navigation, and collecting magnetic and Wi-Fi fingerprints. Acquiring the map involves obtaining blueprints or manual measurements using lasers and meters, ensuring reliability for testing routes and assessing real-world performance.
- "Building Map Graph Constructions" section - In our navigation engine development, constructing a building map graph is crucial. This graph connects key locations, creating a structured framework for efficient route planning within the building. It enhances navigation and decision-making, providing a reliable foundation for guiding users seamlessly through indoor spaces.
- Gathering Building Sensors Data: Once we have the building map, the subsequent step in our system involves collecting transition areas, Wi-Fi fingerprints, and magnetic profiles within the building.
- "Maps Transitions Areas Management" - Collecting Areas GPS Coordinates
- "Establishing a WIFI Fingerprints Dataset"
- **"**Establishing a Magnetic Profiles Dataset"
- An overarching database schema is designed to encompass all the engine data, facilitating the referencing of positions as needed: "Database schema" section**.**
- Engine Localization:
    - User Localization Transition From Outdoor to InDoor: This shift from a global location to a site-specific location is described in detail as part of the "Maps Transition Areas Management" process. This initial transition pinpoints the user's location and determines the appropriate map to present.
    - Localization is the process of acquiring a user's location relative to the coordinate system used. The localization of our engine is the body of the user relative to the building map in point of space and time.
    - User Localization inside the Building:
        - INS - Based User localization: Once the user is inside the building, the localization process utilizing smartphone Inertial Measurement Units (IMUs) commences. This is elaborated under the "INS (Inertial Navigation System)" section.

- Wi-Fi-Based User Localization: Our system also leverages Wi-Fi for user localization, and this method is discussed in the "Periodic Wi-Fi Fingerprint Matching" section.
- Magnetic Field-Based User Localization: The use of magnetic fields for user localization is detailed in the "Magnetic Matching (MM)" section.
- Integration Algorithm: The integration of these localization algorithms is described in the "Integration Algorithm" section.

13.4.Engine preprocessing:

13.4.1.Building Blueprint Acquisition:

To acquire the layout of the building, proper authorization is essential. Once the necessary permits are obtained, building measurements can be obtained using one of the following methods:

- Utilizing an existing blueprint of the building's layout.
- Performing manual measurements using laser-based equipment and meters.
- Hybrid approach (combining both).

13.4.2.Building Map Construction:

By processing the building's measurements, we can generate both the visual representation of the building and a logical-digital map. In the case of simple buildings, for the visual map a grid-based map suffices. However, complexities arise when dealing with more intricate structures.

This process is carried out for each unique building, ensuring that building-specific data is used to design the corresponding visual and logical maps, thus optimizing the navigation experience.

Configuration: Map configuration is essential, and it may need to be adaptable to changing requirements. To facilitate this adaptability, functions will be implemented for seamless configuration adjustments. Key aspects of the map configuration include:

Map Scale: Defines the ratio between the size of features on the map and their corresponding real-world dimensions at a given zoom level. This ratio helps in understanding the scale of distances represented on the map.

Height Scale: Refers to the scale used for representing elevation or vertical dimensions. This scale can differ from the horizontal scale and may involve vertical exaggeration to enhance the visibility of height variations.

Map Heading: Indicates the map's orientation relative to the north. This parameter determines how the map is rotated to align with real-world directions.

Map Width and Height: These dimensions can be derived from the measured real-world area. They define the size of the map as displayed on the screen or printed format, ensuring that the map accurately represents the intended area.

13.4.3.Map Visualization:

Visual Map serves as a dynamic representation of a building's layout, enabling users to navigate through various spaces effectively. This visual map is created using detailed blueprints of the building and is crucial for guiding users to their destinations within complex environments.

13.4.3.1.Creating the Visual Map:

1. **Blueprint Integration:** We start by using the building's blueprint as the foundation for creating the visual map. This blueprint provides detailed dimensions and layout information necessary for accurately mapping the interior.
2. **Map Generation:** Using the blueprint, our map generation tool converts the detailed design into a visual format. This involves generating a map that represents the building's floor plan, including walls, rooms, corridors, and points of interest.
3. **Dynamic Rendering with Tiles:** To enhance interactivity and ensure the map updates in real-time as users interact with it, we employ tiles for dynamic rendering.

13.4.3.2.Tiles and Tiles Layers:

Tiles:

Tiles are small, discrete units used to divide a larger map into manageable pieces. This approach allows for efficient loading, rendering, and updating of maps by only processing the visible portions at any given time. By using tiles, the system can improve performance and responsiveness, as it avoids the need to render the entire map at once. Tiles also facilitate smooth zooming and panning by providing different levels of detail as the user interacts with the map. The most common tile types used are vector tiles and raster tiles.

- **Vector Tiles:** Vector tiles represent map data as geometric shapes and text, similar to SVG files. This method supports real-time rendering and styling, allowing for detailed, smooth, and scalable maps. Vector tiles are highly flexible, making them ideal for dynamic and interactive applications. They also maintain quality at any zoom level and allow for real-time updates without reloading entire map images.
- **Raster Tiles:** Raster tiles are pre-rendered images of map sections, such as PNG or JPEG files. These tiles are static and display different levels of detail based on zoom levels. While raster tiles are simpler to implement, they can become pixelated at higher zoom levels and may require more storage space. Raster tiles also lack the flexibility for dynamic updates and interactive features.

While both vector and raster tiles are used in mapping practices, **vector tiles** are generally considered the best choice for indoor navigation systems. They provide superior quality,

scalability, and interactivity, making them more suitable for complex or large buildings. Raster tiles, in contrast, can be inefficient for detailed or large-scale maps due to their static nature and larger file sizes. By choosing vector tiles, we ensure a responsive and adaptable visual map that meets the demands of modern indoor navigation, offering a better user experience with accurate and up-to-date information.

Tiles Layers:

Overview: Tile layers are essential for managing and displaying various map features efficiently. They help render different map elements, such as landmarks, routes, and other data points, with varying levels of detail based on the zoom level and user interactions. Styling layers further enhance the map's visual appearance, allowing for dynamic and customizable displays.

1. User Position: The user's current location is dynamically overlaid on the map and updated in real-time as they move. This dynamic overlay is not pre-rendered but added on-the-fly to ensure accurate navigation and positioning.

2. General Map Features Layers: Tile layers can represent a wide range of map features, including points of interest, natural elements, infrastructure, and more. To ensure modularity and scalability, each feature type has its own dedicated tile layer and corresponding style layer.

3. Style Layers: Style layers control the visual representation of different map features. Each style layer is associated with a specific type of data and can be individually styled to customize how features appear on the map.

Our System's Tile Layers:

- **Infrastructure Tile Layer + Infrastructure Style Layer:** Includes features like trails, walls, doors, roads, yards, and buildings.
- **Point of Interest Tile Layer + Point of Interest Style Layer:** Includes locations such as restrooms, cafeterias, and emergency exits.
- **Navigation Routes Tile Layer + Navigation Route Style Layer:** Displays navigation paths and their associated styling.

13.4.3.3.Navigation Routes in Vector Tiles:

In indoor navigation systems, effectively managing and visualizing navigation routes is critical for guiding users through complex environments. Vector tiles offer a robust framework for this task, enabling dynamic route rendering while utilizing predefined map data to enhance performance and user experience.

To streamline the visualization process, vector tiles often include a comprehensive route layer that encompasses all possible routes within the map. Instead of generating new route visualizations from scratch for each request, the system utilizes this predefined layer. The process involves:

- **Querying Existing Data:** Upon calculating the route, the system queries the predefined route visualization layer to identify and extract the relevant segments that correspond to the calculated path.
- **Highlighting Routes:** The extracted route segments are overlaid on the map, using styles that distinguish the active route from other elements. This might involve applying different colors or thicknesses to the route for clarity.

One of the key reasons for using predefined route data is the complexity of indoor layouts. In many environments, edges and routes may vary in width and other characteristics, making it challenging to generate accurate visualizations in real-time. Predefined routes ensure that these variations are accounted for and consistently represented.

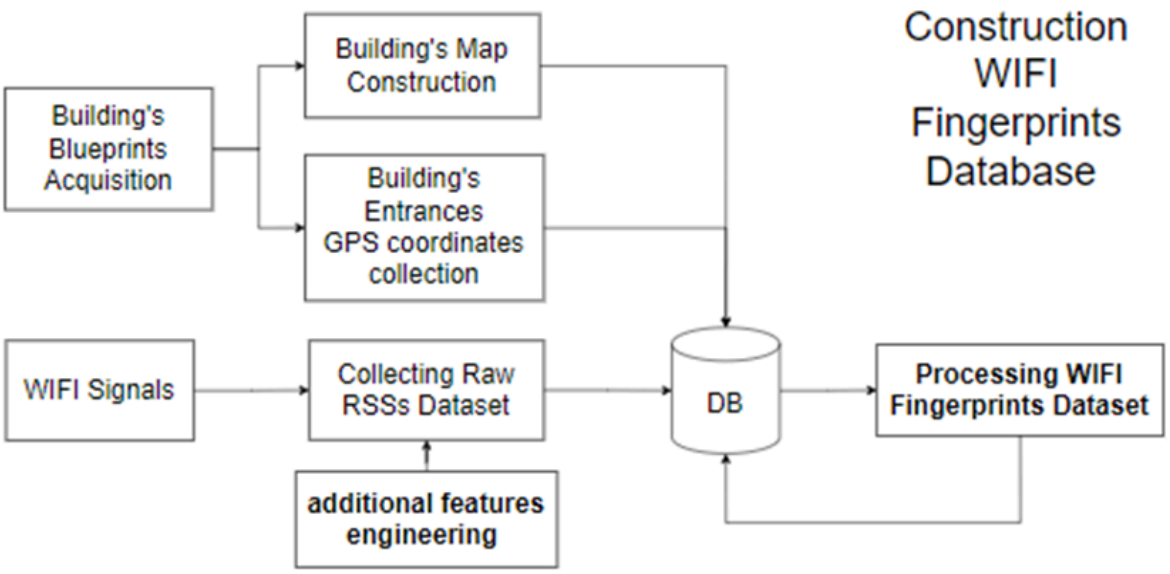
Route Generation: Routes within vector tiles are often generated in advance through either manual placement or automated algorithms. Manual methods involve placing nodes and defining routes based on known pathways and layout features, while algorithms can dynamically generate routes by analyzing the building's layout and obstacles. Proper placement of nodes and consideration of physical barriers are crucial for accurate route generation.

13.4.4.Logical Map - Spatial Graph Representation

Our logical digital map employs a graph-based representation, where each node on the graph represents a unique location within the mapped environment. These nodes serve as vital landmarks, denoting destinations, crossroads, and entrances. The connections, or edges, linking these nodes are defined to match real-world walking distances, measured in meters, ensuring accurate and reliable navigation throughout the mapped space. This digital map forms the basis for implementing graph navigation algorithms that enable efficient route planning and guidance for users.

Annotation: The filling of each node and edge will be done manually, following the database design outlined in the "Building Map Data Collection Schema" section.

13.5.Establishing a WIFI Fingerprints Dataset



[Diagram 13.5 - Wifi Fingerprints Database]

Objective: creation of a database which will be accurately representable of the user position in building space mapped by the RSS signals strength of the Access points at the corresponding position.

In our pursuit of achieving low cost precise indoor user positioning, we harness the existing Wi-Fi signal infrastructure. This intricate process involves a thorough analysis of Received Signal Strength (RSS) emanating from nearby Wireless Access Points (APs) using a user's smartphone network card, antenna, and integrated software program.

Notably, Wi-Fi RSS signals often exhibit inaccuracies within complex indoor environments due to distortions caused by physical obstructions such as walls, users, obstacles, and the terrain itself.

RSS: Signal Strength measured in decibel milliwatts (dBm)

RSS dBm values usually defined as follow:

- -50 dBm: excellent
- -60 dBm: Very good
- -70 dBm: Good
- -80 dBm: Low
- -90 dBm: Very Low
- -100 dBm: No Signal

Access points in a typical Wi-Fi network are generally installed in fixed positions and don't move around once they are deployed. The location of APs is carefully planned and set up to

provide consistent and reliable wireless coverage throughout a building or area. These installations are typically performed by IT professionals or network administrators.

Therefore we have adopted the Finger Matching Model, a widely-accepted navigation technique for estimating user locations. This model necessitates the collection of predefined positions and their corresponding RSS values.

The overall process comprises several pivotal stages:

Accurate Building Map Acquisition: Described Above.

Collecting Raw Fingerprints Dataset:

- Multiple measurements at a single point.
- RSS points at key positions.
- More points the better.

Following the acquisition of the building's layout, we embark on constructing the raw fingerprint dataset. This phase entails the systematic collection of data points at predetermined intervals, chosen based on signal characteristics and requirements. We capture multiple sets of data at each designated point within the 3D space, thus generating a rich dataset comprising multiple RSS readings from various Access Points (APs).

It is important to note that this raw data serves as the foundation for subsequent processing steps. The greater the number of data points we collect, the higher the quality of the final processed data will be. This meticulous data collection process forms the cornerstone of our positioning algorithm, enabling us to enhance accuracy and reliability in indoor user positioning. This can include multiple measurements for the same points at different timestamps as wifi signals tend to fluctuate.

Additional features engineering: is described in the “Raw WIFI Fingerprints Collection Schema” section.

- Additional features should include SSID and BSSID.

Preprocessing Raw WIFI Fingerprints Dataset

establishing a processed WIFI fingerprints dataset.

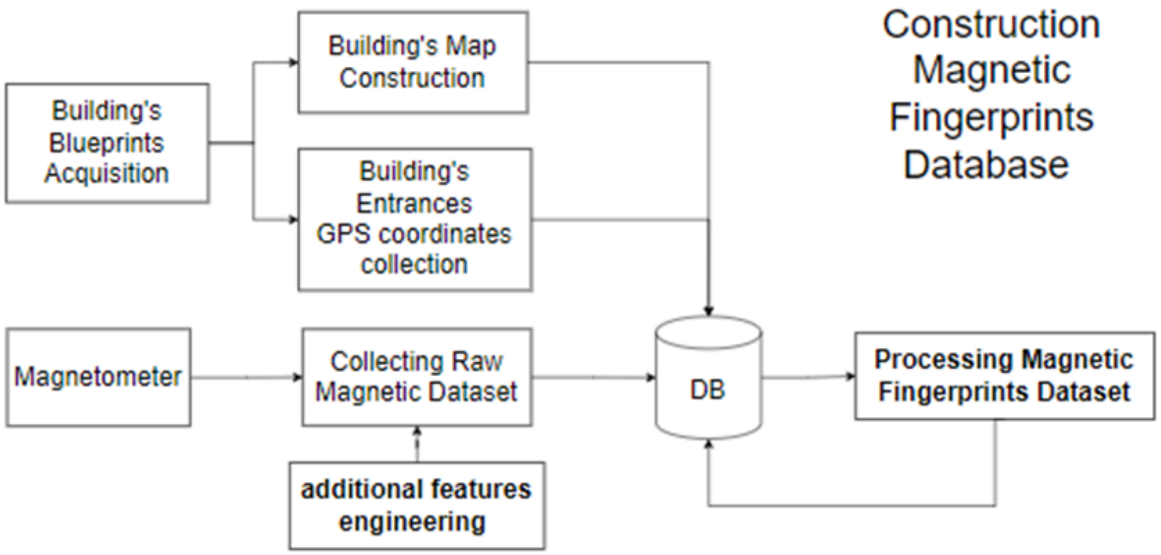
Using the raw fingerprints dataset gathered within the building, we engage on a preprocessing journey that involves employing multiple techniques, notable ones are signal averaging from the Access Points (Aps), networks filtering, access points filtering, threshold filtering, statistical methods, interpolation and machine learning techniques.

Initially we will use the raw RSS values collected, as some techniques require the sequence, but additional features can be added such as described above.

This process culminates in the creation database, yielding position relative to the building map and its corresponding preprocessed RSSs signals : <Position, RSSs, additional features> which results in a signals map.

It's noteworthy that these preprocessing techniques remain dynamic and experimental, subject to potential evolution and refinement over time to ensure the ongoing accuracy and reliability of the database.

13.6.Establishing a Magnetic Fingerprints Dataset:



[Diagram 13.6 -Magnetic Profile Database]

Objective: creation of a database which will be accurately representable of the user position in building space mapped by the magnetometer readings of the magnetic field at the corresponding position.

In our pursuit of achieving low cost precise indoor user positioning, we harness the existing magnetic field within a building. This intricate process involves a thorough analysis for the magnetic field readings.

Magnetic fields, very volatile, can suffer getting closer to other electronics and irons, this can include other smartphones and, in addition, the magnetic field is very weak to changes to placement of electronic devices.

It’s important to note that the magnetic field frequency is much higher than the WIFI sampling frequency, causing the dataset to be a lot bigger.

The overall process comprises several pivotal stages:

Accurate Building Map Acquisition: Described above.

Collecting Raw Magnetics Profiles Dataset:

Following the acquisition of the building's layout, we embark on constructing the raw fingerprint dataset. This phase entails the systematic collection of data points at all building points. We capture multiple sets of data at each designated point with multiple directions, thus generating a rich dataset comprising multiple magnetometer readings at each single point.

The greater the number of data points we collect, the higher the quality of the final processed data will be. This meticulous data collection process forms the cornerstone of our magnetic matching algorithm, enabling us to enhance accuracy and reliability in indoor user positioning.

Additional features engineering: is described in the “Raw Magnetic Profiles Collection Schema” section.

- Additional features should include device’s orientation
- Collecting a point , with people around might not be valid.

Preprocessing Raw Magnetic Profiles Dataset

establishing a processed Magnetic Profiles database. Using the raw fingerprints dataset gathered within the building, we engage on a preprocessing journey that involves employing multiple techniques. This process culminates in the creation database, yielding position relative to the building map and its corresponding preprocessed magnetic readings:

<Position, Magnetic Magnitude, Magnetic Velocity Vector, additional features>

Which results in a magnetic map. It's noteworthy that these preprocessing techniques remain dynamic and experimental, subject to potential evolution and refinement over time to ensure the ongoing accuracy and reliability of the database.

13.7.Map Engine Localization:

Maps Transitions Areas Management:

Objective: This algorithm serves the purpose of identifying whether a user is entering a predefined building within our application. It achieves this by employing GPS coordinates to ascertain the user's location relative to a designated entrance area positioned outside the building, ensuring optimal GPS signal reception. This determination is pivotal for orchestrating the transition to the corresponding map tailored to the building in question. In scenarios where the GPS signal is absent while the user is already inside the building, it also provides a user-friendly manual selection option, granting the user access to the same map. Offset is also introduced because of GPS meter accuracy.

Algorithm:

1. System Initialization:

1. Activate the GPS functionality.

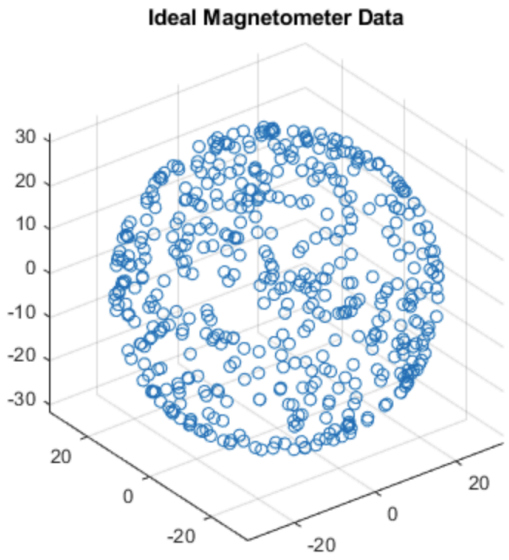
2. **If GPS Signal Available (indicative of the user being in a public area):**
    a. System Display user position in global map
    b. **IF User In Predefined Area/Building Area + offset:**
        i. System display building area map.
3. **Else (GPS Signal Unavailable, indicating the user is already inside a building):**
    a. Allow the user to manually select the building.
    b. Initiate the map associated with the selected building.
    c. Get the user's location using the navigation engine.

13.8.Sensors Overview

Before delving into the core components of the engine, we would like to provide some background on the infrastructure used. Accurate readings from the algorithms require several steps to manage random environmental biases and errors.

In the following sections, we will discuss sensor calibration, sensor fusion, and the application of machine learning techniques to ensure precision and reliability.

13.8.1.Sensors Calibrations:



[Image 13.8.1.Magnetometer Calibration Example]

Sensor calibration, vital for electronic devices like smartphones, enhances measurement accuracy and consistency. This process corrects inherent biases, environmental influences, and manufacturing imperfections in sensors. The primary goal is to bring sensor data as close as possible to true values and ensure measurement consistency across different devices and changing environments.

Calibration methods vary, with options such as algorithmic adjustments and using default calibrated data from Android and iOS systems. However, the latter may not cover all

scenarios perfectly. Combining multiple sensors can further improve accuracy by compensating for biases introduced by other sensors.

In our system, we employ algorithms and sensor combinations to enhance accuracy. Calibration can occur during the initiation stage or as in-run calibration, particularly benefiting gyroscope and accelerometer data. Uncalibrated data also serves purposes like benchmarking, error estimation, and testing to assess sensor performance.

Our System

we will use the various sensors with the default OS calibrated data. Unlike the accelerometer and gyroscope the magnetometer is more susceptible to errors, therefore after collecting the magnetometer readings, we will employ techniques to evaluate magnetometer data which are presented in paper [9].

13.8.1.Sensors Fusion:

Sensor fusion also known as data fusion is a subset of information fusion ,and it's a pivotal technique within sensor technology, it involves the integration of data collected from multiple sensors to improve the accuracy, consistency, and overall reliability of the information obtained.

The concept of sensor fusion centers on the idea that individual sensors, while valuable in their own right, can have inherent biases, inaccuracies, or variances. These discrepancies can arise due to a multitude of factors, including environmental conditions, component tolerances, and manufacturing imperfections. Consequently, relying solely on the data from a single sensor may lead to incomplete or less accurate information.

Sensor fusion algorithms can be broadly classified as such because they focus on combining data from different sensors to enhance the reliability of readings. Their key purpose is to merge and optimize data, often from sensors like accelerometers, gyroscopes, and magnetometers.

While sensor fusion may not be an absolute necessity, it is highly likely to significantly enhance the reliability of data readings. Initially, we work with raw sensor data to gain a clear understanding of the information. As we make advancements, we will adapt to specific sensor fusion techniques tailored to indoor navigation scenarios. These techniques are supported by various algorithms presented in relevant papers that align with our engine's requirements.

The fusion/s most likely will occur **INS** stage, specifically in the **AHRS** stage, yet we are open to the wide variety of solutions, which do exist.

We would like to talk about some of the common options available.

13.8.1.1.Common Sensors Combinations in Indoor Navigation Systems:

- **MARG sensors** – magnetic, angular rate, gravity. (Magnetometer, gyroscope, accelerometer correspondingly.)

13.8.1.2.Common Information Fusions Filters in Indoor Navigation Systems:

These following filters are mentioned in various indoor navigation papers.

- **Extended Kalman filter (EKF):**
  - Strengths: The Extended Kalman Filter is widely used for state estimation in nonlinear systems. It is effective in fusing sensor data for various applications, including navigation and control.
  - Weaknesses: EKF assumes that the system is approximately linear, which can lead to inaccuracies in highly nonlinear systems. It may also be computationally intensive.
- **Unscented Kalman filter (UKF):**
  - Strengths: The Unscented Kalman Filter is a nonlinear variant of the Kalman Filter. It provides a more accurate estimation of state in nonlinear systems compared to the EKF.
  - Weaknesses: The UKF can be computationally more demanding than the EKF.
- **Particle Filter (Monte Carlo Localization - MCL):**
  - Strengths: The Particle Filter, or Monte Carlo Localization (MCL), is a versatile filter that can handle non-Gaussian and highly nonlinear estimation problems. It is particularly effective in scenarios with significant uncertainties.
  - Weaknesses: It can be computationally intensive, and the performance heavily depends on the number of particles used.
- **Quaternion EKF with gradient descent (GD)**:
  - Strengths: Quaternion-based EKF with Gradient Descent is used for orientation estimation, particularly in applications involving IMUs. It offers a stable and efficient solution for attitude estimation.
  - Weaknesses: It may not be as accurate as other methods in scenarios with large disturbances.
- **Madgwick Filter:**
  - Strengths: The Madgwick filter is known for its computational efficiency and suitability for resource-constrained applications. It is capable of providing reasonably accurate orientation estimates, making it a popular choice for real-time systems.
  - Weaknesses: While efficient, it may not achieve the same level of accuracy as more complex algorithms, particularly in scenarios with significant sensor noise.
- **Mahony Filter:**
  - Strengths: The Mahony filter is an extension of the Madgwick filter and offers a good trade-off between computational efficiency and accuracy. It is well-suited for applications that require a balance between performance and real-time processing.
  - Weaknesses: While more accurate than the Madgwick filter, it may still be outperformed by more sophisticated algorithms in dynamic and challenging environments.
- **Complementary Filter-Based AHRS:**
  - Strengths: Complementary filters offer a simple yet effective way to combine sensor data and provide orientation estimates. They are often used in low-power or resource-constrained applications.

- Weaknesses: They may lack the accuracy of more advanced filters and are sensitive to sensor biases and noise.

13.9.Machine Learning:

Machine learning offers tremendous potential for improving indoor positioning by intelligently combining data from various sources to pinpoint user locations with high accuracy. By leveraging algorithms that can learn and adapt from vast amounts of sensor data and complex patterns, machine learning can overcome challenges such as signal obstruction and multipath effects common in indoor environments. This can lead to more precise and reliable user location tracking, enhancing navigation experiences in complex settings like hospitals and malls. However, the adoption of machine learning for this purpose requires significant investments in time, computational power, and specialized expertise. Given our current constraints in budget and resources, we are unable to implement such advanced techniques at this stage. Instead, we will continue to utilize more traditional methods until we can allocate the necessary resources to fully harness the capabilities of machine learning for indoor positioning.

13.10.Inertial Navigation System (INS):

An Inertial Navigation System (INS) is a navigation technology that relies on motion sensors, including accelerometers and rotation sensors known as gyroscopes, combined with specialized software algorithms. This system is designed to continuously compute and update the position, orientation, and velocity of a moving object autonomously, without relying on external references or signals.

At the heart of every INS lies the integration of accelerometers and gyroscopes. These sensors provide critical data for the system to perform dead-reckoning, a method of estimating position and orientation based on changes in motion, particularly rotational changes in space.

While the magnetometer can enhance the system's performance, it remains optional and is considered more of an improvement rather than a core component due to interference,

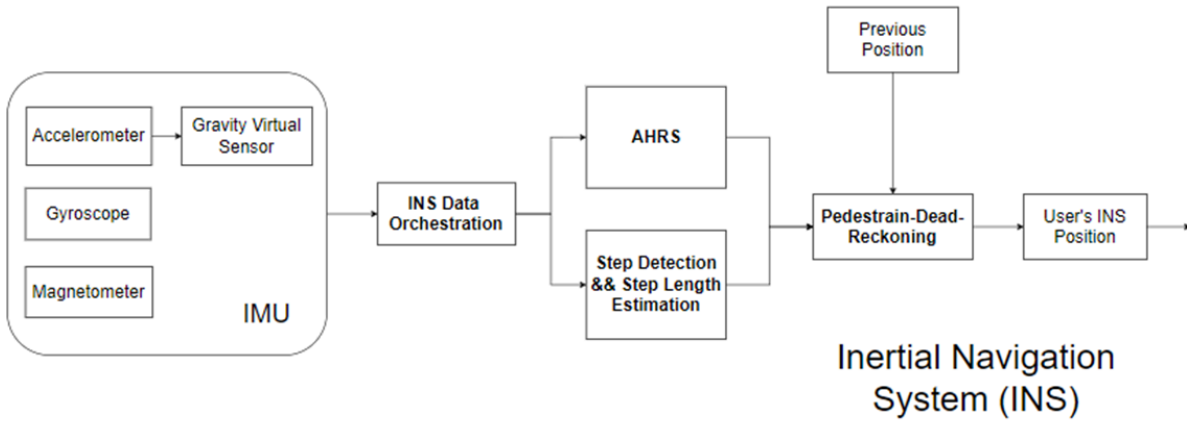
Indoor navigation is not a fixed, one-size-fits-all problem, and the development of Inertial Navigation Systems (INS) can take various paths depending on available resources and specific application needs.

We are actively exploring the diverse structural approaches outlined in the research papers to tailor Inertial Navigation Systems (INS) for our specific needs. The high-level diagram serves as a useful framework, outlining the flow of the INS process. However, it's important to note that certain blocks within this diagram may not always be utilized in our implementation. For instance, while sensor fusion is a valuable technique, we might opt to work with raw data in some scenarios. Additionally, the necessity of fusion might vary depending on the algorithm employed in different blocks, introducing flexibility into our

approach. This adaptability allows us to fine-tune our INS to match the precise requirements of the task at hand.

Here's a high-level overview encompassing a broad spectrum of possible techniques in the realm of Inertial Navigation Systems (INS), ensuring that our INS implementation is precisely tailored to its intended purpose, optimizing its performance and functionality.

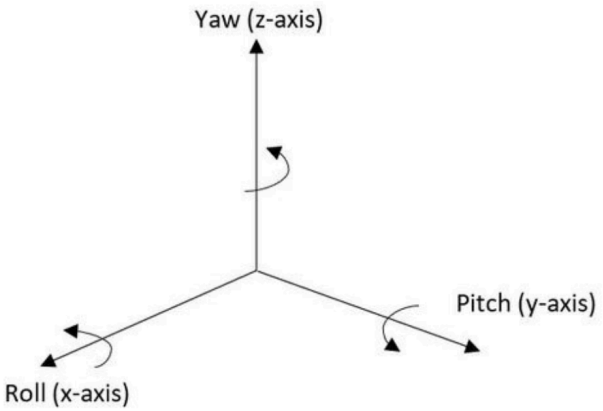
13.10.1.High Level Overview:



[Diagram 3.10.1.INS]

13.10.2.Understanding Spatial Coordination in 3D Space (Euler angles):

Indoor navigation depends on a clear grasp of spatial coordination systems such as azimuth, roll, and pitch. Azimuth measures horizontal angles clockwise from north, while roll and pitch indicate rotations around lateral and longitudinal axes. These standards ensure precise communication and accurate direction interpretation in complex indoor environments, aiding developers and users in effective navigation and spatial positioning.



