Sharon Mor | Yuval Badihi

Advisor – Dr. Naomi Unkelos Spiegel

First Response Team Application

Table of Contents

ABSTRACT

This book proposes creating an advanced web and mobile app to optimize communication and operational efficiency for first response teams, especially in high-risk and recently impacted areas. The proposed tool supports real-time strategic decision-making and comprehensive emergency management. The book details the technical design, diagrams, and research underpinning this tool, including system architecture frameworks. It incorporates insights from interviews with first responders to shape features and functionality. Additionally, it examines the app's potential as a model for leveraging technology to elevate emergency management practices, aiming to significantly enhance responsiveness and effectiveness.

INTRODUCTION

Today's world requires robust emergency management systems, especially in crisis-prone areas. Our project aims to develop an innovative application to enhance communication and decision-making for first response and resilience teams during emergencies.

The application will provide a centralized platform for coordination, with features like alarm creation, task assignment, real-time location tracking, hazard marking systems, and comprehensive task management tools. Built using modern web and mobile technologies like React and React Native, it will differentiate between response and recovery phases, archive incident data, and draw upon established emergency response principles.

This tool promises to revolutionize operations by improving team preparedness, equipping responders with vital capabilities, and overcoming communication barriers in chaotic situations. Key features include centralized coordination platform, alarm creation and task assignment, real-time location tracking, hazard marking systems, differentiation of response/recovery phases, archiving of incident data, and being grounded in proven emergency response practices.

The application's development is driven by the urgent need highlighted during critical incidents like the events of 7.10, which exposed shortcomings in emergency coordination. Extensive dialogue with responders in high-risk areas like the Golan settlements revealed their existing tools are insufficient for emergencies requiring speed, precision, and reliability.

By integrating advanced features into an accessible interface, this application aims to simplify complexities, boost team confidence, and transform how first responders operate during crises. The goal is to create an essential partner that substantially impacts response efficiency and ultimately saves lives.

DEFINITIONS

**First Response Team (FRT):** Group of trained individuals, including professionals and volunteers, who provide immediate assistance during emergency situations. They act as the first line of defense, delivering urgent aid and coordination until additional support arrives.[5]

**Community Emergency Response Teams (CERT):** Trained civilian volunteers who assist professional responders during emergencies across sectors like health, education, and logistics. The CERT program teaches basic skills like fire safety, light rescue, first aid, and volunteer management. CERT members provide immediate support in these areas until responders arrive, enhancing a community's resilience and response capabilities in crises.[6]

|  |  |  |  |
| --- | --- | --- | --- |
| **Avuka Squad** | **Zello** | **AVIA** | **Feature** |
|  | ✔ | ✔ | IOS Support |
| ✔ | ✔ | ✔ | Android Support |
|  | ✔ | ✔ | Web/PC Support |
| ✔ |  | ✔ | Management App |
|  | ✔ | ✔ | Resident's App |
|  |  | ✔ | Wellness Check |
|  |  |  | Automatic Dialing Emergency Services |
| ✔ |  | ✔ | Google Maps |
| ✔ |  | ✔ | Dynamic Map |
| ✔ |  | ✔ | GPS Tracking |
| ✔ |  |  | Sectors In Map |
| ✔ |  |  | Freehand Marking |
| ✔ |  |  | Navigation Option |
|  |  |  | 3D Map |
| ✔ | ✔ |  | Push-To-Talk |
| ✔ | ✔ | ✔ | Chat |
| ✔ | ✔ | ✔ | SOS Alarm |
| ✔ | ✔ | ✔ | Alert For FRT Activation |
| ✔ |  | ✔ | Predefining Possible Emergency Events |
| ✔ |  |  | Execute Exercise |
| ✔ | ✔ | ✔ | Defining Groups |
| ✔ | ✔ | ✔ | Operation Logs |
| ✔ |  | ✔ | Message Board |
|  |  | ✔ | Forces Count |
|  |  |  | Performance Monitoring |
|  |  |  | Integration With Cameras |
|  |  |  | Smooth Registration |

EXISTING TOOLS

|  |  |
| --- | --- |
| Home | Avuka-Squad System | **Avuka Squad:** System suite was designed and developed to supply a comprehensive location based C2 [1] (Command and Control) solution for organizations with complex security needs. [7] |
| לנצח את האירוע עם טכנולוגיה מתקדמת - AVIA SECURITY | **AVIA:** System operates as an emergency population management system that enables control over complex security and civilian events.[8] |
| ‪Zello PTT Walkie Talkie – At the ECOM App Library | Get it here‬‏ | **Zello:** Zello is a modern and customizable walkie-talkie app (and more) that lives on any smart device.[9] |

*[1] https://intelligence.airbus.com/industries/defence/c2*

The existing tools like Avuka Squad, AVIA, and Zello provide some features for first response teams, but lack a comprehensive, integrated solution that meets all their diverse needs.

To address this gap, the proposed First Response Team application aims to develop a unified platform that combines vital features like communication tools, mapping, task assignment, and real-time coordination in a user-friendly interface.

The key differentiators are:

* Simplicity for use in high-stress situations.
* Features tailored based on extensive interviews with first responders.
* Leveraging modern web and mobile technologies for scalability and accessibility.
* Advanced data handling, search capabilities, and secure authentication.