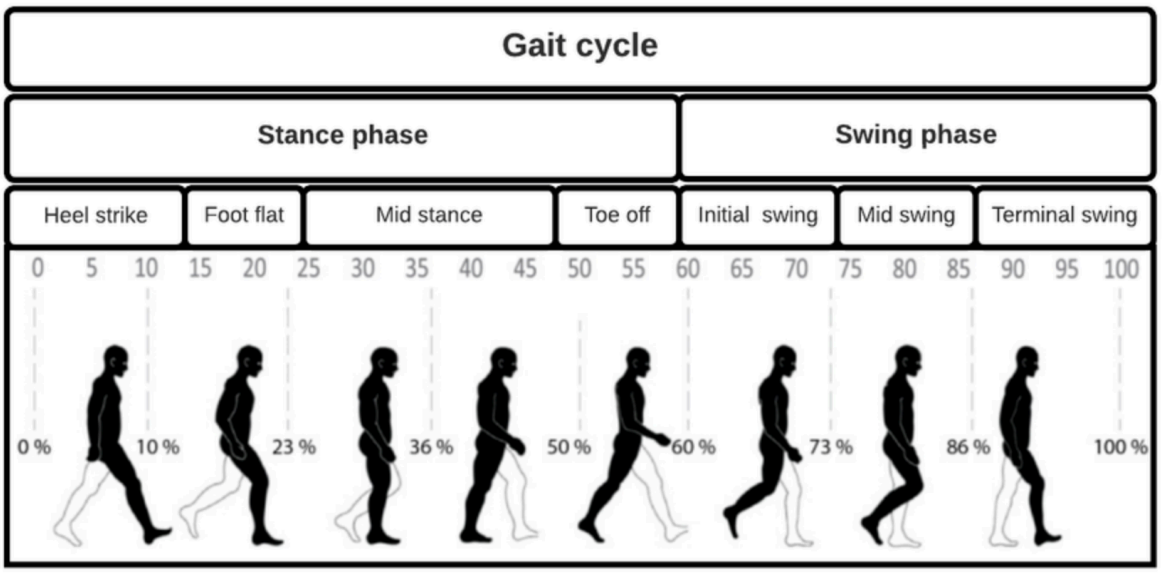
[Diagram 13.10.2 - Euler Angles]

13.10.3.INS Data Orchestration:

Efficiently overseeing sensor data, prioritizing error reduction and noise minimization. This procedure frequently encompasses methods such as filtering, calibration, and noise reduction. Given our specific need for a motion-free smartphone system, some conventional techniques may not be applicable to our case. In our initial deployment, we will exclusively employ **Sensor Calibration.**

13.10.3.Pedestrian Dead Reckoning - PDR

In the realm of navigation, Dead Reckoning (DR) is a technique used to determine the current position of a moving object. It relies on a known initial position (a "fix") and incorporates estimates of speed, heading, and elapsed time. However, since DR is a relative navigation method and MEMS IMUs (Micro-Electro-Mechanical Systems Inertial Measurement Units) are prone to high errors, it is susceptible to accumulating inaccuracies. To address this issue, Pedestrian Dead Reckoning (PDR) was developed, specifically tailored for pedestrian navigation. PDR leverages human motion information, particularly the gait cycle (GC), to improve its accuracy.



[Image 13.10.3.Gait Cycle]

PDR encompasses four primary components: spatial orientation estimation, step detection, heading estimation, and step length estimation. Each of these elements plays a critical role in the PDR system and will be discussed in detail.

PDR systems can be divided into two main categories: strapdown systems(device is attached to the body) and free-moving systems (handheld devices). Strapdown PDR systems make use of the human gait cycle, but in our specific system,which is not a strapdown system, utilizing the human gait cycle is feasible primarily when the device remains stationary relative to the user's body. However, accommodating free movement introduces significant complexity.

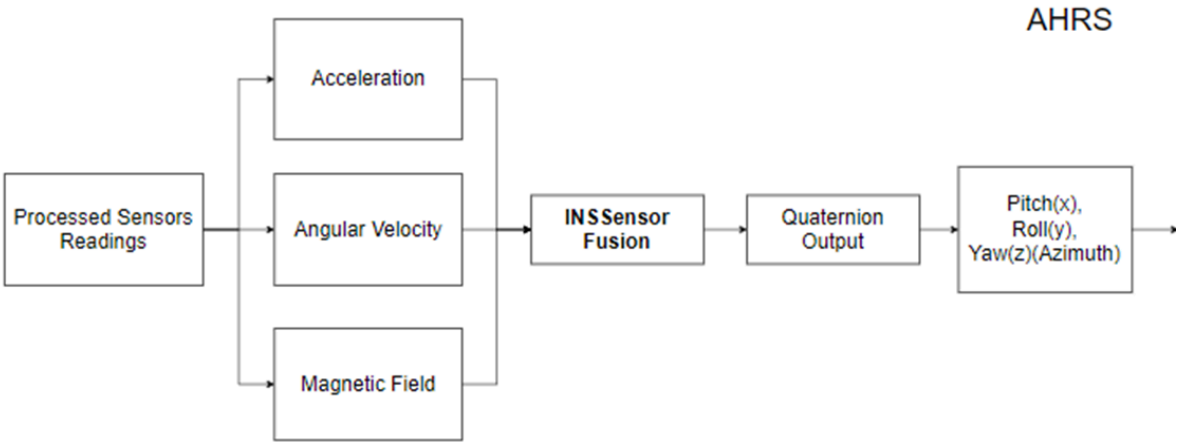
However, despite the challenges, PDR provides distinct advantages over traditional DR methods in pedestrian navigation systems. Therefore, PDR is a suitable choice for our system, and the algorithm is defined as follows:

$$\boldsymbol{Position} = \boldsymbol{PrevPositon} + \begin{bmatrix} \boldsymbol{SL} \cdot \boldsymbol{cos}\,\psi \\ \boldsymbol{SL} \cdot \boldsymbol{sin}\,\psi \end{bmatrix}$$

$$\psi = \text{yaw(Azimuth)}$$

$$\text{SL} = \text{Step Length}$$

13.10.3.AHRS – attitude and heading reference system:



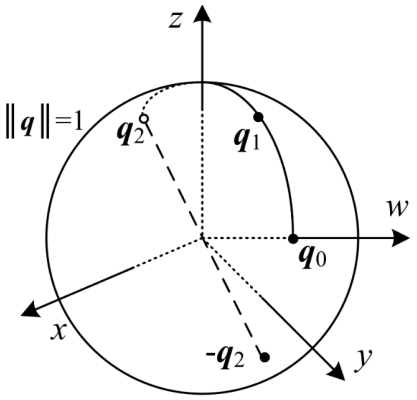
[Diagram 13.10.3.AHRS]

Estimating the user's orientation in space, utilizing the Inertial Measurement Units (IMUs) within the device, is a crucial aspect of our navigation system. The orientation estimation is fundamental for understanding the user's position and movement. One essential point to emphasize is that the device's heading direction may not always align perfectly with the user's actual heading, necessitating a thoughtful approach to handling this misalignment. In this context, a variety of AHRS methods are available, each tailored to the specific requirements and constraints of diverse applications.

However, for indoor navigation, one widely adopted and particularly robust approach is the **quaternion-based** methods. This method excels in providing accurate and efficient

estimations of the device's attitude, making it an ideal choice for tracking user movements in confined spaces.

Quaternions are preferred due to their ability to avoid gimbal lock (loss of one degree of freedom in a multi-dimensional mechanism at certain alignments of the axes), a phenomenon that limits the accuracy and stability of Euler angle representations, particularly in highly dynamic situations.



[Diagram 13.10.3.Quaternions]

The processed sensors readings that came after the INS data orchestration, using this data, we get the angular velocity from the gyroscope, magnetic field magnitude and vector, the acceleration from the accelerometer and the gravity from the virtual gravity sensor , using those information , we can calculate the attitude of the device.

Heading direction: The system assumption is that the user heading direction (azimuth), is the same as the device heading direction.

Initial localization direction: Initial position in the building can be acquired by using either GPS/GNSS outdoors, which also provides heading , by using WI-FI indoors, or by using WI-FI with Magnetic field.

13.10.3.1.Heading Direction

In Order to get smartphone’s heading direction or realign heading a few choices arise:

- Option 1 Magnetometer(Not Very good, Easiest): Upon entering an entrance, the initial entrance heading direction, will equate to the user heading direction. - Probable, not bulletproof, yet might be sufficient.

$$m_{magnitude} = \|\vec{m}\|_2 = \sqrt{m_x^2 + m_y^2 + m_z^2}$$

$$magnetic\ field\ vector = \frac{\vec{m}}{\|\vec{m}\|_2} = \left[\frac{m_x}{m_{magnitude}}, \frac{m_y}{m_{magnitude}}, \frac{m_z}{m_{magnitude}}\right]$$

$$initial\ north\ based\ \psi = atan2\left(\frac{\frac{m_y}{m_{magnitude}}}{\frac{m_x}{m_{magnitude}}}\right) = atan2\left(\frac{m_y}{m_x}\right)$$

- Option 2 Orientation: using roll, pitch, yaw after sensor fusion and get the yaw to use that as the heading of the user.
- Option 2 respective to node predefined directions: Upon entering a node, the user heading direction, will equate to closest the available node heading. (requires adding possible heading directions for each node)
- Option 3 vector subtraction: After getting the user's position in the map, waiting for a new position, thus requiring the user's vector.
- Option 4 PCA: PCA algorithm in order to get the heading direction, requiring a way to determine forward or backward. Presented in reference [2].

The attitude mechanization suffers from drift from measurement integration and imposes the need for periodic attitude corrections and realignment between the body frame and the site frame. The alignment approach can be divided into 2 phases, the first one is leveling where tilting angles –roll and pitch – are estimated, and then the azimuth is corrected from projected magnetic measurements in the site frame. For tilt compensation the accelerometer measurements are used to extract gravity information for the leveling process

13.10.4.Step Detection && Step Length Estimation:

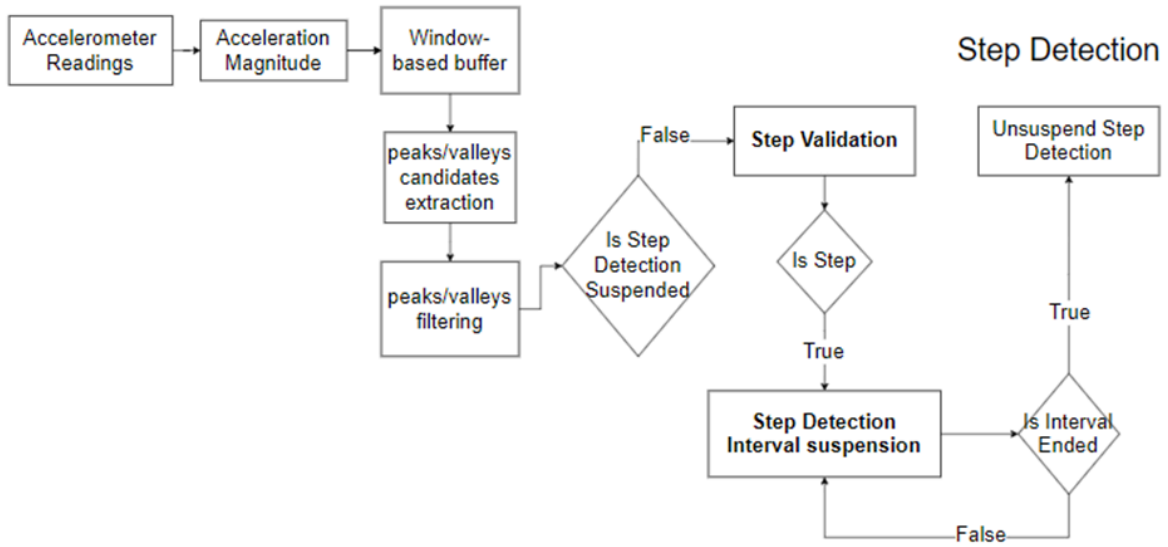
Step detection and step length estimation is the main component in the INS system for pedestrians, utilized by the PDR mechanism in the INS.

PDR requires a robust step detection algorithm, the effect of missing a step or fake step count can vary on the performance of the INS system.

In a fixed step length, getting a wrong step length or non-consistent step detection will increasingly raise the drift error by time, in a standalone INS system.

Step detection algorithms typically rely on pattern detection in sensor measurements. In a strapdown system where the GC is utilized, a sensor attached to the user body will greatly benefit from capturing the motions, but in our system, where the smartphone is free-moving, the complexity raises.

13.10.4.1.Step Detection:



[Diagram 3.9 Steps Detection]

The stepping orientation of the user's is determined by the threshold of the magnitude of the peaks and valleys in the accelerometer peaks and valleys data. The higher being running, the lower being walking.

Step detection of the user from his smartphone, while allowing the smartphone to move freely, commonly requires incorporating the peaks and lows readings from the accelerometer. The widely known algorithms such as ZURP and MARU do not hold with the free-moving smartphone. Each accelerometer peak in magnitude is a candidate (can be false positive) for a step. These peaks and lows methods are implemented using accelerometer magnitude window-based intervals.

Window-based buffer:

Collection of the recent accelerometer values. The variation of the magnitude can be volatile during the progression of the user, a fixed larger window size may not be able to handle the variations in the recent magnitudes statistics. Therefore a smaller fixed sized window is desired

Peaks/Valleys Candidates Extraction:

Peak candidate: $\overrightarrow{\|a_k\|} > \max\left(\overrightarrow{\|a_{k-1}\|}, \overrightarrow{\|a_{k+1}\|}\right)$

Valley candidate: $\overrightarrow{\|a_k\|} < \min\left(\overrightarrow{\|a_{k-1}\|}, \overrightarrow{\|a_{k+1}\|}\right)$

Peaks/Valleys Filtering:

Every peak candidate is validated by checking the time distance to the recent peak using the following threshold.

$B = constant$
μ = average of the time interval between peaks/valleys.
σ = standard deviation of the time interval between peaks/valleys

$$Threshold_{peak} = \mu_{peak} - \frac{\sigma_{peak}}{B}$$

$$Threshold_{valley} = \mu_{valley} - \frac{\sigma_{valley}}{B}$$

Step Validation:

$$d_n = \iint_{k=start}^{end} \overrightarrow{\|a_k\|} > threshold_{step} = \frac{0.6}{k} \sum_{t=1}^{k} d_{n-t}$$

$nth\ step\ displacement$

Step Detection Interval Suspension:

Suspend the step validation operation for a time frame. Might not be used. Since the update is initially adaptive and not time based, it will be none.

13.10.4.2.Step Length Estimation:

$$SL = K_{step} \times \sqrt[4]{a_{y\_max} - a_{y\_min}}$$

$$K_{step} = 0.68 - 0.37 \times v_{step} + 0.15 \times v_{step}^2$$

Step length estimation, is an important step in PDR algorithm which decides the distance the user takes at each step cycle. There are various ways of estimation in step length, such as machine learning algorithms, statistical methods, windows, zero-crossing , an immutable value, and immutable value by correlation of sex, age, or height.

The paper [4] presented an easy model using Weinberg Model with a personalized set of constraints.

Step length can be obtained by simple approximation by age, sex and height. As shown in paper [2] multiple references for studies that show correlation between those parameters.

13.11.WIFI Fingerprints Matching and Magnetic Profiles Matching Base Structure:

Within our indoor navigation system, Wi-Fi fingerprint matching and magnetic profiles matching serve as integral components. They both share a unified structure designed to facilitate the accurate localization of users within indoor environments. This unified approach ensures consistency and reliability throughout the system. Our decision to divide the matching algorithms into sub-components is underpinned by several key benefits. Firstly, it promotes consistency and offers precise control over the matching process. Additionally, this approach encourages code and component reusability, simplifying development and maintenance. By creating a clear development path through sub-components, we enhance the transparency of the entire process, allowing for easier issue identification and resolution. This structure not only drives precision but also boosts efficiency, collectively providing a robust foundation for our indoor navigation system. Ultimately, it guarantees that Wi-Fi fingerprint matching and magnetic profiles matching work seamlessly together to deliver accurate and reliable location data.

1. **Data Orchestration:** Data orchestration encompasses the preprocessing of sensor data, including steps such as normalization, filtering, scaling and buffering. This phase ensures that the data is prepared for meaningful comparison.

2. **Similarity Metric:** Similarity metrics are fundamental to these algorithms. They quantify the similarity between two data points, helping us assess how closely they align. Commonly used metrics include Euclidean distance, cosine similarity, and Jaccard index.

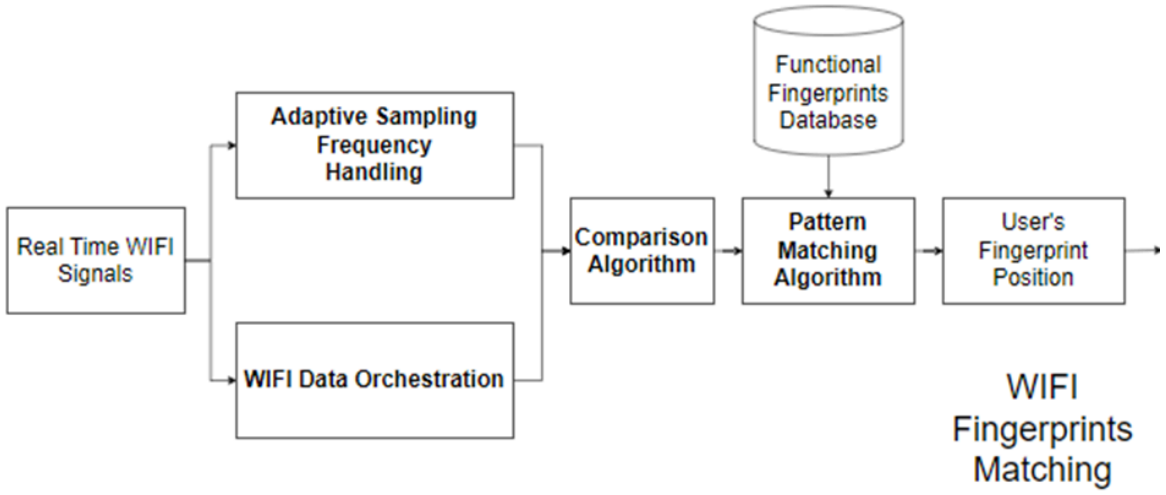
3. **Comparing:** The comparison step is essential for these matching algorithms and offers two distinct options:

    a. **Point To Point Comparison:** This method involves the direct comparison of individual data points between real-time readings and database entries. It assesses the similarity between two single data points, indicating how closely they correspond.

    b. **Sequence Comparison:** Sequence comparisons focus on evaluating the similarity between data sequences, rather than individual data points. Several sub-options exist, such as time interval-based comparisons, where sequences are compared over specific time periods. Rolling window comparisons involve assessing data sequences within sliding windows, enabling a more dynamic matching process. The choice of comparison method depends on the context of the application and the desired level of granularity.

4. **Matching:** Matching algorithms follow the comparison step and employ the comparison output to establish the best match between real-time data and database entries. These algorithms play a crucial role in pinpointing the most accurate location estimation.

13.12.Periodic WIFI Fingerprints Matching:



[Diagram 3.10 - WIFI Fingerprints Matching]

Objective: periodic fingerprint matching to correct user's position and reset position drift error from INS subsystem.

To achieve accurate positioning and reduce drift errors from the Inertial Navigation System (INS), our objective is to implement periodic fingerprint matching. Wi-Fi fingerprinting offers versatile algorithms, including triangulation, historical data usage, comparison methods, and machine learning techniques. These approaches enable the system to periodically compare Received Signal Strength (RSS) values from the user's device with stored reference RSS values. By identifying the closest matches, the system corrects the user's position, thereby enhancing accuracy and minimizing drift errors.

Moreover, Wi-Fi fingerprint matching plays a crucial role in refining the search area for magnetic matching. This optimization further enhances the overall accuracy and performance of the localization process.

It's important to note that each smartphone has a limit on the number of scans it can perform within a given timeframe. This limitation restricts algorithms that rely heavily on frequent scans. Therefore, adjustments must be made to align with these constraints or consider alternative approaches. For instance, on Android 9 and above, the limit is set to four scans every two minutes.

13.12.1.WIFI Data Orchestration

Wireless signals introduce notable variations in Received Signal Strength (RSS), often leading to inconsistencies when matching the real-time data to the fingerprints database data. To surmount this challenge a few techniques will be used:

Threshold-Based Filtering:

This mechanism identifies and eliminates unreliable signals within a defined area. Our engine uses three threshold mechanism at this order, respectfully:

1. **Filtering by Number of Observed APs** (sequence quality control): (paper[4])

Filtering by the number of observed Access Points (APs) is a technique used to assess the reliability and relevance of Wi-Fi networks based on how many APs a device can detect. It helps determine the quality of the Wi-Fi signal environment and can serve as a criterion for whether or not to trust the information provided. The steps involved are as follows:

Configuration: APThreshold = 3 (initial value):

Procedure: IF CountAPs (RSSs) < APThreshold (paper [4]): Remove RSSs sequence (skip)

2. **Filtering by RSS Threshold** (signal quality control):(Paper[1], paper[4])

Filtering RSS values below a threshold is primarily used for signal quality control.

Configuration: RSSThreshold = -100 (initial value)

Procedure: IF RSS < RSSThreshold: Remove unreliable RSS signals. (Skip)

3. **Filtering by Distance** (defining a spatial range.): (paper[4])

Filtering by distance, is used to define a region of interest or to identify devices or access points within a certain proximity. It's about defining a spatial boundary based on the estimated or measured distances. Meaning only caring about signals from access points within DistanceThreshold meters, regardless of their RSS values.

Calculate RSS access point distance using Free Space Path Loss is a common technique, but in indoor system where there obstacles in your path, diffraction effects and the AP varying its power output to save energy, it is impractical to obtain a reliable distance figure just looking at RSS value, A reliable way to measure the distance is using the time-of-flight (TOF), this means calculating the roundtrip time between the device and the access point.

Configuration: DistanceThreshold = -10m (initial value)

Procedure: IF CalculateRSSDistance (RSS) < DistanceThreshold: Remove out of bound RSS. (Skip)

Calculate distance:

$$Distance = RTT * \frac{C'}{2}$$

$$C' = speed\ of\ light\ \cong 299{,}792{,}458 \frac{\mathrm{m}}{\mathrm{s}}$$

$$RTT - round\ trip\ time$$

Rolling Window RSSs Sequence Buffer:

Configuration: MSL – Max Fingerprints Sequence Length.

Initially: MSL = 5.

Buffering the last MSL values for sequence matching.

Considerations:

1. TTL- Time to live for each element inside the buffer.
2. TTS- time to stay, TTL – time to live for each element inside the buffer.
3. Every element that expire from TTL/TTS:
    a. Averaging their RSSs values for each distinct AP, thus reducing the buffer size.
    b. Deleting from the buffer.
4. Mirror buffer values before normalization, then apply normalization at each new RSS input.

Sequence Normalization:

Normalizing the sequence buffer.

Initially: mean-standard deviation each dimension (paper [1])

Considerations:

1. Min-max
1. Vector Normalization
2. DBm value scaling
3. Logarithmic transformation

Adaptive Sampling Frequency Handling:

Objective: adapting to low device Wi-Fi scanning sampling frequency to System by filling with prediction.

It is essential to acknowledge that smartphone scanning demands substantial computational resources. Many manufacturers impose stringent limitations on Wi-Fi sampling frequencies to conserve power. To mitigate the potential for positioning inaccuracies resulting from low sampling frequencies, we have adopted some techniques.

- The required System WIFI scanning frequency set to.
- The device WIFI scanning frequency denoted as.
- The Wi-Fi scanning frequency is measured in MHz:

Initially: None, after some development Adaptive Interpolation RSSs Approximation.

Configuration: The WIFI-scanning frequency the system requires for efficient localization, considering the computational complexity and positioning accuracy.

CalculatingF1: Calculating f1 involves several factors, including the distance between access points, signal strength of access points, smartphone velocity, and heading direction. We will examine various options for setting this parameter.

Alternatively, f1 can be a fixed value that suits most scenarios. However, to accurately obtain the user's velocity and direction, irrespective of the phone's orientation in space, we require a fully operational Inertial Navigation System (INS).

Adaptive Interpolation RSSs Approximation Algorithm::

Interpolation technique adapting to the user's device between access points. That presented in paper [1]:

The WIFI scanning frequency of the user smartphone is denoted as

The WIFI scanning frequency set to which represents the amount of scans needed to get accurate positioning , therefore it is highly influenced by the smartphone's velocity, and the access points distances from each other access point.

In each sampling frequency interval, each periodic first APs results are cached, until the next interval of WIFI scanning the cached results will update as follow:

$$N = floor\left(\frac{f1}{f2}\right) \quad N > 0$$

N – The number of interpolations needed until the next interval. (Floor = round down)

For each access point in current RSSs sequence, update its RSSs values by using this formula:

$$RSSs.AP.RSS_j = \frac{N-i}{N} * previous\ AP.RSS_j + \frac{i}{N} * current\ AP.RSS_j$$

$$note: the\ lower\ the\ signal\ the\ close\ to\ the\ access\ point$$

$Func\ AdaptiveWIFISampling()$:

- $f1 = \boldsymbol{getf1Value}()$
- $f2 = getf2Value()$
- $N = floor\left(\frac{f1}{f2}\right)$
- $i = 1\ \ (i\ denote\ the\ ith\ interpolation\ signal)$
- $prevRSSs = empty\ list$
- $currentRSSs = empty\ list$
- $scan = True$
- $Interval = f2 * 10^6$
- $\boldsymbol{WHILE}\ True$:
  - $currentf1 = \boldsymbol{getf1Value}()$
  - $\boldsymbol{IF}\ f1\ != currentf1$:
    - $f1 = currentf1$
    - $N = floor\left(\frac{f1}{f2}\right)$
    - $i = 1$
  - $RSSs\ = \boldsymbol{GetRSSSignals}(f2, N, i, scan, prevRSSs, currentRSSs)$
  - $\boldsymbol{IF}\ scan$: (the RSS values are scanning values)
    - $scan = False$
    - $prevRSSs = currentRSSs$
    - $currentRSSs = RSSs$
    - $i = 1$
  - $\boldsymbol{ELSE}$: (the RSSs values are approximations)
    - $Interval -= time$
    - $i\ += 1$
    - $IF\ Interval = 0$:
      - $Interval = f2 * 10^6$
      - $scan = True$

$\textbf{GetRSSSignals}(f2: integer\ , N: interger\ , i: integer\ , scan: boolean, prevRSSs: list, currentRSSs: list)$

- $IF\ scan$:
  - $RSS_{scan} = WIFIScan()$
  - $return\ RSS_{scan}$
- $ELSE$:
  - $RSSs_{approx} = CurrentRSSs$
  - $FOR\ index, currentRSS\ IN\ currentRSSs$:
    - $prevRSS = 0$
    - $ap = currentRSS.AP$
    - $current = currentRSS$
    - $FOR\ prevRSS\ IN\ prevRSSs$:
      - $IF\ ap = prevRSS.AP$:
        - $prevRSS = prevRSS.RSS$
    - $RSSs_{approx}[index].RSS = \frac{N-i}{N} * prevRSS + \frac{i}{N} * current.RSS$
    - $return\ RSSs_{approx}$

13.12.2.Comparing:

In Wi-Fi fingerprint matching, the comparison algorithm is employed to assess the connection between the database fingerprints and the measured signals. Single point matching is a straightforward method for computing the distance between the measured signal and the corresponding fingerprint in the database. This method involves comparing similar signals in the database to approximate a user's location. However, due to the inherent noise in collected wireless signals, single point matching often leads to mismatches.

In contrast, sequence matching relies on signals from multiple sampling points and incorporates historical data in its comparison process. The outcome of sequence matching is influenced by all the sampled points in the sequence. Even if there is a partial point mismatch, sequence matching has the potential to yield accurate results. Consequently, when compared to single point matching, sequence matching tends to minimize mismatches.

Initially: MDTW Algorithm (paper [1], paper [5]):

MDTW Algorithm: (paper [1]):

> DTW algorithm – dynamic time warping algorithm – a time series algorithm that measures the similarity between two likely unequal temporal sequences. DTW either extends or shortens the sequence of measurements to reach the same length as the reference sequence
>
> In the indoor positioning system, DTW is often used to calculate the distance between two one-dimensional signals
>
> For Wi-Fi fingerprint matching, each AP signal represents one-dimensional information. Multi-dimensional signals are collected because of the existence of multiple APs. Therefore, Wi-Fi fingerprint sequence matching is a multi-dimensional fingerprint matching.

This approach extends the fingerprint from one dimension to multiple, and thus enriches its diversity. Meanwhile, the multi-dimensional dynamic time warping method is introduced for matching multi-dimensional fingerprints with inaccurate profile lengths. Compared with the traditional single-point matching method, the profile-matching method provides a more reliable solution especially in environments with sparse distributed access points and a more accurate initial position.

Each sequence is a length of matrix dimensions, a similarity metric is measured in each point, then using dynamic programming to accumulate the distances, the smallest value in the accumulated rows and columns are the most similar between the sequences.

$$\boldsymbol{CalculateDTWAccumulativeDistanceMatrix}\begin{pmatrix} func\ \boldsymbol{SimilarlyMetric}, \\ sequence\ a, \\ sequence\ b \end{pmatrix}:$$

- $n = a.length$
- $m = b.length$
- $\boldsymbol{IF}\ n \geq m$:
    1. $dmatt = new\ matrix\ (n+1, m+1).fill(0)$
    2. $func\ \boldsymbol{distance} = (x, y) \rightarrow \boldsymbol{SimilarlyMetric}(a(x, y)), b(x, y))$
- $\boldsymbol{ELSE}$:
    1. $dmatt = new\ matrix\ (m+1, n+1).fill(0)$
    2. $func\ \boldsymbol{distance} = (x, y) \rightarrow \boldsymbol{SimilarlyMetric}(b(x, y)), a(x, y))$
- $\boldsymbol{pmatt} = \begin{pmatrix} \boldsymbol{distance}(\boldsymbol{0}, \boldsymbol{0}) & \cdots & \boldsymbol{distance}(\boldsymbol{0}, \boldsymbol{m}) \\ \vdots & \ddots & \vdots \\ \boldsymbol{distance}(\boldsymbol{n}, \boldsymbol{0}) & \cdots & \boldsymbol{distance}(\boldsymbol{n}, \boldsymbol{m}) \end{pmatrix}$
- $dmatt(1,1) = pmatt(1,1)$
- $\boldsymbol{FOR}\ i = 2\ \boldsymbol{TO}\ n\ \boldsymbol{DO}$:
    1. $dmatt(i, 1) = pmatt(i, 1) + dmatt(i-1, i)$
    2. $\boldsymbol{FOR}\ j = 2\ \boldsymbol{TO}\ m\ \boldsymbol{DO}$:
        - $dmatt(1, j) = pmatt(1, j) + dmatt(1, j-1)$
        - $dmatt(i, j) = pmatt(i, j) + min([$
            - $dmatt(i-1, j)$,
            - $dmatt(i-1, j-1)$,
            - $dmatt(i, j-1)$
        - $])$
    3. $\boldsymbol{END\ FOR}$
- $\boldsymbol{END\ FOR}$
- $\boldsymbol{IF}\ n \geq m$:
    1. $distance = dmatt[n][m]$
- $\boldsymbol{ELSE}$:
    1. $distance = dmatt[m][n]$
- $\boldsymbol{RETURN}\ dmatt$,distance

Similarity Metric: Initially: Similarity Metric (paper [1]):

$$\boldsymbol{SimilarlyMetric}(sequence\ a, sequence\ b) \rightarrow \sqrt[2]{\sum_{k=1}^{p}\left(\boldsymbol{a}(\boldsymbol{k},\boldsymbol{i}) - \boldsymbol{b}(\boldsymbol{k},\boldsymbol{j})\right)^2}$$

$$where\ \ p\ is\ the\ \max\ number\ \ of\ RSSs\ in\ certain\ point\ of\ time\ ,\ between\ the\ RSSs$$

Considerations:

- Adding access point BSSID to the RSS vector and adding weights with the most weight to the AP BSSID
- Adding network SSID, access point BSSID to the RSS vector and adding weights with the second most weight to the Network SSID, and the most weight to BSSID

13.12.3.Matching:

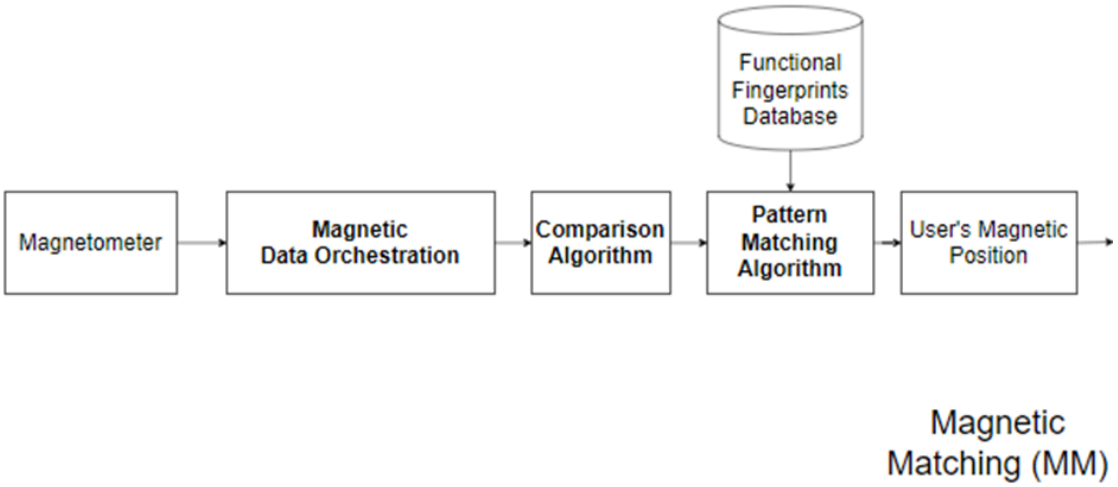
Initially: KNN - K-Nearest Neighbors, K = 1

KNN is a simple yet effective algorithm that works based on the concept of similarity. KNN identifies the k-nearest data points in the database to the new data point using a distance metric.

KNN is widely accepted across studies in indoor navigation context, this is a valuable pattern matching tool in this field.

Although KNN can be used as a standalone approach for Wi-Fi fingerprint matching, KNN can be used alongside other methods, combining the MDTW with KNN (paper [1]) , which means the distance metric between sequences will be the MDTW outputs between the sequences.

13.13.Magnetic Matching (MM):



[Diagram 3.11 - Magnetic Matching]

13.13.1.Magnetic Data Orchestration:

Configuration**:** MM Interval

Rolling Window Magnetic Sequence Buffer:

Configuration: MPSL – Max Profiles Sequence Length.

Initially: MPSL = 10.

Buffering the last MPSL values for sequence matching.

Considerations:

1. TTL- Time to live for each element inside the buffer.
2. TTS- time to stay, TTL – time to live for each element inside the buffer.
3. Every element that expire from TTL/TTS:
    a. Averaging their magnetic values, thus reducing the buffer size.
    b. Deleting from the buffer.
4. Mirror buffer values before normalization, then apply normalization at each new RSS input.
5. normalization

Threshold Filtering:

Standard-Deviation and Change Filtering (quality control): (paper [4])

Configuration: MagValuesMaxLength = 10s (initial values).

MagChangeThreshold = 4

MagStdThreshold = 4

Procedure:

$$diff_T = maximunValue - minimumValue \; at \; Time \; T$$

$$Std_T = Standard \; Deviation \; at \; Time \; T$$

$$\textbf{IF} \; diff_T < MagChangeThreshold \;\; \textbf{OR} \; Std_T < MagStdThreshold:$$

Skip MM matching.

13.13.2.Comparing & Matching:

Similarity Metric:

$$\textbf{SimilarlyMetric}(sequence \; a, sequence \; b) \rightarrow \sum_{k=1}^{p}(|\boldsymbol{a}(\boldsymbol{k},\boldsymbol{i}) - \boldsymbol{b}(\boldsymbol{k},\boldsymbol{j})|)$$

$$where \;\; p \; is \; the \max \; number \; of \; magnetic \; profiles \; in \; certain \; point \; of \; time \; , between \; the \; sequences$$

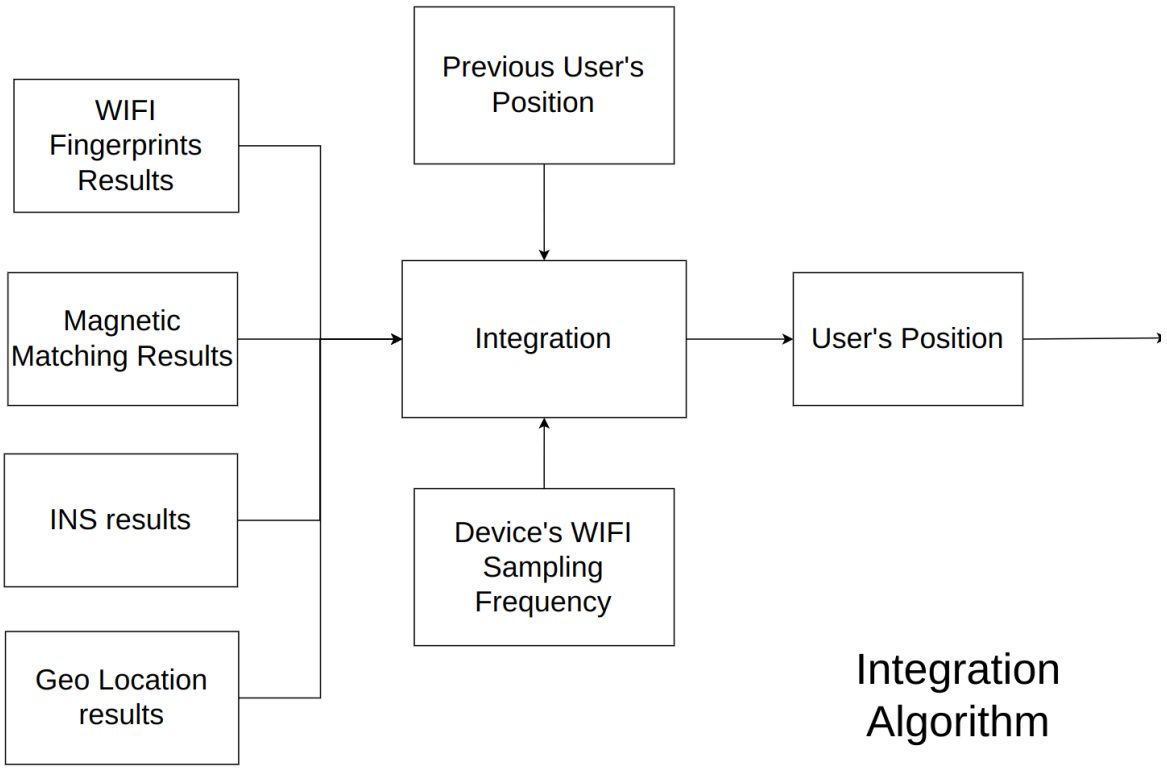
Option 1: Comparing sequences with multi-dimensional DTW algorithm with matching KNN, as shown in the WIFI fingerprints matching.

Option 2: KNN method.

Option 3: Adaptive KNN method.

Initially: Option 2 KNN method

13.14.Integration Algorithm:



[Diagram 3.11 - Integration Algorithm]

The integration algorithm involves the combination of Wi-Fi fingerprints, magnetic matching results, and INS (Inertial Navigation System) data using the prior user's location and the sampling frequency of their device's Wi-Fi signals to determine the user's current position. It's worth noting that there may be instances where the results from Wi-Fi and magnetic field data cannot be effectively integrated into the algorithm, depending on the specific implementation.

For our initial implementation, we drew inspiration from a research paper [4], which outlines three distinct algorithm structures. We have chosen to utilize the third structure as it has demonstrated the most promising outcomes. While the individual implementations of each component within the integration may vary, we have concluded that these variations should not fundamentally impact the overall architectural framework.

Integration:

INS results: older position, new position

IF WIFI meets requirments && WIFI results meets requirments:

- *WIFI results:* ***PositionSpace.***

IF MM meets requirments && MM results meets requirments:

- *MM results:* ***PositionSpace.***

$IntegrateResults(INS, WIFI, MM) \rightarrow$ ***Position:***

- $results =$ *INS results*
- *IF WIFI:*
  - $results = IntegrateResults(INS.WIFI)$
  - *IF results = one position:*
    - *return results*
- *IF MM:*
  - $results = IntegrateResults(results.MM)$
  - *IF results = one position:*
    - *return results*
- *return results*

Initial Integration:

$IntegrateResults(INS, WIFI) \rightarrow$ ***PositionSpace or Position:***

- $MDTW - WLS\ (weighted\ least\ squares) -$ *lagrange. presented in paper* [1]
- *With modification of* $p =$ *leasts squares p num of positions*

$IntegrateResults(INS\ WIFI, MM) \rightarrow$ ***Position***

- *MM search the INS WIFI positions. using* 1NN
- *Return the MM search position*

13.15.Navigation Algorithm:

Navigation algorithms using graphs are fundamental for computing optimal paths and routing solutions in various applications, from indoor navigation to complex transportation networks. In this context, a graph is a mathematical representation consisting of nodes (vertices) and edges (connections) that depict the layout of the environment. Navigation algorithms leverage these graphs to efficiently determine the shortest or most effective route between two points.

In addition to traditional graphs, flow graphs can also be used, where edges have capacities representing the maximum possible flow between nodes. While flow graphs are highly effective for optimizing network flows and managing capacities in complex systems, they are often considered overkill for indoor navigation due to the simpler nature of the environments. For indoor settings, standard graph-based algorithms like Dijkstra's or A* are typically sufficient to provide efficient and practical routing solutions.

Our System Navigation to Destination Algorithm:

Based on Dijkstra's shortest path

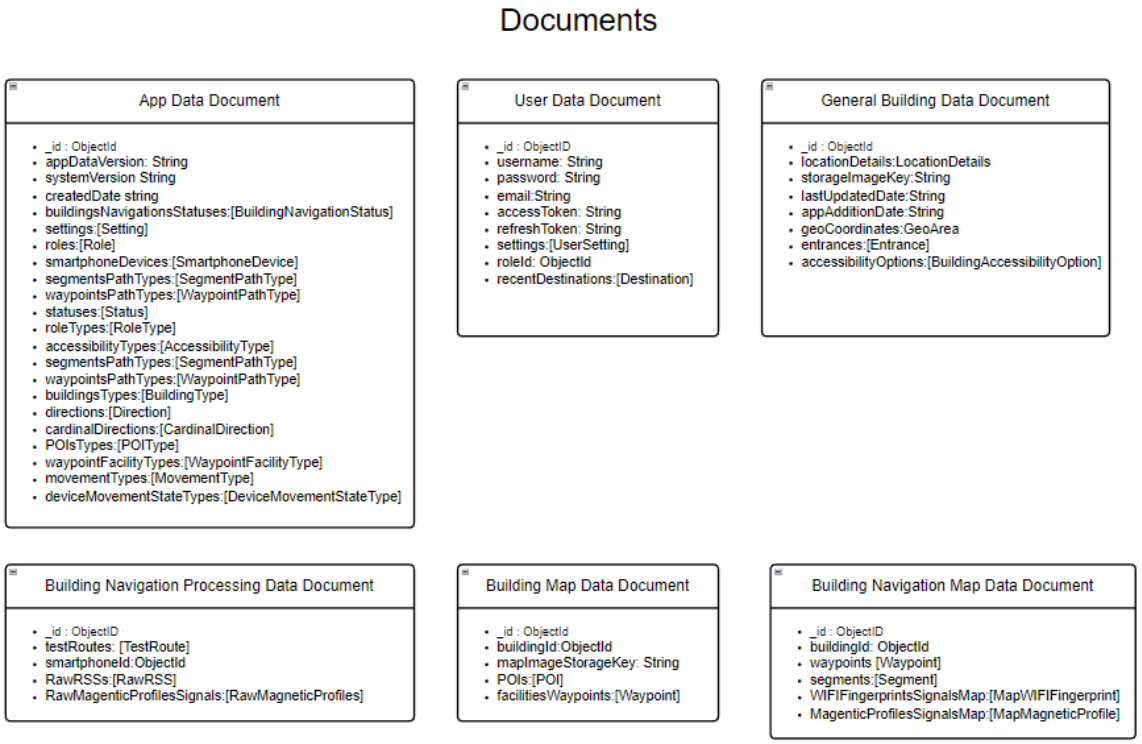
Input: source point, destination point, building map's undirected graph, route configurations, user accessibility settings, stops.

- Closest graph vertex to the source point is the initial graph location.
- Map's graph annotated as follow: vertex = v, edge = e , g = graph
- Based on the user's accessibility options, mark all user's unavailable accessibility edges as visited. (preparing for Dijkstra)
- For each stop vertex v create a new graph g (i+1) and connect the stop vertex v edges to v (i+1) instead of v, with directed arrow. Destination point is d (i+1). Now in order to get d (i+1) you must pass through those bridges, which have no option to go back.
- Run Dijkstra.

Output: route = list of nodes, edges

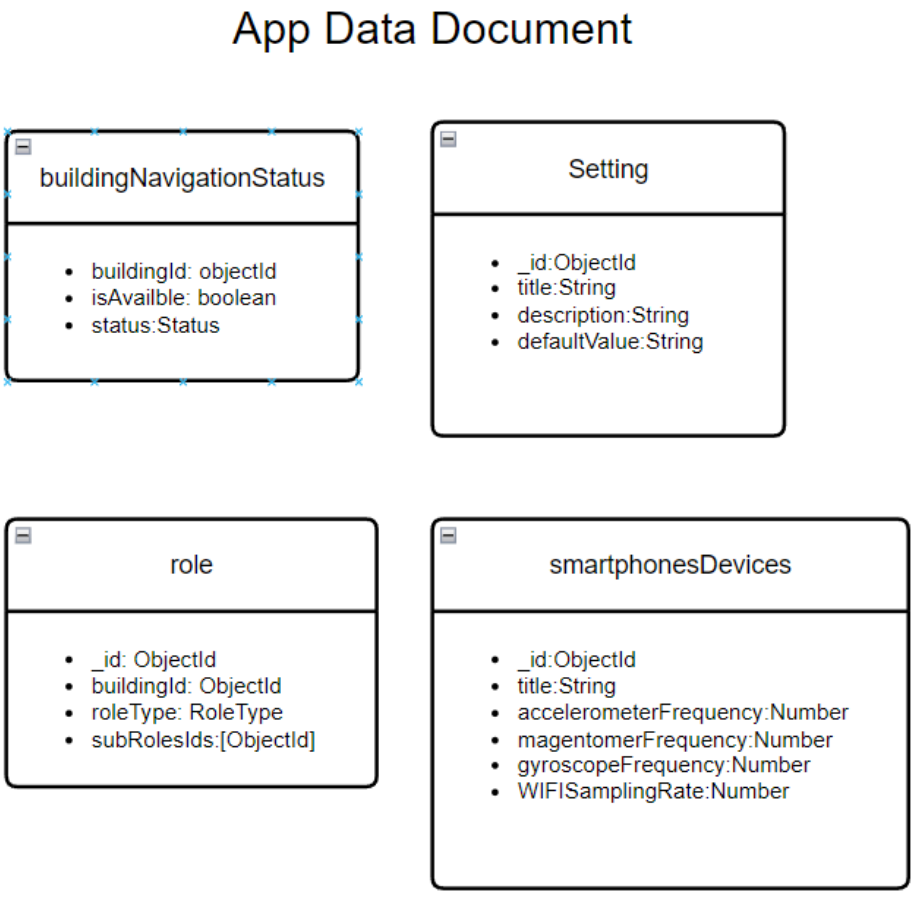
13.16.Database Design Schemas diagrams:

13.16.1.Mongo documents(documents-based noSQL):



[Schema 13.16.1.Documents]

13.16.2.App Data Schema:



[Schema 13.16.2.App Data Document]

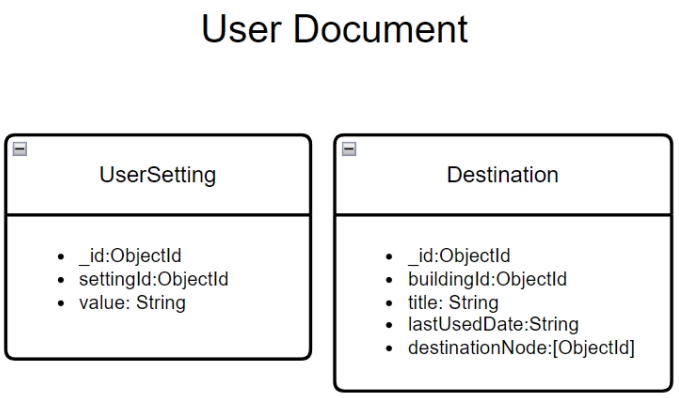
The App Data Collection Schema is designed to manage critical metadata related to our system's configuration and settings. It serves as a repository for essential information that guides the behavior and appearance of our application. This metadata collection plays a pivotal role in ensuring the accuracy and consistency of our system's operations.

Fields Descriptions

- **BuildingNavigationStatus:** The building object denotes a building availability, production status, development stage, failures, suspensions, pending actions, and requests awaiting administrative approval.
- **Setting:** application settings definitions
- **Role:** roles with permissions , a building has users with special permission available to configure building information, which includes operations updating information, changing availability of POIs or segments.
- **SmartphoneDevice:** devices and their configurations used to create our data infrastructure.
- **statuses:**informational annotation that categorizes buildings status based.
- **roleTypes:**informational annotation that categorizes roles based on their intended purposes.

- **AccessibilityType:** informational annotation that categorizes accessibility options.
- **SegmentPathType:** informational annotation that categorizes segment path type.
- **WaypointPathType:** informational annotation that categorizes way point path type.
- **BuildingTypes:** informational annotation that categorizes buildings based on their intended purposes.
- **Directions:** relative directions to the used map.
- **CardinalDirections:**true directions.
- **POIType:** informational annotation that categorizes points of interest based on their intended purposes.
- **WaypointFacilityType:**type of facility for a point, can be None.
- **MovementType:** user movement data annotation for testing route.
- **DeviceMovementStateType:** device movement data annotation for testing route.

13.16.3.User Schema:



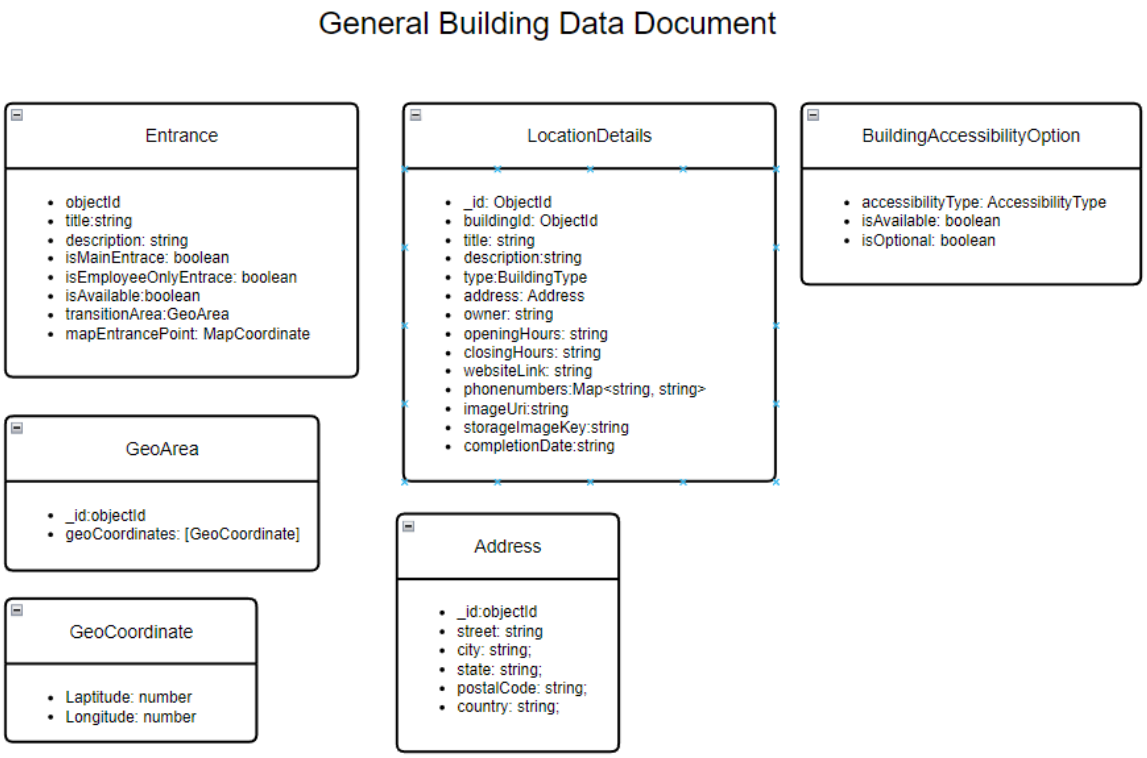
[Schema 13.16.3.User Document]

The Users Collection Schema defines the structure of user data within our indoor application navigation system.

Fields Descriptions

- **UserSetting:** user configuration for application setting.
- **recentDestinations:** latest recent destination user has used.

13.16.4.General Building Data Schema:



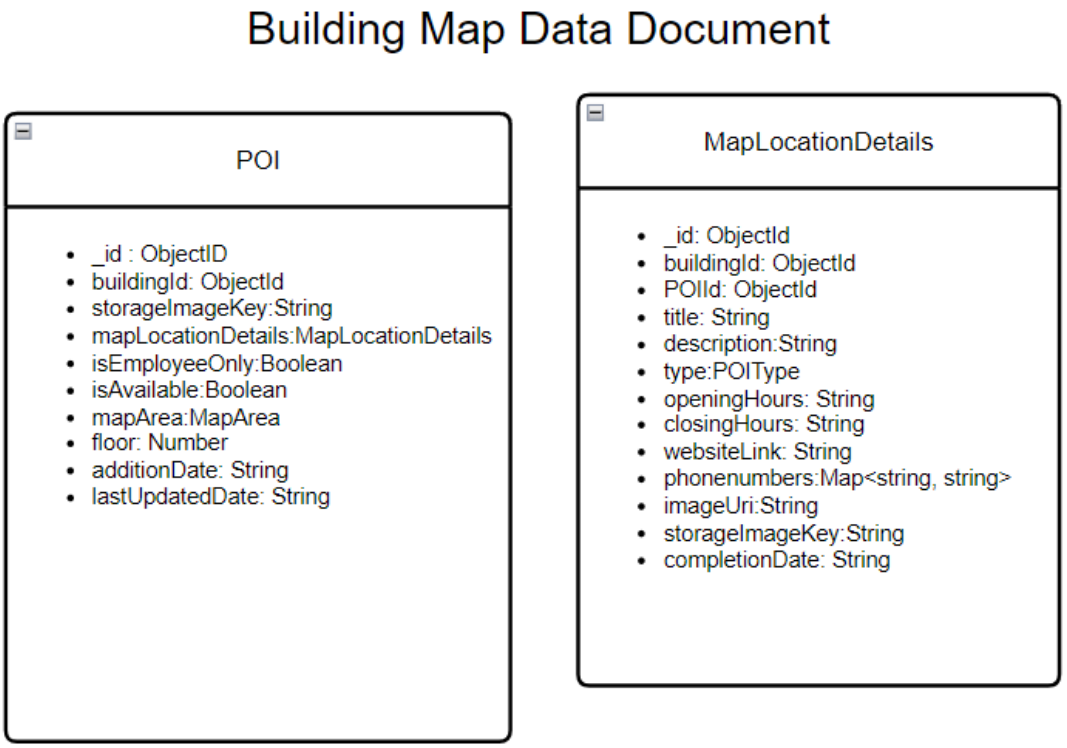
[Schema 13.16.4.General Building Data Document]

The Global Buildings Data Collection Schema is designed to store essential information about buildings, including their identification, characteristics, entrances and geographic coordinates. It plays a crucial role in our system for global localization, helping users to access information about the buildings worldwide.

Fields Descriptions

- **Entrance:** building entrance location ,transition area, data
- **GeoArea**: Global GPS coordinates Area
- **GeoCoordinate**: Global GPS coordinates point
- **BuildingAccessibilityOptions:** building available accessibility options.
- **Address:** address of the building.
- **LocationDetail:** building general information details.

13.16.5.Buildings Map Data Schema:



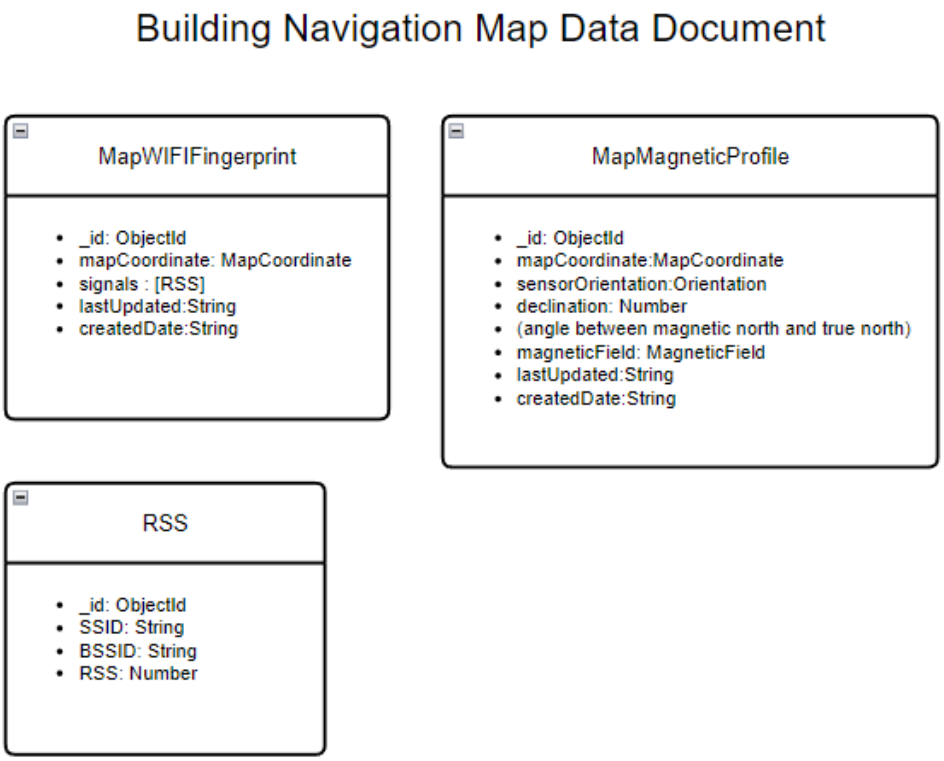
[Schema 13.16.5.Building Map Data Document]

This schema is designed to store detailed visual data about buildings, including maps, points of interest (POIs), and associated information. It enables our application to provide users with visual representations of buildings, along with information about various points of interest within those buildings.

Entity Descriptions

- **POI:** point of interest important data.
- **MapLocationDetails:** point of interest general information details.

13.16.6.Buildings Navigation Map Schema:



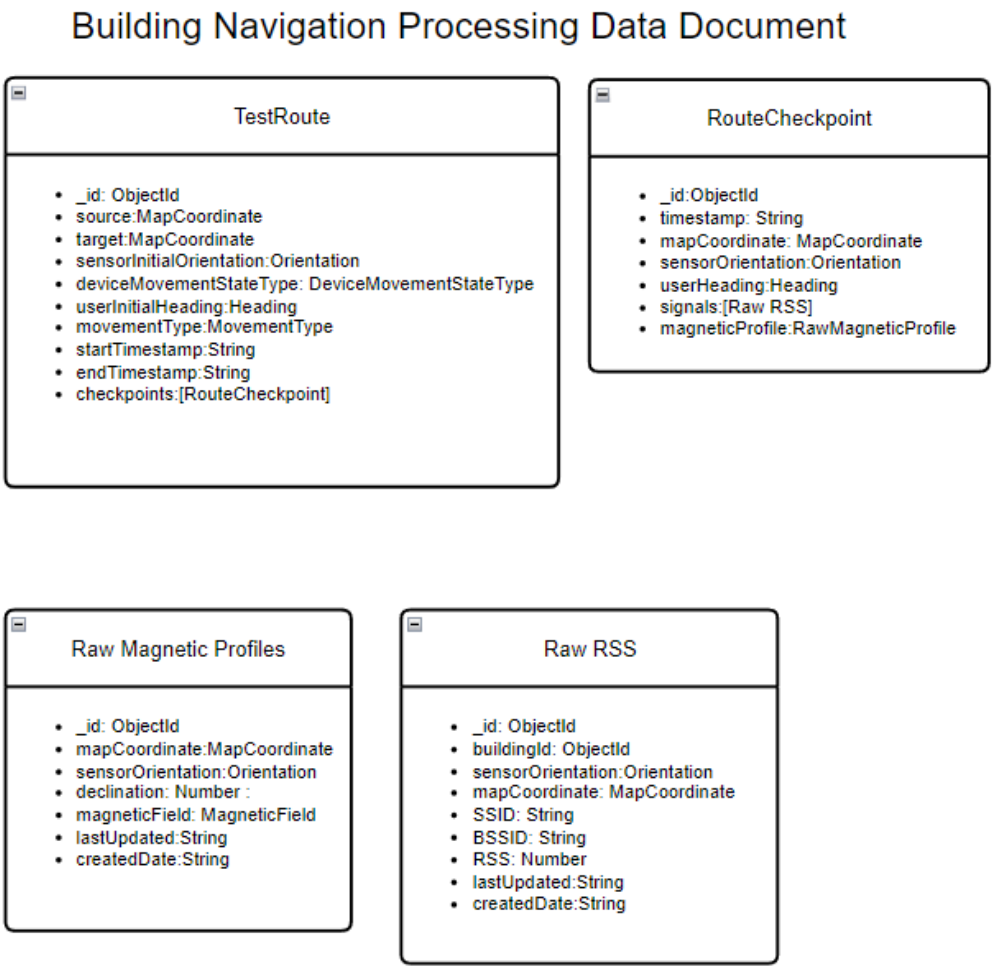
[Schema 13.16.6.Building Navigation Map Data Document]

The Buildings Maps Navigations Collection Schema is designed to store data related to building navigation, including segments and stops, to facilitate routing and navigation within buildings. It forms the foundation for our indoor navigation system, allowing users to find their way efficiently within complex structures.

Fields Descriptions

- **RSS:** processed rss ready for localization.
- **MapWIFIFingerprint:** list of RSSs used in localization.
- **MapMagneticProfile:** magnetic field data by map coordinate.

13.16.7.Building Navigation Processing Data Document:



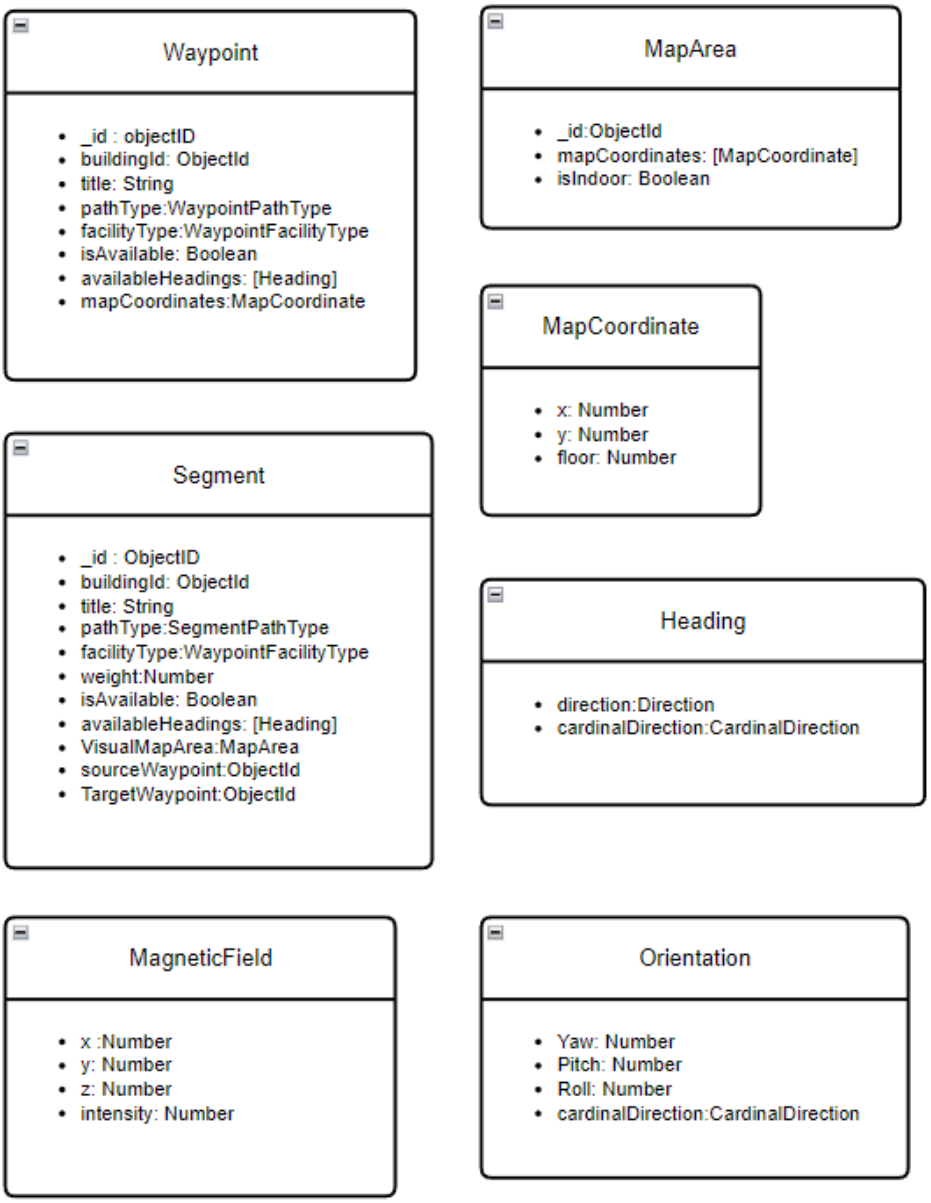
[Schema 13.17.7.Building Navigation Processing]

The processing data documents contain data about routing and the raw sensors data that will act as an anchor point for the various processed system configurations that will be generated through testing. It is important to note that the raw collection itself is not retained in the system; instead, it serves as the starting point for creating the processed collections, and might contain excess data.

Fields Descriptions

- **TestRoute:** route taken while collecting data, used for testing and adjustments for our engine.
- **RouteCheckpoint:** checkpoint collected at real time, part of the test route data.
- **RawRSS:** raw RSS values, collected during the test route.
- **RawMagneticProfiles**:raw RSS values, collected during the test route.

13.16.8.Building Navigation Common Nested Schemas:



[Schema 13.16.8- Building Navigation objects]

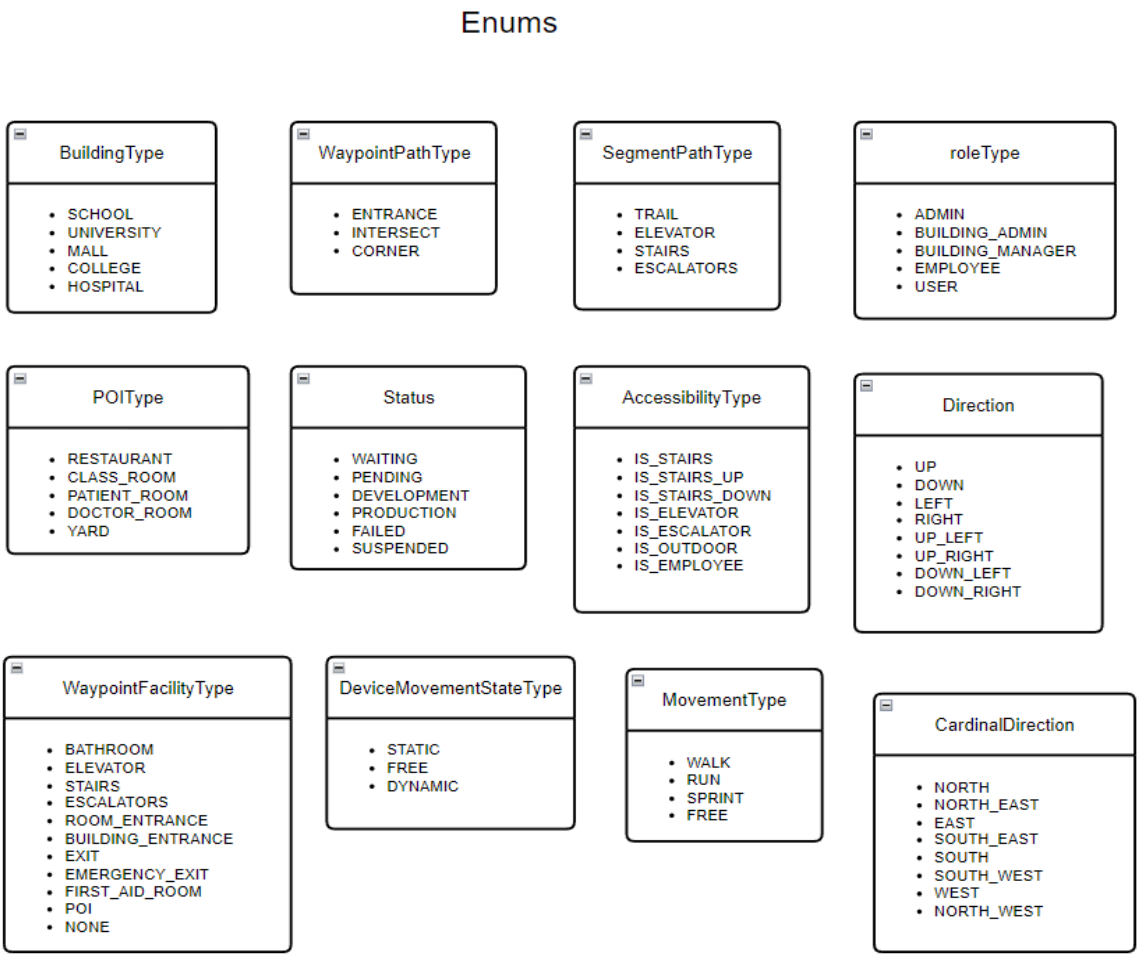
Shared entities between building navigation map, building navigation processing

Entities Descriptions:

- **Waypoint:(**nodes**)** define the entity object of waypoint.
- **MapArea:** Coordinates area ,relative to a building map.

- **Segment:(**edges**)** define the entity object of a segment which represents a connection area between waypoints.
- **MapCoordinate**:Coordinates area ,relative to a building map.
- **Heading**:represent a 2d heading direction of a user.
- **MagneticField**:define the entity object Magnetic fields that hold magnetic fields data.
- **Orientation**:define the entity object Orientation , which represents the heading direction in the 3d place.

13.17.Constants/Enums:



[Table 3.19 - Enums]

14.Risks:

Metric	Power **(Probability * Severity)**	**1 - 5: Very Low**
Description	The impact the risk has on our project	**5 - 10: Low**
		10 - 15: Medium
Power Levels		**15 - 20: High**
		20 - 25: Very High

Metric	Probability	**1: Very Low**
Description	The probability of the risk happening on our project	**2: Low**
		3: Medium
Probability Levels		**4: High**
		5: Very High

Metric	Severity	**1: Very Low**
Description	How detrimental the risk is to the system	**2: Low**
		3: Medium
Severity Levels		**4: High**
		5: Very High

14.1.Initial Risk Assessment:

Risk	Probability (1-5)	Severity (1-5)	Power (Prob * Severity) (1-25)	Impact	Mitigation Strategy
Lack of adherence to project timeline	2	2	4	Significant delays, compromised delivery in time	Implement effective project management strategies to monitor and track progress
Inadequate initial planning	3	1	3	Potential project setbacks and rework	Conduct thorough initial planning and identify potential risks and challenges
Lack of focus on project core objectives	1	1	1	Potential deviation from project goals and objectives	Establish clear project objectives and regularly review progress towards those objectives
Failure in implementing a functional navigation system with chosen technology	3	5	15	Significant risk potential of change the technology used in our project	Heavily rely on literature researches, and testing the infrastructure early at the project development
failure in accuracy in the navigation system with the chosen technology	3	5	15	Significant risk potential of change the technology used in our project	Heavily rely on literature researches, and testing the infrastructure early at the project development
Non-compliance with functional requirements	3	3	9	Impact on system functionality and user satisfaction	Establish comprehensive requirements gathering and validation processes
Incomplete GUI presentation	2	1	2	Reduced user experience and potential usability issues	Prioritize and allocate sufficient resources for GUI development and testing
Functional failures and task deferrals	2	1	2	Impaired system functionality and project delays	Implement proactive task management strategies to minimize functional failures

Risk	Probability (1-5)	Severity (1-5)	Power (Prob * Severity) (1-25)	Impact	Mitigation Strategy
Human factors (illness, overload, uncontrollable)	1	1	1	Potential disruption and resource constraints	Allow for sufficient contingency plans and resource allocation to address potential setbacks
Difficulty in acquiring required knowledge	2	1	2	Potential gaps in understanding and delays in project execution	Provide adequate time for learning and skill development to address knowledge gaps
Technological barriers	2	3	6	Minimal impact on project progress	Conduct thorough research and testing to identify and address potential technological barriers

[Table 14.1.Initial Risk Assessments]

14.2.After Engineering Report Risk Assessment:

Risk	Probability (1-5)	Severity (1-5)	**Now** Power (Prob * Severity) (1-25)	**Before** Power (Prob * Severity) (1-25)	Change	Impact	Mitigation Strategy
Lack of adherence to project timeline	2	2	4	4	Same	Significant delays, compromised delivery in time	Implement effective project management strategies to monitor and track progress
Inadequate initial planning	3	1	3	3	Same	Potential project setbacks and rework	Conduct thorough initial planning and identify potential risks and challenges
Lack of focus on project core objectives	1	1	1	1	Same	Potential deviation from project goals and objectives	Establish clear project objectives and regularly review progress towards those objectives
Failure in implementing a functional navigation system with chosen technology	3	5	15	15	Same	Significant risk potential of change the technology used in our project	Heavily rely on literature researches, and testing the infrastructure early at the project development
failure in accuracy in the	3	5	15	15	Same	Significant risk potential of change the	Heavily rely on literature researches, and

Risk	Probability (1-5)	Severity (1-5)	**Now** Power (Prob * Severity) (1-25)	**Before** Power (Prob * Severity) (1-25)	Change	Impact	Mitigation Strategy
navigation system with the chosen technology						technology used in our project	testing the infrastructure early at the project development
Non-compliance with functional requirements	3	4	10	9	Higher	Impact on system functionality and user satisfaction	Establish comprehensive requirements gathering and validation processes
Incomplete GUI presentation	2	1	2	2	Same	Reduced user experience and potential usability issues	Prioritize and allocate sufficient resources for GUI development and testing
Functional failures and task deferrals	2	1	2	2	Same	Impaired system functionality and project delays	Implement proactive task management strategies to minimize functional failures
Human factors (illness, overload, uncontrollable)	2	1	2	1	Higher	Potential disruption and resource constraints	Allow for sufficient contingency plans and resource allocation to address potential setbacks

Risk	Probability (1-5)	Severity (1-5)	**Now** Power (Prob * Severity) (1-25)	**Before** Power (Prob * Severity) (1-25)	Change	Impact	Mitigation Strategy
Difficulty in acquiring required knowledge	2	1	2	2	Same	Potential gaps in understanding and delays in project execution	Provide adequate time for learning and skill development to address knowledge gaps
Technological barriers	2	3	6	6	Same	Minimal impact on project progress	Conduct thorough research and testing to identify and address potential technological barriers
Fail to create system tools.	1	5	5	-	New	Uncertainty system functioning (testing)	Provide adequate time for developing the tools.
fail to create a building map.	1	5	5	-	New	Unable to showcase the system in a building	Provide adequate time for creating a building map.
Failure to get building owner’s permissions for measurements	1	2	2	–	New	Unable to showcase system in desired demo location	Use different locations for the demo purposes.

Risk	Probability (1-5)	Severity (1-5)	**Now** Power (Prob * Severity) (1-25)	**Before** Power (Prob * Severity) (1-25)	Change	Impact	Mitigation Strategy
Inaccurate sensor readings.	2	2	4	–	New	Increase development time by some margin	Comparison between different devices
Fail to calibrate sensors readings	2	5	10	-	New	Unable to provide a stable accuracy across different devices	Use known calibrations methods, while testing according to the research papers.

[Table 14.2.Secondary Risk Assessments]

Limitations:

1. **Reliance on WiFi signals:** The accuracy of indoor navigation may be affected by the availability and strength of WiFi signals in certain areas of the building.
2. **Reliance on Magnetic signals:** The accuracy of indoor navigation can be impacted by the availability and obscuration of magnetic fields in certain areas of the building.
3. **Sensor compatibility:** The app's performance depends on the compatibility and functionality of smartphone sensors, which can vary across different devices. Issues may arise due to malfunctioning sensors, missing sensors, outdated sensors or uncalibrated sensors..
4. **Complex indoor environments:** Navigating in highly complex indoor environments with multiple levels, intricate layouts, or signal interference can pose challenges for accurate positioning.
5. **Data accuracy and maintenance:** The accuracy of floor maps and points of interest relies on the availability and maintenance of up-to-date data, which may require continuous updates and verification.

15.The Testing plan

This document details the testing strategy and procedures for the prototype of our indoor navigation system. The testing activities related to the localization engine will be conducted at Afeka College.

15.1.Test Objectives

- Verify the functionality of the user interface , interface integrations and user experience.
- Validate the accuracy of sensor data integration.
- Ensure proper communication between the frontend and backend applications.
- Confirm the correctness of server-side functionalities, including search, map rendering, navigation and localization algorithms.
- Validate the data storing and retrieval processes in the database.
- Verify the expected flow of the system.
- Test user positioning accuracy and responsiveness.

15.2.Test Scope

- Front-end application UI/UX.
- Sensors Layer data integration and processing.
- Communication between frontend and backend applications.
- User Authentication and User data management.
- Indoor navigation accuracy
- Detailed Examination of Building Layout, including Map and Points of Interest (POI).
- Database operations.

15.3.Test Environments

- Frontend: Android and iOS smartphones/simulators.
- Backend: Postman for API testing and Docker for containerized deployment.
- Database: MongoDB Atlas for testing connection and storage with the backend application.

15.4.Test Approach

15.4.1.Unit Testing

Unit testing will focus on individual components of the system, including UI components, sensor data integration modules, backend functionalities, and database operations. Each unit will be tested in isolation to ensure that it performs as intended.

15.4.2.Integration Testing

Integration testing will validate the interactions between different units and components. This includes testing the communication between the frontend and backend applications, ensuring proper data flow, and confirming the integration of sensor data. Integration tests will be conducted using both real devices and simulators.

15.4.3.System Testing

System testing will evaluate the overall functionality and performance of the indoor navigation system. This includes end-to-end testing of the user interface, complete sensor data integration, communication between frontend and backend, and the correctness of server-side functionalities. System testing will simulate real-world scenarios to ensure the system's robustness.

15.4.4.Localization And Navigation Testing , Metrics & Evaluations

The final evaluation of our system will be presented after the deployment stage, the metrics are described from paper [6]:

- **Scalability (2/5):** a lot of preparations such as creating the building map, and collection of the data points reduce the scalability of our system. (**system requirement**).
    - Excellent Accuracy: RMSE less than 1 meter.
    - Good Accuracy: RMSE between 1 meter and 5 meters.
    - Fair Accuracy: RMSE between 5 meters and 10 meters.
    - Poor Accuracy: RMSE greater than 10 meters.
- **Complexity:** Assessing complexity can be challenging, as it encompasses various elements including implemented algorithms, integrated components, individual time investments, and total man-hours. To streamline evaluation, we aim to simplify by focusing on the total man-hours invested.
    - Above 300 Hours - Very High
    - Around 300 hours - High
    - Around 200 Hours - Medium
    - Around 100 Hours - Easy
- **Robustness:** is measured by the duration of continuous navigation without encountering any unexpected scenarios. The goal is to achieve a minimum of 10 minutes of uninterrupted navigation.
    - Goal: 10 minutes
    - Below Expectations: Less than 10 minutes, but more than 1 minute
    - Bad: 1 minute or less.
- **Reliability:** is measured by how often unexpected scenarios occur.
    - Goal: 1 in 20 usages.
    - Great: 1in 10 usages.
    - Good: 1in 5 usages.
    - Bad: 1 in 2 usages

15.4.5.Accuracy Testing:

A <position, RSS> dataset and a <position, magnetic feature> database are generated simultaneously in this phase. A walk-survey method is adopted. A surveyor is asked to hold the smartphone horizontally and walk along pre-designed survey trajectories that have landmarks (e.g., corners, intersections, and other preset landmarks), and record the time of passing every landmark. The coordinates of landmarks can be obtained from a digital indoor map, while the true orientation of links between adjacent landmarks can be calculated by using these coordinates.

Accuracy**:** measured by the accuracy of the user position in the map: (Actual vs Expected Position using RMSE with meters as unit), with visual showcase.

15.5.Test Cases

Test Case: Application Launch

- Input: Open the indoor navigation application
- Expected Output: Application launches successfully without errors and displays the main screen or landing page.

Test Case: User Authentication

- Input: Enter valid username and password
- Expected Output: User is authenticated and granted access to the application's features.

Test Case: Invalid Authentication

- Input: Enter invalid username or password
- Expected Output: User authentication fails, and an appropriate error message is displayed.

Test Case: Map Rendering

- Input: Select a building or venue for navigation
- Expected Output: Indoor map of the selected building or venue is rendered accurately on the screen, displaying relevant landmarks and points of interest.

Test Case: Search Functionality

- Input: Enter a specific location or point of interest in the search bar
- Expected Output: Relevant search results are displayed, allowing the user to select the desired location for navigation.

Test Case: Route Calculation

- Input: Select a starting point and destination on the indoor map
- Expected Output: The application calculates the shortest route between the selected points and displays turn-by-turn navigation instructions.

Test Case: Turn-by-Turn Navigation

- Input: Start navigation to a destination
- Expected Output: Turn-by-turn navigation instructions are provided at each step, guiding the user along the selected route.

Test Case: Offline Navigation

- Input: Disable internet connection

- Expected Output: The application continues to provide navigation functionality using offline maps and stored data, with limited or no reliance on internet connectivity.

Test Case: Voice Guidance

- Input: Enable voice guidance for navigation
- Expected Output: Clear and accurate voice prompts are provided to guide the user through each step of the navigation process.

Test Case: Arrival Notification

- Input: Reach the destination
- Expected Output: The application notifies the user when they have reached the destination, providing confirmation of successful navigation.

Test Case: User Interface (UI)

- Input: Interact with various UI elements (buttons, menus, etc.)
- Expected Output: UI elements respond appropriately to user input, providing intuitive and smooth navigation through the application.

Test Case: Compatibility

- Input: Install and run the application on different devices (Android and iOS)
- Expected Output: The application functions correctly and consistently across various devices and platforms, maintaining its usability and performance.

16.Literature review

Advances in Indoor Positioning and Indoor Navigation

The literature abstract discusses the challenges and advancements in indoor navigation systems. It emphasizes the absence of an overall solution and the potential impact of such

systems in various fields. The abstract acknowledges the challenge of comparing and evaluating different indoor positioning systems. The mentioned papers in the special issue address this issue, providing methodologies for effective system evaluation and comparison. These methodologies can assist you in assessing and selecting the most suitable components or techniques for your project's navigation system. Overall, the summarized literature provides insights into various aspects of indoor navigation systems. These insights can help guide our project's design decisions and contribute to the development of a cost-effective and efficient indoor navigation system.

Real-Time Map Matching with a Backtracking Particle Filter Using Geospatial Analysis

From article 1 in “1.2. Magnetic Field and Inertial Systems” , refer us to an interesting article about implementing algorithms using the inertial IMU in smartphones , in an indoor navigation.

This literature review focuses on an article that explores the use of inertial odometry, based on the inertial measurement unit (IMU) values embedded in smartphones, as a localization method for indoor navigation. The article highlights the advantages of inertial odometry, its application in pedestrian positioning, and the need for map-matching to correct cumulative errors. The paper presents a real-time map-matching approach using backtracking particle filters combined with geospatial analysis . This approach reduces the complexity of spatial queries and provides flexibility in incorporating different types of spatial constraints. The primary goal is to generalize the algorithm to accommodate various odometry data calculated by different sensors

This article presents valuable insights into utilizing inertial odometry and mobile sensors for indoor navigation, which aligns with the objectives of our own project. The findings emphasize the wide accessibility of inertial odometry in many devices, including smartphones, making it a viable approach for pedestrian positioning.The description of the map-based optimization technique, specifically utilizing transition areas between floors, provides valuable insights for improving accuracy in complex building structures. This knowledge can inform the implementation of my own indoor navigation system.

Overall, this article serves as an important reference for our indoor navigation project. It highlights the advantages of inertial odometry, presents a map-matching approach, and demonstrates its effectiveness in achieving accurate positioning in complex environments. The findings and developments discussed in the article will guide the implementation and optimization of our own indoor navigation system, contributing to its success

Analysis of Magnetic Field Measurements for Indoor Positioning

From article 1 in “1.2. Magnetic Field and Inertial Systems”, refer us to an interesting article about navigation for analysis of magnetic fields.

The article discusses the feasibility and limitations of relying solely on magnetic field measurements for indoor positioning. It highlights the potential of using mobile sensors and magnetic fields for indoor navigation, emphasizing the importance of calibration and appropriate data processing techniques.

This article provides valuable insights into utilizing mobile sensors and magnetic fields for indoor navigation, aligning with the objectives of our own project. The findings of the article support the feasibility of using magnetic field measurements from mobile sensors for positioning in indoor environments. The analysis of statistical characteristics and the proposed methods for anomaly elimination and calibration offer practical approaches to improve the accuracy and reliability of magnetic field-based indoor navigation.

The comparison of homogeneous and heterogeneous devices using various machine learning methods helps in understanding the performance variations and identifying the suitable techniques for our project. It informs us about potential challenges, such as interference from ferromagnetic materials, which we should consider in the implementation of our indoor navigation system.In summary, this article serves as a valuable reference for our indoor navigation project, providing insights for the magnetic field measurements, techniques , and the performance evaluation of magnetic field-based positioning. It offers valuable conclusions regarding the feasibility and limitations of using mobile sensors and magnetic fields for indoor navigation, guiding the direction of our project implementation.

Smartphone-Based Inertial Odometry for Blind Walkers

From article 1 in : “1.5. Location Systems Evaluation and Comparison”, refer us to interesting article about navigation for blind walkers

This literature review focuses on an article that explores the use of pedestrian tracking systems implemented in regular smartphones for assisting blind individuals in wayfinding and backtracking. . The contribution presents a comparative assessment of various algorithms that utilize inertial sensors for pedestrian tracking. The article considers two situations of interest. In the first situation, where a map of the building is unavailable, the study assumes that users navigate through a network of passages intersecting at 45° or 90° angles. A new two-stage turn detector, combined with an LSTM-based step counter, is proposed to accurately reconstruct the path traversed by blind individuals. This approach is compared with RoNIN, a state-of-the-art deep learning-based algorithm. In the second situation, where a map is available, providing strong prior information on possible trajectories, the study experiments with particle filtering. An additional clustering stage based on mean shift is introduced to refine the results. The performance of these algorithms is evaluated and compared.

This article provides valuable insights into the development of inertial sensor-based pedestrian tracking systems in indoor navigation, which is directly relevant to our own project. The proposed two-stage turn detector and LSTM-based step counter offer a promising approach for accurately reconstructing the path traversed by blind individuals in situations where a map is not available. Comparisons with state-of-the-art algorithms demonstrate the effectiveness of this approach. In situations where a map is available, the study explores the use of particle filtering combined with mean shift clustering to improve

trajectory estimation. Overall, this article serves as a valuable reference for our indoor navigation project, providing insights into algorithmic approaches and considerations for pedestrian tracking systems. The findings and methodologies presented in the article will support the development and optimization of our own navigation system.

MeshMap: A Magnetic Field-Based Indoor Navigation System With Crowdsourcing Support

This literature review focuses on a paper that introduces MeshMap, an indoor navigation system based on magnetic fields and crowdsourcing. MeshMap leverages users' smartphones for easy and cost-effective indoor navigation, eliminating the need for pre-deployed infrastructure such as Bluetooth, or WiFi. By utilizing magnetic fields and crowdsourcing, MeshMap offers applicability, stability, and resilience to occlusions caused by human bodies and other barriers.

The proposed approach in this paper incorporates two key features. Firstly, crowdsourcing is to create a magnetic fingerprints database by merging sensor data from multiple users' different paths.

Secondly, a dynamic time warping-based matching algorithm to perform magnetic field time-series matching and position correction, considering various factors such as users' walking behaviors, magnetic abnormal positions, corners, and more.

The implementation and testing of MeshMap have demonstrated its capability to achieve real-time positioning and navigation. The testing results reveal that MeshMap maintains high accuracy. The introduction of MeshMap, an indoor navigation system based on magnetic fields and crowdsourcing, presented in this paper interests us greatly for the implementation of our indoor navigation project.

Overall, this paper on MeshMap provides valuable inspiration and guidance for our own indoor navigation project. The utilization of magnetic fields, crowdsourcing, and dynamic time warping-based matching algorithms can significantly contribute to the success of our project to develop a cost-effective and infrastructure-free solution , making it highly accessible for users with smartphones.

Sensors and Sensing Technologies for Indoor Positioning and Indoor Navigation

This literature review presents the significant advancements made in the field of indoor positioning and navigation since 2010.

This special issue focuses on the recent developments in sensors and sensing technologies specifically tailored for indoor positioning and navigation networks. The included papers provide valuable insights into the implementation, modeling, and integration of novel technologies and applications in this domain.

The papers featured in this special issue contribute to the ongoing efforts in advancing indoor positioning and navigation. They showcase the progress made in leveraging sensor technologies and innovative approaches to enhance the accuracy, reliability, and usability of indoor localization systems. The issue also empathizes the growing importance and potential applications of indoor localization across various domains. The inclusion of location-based services, indoor maps, and 3D building models in the papers demonstrates the need for comprehensive spatial information to support accurate indoor navigation.

Incorporating such features into our project can enhance the user experience and enable seamless navigation within complex indoor environments. The focus on self-contained sensors, wearable devices, and multi-sensor systems highlights the importance of leveraging multiple and diverse data sources for indoor positioning. Integrating these technologies into my project can enhance accuracy and reliability by combining data from different sensors and modules.

This paper provides various research papers related to the indoor navigation field , which provides us great insights that offer valuable guidance for future research in development in our project. Incorporating the findings and advancements discussed can contribute to the success and effectiveness of our own indoor navigation project, ensuring accurate, reliable, and user-friendly navigation experiences in indoor environments.

Hybrid Wireless Fingerprint Indoor Localization Method Based on a Convolutional Neural Network

This research paper focuses on improving indoor navigation accuracy using an approach called Hybrid Wireless fingerprint (HW-fingerprint) combined with a convolutional neural network (CNN).

The HW-fingerprint enhances the expression of indoor environment characteristics by incorporating the ratio relationship between different received-signal-strength-indicator (RSSI) values from important access points (APs) along with the RSSI itself. By combining the ratio of RSSIs and the RSSI values themselves, the HW-fingerprint captures more detailed information about the indoor environment.

The article elaborate the HW-fingerprint approach benefits, such as taking into account weather changes and moisture levels , the article continue to elaborate on the purpose of CNN architecture which, effectively learns important features from the complex HW-fingerprint for indoor locations , This allows for more accurate and robust indoor location classification, reducing the need for manual feature extraction.

The proposed methods are to construct the HW-fingerprint both in an offline and online phase. The deep-learning method is to learn the features of the HW-fingerprint and predict the indoor location. In the offline data-processing module,using (MAC) address sequence of the relevant AP which is used to construct the ratio fingerprint, and also to match to the corresponding online HW-fingerprint.

The proposed method does not require prior knowledge of the indoor environment layout or the deployment of specific hardware. This makes it easier and more cost-effective to implement in our indoor navigation application.

By incorporating the HW-fingerprint and CNN-based localization, our indoor navigation project can provide users with accurate and reliable positioning information. This improves the overall user experience, making it easier for people to navigate indoor spaces, find specific locations, and optimize their movement within buildings. This article offers us valuable insights and considerations in designing and implementing our own indoor navigation project.

Map Assisted PDR/Wi-Fi fusion for indoor positioning using smartphone

This literature review focuses on a map-assisted pedestrian navigation system designed for smartphone users. The system combines map information, IMU-based Pedestrian Dead Reckoning (PDR) which is MEMS based, and Wi-Fi localization using fingerprinting methods. This literature explains the uses of PDR ,utilizing smartphones, as well as the unavoidable accuracy errors using this technique on its own. PDR utilizes smartphone sensors to estimate step detection, step length, and heading during pedestrian movement. However, inherent errors in these algorithms, such as step length uncertainties, magnetic disturbances in indoor environments, and unstable smartphone positions, can lead to inaccuracies. Previous approaches have proposed Wi-Fi fusion or map matching techniques to enhance PDR accuracy. However, these methods faced challenges in fault matching and creating accurate maps for hall areas where pedestrians may traverse diverse trajectories. To address the issues, this paper introduces a Virtual Link (VL) algorithm with a Virtual Track (VT) to resolve the map structure problem in hall areas and improve accurate link selection. Additionally, an Extended Kalman Filter (EKF) is employed to estimate pedestrian position and IMU sensor errors. The EKF leverages map information to estimate errors associated with step length estimation, heading estimation, and IMU sensor readings during pedestrian dead reckoning.

The article presents a detailed explanation of designing such a system, providing us insights and techniques to use in our own project.Incorporating the insights and techniques into our indoor navigation project can significantly enhance the accuracy and reliability of the smartphone-based navigation system. The findings serve as valuable guidance for developing and designing an efficient and robust navigation solution for pedestrians in indoor environments.

Related work (additional literatures):

Many implementations are existing in indoor navigation areas, In order to emphasize that indoor navigation is a versatile field with numerous available methods.

A brief explanation of the additional referenaces papers is presented. Hopefully will help understand how diverse and different the options are:

Reference [1], present a hybrid approach using INS/WIFI indoor localization, available with all positioning of the smartphone such as calling, handheld and in pocket. The presented

paper shows improvement of localization with the hybrid model rather than each as a standalone. The study uses an algorithm for the fingerprints matching called MDTW-based WSL –this algorithm describes a multi-vector version of the dynamic time warping algorithm , the weight least square is used to weight the distances output from the MDTW to aid the PDR update. The results are correlated to the number of access points available. The results also show that certain motion states of the smartphone will have more position errors then others.

Reference [2], presents an INS as a standalone system, the purpose of this study is to enhance and extend the short-term reliability of PDR systems for smartphones as a standalone system through an enhanced step detection algorithm, a periodic attitude correction technique, and a novel PCA-based motion direction estimation technique. This paper study had a big influence on our INS system, and will be expended at the INS system stage.

Reference [3] presents a standalone INS using accelerometer ,gyroscope and the virtual gravity sensor from android, the method require smartphone with static relative position to the body, the study specifically focuses on hand-held position of smartphone , for usage of matters which the user responsibility is high which will keeps his static position as is. The study was an interesting simpler implementation than the rest of cited papers.

Reference [4] presented three ways of structuring the architecture of the localization system using INS, WIFI fingerprinting and magnetic matching. Three-level quality control mechanism is shown.

Reference [5], the paper, is very unique and has very easy implementation. It's based on the assumption of knowing the route the user is going to take, which will require the user to choose his route, meaning assisting in the navigation process, but not in the live localization feature. This algorithm presented is using only the device sensors and a building map, its uses the corners and predefined steps directions corresponding to the direction available, to align user movement to the defined direction, to reduce the positional error, while still incorporating iterative algorithm (First Fit algorithm and Best Fit algorithm) to handle unpredicted behavior. This paper caused us to add a direction map for each building's map, in such a way that each block in the map will have his own set of possible directions. This assumption can be utilized in our navigation system, in correcting the positioning of the user. This paper some INS techniques that will be elaborated later on. Also the corners map direction alignment, which can be useful in itself, during routing navigation, after estimating the user's location, to reduce the error.

Reference [6] is of utmost significance as it represents the current pinnacle of knowledge in the field. This comprehensive study provides an in-depth and profound understanding, offering a broader perspective that guides our search for relevant data in this domain. The research presented in this paper not only validates our concept of transitioning areas indoors to reduce complexity but also equips us with a wealth of metrics essential for testing our system effectively.

Reference [7] plays a pivotal role in our quest to integrate Inertial Navigation System (INS) and Magnetic Matching (MM) for smartphone positioning in indoor environments. The

insights provided by this paper, focusing on ambient magnetic fields, enrich our understanding and contribute significantly to the seamless integration of INS and MM. Although the paper doesn't specifically address Wi-Fi integration, its emphasis on enhancing accuracy through magnetic field analysis aligns with our broader goals in refining our indoor navigation system.

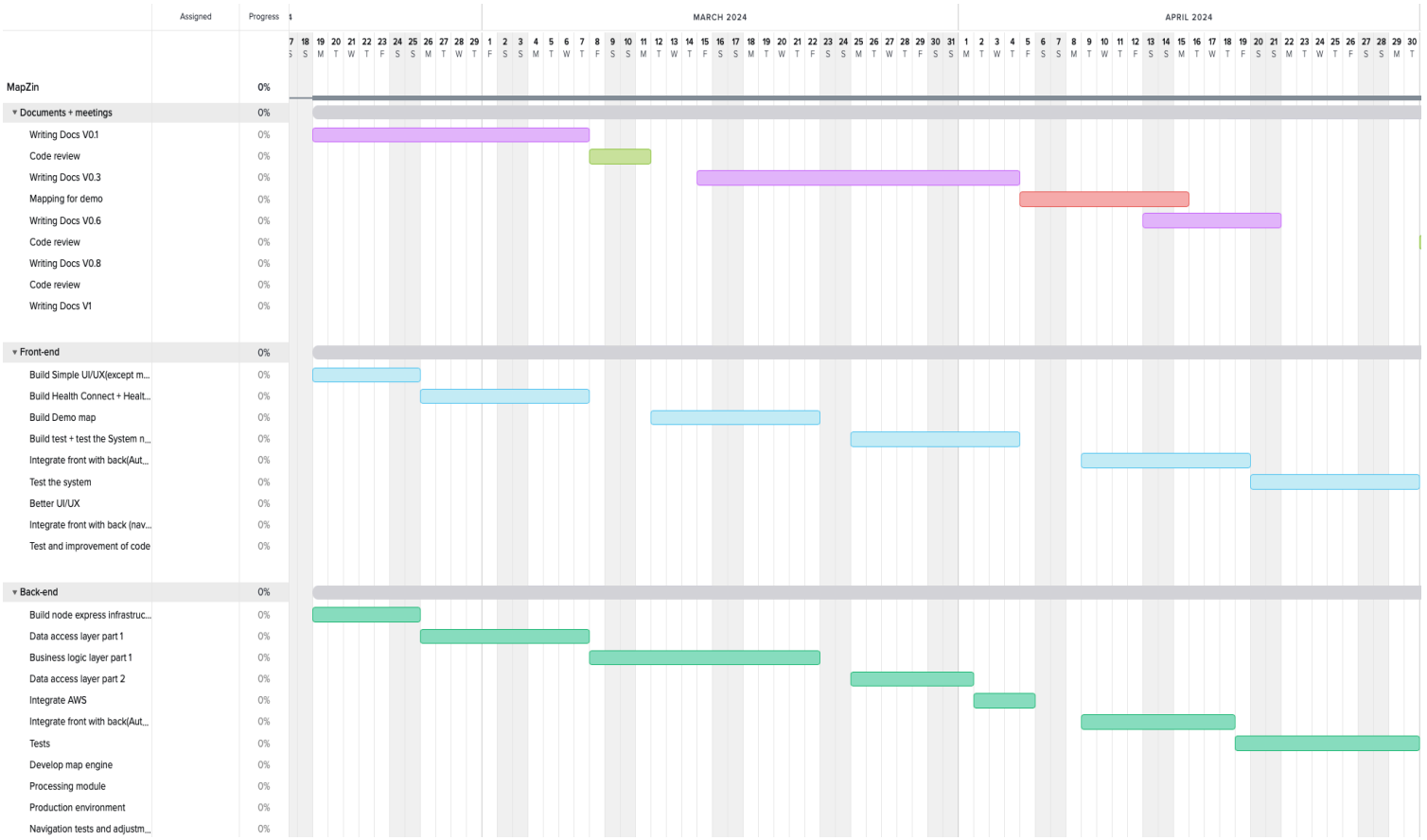
Reference [8] a survey on indoor navigation techniques, was instrumental in our project. Utilizing both Wi-Fi and other methods, the survey guided our decision-making process. The comparative analysis of route algorithms across studies proved invaluable, helping us choose the most fitting algorithm for our system's navigation.

Reference [9] introduces a novel iterative calibration algorithm for triaxial magnetometers, emphasizing enhanced accuracy and reliability with just nine distinct measurements. The proposed method outperforms conventional models, demonstrating faster convergence suitable for real-time applications. The findings contribute valuable insights to our calibration techniques, particularly in achieving improved precision for triaxial magnetometers.

17.Progress Tracking: Project management

Initial sprints plan (Gantt):

This Gantt chart was initially intended for two people, as my teammate and I had split the tasks. However, since we are no longer working together, the final output may not fully reflect our initial expectations. The chart is also quite naive and overly general, lacking the explicit details necessary for such a complex indoor navigation project. For example, tasks like “Backend build skeleton” , are too vague, as they could refer to WebSockets, navigation modules, and more. The UI work was substantial, involving map display, map tools, and gesture handling, which were particularly challenging. Additionally, the data mining screens and data collection further complicated the project. This chart demonstrates our collaborative effort and time management up to that point but does not fully capture the intricacies and challenges we encountered.



[Image 17.Gantt]

- Sprint 1: 19.2 -26.2:
  - Doc- Split work on document for all the project
  - Front - simple UX (without map view)
  - Back - build skeleton
- Sprint 2: 26.2 -7.3:
  - Doc- finish doc skeleton and team know what need to be done

    - Front - Get data from sensors - Android + IOS
    - Back - Start Data access layer
- Sprint 3: 8.3- 11.3:
    - 1.Code review + and progress update
- Sprint 4: 12.3 - 22.3:
    - Doc- add all need data
    - Front - Building main screen to present
    - Back - working on the BLL
- Sprint 5: 23.3 - 4.4:
    - 1.Doc- add all need data
    - 2.Front- Building nav view on the map
    - 3.Back- finishing Data access layer
- Sprint 6:5.4 - 16.4:
    - 1.Mapping area in Afeka and insert all data to system
- Sprint 7: 17.4- 30.4:
    - 1.Doc: 80% need to be writen
    - 2.Front- building unit test + start to integrate with back
    - 3.Back-end: building unit test + start to integrate front
- Sprint 8: 1.5-9.5
    - 1.code review
- Sprint 9: 10.5-27.5:
    - 1.Doc- check each other work
    - 2.Front- UI/UX
    - 3.Back-end: AWS integration
- Sprint 10: 28.5 - 10.6:
    - Doc- v0.8 should be written include all data written
    - Front- Develop nav view
    - Back-Develop map engine
- Sprint 11: 12.6- 24.6:
    - Doc- all should be written (non- final version)
    - Front- Step by Step nav
    - Back-Processing module
- Sprint 12: 24.6 - 3.7:
    - Doc- Done
    - Front- test’s
    - Back- test’s

18.Implementation:

The details of this project are comprehensively documented in this report, covering its objectives, scope, and expected outcomes. Now, our focus shifts to the implementation phase where we detail how the project was planned and executed. This section will delve into the strategic decisions, methodologies adopted, and the planned versus actual outcomes. By comparing our initial plans with the achieved results, we aim to highlight the journey from conception to realization, emphasizing key insights gained and lessons learned throughout the development process

18.1.Preface:

The focus of this project prototype was to prioritize essential functionalities such as map manipulation, navigation, and basic localization algorithms. Initial development efforts concentrated on establishing robust infrastructure for data streaming from Android devices and WebSocket connections. Screens like user profiles and settings were deprioritized initially to concentrate on core features. As we move forward, our goal is to refine existing algorithms for improved performance in live directions and localization, emphasizing flexibility and scalability in algorithm updates.

18.2.Features and Limitations

Languages:

- **English Only:** Due to time constraints and prioritization, support for other languages is not available.

Settings:

- **No Customization Options:** There are no settings for themes or styling as they were not prioritized and time did not permit their implementation.

Users:

- **No Authentication:** The project does not include login or registration features because they were not prioritized and there wasn't enough time to develop them.

18.3.Backend Overview:

18.3.1.Initial Backend Technology Stack:

Please note that while we outline our Backend Technology Stack, we do not commit to utilizing all listed components:

- **Framework:** Express.js + Node.js
- **Express Libraries:**
    - Multer – for files
    - Aws-sdk

    - Cors
    - Helmet – security related http headers
    - Cookie-parser
    - Express-validator
    - Express-rate-limit
    - Passport + Passport-google-oauth20
    - Morgan - logging
    - Body-parser
    - Mongoclient + Mongoose
    - Socket.io - for real time communication
    - dotenv
- **Containerization:** docker
- **Files Storage:** Amazon S3
- **Computation and Caching:** Redis - for caching and computing across multiple server instances. Caching such as recent routes, blocked ips, sessions , buildings data. Computations such as tiles processing though api, so requests won't standby.
- **Development Only**:
    - Localstack (AWS local emulator):
        - s3 - for files
        - Cloudwatch - logging
        - Ec2 - emulate hosting and dynamic scaling
        - Cloudformation - for configuring cloud resources using yaml.
        - Api Gateway - for handling requests, also has web sockets.
        - Iam - access control permissions.
    - Swagger-ui-express
    - Dev tools:
        - Aws-cli
        - Localstack
        - Docker compose
- **Production**:
    - PM2 for automatic restarts, logging, multiple instances, and zero downtime
    - Nginx and Apache for load balancing across instances
    - Let's Encrypt SSL for free SSL certificates, renewed daily with Ubuntu crontab
    - EC2 AWS cloud hosting for dynamic scaling
- **Testing:**
    - Testing framework: Jest
    - Testing lib helpers: Supertes
- **CI/CD (continuous integration, continuous deployment):**
    - Automated testing Jenkins
    - Version control bitbucket
    - Monitoring and logging: Prometheus + grafana, and AWS cloud watch.
- **Database:** MongoDB (DocumentDB with MongoDB Compatibility)

18.3.2.Prototype Backend Technology stack

- **Framework:** Express.js + Node.js

- **Express Libraries:**
    - Multer: for uploading maps (but i used the maps which seeded from local folder)**(NOT USED YET)**
    - Aws-sdk: did not upload the map to s3, or use ec2.**(NOT USED YET)**
    - Cors
    - Helmet **(NOT USED YET)**
    - Cookie-parser: in mobile there are no cookies , only tokens are used.**(REMOVED)**
    - Express-validator: was not used in all endpoints.(**PARTIAL**)
    - Express-rate-limit: was not used in all endpoints. (**PARTIAL**)
    - Passport + Passport-google-oauth20: No Users **(NOT USED YET)**
    - Morgan - logging: used Winston instead. **(REMOVED)**
    - Winston - logging (**ADDED**)
    - Body-parser
    - Mongoclient + Mongoose
    - Socket.io: Used Websockets instead. **(REMOVED)**
    - ws (websockets) (**ADDED**)
    - dotenv
- **Containerization:** docker
- **Files Storage:** Amazon S3 (**IMPLEMENTED BUT NOT USED**)
- **Computation and Caching:** Redis**(NOT USED YET)**
- **Development Only**:
    - Localstack (AWS local emulator): (**IMPLEMENTED BUT NOT USED**)
        - s3 - for files (**IMPLEMENTED BUT NOT USED**)
        - Cloudwatch - logging **(NOT USED YET)**
        - Ec2 - emulate hosting and dynamic scaling **(NOT USED YET)**
        - Cloudformation - for configuring cloud resources using yaml. **(NOT USED YET)**
        - Api Gateway - for handling requests, also has web sockets. **(NOT USED YET)**
        - Iam - access control permissions. **(NOT USED YET)**
    - Swagger-ui-express
    - Dev tools:
        - Aws-cli **(NOT USED YET)**
        - Localstack (**IMPLEMENTED BUT NOT USED**)
        - Docker compose
- **Production**:
    - PM2 for automatic restarts, logging, multiple instances, and zero downtime(**IMPLEMENTED BUT NOT USED**)
    - Nginx and Apache for load balancing across instances (**NOT USED YET**)
    - Let's Encrypt SSL for free SSL certificates, renewed daily with Ubuntu crontab (**NOT USED YET**)
    - EC2 AWS cloud hosting for dynamic scaling (**NOT USED YET**)
- **Testing:**
    - Testing framework: Jest (**NOT USED YET**)
    - Testing lib helpers: Supertes (**NOT USED YET**)
- **CI/CD (continuous integration, continuous deployment):**

    - Automated testing Jenkins (**NOT USED YET**)
    - Version control bitbucket (**NOT USED YET**)
    - Monitoring and logging: Prometheus + grafana, and AWS cloud watch (**NOT USED YET**).
- **Database:** MongoDB (DocumentDB with MongoDB Compatibility)

18.3.3.Initial Backend Folder Structure:

This was the initial folder structure of the project. However, it was changed as it was not scalable and made the code difficult to locate and maintain.

BackEnd/
├── Core/
├── Config/
├── Routes/
├── Middlewares/
├── Controllers/
├── Services/
├── Repositories/
├── Dao/
├── Entities/
├── Storages/
├── Integrations/
├── Processing/
├── Utils/
├── Exceptions/
├── Constants/
└── tests/
└── ...

18.3.4.Prototype Backend Folder Structure:

Moved to a domain-based folder structure for improved scalability and organization.

├── **scripts**
├── **.docker**: all the docker files
├── **dist:** typescript build
├── logs:
├── src
│   ├── **common**: shared definitions , constants ,interfaces and utilities.
│   │   ├── constants (e.g., http codes).
│   │   ├── enums (e.g., Cardinal Direction)
│   │   ├── interfaces (e.g., MapNode)
│   │   └── **utils**: all utilities organized in folders
│   │   ├── graph

│ │ ├── POI
│ │ ├── routes
│ │ ├── math (e.g., distance.ts, ray-casting.ts)
│ │ ├── localization (e.g., inside-building.ts, floor.ts)
│ │ ├── sensors
│ │ ├── svg
│ ├── **config:** all configurations for system, map engine and 3rd party services.
│ ├── **core:** the main components of the backend , mostly built just like the documentation presented.
│ │ ├── **geolocation:** everything related to geolocation manipulation and matching
│ │ ├── INS
│ │ │ ├── AHRS
│ │ │ ├── IMU
│ │ │ ├── orchestration
│ │ │ ├── step-detection
│ │ │ ├── step-length-estimation
│ │ ├── integration
│ │ │ └── MapEngineIntegration.ts
│ │ ├── magnetic
│ │ │ └── MagneticMatching.ts
│ │ ├── navigation
│ │ │ ├── graph
│ │ │ ├── path-finding
│ │ ├── sensor-fusion
│ │ ├── wifi
│ ├── **data:** anything static data.
│ ├── **domains**
│ │ ├── app-data
│ │ │ ├── api
│ │ │ ├── data-access
│ │ │ ├── domain
│ │ │ └── index.ts
│ │ ├── buildings
│ │ │ ├── **data: domain about general data about the building**
│ │ │ ├── **embedded**: shared
│ │ │ ├── **graph: domain about graph data about the building**
│ │ │ ├── **magnetic: domain about magnetic data about the building**
│ │ │ ├── **map: domain about graph data about the building**
│ │ │ ├── **processing: domain about processing data about the building**
│ │ │ ├── **wifi: domain about wifi data about the building**
│ │ │ └── index.ts
│ │ ├── localization
│ │ │ ├── api
│ │ │ ├── domain
│ │ │ └── index.ts

```
│   │   ├── navigation
│   │   │   ├── api
│   │   │   ├── domain
│   │   │   └── index.ts
│   │   ├── users
│   │   │   ├── api: endpoints, controller
│   │   │   ├── domain: service
│   │   │   ├── data-access: mongo schemas in the database.
│   │   │   └── index.ts
│   │   └── index.ts
│   ├── exceptions: central error handling
│   │   ├── app-error.ts
│   │   ├── errorHandler.ts
│   │   └── index.ts
│   ├── handlers
│   │   └── MemoryFileHandler.ts
│   ├── lib: cross cutting concerns
│   │   ├── env
│   │   └── logger
│   ├── middlewares: shared decorator for apis
│   ├── processing
│   ├── seeding: data that inserted if is not presented in the database
│   │   ├── afeka
│   ├── websockets:
│   │   ├── navigation:  WebSockets specific for navigation.
│   │   │   ├── NavigationProcessor.ts
│   │   │   ├── NavigationRequest.ts
│   │   │   ├── NavigationResponse.ts
│   │   │   ├── NavigationWebSocketEvent.ts
│   │   │   └── WebSocketNavigationHandler.ts
│   │   ├── index.ts
│   │   ├── IWebSocketEvent.ts
│   │   └── WebSocketManager.ts
│   ├── app.ts
│   ├── db.ts: database connection
│   ├── index.ts: application entry point
│   └── swagger.ts
```

18.4.Frontend Overview:

18.4.1.Initial Front End Technology Stack:

Please note that while we outline our Frontend Technology Stack, we do not commit to utilizing all listed components.

The various sensor readings subscribed using the observer design pattern.

- **Framework:** React native
    - React Native Notable Libraries:
        - React Navigation
        - redux
- **Apple Step Detection:**
    - Healthkit
    - Health Connect
- **Android Step Detection:**
    - Samsung health
- **Testing:** Jest

18.4.2.Prototype Front End Technology Stack:

The various sensor readings subscribed using the observer design pattern.

- **Framework:** React native
    - React Native Notable Libraries:
        - React Navigation
        - Rxjs (**ADDED**)
        - Redux (**ADDED EXTRA**)
    - Android Source Code:
        - Wifi Package (**ADDED AND IMPLEMENTED**)
        - Sensor Package (**ADDED AND IMPLEMENTED)**
- **Apple Step Detection (REMOVED):**
    - Healthkit
    - Health Connect
- **Android Step Detection (REMOVED):**
    - Samsung health
- **Testing:** Jest **(NOT USED YET)**
- **Platform:** only android. (**MODIFIED**)

18.4.3.initial Frontend Folder structure:

```
FrontEnd/
│
│   ├── components/
│   │   ├── Common/
│   │   ├── CustomHooks/
│   │   ├── MapBuilding/
│   │   ├── UserPath/
│   │   ├── UserProfile/
│   │   ├── UserPathChose/
│   │   ├── Directions/
│   │   └── POI/
│   ├── screens/
│   │       ├── HomeScreen/
│   │       ├──  LoginScreen/
│   │       ├── RegisterScreen/
│   │       ├── SelectBuildingScreen/
│   │       ├── ProfileScreen/
│   │       └── MainScreen/
│   ├── models/
│   ├── services/
│   ├── utils/
│   ├── navigation/
│   └── redux/
└── App.js
```

```
│   ├── tests/
│   │   ├── components/
│   │   │   ├── Common/
│   │   │   ├── CustomHooks/
│   │   │   ├── MapBuilding/
│   │   │   └── UserPath/
│   │   ├── screens/
│   │   │   ├── HomeScreen/
│   │   │   ├── LoginScreen/
│   │   │   ├── RegisterScreen/
│   │   │   ├── SelectBuildingScreen/
│   │   │   ├── ProfileScreen/
│   │   │   └── MainScreen/
│   │   ├── models/
│   │   ├── services/
│   │   ├── utils/
```

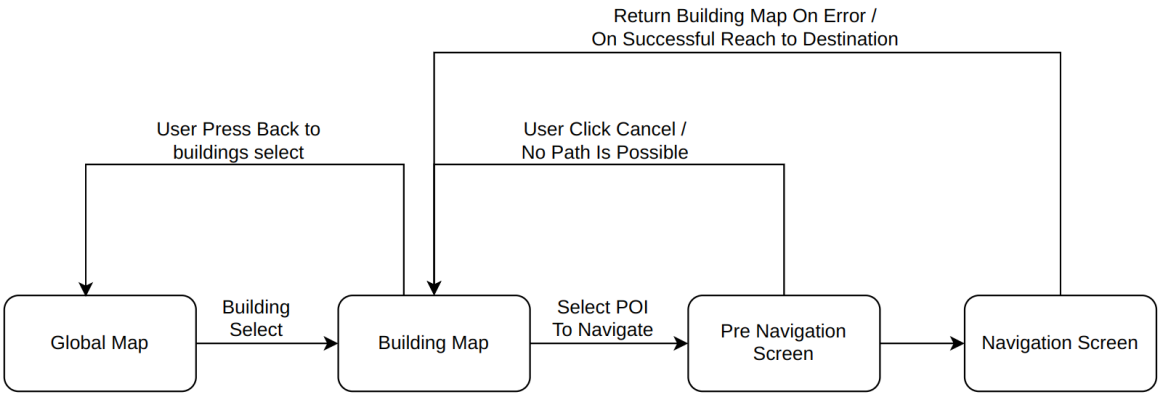
```
│  │  ├── navigation/
│  │  └── App.test.js
└──└── ...
```

18.4.4.Prototype frontend folder structure:

```
./frontend
├── Client
│   ├── ios
│   ├── android/app/src/main/java/* - modules implemented to get data from android phone
│   ├── src
│   │   ├── app - "redux toolkit" convention folder called 'app' , all redux slices.
│   │   │   ├── active - the active building data slice
│   │   │   ├── admin - admin related api, routes, points and graph data.
│   │   │   ├── building - all buildings related data
│   │   │   ├── map - the maps of the selected building
│   │   │   ├── navigation - navigation slice for routing
│   │   │   ├── user - unused
│   │   │   ├── logger.ts – redux logger
│   │   │   ├── status.ts - state of slice
│   │   │   └── store.ts - redux main store object
│   │   ├── assets: frontend only images (user arrow, etc..)
│   │   ├── components: (general component that are shared across multiple screens)
│   │   ├── config: (Configuration such as server url)
│   │   ├── constants: (enums and constants)
│   │   ├── contexts: (shared state across multiple components, used for UI based components like messages)
│   │   ├── externals: (facade for external services)
│   │   │   └── map-service-provider.js (redirect to different application to get directions to the building by using the phone map provider if exists)
│   │   ├── hooks: (react states with logic)
│   │   ├── layouts: (the main layouts of the screen, map overlay, map layout , bottom drawer etc…)
│   │   ├── main: (the main factory  of the application after initialization)
│   │   ├── navigation: ( screen navigation logic )
│   │   ├── core: (partial map engine for client , not during navigation, without websockets)
│   │   ├── screens ( all the application screens, categorized by domain)
│   │   │   ├── admin: (admin related screen , route collection screen, points collection screen)
│   │   │   ├── auth: (login and signup screen)
│   │   │   ├── errors (error screens)
│   │   │   ├── general (general screens, loading screen,settings, etc..)
│   │   │   ├── main (all the application screens, each screen in main , got his entry point and all the screen specific components in folder called  components):
│   │   │   │   ├── components (shared components only in all main)
│   │   │   │   ├── building-global-map
```

│ │ │ │ ├── **building-map**
│ │ │ │ ├── **building-pre-navigation**
│ │ │ │ ├── **building-navigation**
│ │ │ │ ├── **building-search**
│ │ │ │ ├── **building-pois-search**
│ │ │ └── **modals: (**general pop ups ,permissions,errors,location services, etc…**)**
│ │ ├── **sensors: (** Integration of Android sensors via React Native for sensor data handling **)**
│ │ ├── **server (**server related interfaces**)**
│ │ │ ├── **api-client.ts** (all rest api calls)
│ │ │ ├── **navigation-ws-client.ts (**based on ws-client, a specific implementation for navigation and localization, with predefined events, and calls**)**
│ │ │ └── **ws-client.ts (**server websocket)
│ │ ├── **services (**all the streams services using RXJS, singleton, observables**)**
│ │ │ ├── **GpsService.ts** (streaming GPS data with the defined configurations)
│ │ │ ├── **SensorsService.ts (**streaming Sensor data by choice with the defined configurations**)**
│ │ │ ├── **UserIndoorPositionService.ts** (streaming the user position to a component that request this, uses userSpatialDataStream for data, and partial map engine for the logic)
│ │ │ ├── **UserSpatialDataStreamService.ts** (streaming all the spatial data needed for localization and navigation, uses the other services to require data, and merge them accordingly)
│ │ │ └── **WifiService.ts** (streaming all ready WI-FI data)
│ │ ├── **static-maps: (**used for global map, israel map**)**
│ │ ├── styles
│ │ └── utils
…

18.5.GUI - Application Screens:



[Diagram 18.5.Application Screens]

18.5.1.Global Map Screen:

- **Overview:** Displays a comprehensive map of Israel with clickable buildings to view details.
- **User Location:** Shows the user's global location in relation to buildings, including proximity.
- **Auto-Detection:** Automatically detects if the user is within a building's coordinates.
- **Search Functionality:** Allows users to search for buildings.

18.5.2.Building Map Area:

- **Detailed View:** Users can enter a building map to see all points of interest (POIs).
- **Interaction:** POIs can be selected by clicking or searching.
- **Map Controls:** Users can pan, zoom in, rotate by 90 degrees, center the map on their location, switch floors, and lock the position on the user.
- **Navigation:** Users can initiate navigation to a POI by clicking on it.

18.5.3.Pre-Navigation Screen:

- **Path Display:** After selecting "navigate" on a POI, the path across multiple floors is shown, including distance and estimated time.
- **Feasibility Check:** If the path is not feasible based on (currently unimplemented) preferences, the user is notified and returned to the building map.
- **Confirmation:** Users can confirm to proceed to the navigation screen or cancel to return to the building map.

18.5.4.Navigation Screen:

- **Live Directions:** Provides real-time directions with the user's location displayed on the map.
- **WebSocket Updates:** The user receives live updates via WebSockets.
- **Recalculation:** If the user deviates from the path beyond a certain threshold, the route is recalculated and a new route is provided.

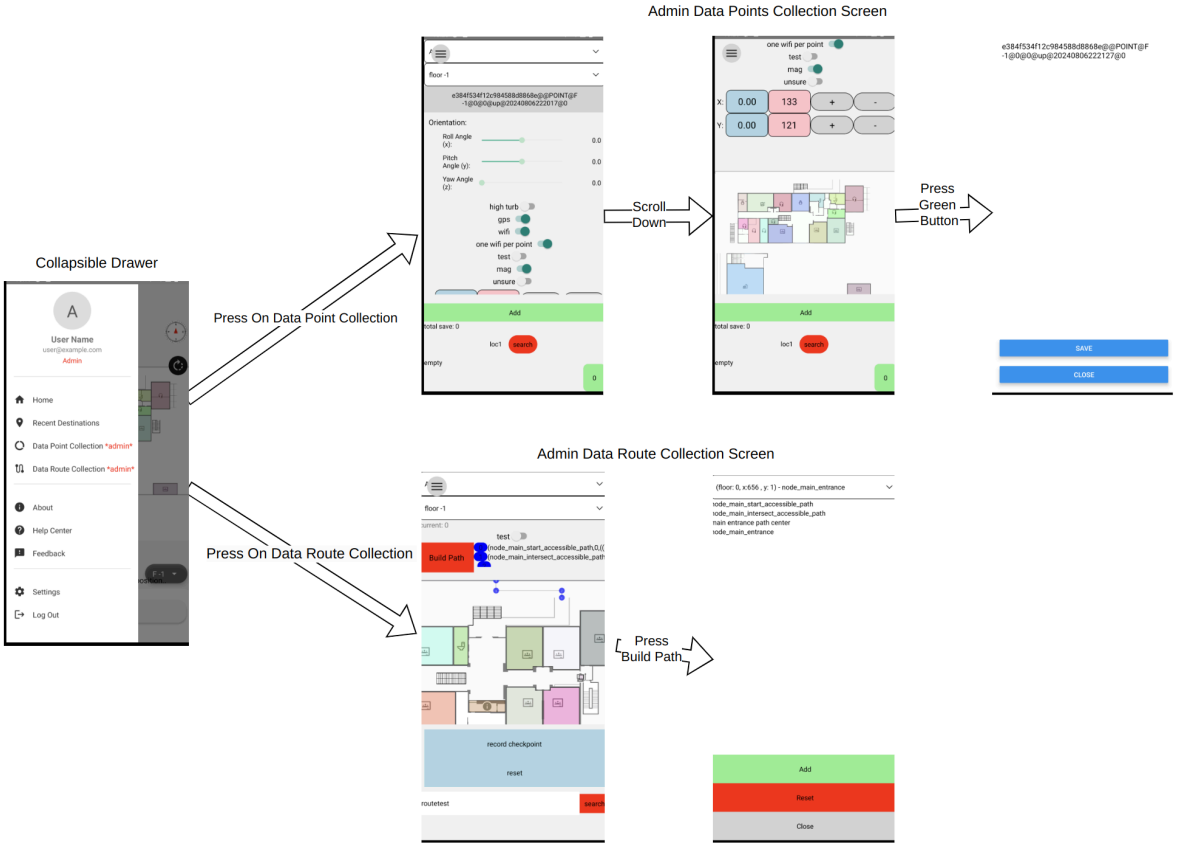
18.6.UI/UX Flowchart:

18.6.1.Admin Screens:

From drawer navigation only, you can navigate to the admin screens.

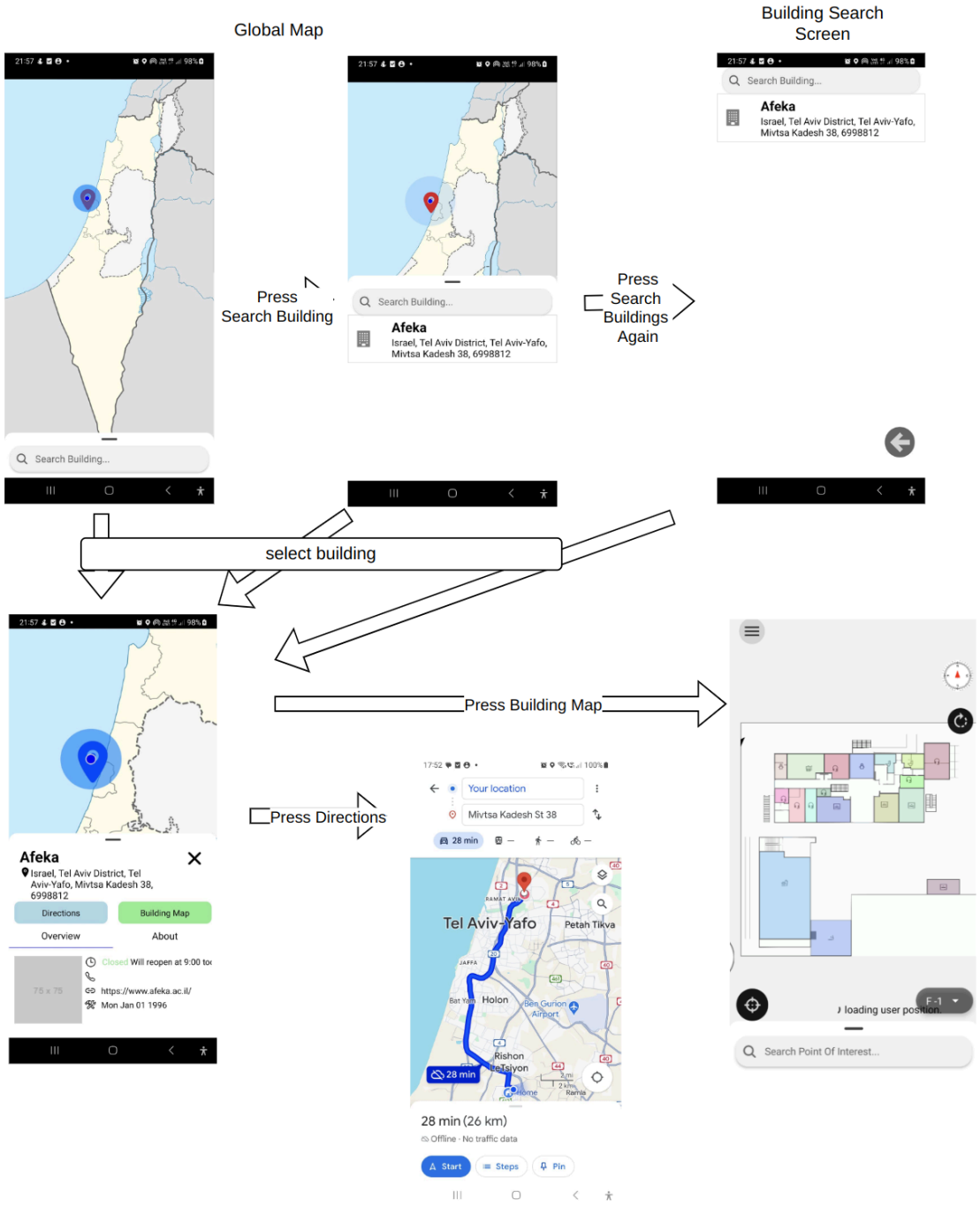
Admin data points collection screen:
Screen to collect data at various points in selected building and selected floor

Admin Data route collection screen:
Screen to collect routes data at various route



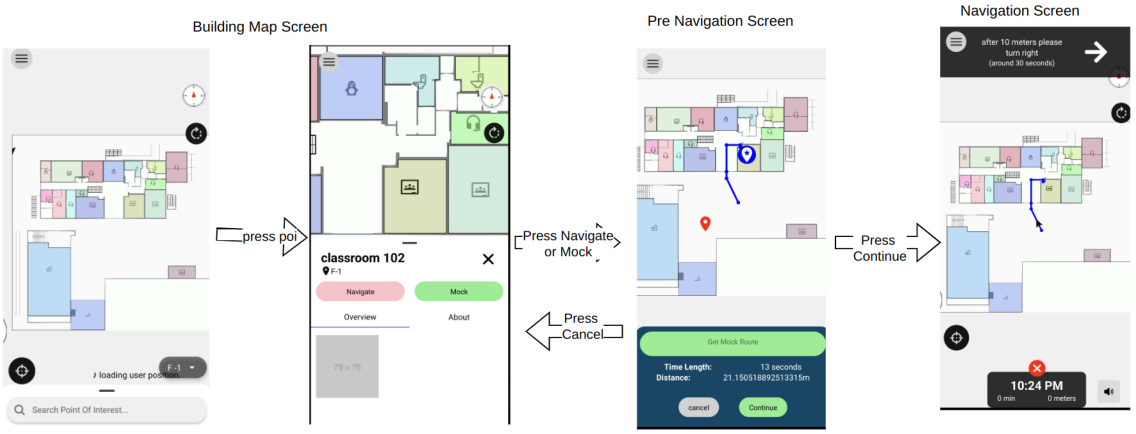
[Diagram 18.6.1.Admin Screens Navigation]

18.6.2.Global Maps Screens:



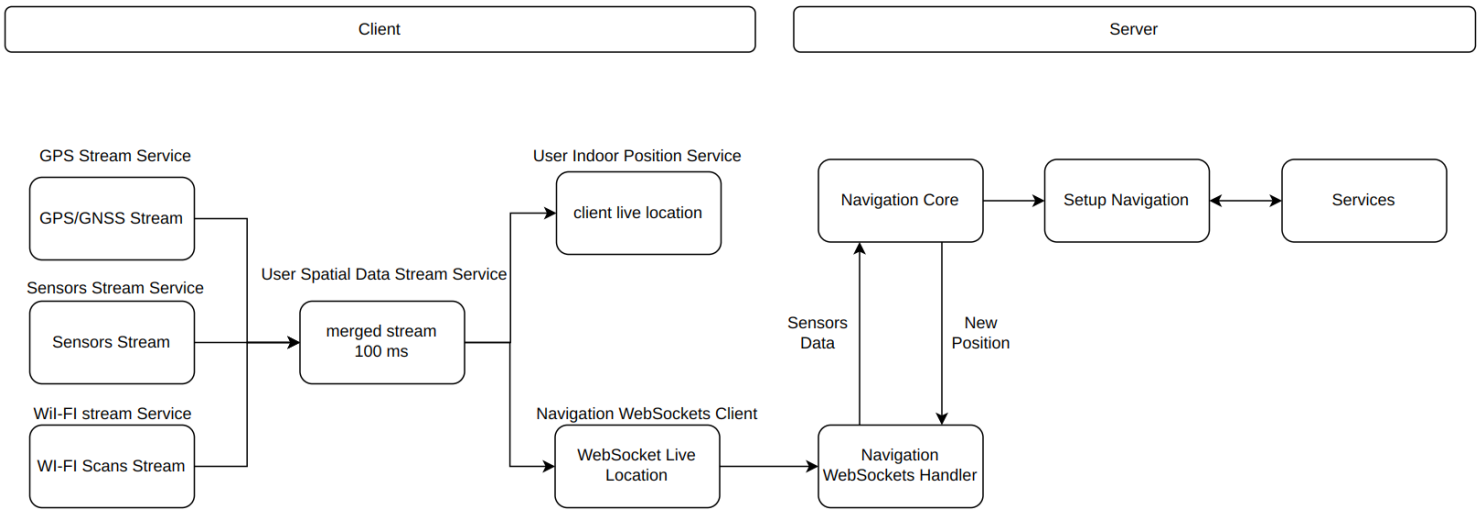
[Diagram 18.6.2.Global Map Screens]

18.6.3.Building Map Screens:



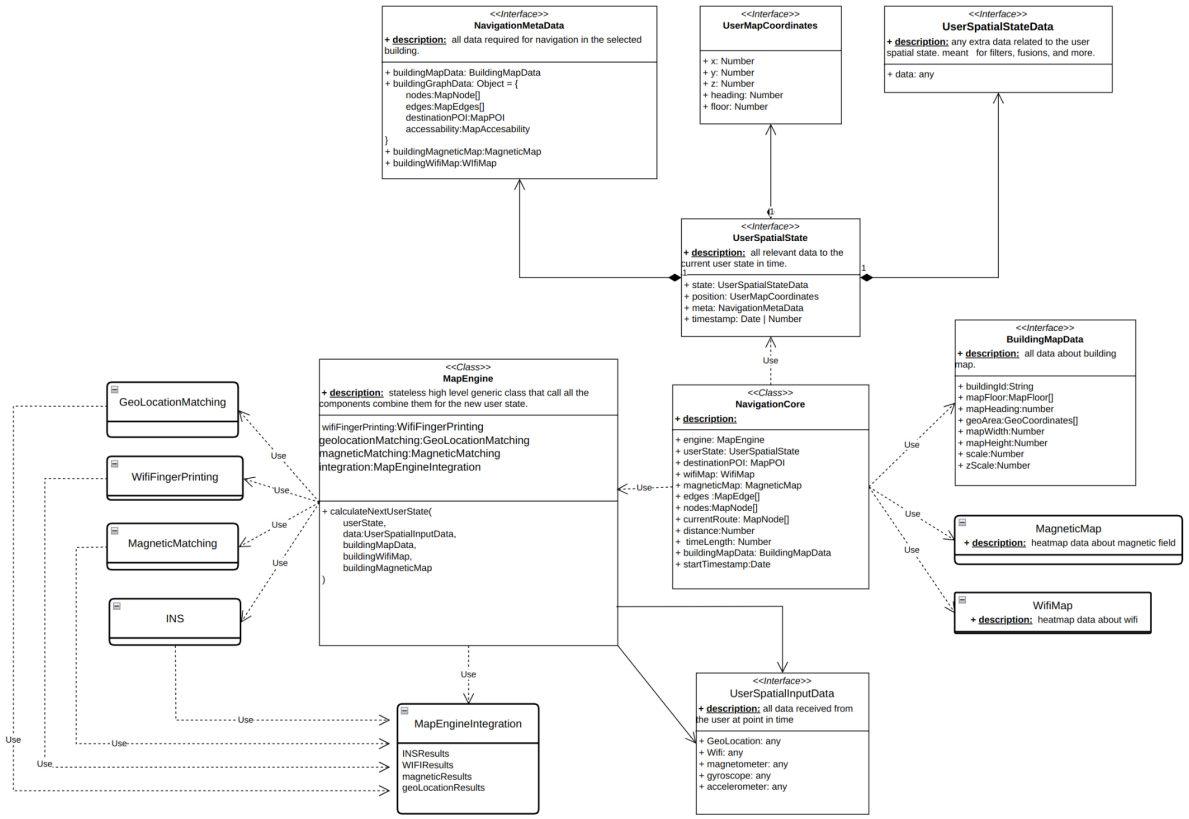
[Diagram 18.6.3.Building Map Screens]

18.7.Main Navigation Flow:



[Diagram 18.5.Navigation Flow]

18.8.Navigation and Localization Class Diagram:



[Diagram 18.5.Navigation And Localization Classes Diagram]

18.9.Algorithm Overview:

18.9.1.Prototype Navigation

Our navigation algorithm utilizes Dijkstra's Shortest Path algorithm with preferences for accessibility. For instance, users can specify preferences such as avoiding stairs when navigating to Points of Interest (POIs) from their current location.

Implementation Details:

1. **Input Handling:**
    - Retrieve user location and destination POI from the client.
    - Fetch nodes and edges specific to the building ID.
2. **Identifying Destination Node:**
    - Determine the nearest node to the current location that corresponds to the destination POI ID. This node becomes the destination node.
3. **Graph Preparation:**
    - Filter out edges classified as stairs for accessibility preferences.
    - Introduce a virtual node representing the user's current location and integrate it into the graph structure.
4. **Path Calculation:**
    - Identify the nearest node to the virtual current location node.
    - Establish a virtual edge connecting the current location node to this nearest node.
    - Incorporate the filtered edges into the graph.
    - Remove any isolated nodes that are no longer connected in the graph.
5. **Route Calculation:**
    - Execute the shortest path calculation from the current node to the destination node.
    - If a valid path is found, generate an SVG representation of the path, including time length and total distance.

Real-Time Navigation: During real-time navigation, graph data is fetched only once, leveraging persistent WebSocket connections for subsequent operations. This optimizes performance by minimizing data retrieval operations.

This approach ensures efficient navigation while adhering to user preferences and leveraging real-time data connectivity for seamless user experiences.

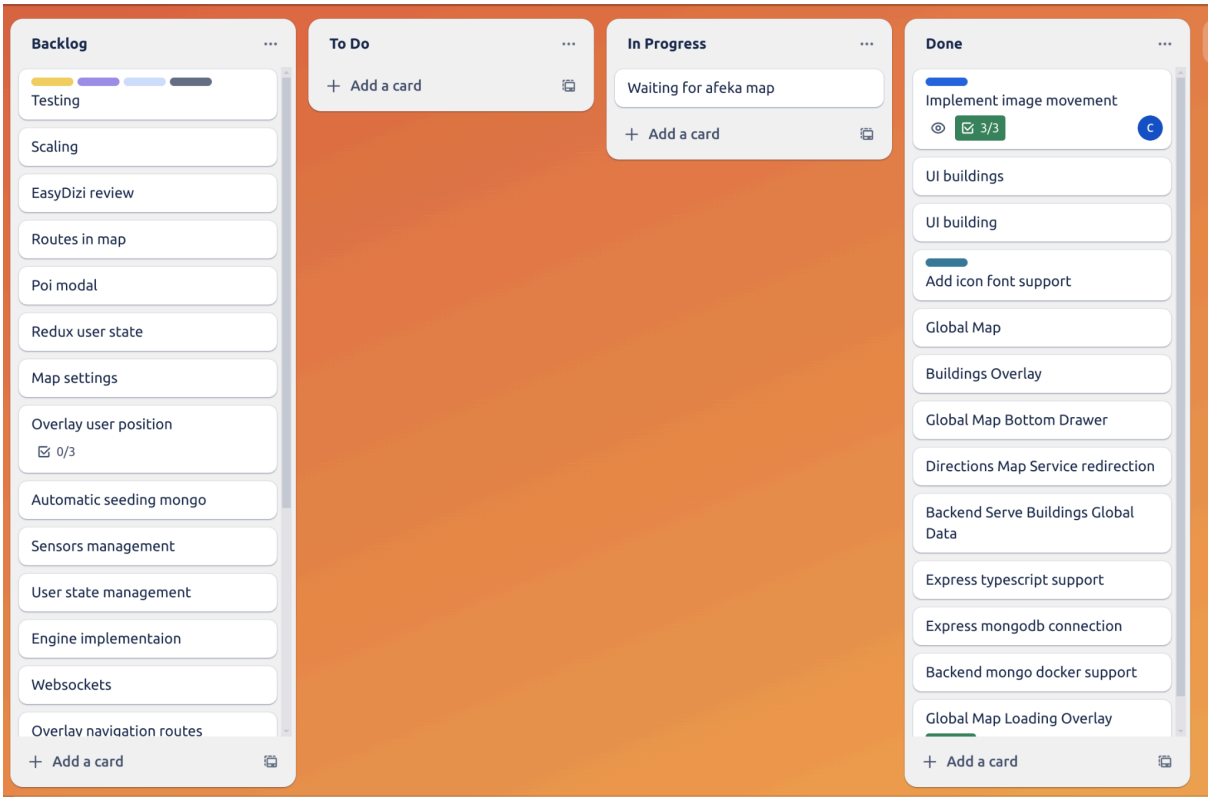
18.8.2.Prototype Map Engine Algorithms Configurations:

- **WI-FI Sampling Interval =** 1 scan in 30 sec.
- **Accelerometer Sampling Interval** = 100 MS
- **Magnetometer Sampling Interval** = 100 MS
- **Gyroscope Sampling Interval** = 100 MS
- **Sensors Orchestration:** No Calibrations.

- **User position:** we will use Quaternion based Extended Kalman Filter (EKF) to recursively update the position state from the various sensors. We will use the user's position in the state after calculating the user's position from the smartphone coordinate system. (not working well yet)
- **Step detection window-based buffer initial time** - Initially: 5 seconds (with accelerometer readings interval, should give a fixed size window)
- **The integration:**
    - **GNSS/GPS:** If gps accuracy is high, and distance from the current position and the new gps position is above a certain threshold, realignment using the GPS location.
    - **WI-FI:** wifi data available, process wifi scan data. And use that to get the next location. (meaning not using the prev wifi, only the current, for now).
    - **Magnetic:** was not implemented yet.
    - **INS:** always uses INS.
- **WIFI Number Observed Access Points sequence quality control**: APThreshold = 3
- **WIFI signal quality control :** RSSThreshold = -75
- **WIFI spatial range DistanceThreshold** = Not Implemented Yet
- **WIFI Rolling Window RSSs Sequence Buffer:** 0
- **WIFI Sequence Normalization:** None
- **WIFI Matching:** Not Implemented Yet
- **WIFI Similarity Metric:**Not Implemented Yet
- **WIFI Adaptive Sampling Frequency Handling:**Not Implemented Yet
- **WIFI Comparing:** Not Implemented Yet
- **MM Rolling Window Magnetic Sequence Buffer:** Not Implemented Yet
- **MM Standard-Deviation and Change Filtering (quality control):**Not Implemented Yet
- **MM Comparing & Matching:** Not Implemented Yet

19.Development and Progress:

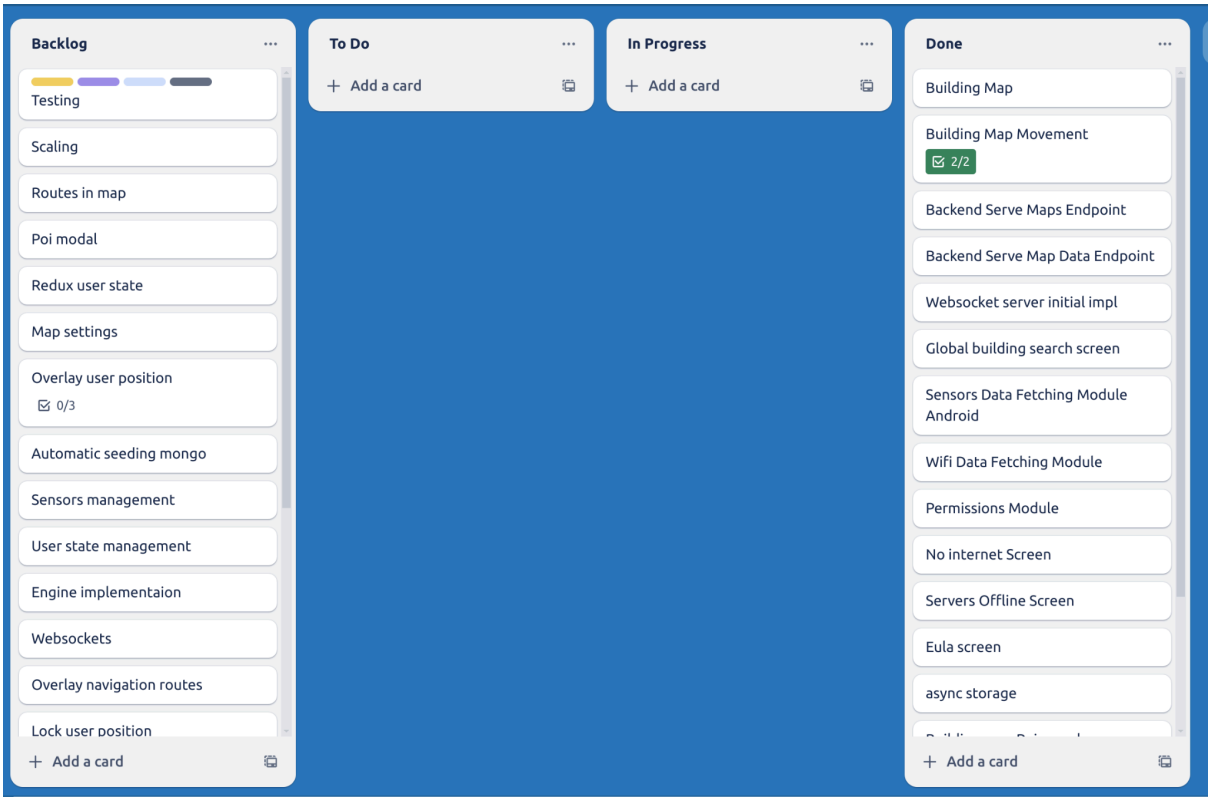
19.1.Sprint 1.3 - 14.3:

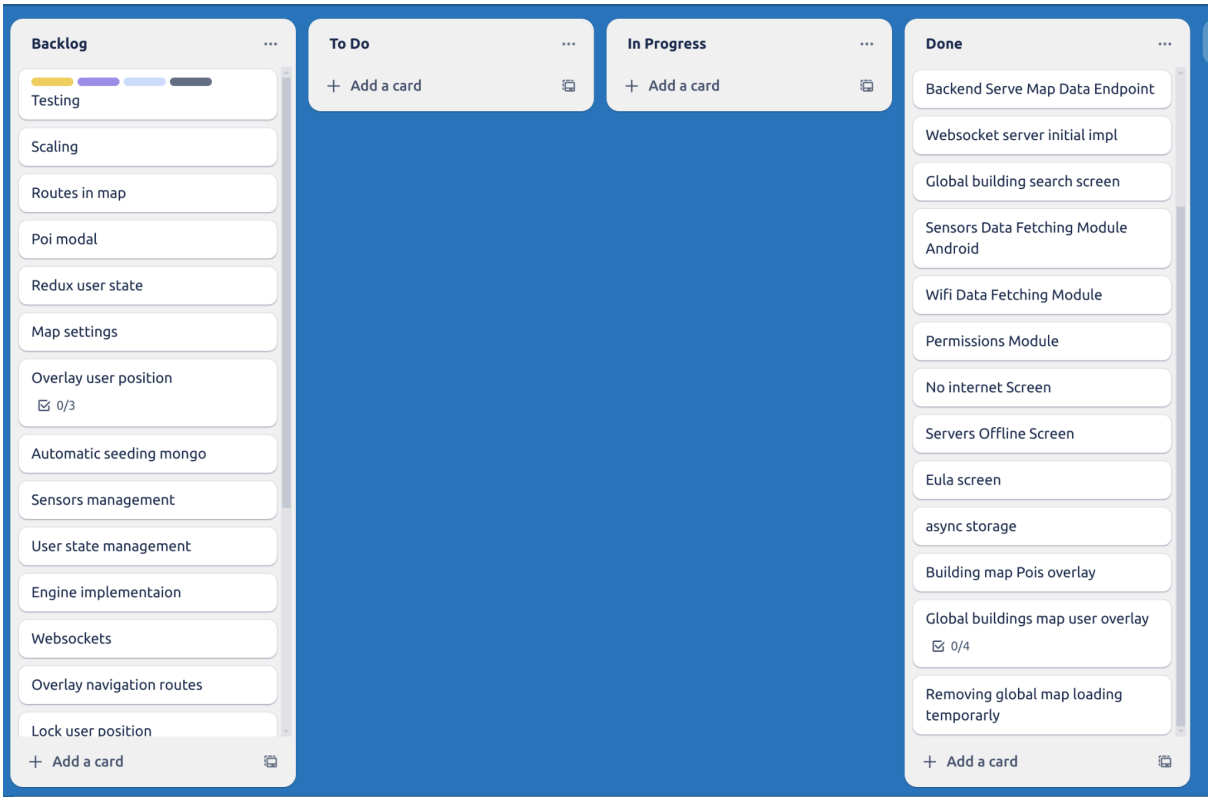


[Image Trello 19.1.Sprint 1.3-14.3]

- Began by waiting for the building blueprints and floor maps from Afeka.
- Created the backend infrastructure with an Express server, MongoDB, and Docker.(1)
- Faced issues with TypeScript and Docker, so postponed Docker support.
- Implemented Map Redirection for map service provider in the phone (was removed for meantime).
- Created a Global Map UI.
- Implementing getting data from wifi networks in android code
- forking react-native-sensors's android code modifying the code to be more explicit , and errorless (later a big rework) .

19.2.Sprint 14.3 - 1.4:

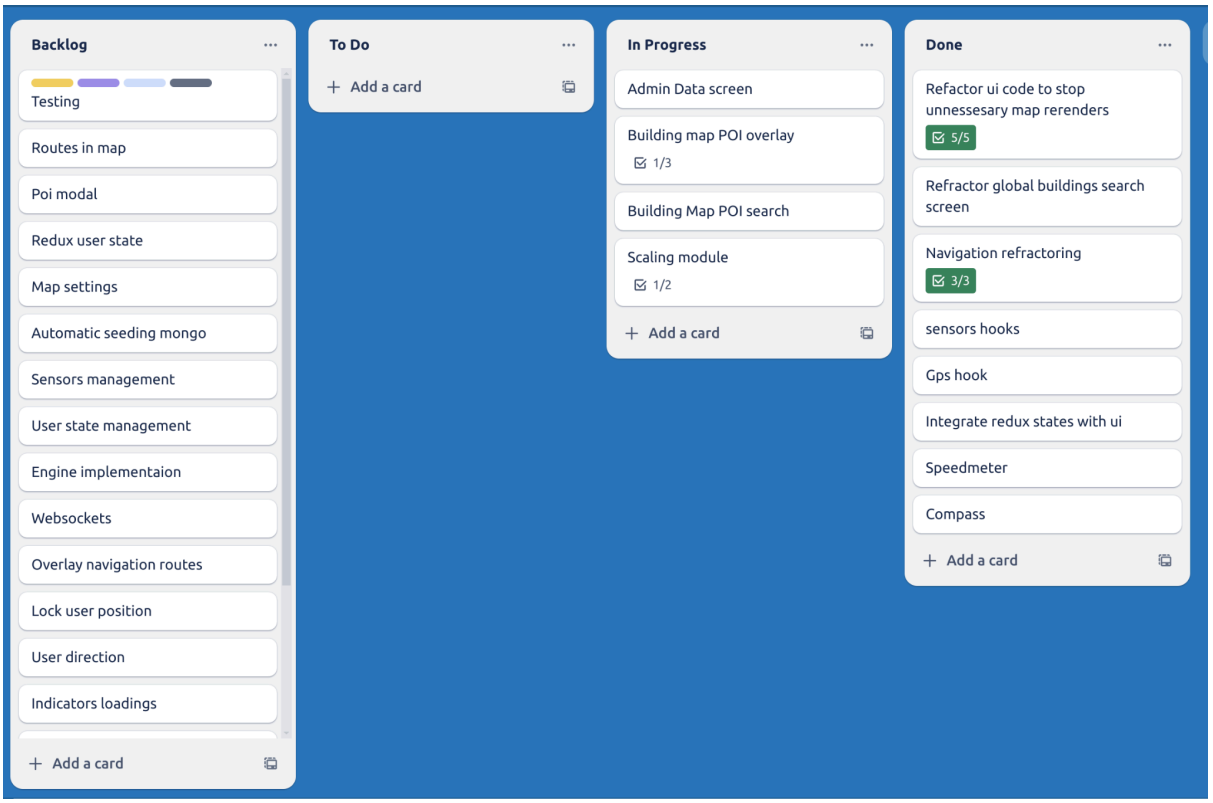




[Image Trello 19.2.Sprint 14.3-1.4]

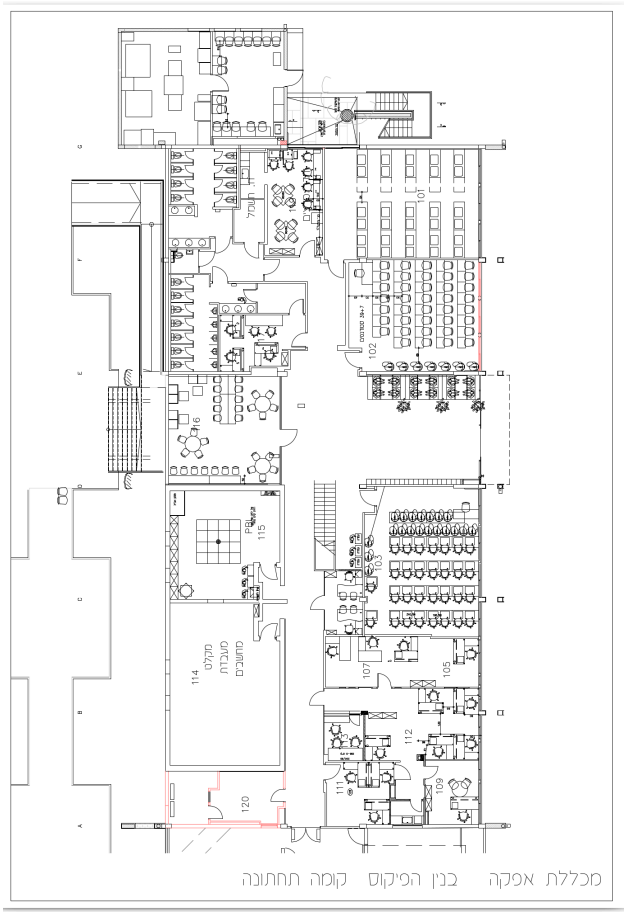
- Created the backend infrastructure with an Express server, MongoDB, and Docker.(2)
- Built the initial React Native UI.
- Implemented a feature to list and showcase buildings in the application.
- Permissions, Error, Loading Screens, for indicators in development.
- Creating pulsing circle user location in the global map.
- left drawer was removed for sake of simplicity

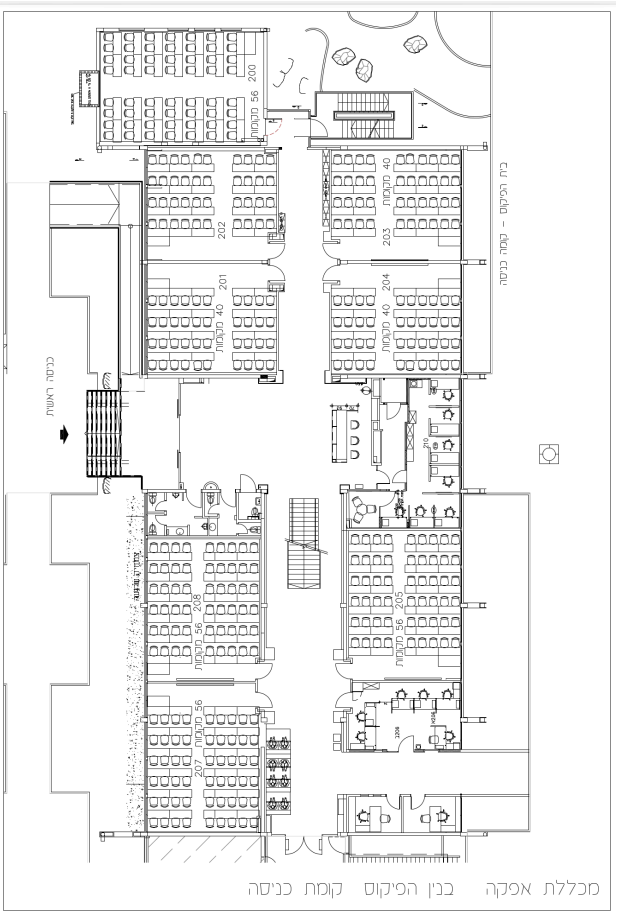
19.3.Sprint 1.4 - 14.4:



[Image Trello 19.3.Sprint 1.4-14.4]

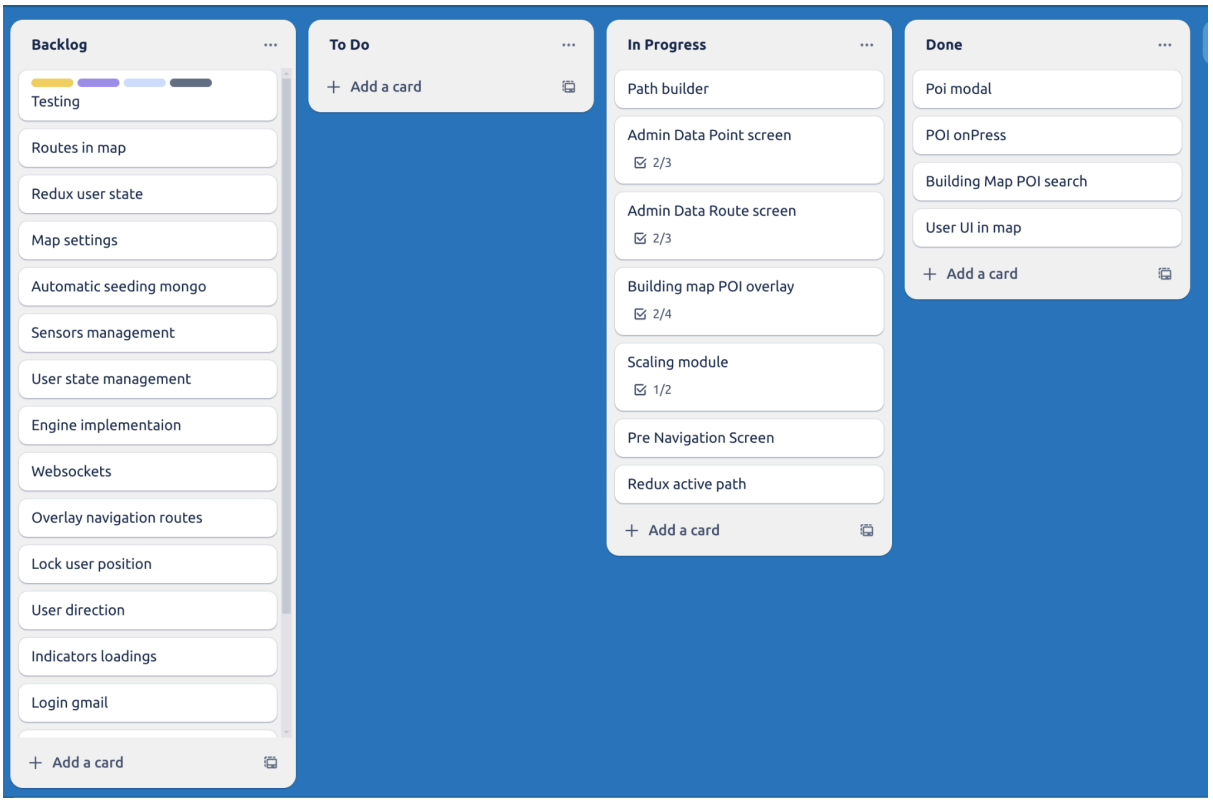
- Had a complete refactor for the frontend components, for creating more modular components (1).
- Creating redundant hooks(which was not used).
- Added redux (complete rewrite of component data fetching).
- Creating the admin data screens, scaling module (which was postponed) .
- Received the Afeka floor's maps for floor 0, and floor -1 (for the record: inaccurate , outdated, partial maps, but are good enough):





[Image 19.3.Buildings Blueprints]

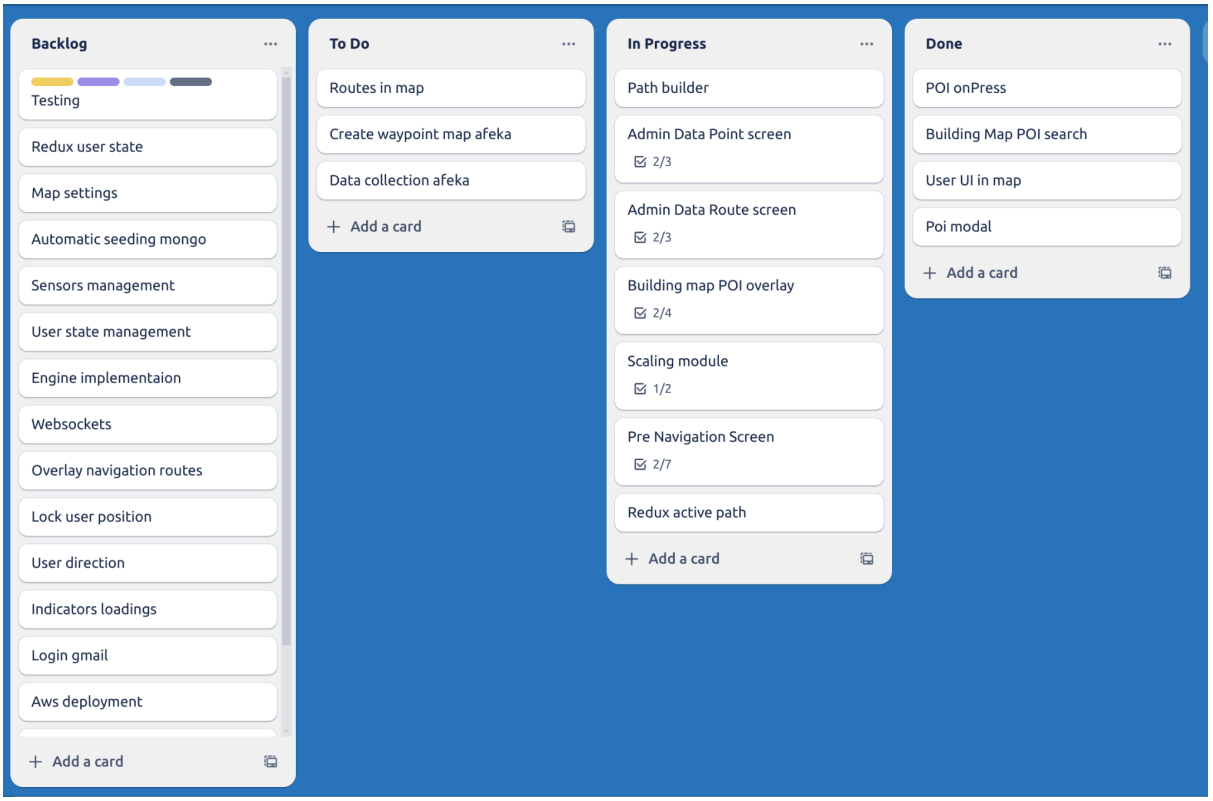
19.4.Sprint 14.4 - 1.5:



[Image Trello 19.4.Sprint 14.4 - 1.5]

- Create a path builder for custom routing , yet was abandon for now, since navigating for one location should be the first step (1)
- Creating Admin screens - almost finished, for collecting data about the routes, and building wifi.
- Meanwhile photoshopping the floor's maps
- Big rework of sensor android code. After being completely immersed in the documentation of android: https://developer.android.com/develop/sensors-and-location/sensors/sensors_overview

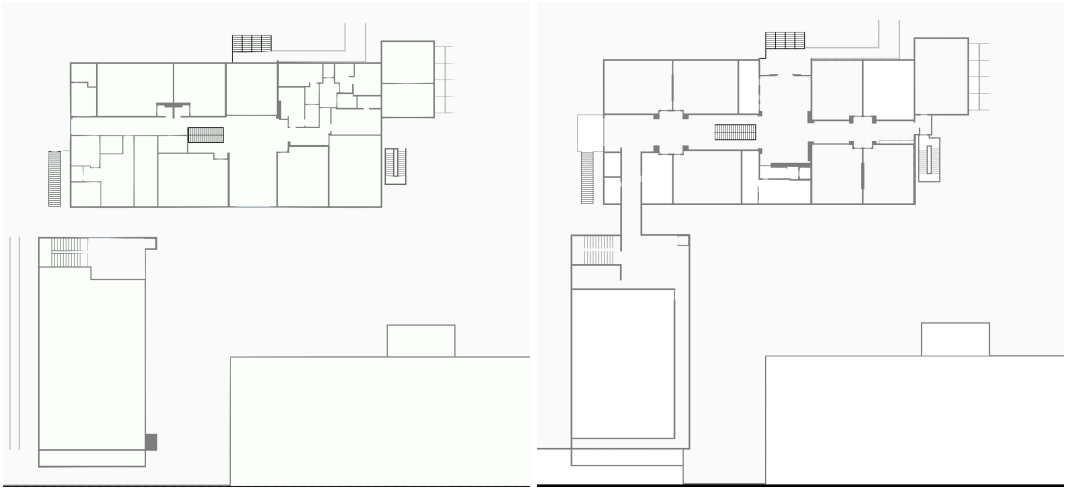
19.5.Sprint 1.5 - 14.5:



[Image Trello 19.5.Sprint 1.5-14.5]

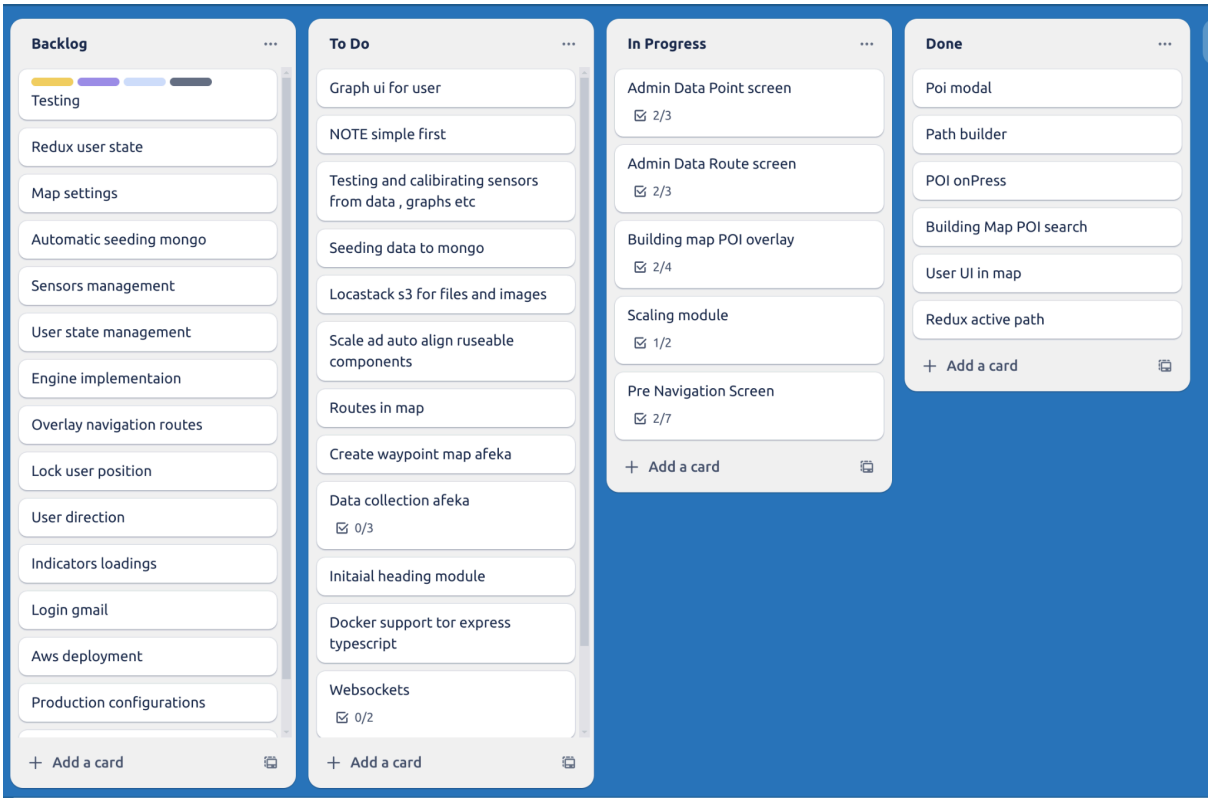
- Create a path builder for custom routing , yet was abandon for now, since navigating for one location should be the first step (2)
- Finished POI modals
- Created User Arrow Icon in Map.
- Manually collecting Afeka map's measurements. (measuring centimeters).
- Photoshop Map using photopea, and map logic so they are aligned (same width , same height, stairs in same place etc…):

Floor -1 floor 0



[Image 19.5.Building Maps]

19.6.Sprint 14.5 - 1.6:



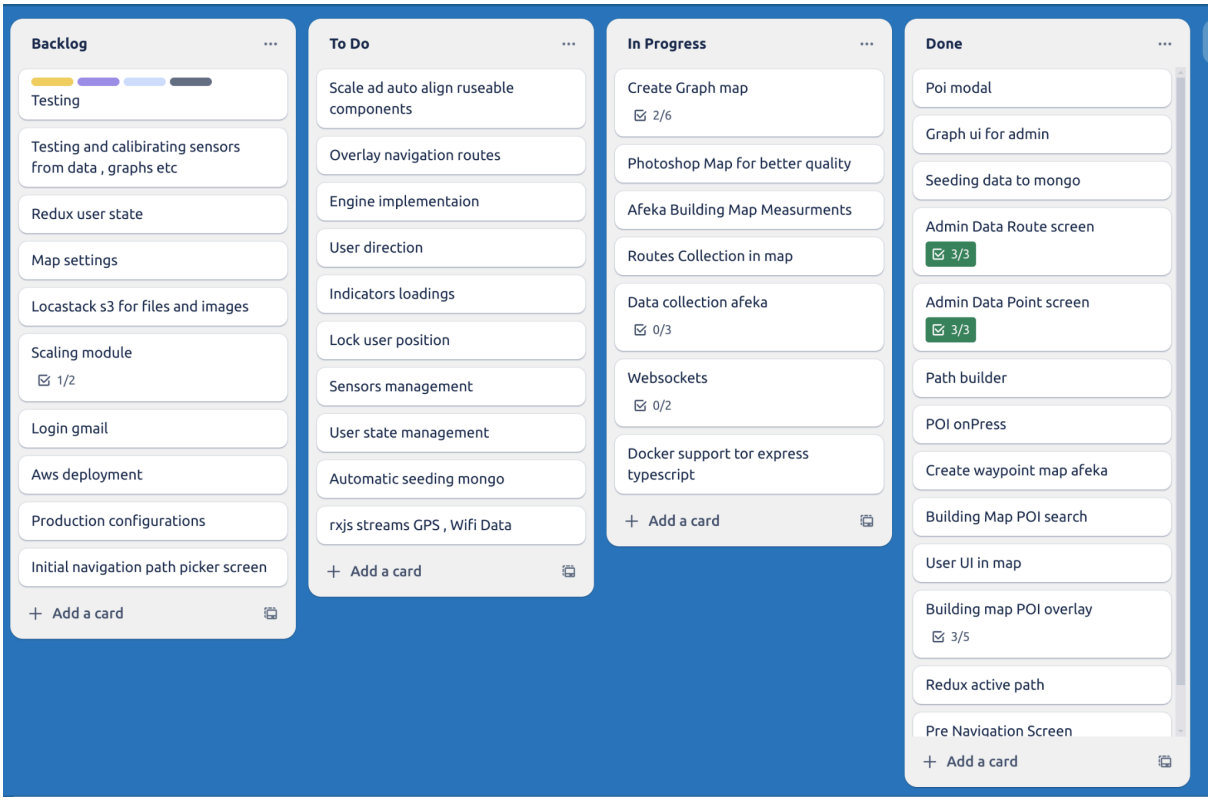
[Image Trello 19.6.Sprint 14.5 - 1.6]

- Working hard on the admin screen, making sure add all data is collected: wifi gnss/gps,and sensors which includes all the physical sensors and also virtual

sensors(which are sensors that has are not actual sensors, but using the basic sensors as a base):

- Physical Sensors:
    - Magnetometer, magnetometer uncalibrated ,
    - Accelerometer, gyroscope, gyroscope uncalibrated ,
- Virtual:
    - Rotation vector
    - Gravity vector
    - Linear acceleration vector
- And many variables that might be needed for later on just in case.
- Noticing a performance hit with lower sampling rate under 100ms.

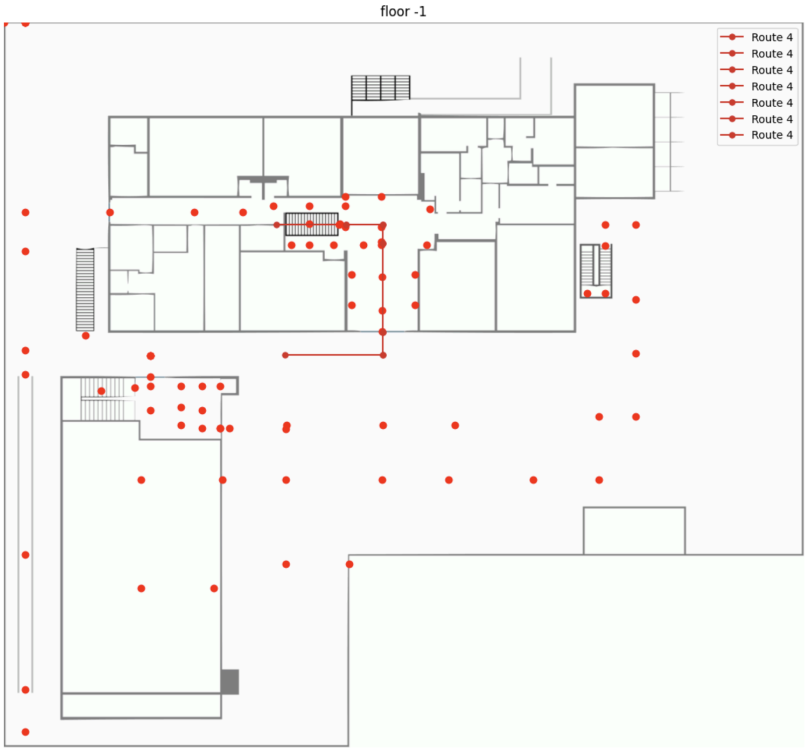
19.7.Sprint 1.6 - 14.6:

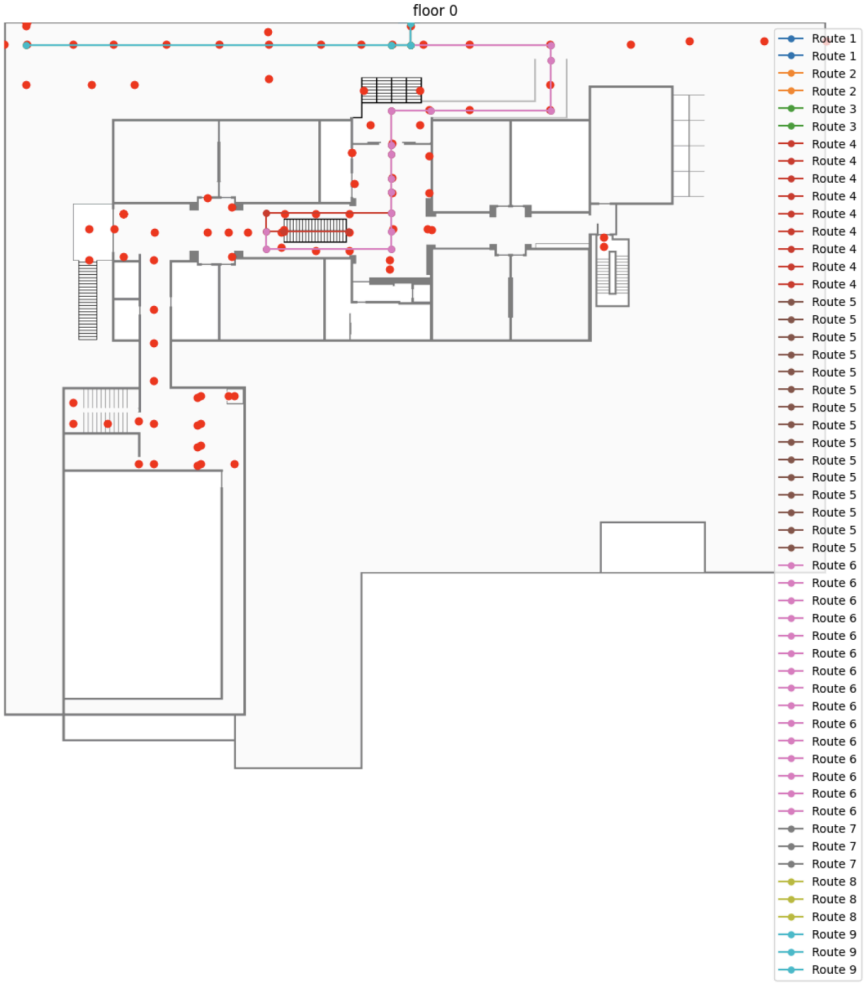


[Image Trello 19.7.Sprint 1.6 - 14.6]

- Start learning how to use the RXJS library for observer reactive streams.

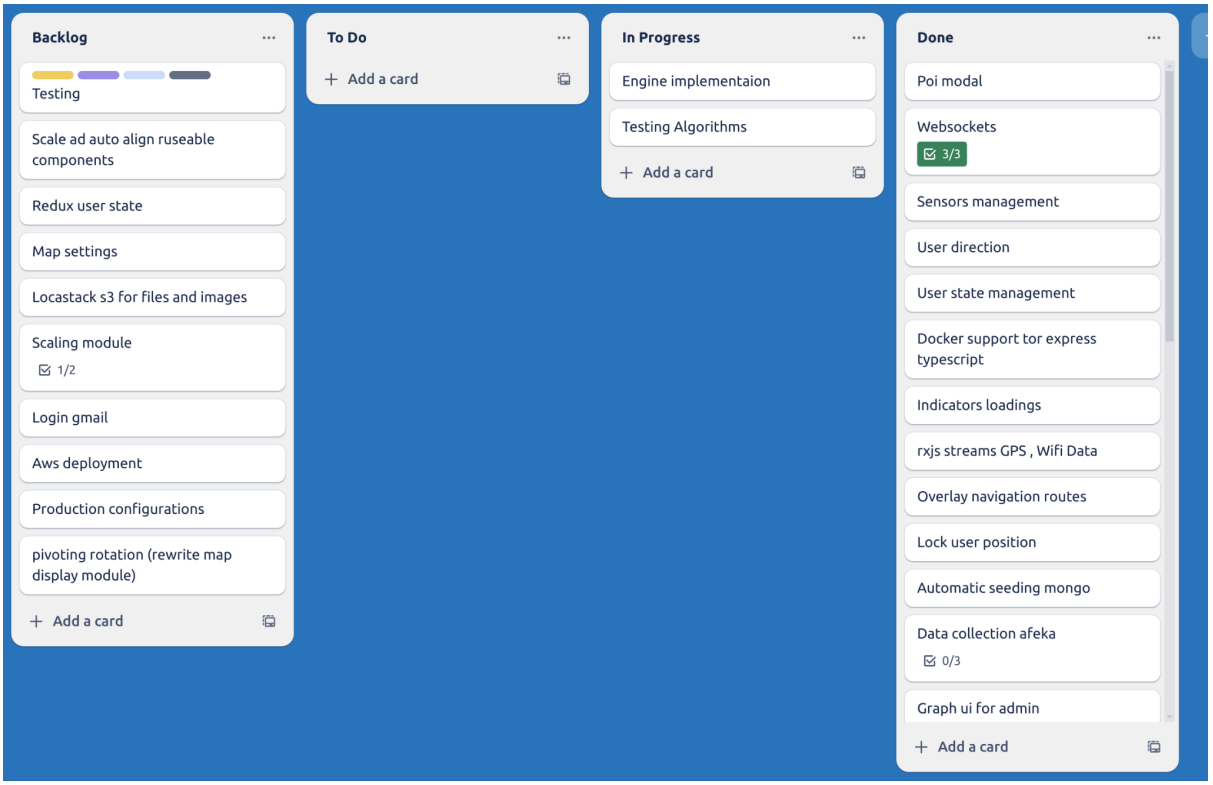
- Complete sensors , wifi , GPS , streams refractor
- Initial implementation of the map engine, (without algorithms)
- Finish admin screens
- Manually collects map data, collecting many points and routes (many problems arised , like crashes, and data loss).
- The collected wifi points per floor: (there are some errors, such as the 2 points at the top left at floor -1).
- The collected routes (need to take them again):


floor -1
Route 4
Route 4
Route 4
Route 4
Route 4
Route 4
Route 4


floor 0
Route 1
Route 1
Route 2
Route 2
Route 3
Route 3
Route 4
Route 4
Route 4
Route 4
Route 4
Route 4
Route 4
Route 4
Route 4
Route 5
Route 5
Route 5
Route 5
Route 5
Route 5
Route 5
Route 5
Route 5
Route 5
Route 5
Route 5
Route 5
Route 5
Route 6
Route 6
Route 6
Route 6
Route 6
Route 6
Route 6
Route 6
Route 6
Route 6
Route 6
Route 6
Route 6
Route 6
Route 7
Route 7
Route 7
Route 8
Route 8
Route 8
Route 9
Route 9
Route 9

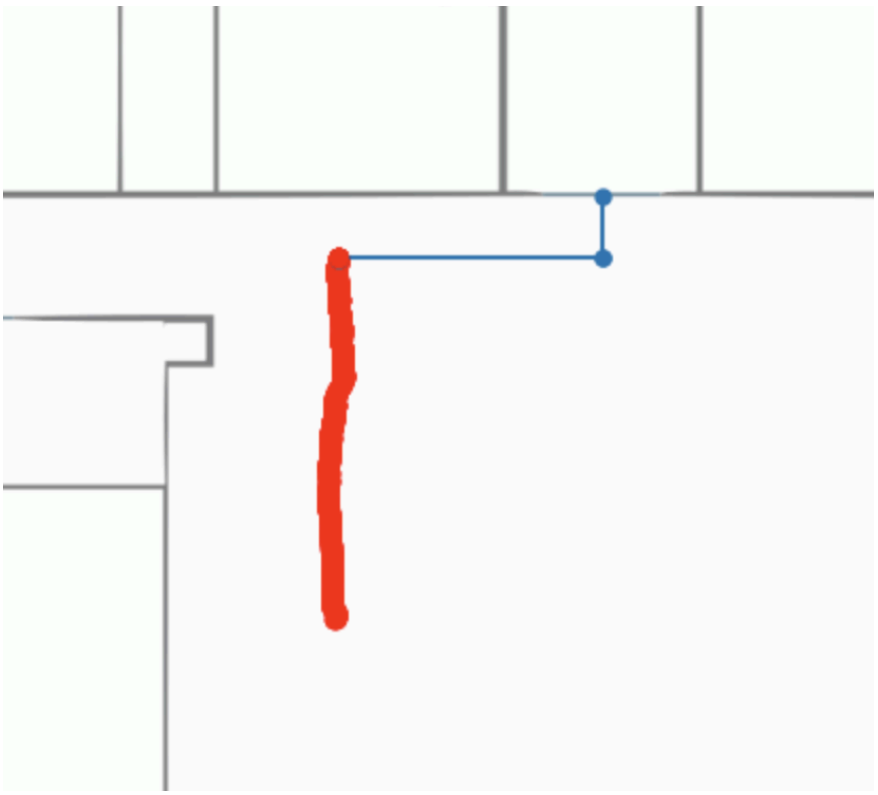
[Image 19.7.Routes And Points Collected]

19.8.Sprint 14.6 - 1.7:



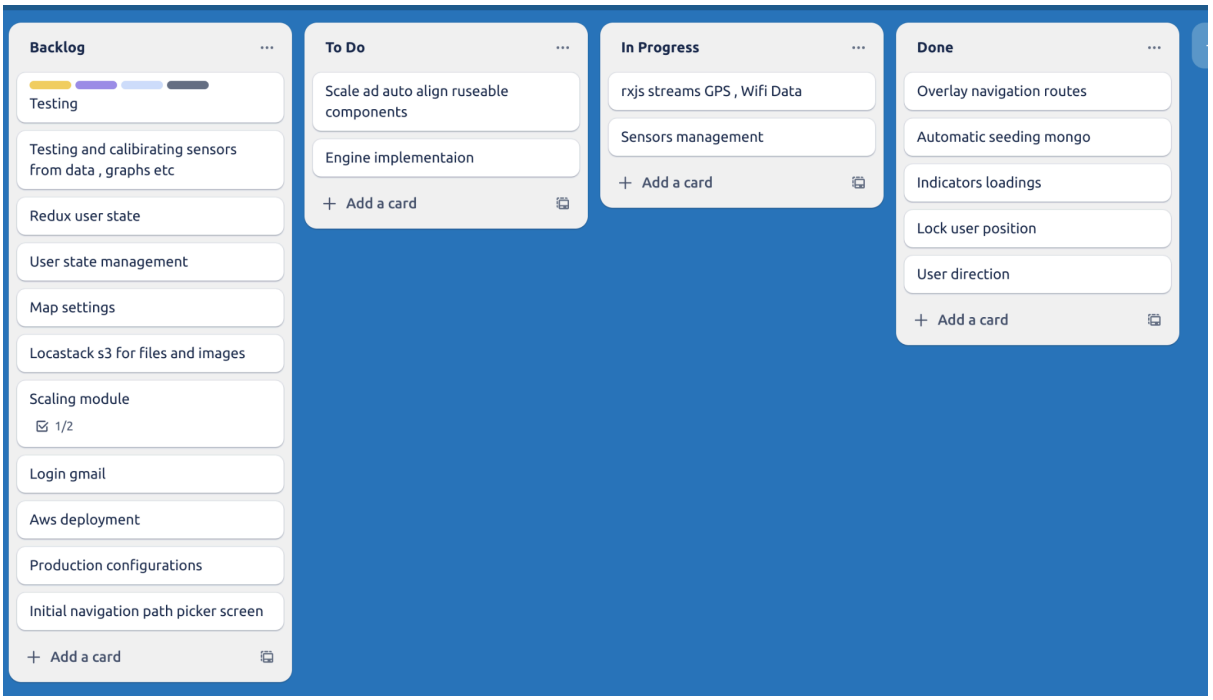
[Image Trello 19.8.Sprint 14.6-1.7]

- Finished websockets
- Finished all the basic functionality required.
- Working on the implementation of the navigation engine , in terms of math, adjusting modules , and testing algorithms.
- Fully complete the basic application
- First test routes with quaternion EKF:

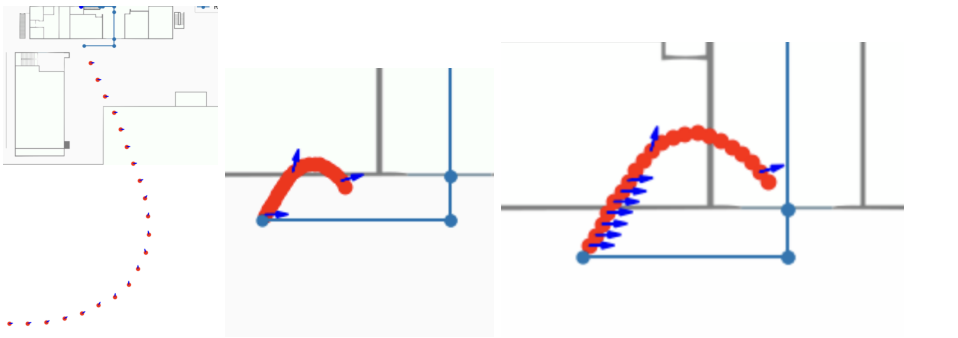


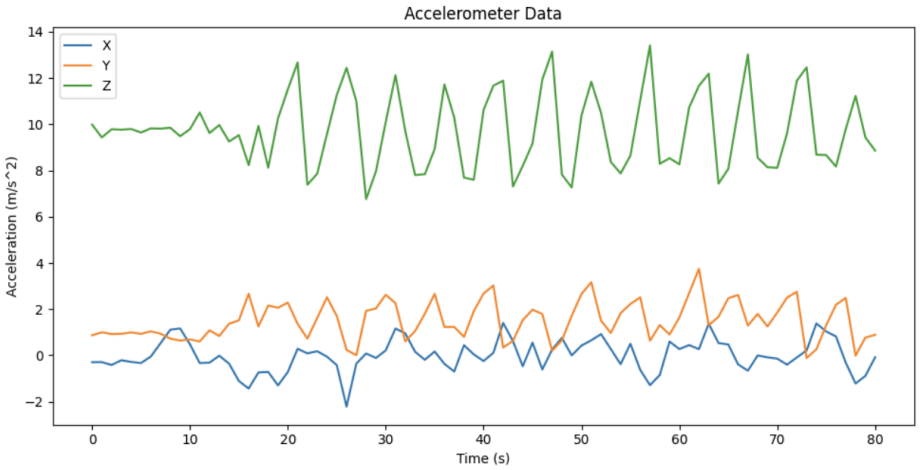
[Image 19.8.Example Path]

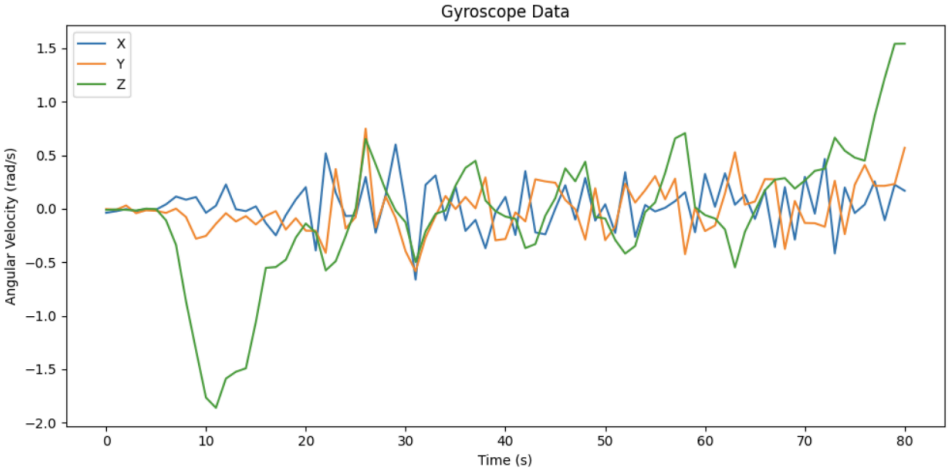
19.8.Sprint 1.7-14.7

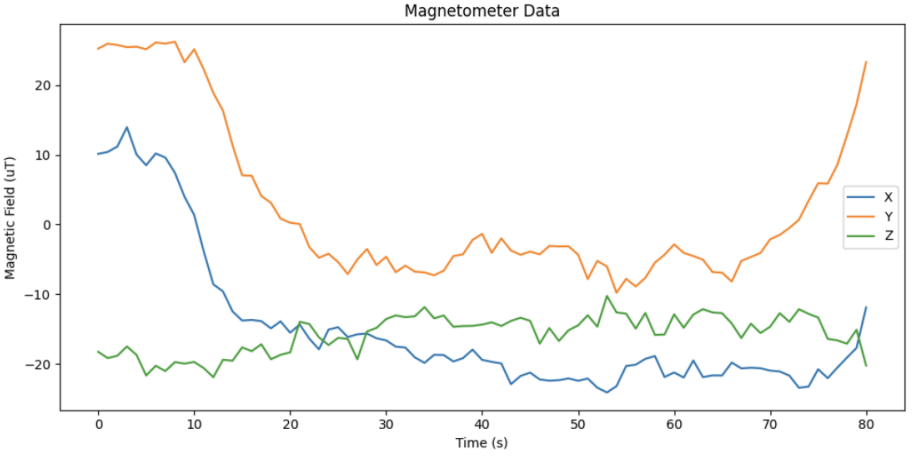


- Finish navigation routes
- FInished animations for user positioning
- Finished Auto Seeding
- Finished Positioning UI features
- Testing Routes:

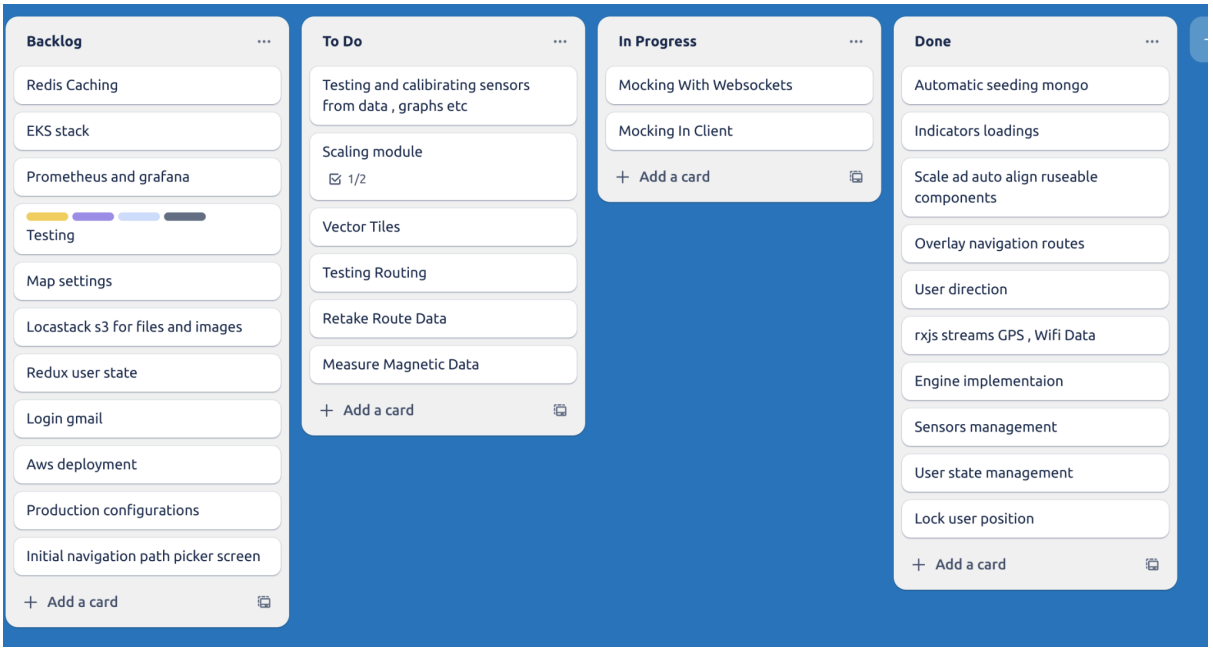



Accelerometer Data
X
Y
Z
Acceleration (m/s^2)
Time (s)


Gyroscope Data
X
Y
Z
Angular Velocity (rad/s)
Time (s)

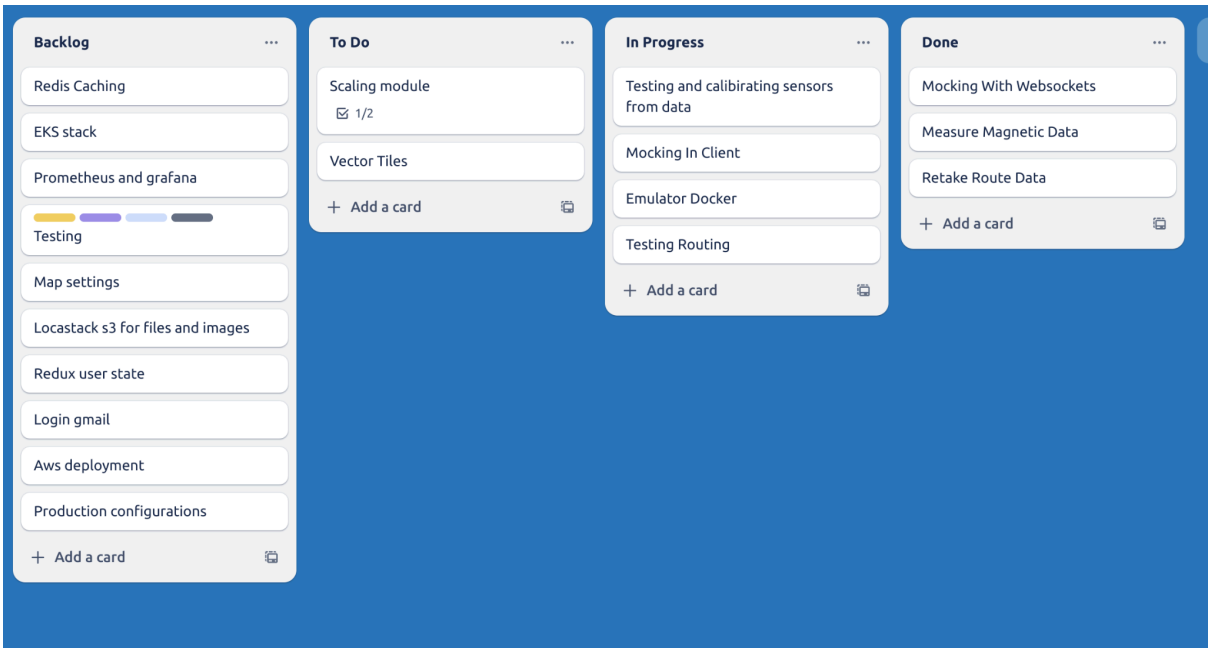

Magnetometer Data
X
Y
Z
Magnetic Field (uT)
Time (s)

19.9.Sprint 14.7 - 1.8:



- Finished sensors management
- Finished all RXJS streams
- Refactor a lot of the code for reusable component in the frontend
- Creating Mocking Since I need more time to implement the system .

19.10.Last Sprint 1.8 - 8.8:



- Creating mock route with websockets - realizing it not a good idea, since the websockets are not returning in consistent order , i have made the websockets stateful for the mocking implementation , unlike in real navigation where the websockets are stateless, therefore it cannot be used for perfectly mocking the positioning in real time, meaning the mocking needs to be done by the client.
- Retook Navigation Routes and Took partial area magnetic map.

20.Discussion

Our project focuses on developing a comprehensive indoor navigation system, which necessitates a strong reliance on research papers to ensure its functionality. Due to budget constraints ,the absence of pre-existing infrastructure and our desire for an affordable system without having pre-deploying infrastructure ready , we are limited to utilizing the available hardware, namely the mobile's inertial measurement units and WiFi. However, in isolation, these technologies may not perform optimally without the implementation of algorithms and techniques described in relevant research papers. Instead of reinventing existing solutions, we will leverage these papers to identify and implement the most cost-effective navigation features. Our approach involves testing different systems based on insights from the research papers, evaluating their performance and functionality, and ultimately building a seamless and efficient indoor navigation system. To minimize computational load and enhance our interaction with the mobile sensors, we will manually collect building data to construct a 3D model of the selected building. It is important to note that we will seek the building owner's permission for this data collection process.

As we delved deeper into our indoor navigation app's complexities, disagreements arose regarding navigation methods and documentation structure. To address this, we opted for a trust-based approach, dividing responsibilities between team members. One will oversee front-end decisions, while another will manage server-side matters.

This decision reflects our commitment to leveraging individual expertise and fostering collaboration. By doing so, we aim to streamline decision-making and ensure clarity in development efforts. With this division of responsibilities, we're poised to overcome challenges and deliver a top-notch navigation solution.

In the final prototype stage, despite exceeding our timeline due to the challenges outlined in the issues section, our project continues to progress steadily. We are committed to maintaining a focus on quality over quantity, and we anticipate that the application will excel upon completion. We remain enthusiastic about our journey and ongoing development efforts.

21.Summary and conclusions

The charter has helped us better understand the core aspects of navigation systems. By studying research papers and considering the system's architecture, we have gained valuable insights and identified areas where our knowledge is lacking.Despite the challenges posed by the complexity of the system and our constraints, we have developed a solid understanding of how our system will be built and the potential challenges we may face in the future.

Incorporating insights from the Engineering report, our understanding of navigation systems' core aspects has significantly deepened. Synthesizing findings from research papers and analyzing the system's architecture has honed our understanding and identified areas where further knowledge is needed. Despite the challenges posed by system complexity and constraints, we've developed a solid grasp of how our system will be constructed and the potential hurdles we may encounter down the road.

During the prototype stage, our team operated independently, and I proceeded as a solo effort, finding Trello sufficient without the need for additional tools like Jira. The challenges involved are immense, and I encourage others to attempt this endeavor, as it involves extensive mathematical and practical issues across various domains—from manual labor and unpredictability to hardware intricacies and API development for building management. Despite the complexities, this project has been an enriching experience that I thoroughly enjoy, and I am committed to its ongoing development. It's a highly flexible project where the possibilities are virtually limitless.

22.Future Developments

Alpha Tests: No tests were run yet.

4/7/2024:

- Continuing the localization until reaching a clear path (does not need to have good accuracy).
- Finishing UI:
    - Refractor the map display to include all the features I desired.
    - Styling the layout.
    - Dark mode.
- Adding user base, and settings with their preferences, adding gmail support.
- Application flow testing - that everything works as intended.
- Automations and testings
- Production ready environment , with automated deployment.
- take more data, and routes, and more accurate measurements
- Raising engine accuracy.
- Voice directions feature.
- Location share feature.
- Admin/Building manager API for managing the building data.
- better global map.
- Publish application to google app store.
- Vector tiles for the maps.

Acknowledgments

- Revital Marom for product guidance
- Yoram for organizing the team structure
- Stack overflow
- Google Scholar
- Github
- Myself

23.References

[10]. Perez-Navarro, A., R. Montoliu, and J. Torres-Sospedra. 2022. "**Advances in Indoor Positioning and Indoor Navigation**." *Sensors* 22 (19): 7375. doi:10.3390/s22197375. [Link]

[11]. Harder, D., H. Shoushtari, and H. Sternberg. 2022. "**Real-Time Map Matching with a Backtracking Particle Filter Using Geospatial Analysis.**" *Sensors* 22 (9): 3289. doi:10.3390/s22093289.[Link]

[12]. Ouyang, G., and K. Abed-Meraim. 2022. "**Analysis of Magnetic Field Measurements for Indoor Positioning**." *Sensors* 22 (11): 4014. doi:10.3390/s22114014. [Link]

[13]. Ren, P., F. Elyasi, and R. Manduchi. 2021. "**Smartphone-Based Inertial Odometry for Blind Walkers.**" *Sensors* 21 (12): 4033. doi:10.3390/s21124033. [Link]

[14]. Chen, L., J. Wu, and C. Yang. 2020. "**MeshMap: A Magnetic Field-Based Indoor Navigation System With Crowdsourcing Support**." *IEEE Access* 8: 39959-39970. doi:10.1109/ACCESS.2020.2974901. [Link]

[15]. "**Sensors and Sensing Technologies for Indoor Positioning and Indoor Navigation**." 2020. *Sensors* 20 (20): 5924. doi:10.3390/s20205924. [Link]

[16]. Liu, Z., B. Dai, X. Wan, and X. Li. 2019. "**Hybrid Wireless Fingerprint Indoor Localization Method Based on a Convolutional Neural Network.**" *Sensors* 19: 4597. doi:10.3390/s19204597. [Link]

[17]. Lee, M. S., H. Ju, and C. G. Park. 2017. "**Map Assisted PDR/Wi-Fi Fusion for Indoor Positioning Using Smartphone.**" *International Journal of Control, Automation, and Systems* 15 (2): 627-639. doi:10.1007/s12555-015-0342-2. [Link]

Additional references:

[1]. Chen, Jian, Gang Ou, Ao Peng, Lingxiang Zheng, and Jianghong Shi. 2018. "**An INS/WiFi Indoor Localization System Based on the Weighted Least Squares**" *Sensors* 18, no. 5: 1458. [Link]

[2]. Khedr, Maan, and Naser El-Sheimy. 2021. "**S-PDR: SBAUPT-Based Pedestrian Dead Reckoning Algorithm for Free-Moving Handheld Devices**" *Geomatics* 1, no. 2: 148-176. [Link]

[3]. Li, Zheng, Zongheng Wu, Wencheng Zhou, Shaolin Weng, and Huiru Zheng. "**A Smartphone Based Hand-Held Indoor Positioning System.**" Lecture notes in electrical engineering, January 1, 2016. [Link]

[4]. Li, You, Yuan Zhuang, Peng Zhang, Haiyu Lan, and Naser El-Sheimy. “**An Improved Inertial/Wifi/Magnetic Fusion Structure for Indoor Navigation**.” Information Fusion. Elsevier BV, March 1, 2017. [Link]

[5]. J. Á. B. Link, P. Smith, N. Viol and K. Wehrle, "**FootPath: Accurate map-based indoor navigation using smartphones,**" 2011 International Conference on Indoor Positioning and Indoor Navigation, Guimaraes, Portugal, 2011, pp. 1-8, [Link]

[6]. Retscher, Günther. 2023. "**Indoor Navigation—User Requirements, State-of-the-Art and Developments for Smartphone Localization**" Geomatics 3, no. 1: 1-46. [Link]

[7]. Kuang, Jian, Xiaoji Niu, Peng Zhang, and Xingeng Chen. 2018. "**Indoor Positioning Based on Pedestrian Dead Reckoning and Magnetic Field Matching for Smartphones**" Sensors 18, no. 12: 4142. [Link]

[8]. E. J. Alqahtani, F. H. Alshamrani, H. F. Syed and F. A. Alhaidari, "**Survey on Algorithms and Techniques for Indoor Navigation Systems,"** 2018 21st Saudi Computer Society National Computer Conference (NCC), Riyadh, Saudi Arabia, 2018, pp. 1-9, [Link]

[9]. IEEE Journals & Magazine | IEEE Xplore. “**A Fast Calibration Method for Triaxial Magnetometers,**” November 1, 2013. [Link]

24.Appendices

24.1.Glossary

Mathematics Definitions:

- $\|\cdot\|_2$ – L2(Euclidean)norm
- $Quaternion = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \left(q_w \in \mathrm{R}, \overrightarrow{(q_x, q_y, q_z)} \in R^3\right) = q_w + iq_x + jq_y + kq_z$
- $Attitude\ Angles\ Format - [roll \quad , pitch \quad , yaw]^T = [\varphi \quad , \vartheta \quad , \psi\ ]^T$
- $Gravity\ Vector = \vec{G} = [0, 0, -g]$
- **Quaternion Multiplication - ⊗:**
  - $a = [a_w, a_x, a_y, a_z], b = [b_w, b_x, b_y, b_z], r = a \cdot b = [r_w, r_x, r_y, r_z]$
  - $r_w = a_w b_w - a_x b_x - a_y b_y - a_z b_z$
  - $r_x = a_w b_x + a_x b_w + a_y b_z - a_z b_y$
  - $r_x = a_w b_y - a_x b_z + a_y b_w + a_z b_x$
  - $r_x = a_w b_z + a_x b_y - a_y b_x + a_z b_w$
- **Calculating rotation matrix from quaternion:**
  - $Rotation\ Matrix = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_w q_y + q_x q_z) \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_w q_x + q_y q_z) & 1 - 2(q_x^2 + q_y^2) \end{bmatrix}$
- **Rotating Vector with Rotation Quaternion**:
  - $conjugate$ of q: $q^* = (q_w, -q_x, -q_y, -q_z)$
  - The inverse of quaternion q: $q^{-1} = \frac{q^*}{|q|^2}$
  - For rotation quaternion q: $q^{-1} = q^* = (q_w, -q_x, -q_y, -q_z)$
  - Quaternion rotation requires two quaternion multiplications.
    - $\vec{V} = q \otimes \vec{V} \otimes q^{-1}$
- **Quaternion to Euler's angles:**
  - Atan2 = calculating an angle that can be between -180 and 180 degrees.

Yaw/Azimuth (around the z-axis) – represent the horizontal rotation (heading direction):

$$\psi = atan2\left(\frac{2(q_x q_y + q_w q_z)}{q_w^2 + q_x^2 - q_y^2 - q_z^2}\right)$$

Pitch (around the y-axis) - represents the vertical or elevation rotation:

$$\vartheta = \mathrm{asin}\left(2(q_w q_y - q_x q_z)\right)$$

Roll (around the x-axis) - represents the tilt:

$$\varphi = \arctan2\left(\frac{2(q_w q_x + q_y q_z)}{q_w^2 - q_x^2 - q_y^2 + q_z^2}\right)$$

- AHRS – attitude and heading reference system
- KF – kalman filter
- EKF – extended kalman filter
- MM – Magnetic Matching.
- RP – Reference Point.
- INS - Inertial Navigation System.
- DCS – Device Coordinates System.
- BCS – Body Coordinates System.
- SCS – Site Coordinates System.
- Strapdown system – a system which the smartphone has a static position relative to the user's position.
- ECEF – Earth-Centered, Earth Fixed Coordinates System
- LRCS – Local Reference Coordinate System.
- BCS – Building Coordinates System.
- AP – Access Point.
- GC – Gait Cycle
- ZUPT – zero velocity update.
- $Accelerometer\ Data = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$
- $Gyoscope\ Data = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$
- $Magnetometer\ Data = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix}$
- $Gravity\ Data = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}$

- **Gyroscope readings at time t:**
  - $\overrightarrow{\omega_t} = [\omega_{xt} \quad , \omega_{yt} \quad , \omega_{zt}]$
  - $\|\overrightarrow{\omega_t}\|_2 = \sqrt[2]{\omega_{xt}^2 + \omega_{yt}^2 + \omega_{zt}^2}$
  - $Gyroscope\ Interval = T_{\omega s}$
- **Acelerometer readings at time t:**
  - $\overrightarrow{a_t} = [a_{xt} \quad , a_{yt} \quad , a_{zt}]$
  - $\|\overrightarrow{a_t}\|_2 = \sqrt[2]{a_{xt}^2 + a_{yt}^2 + a_{zt}^2}$
  - Acelerometer $Interval = T_{as}$
- **Gravity readings at time t:**
  - $\overrightarrow{g_t} = [g_{xt} \quad , g_{yt} \quad , g_{zt}]$
  - $\|\overrightarrow{g_t}\|_2 = \sqrt[2]{g_{xt}^2 + g_{yt}^2 + g_{zt}^2}$
  - Gravity $Interval = T_{gs}$
- **Magnetometer readings at time t:**
  - $\overrightarrow{m_t} = [m_{xt} \quad , m_{yt} \quad , m_{zt}]$
  - $\|\overrightarrow{m_t}\|_2 = \sqrt[2]{m_{xt}^2 + m_{yt}^2 + m_{zt}^2}$
  - Magnetometer $Interval = T_{ms}$
- ***Angle Increment*:**
  - $Gyroscope\ Interval = T_s$

$$\Delta\theta_{xt} = \omega_{xt} \cdot T_s$$

$$\Delta\theta_{yt} = \omega_{yt} \cdot T_s$$

$$\Delta\theta_{zt} = \omega_{zt} \cdot T_s$$

$$\overrightarrow{\Delta\theta_t} = [\Delta\theta_{xt} \quad , \Delta\theta_{yt} \quad , \Delta\theta_{zt}]$$

$$\left\|\overrightarrow{\Delta\theta_t}\right\|_2 = \|\overrightarrow{\omega_t}\|_2 \cdot T_s$$

- **Updating Quaternion:**
  - After getting the new updated quaternion from sensor fusion:
  - $0.5 \cdot q \otimes \theta$