By offering a complete toolset streamlining communication, situational awareness, and decision-making, this application promises to revolutionize emergency management practices.

**Application - גם חלק מהסקירה  
לשלב רפרנסים**

**Web Application and Mobile Application Development**

In today’s technological landscape, effective and intuitive applications are crucial for the efficient management of first response teams. Our system includes both web and mobile applications, tailored to meet the diverse needs of the team. By integrating these platforms, we leverage the unique advantages of each to enhance overall system functionality.

The web application provides a stable and scalable platform suitable for complex data management and operational oversight. Conversely, the mobile application offers unparalleled convenience, ensuring that first responders stay connected and well-informed even during active operations.

Together, these applications support enhanced communication, seamless data sharing, and more effective emergency management, boosting the effectiveness of the response team.  
  
**Web Application Development**

Web applications are becoming more popular due to their ease of use and maintenance. These applications are stored on cloud or dedicated servers, ensuring they are always available with a very low risk of data loss. Anyone with a device that has a web browser can access these applications, which is especially useful for systems that need quick updates or immediate access to data, like first response team control systems.

Developing a web application is generally quick, allowing teams to have a working product ready in a short amount of time. The availability of numerous libraries and third-party tools today eases the development process, enhancing both productivity and efficiency. Moreover, web applications facilitate effective teamwork by enabling easy collaboration and swift updates, which are essential in emergency response scenarios where coordination and timely action are crucial.

However, there are challenges:

Performance: Web applications depend on the browser's performance. Large-scale applications can be slow and may experience downtime or crashes.

Security: The reliance on Internet connectivity increases vulnerability to security breaches. Leveraging established technologies with proven security protocols helps mitigate these risks significantly. Despite these precautions, the threat of hacking and data theft remains a concern, underscoring the need for continuous security enhancements and monitoring.

**Mobile Application Development**

Mobile applications offer distinct advantages for a first response team application, primarily through enhanced accessibility and user engagement. Mobile apps can provide tailored experiences optimized for handheld devices, crucial for first responders who rely on quick, easy-to-navigate interfaces in emergency situations.

The development of mobile applications allows for:

Offline Access: Unlike web applications, mobile apps can offer limited functionality even without an Internet connection, crucial during emergencies in remote or network-compromised areas.

Notifications: Mobile apps can send instant notifications to users, which is vital for updating first responders about urgent situations or new information.

Enhanced Security: Mobile platforms often incorporate robust security measures that protect sensitive data, crucial for an application handling critical first response information.

Like their web counterparts, mobile applications face certain drawbacks:

Compatibility: Developers must ensure the application runs smoothly across various devices and operating systems, which can complicate development.

Maintenance: Mobile applications require regular updates to address system vulnerabilities and compatibility issues, which can increase operational burdens.

In conclusion, integrating both web and mobile applications for controlling first response teams ensures maximum coverage and functionality. Each platform complements the other by addressing specific needs and operational scenarios, thereby providing a comprehensive toolset for effective emergency management.

**Engineering Process**

We selected a microservices architecture after a comprehensive analysis and design process. By thoroughly investigating the problem, reviewing relevant trends, conducting interviews, and analyzing collected data, we gained insights into the key requirements for our first response and emergency settlement resilience application. Creating UML class diagrams further solidified our understanding of the system's components and their interactions. This structured approach highlighted the need for a flexible and scalable architecture that could accommodate the identified modules while enabling independent development, testing, and deployment. A microservices architecture perfectly aligns with these goals, allowing us to break down the system into smaller, decoupled services that can be developed, tested, and scaled independently. By adopting microservices, we can efficiently implement the requirements, establish project expectations, and develop a comprehensive testing strategy tailored to each service. This architectural choice directly emerged from our systematic analysis and design process, ensuring our application meets the identified requirements while providing the scalability and maintainability crucial for first response and emergency settlement operations. Additionally, our team had prior experience working with microservices architectures, as well as the associated front-end technologies like React and React Native, which further influenced our decision.

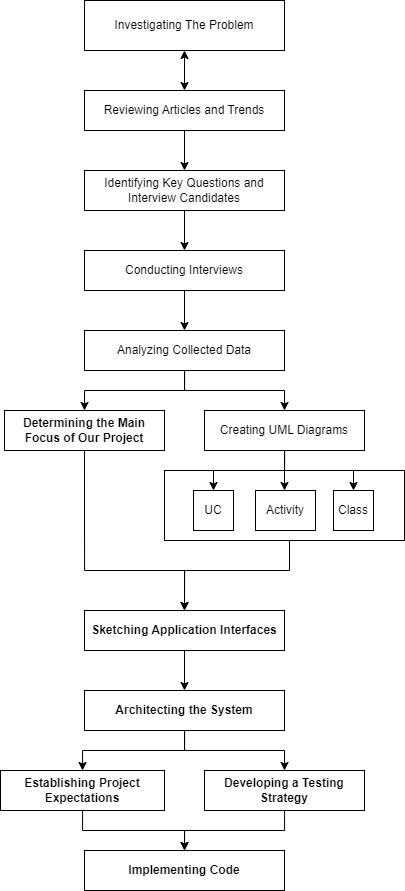


Fig x – Progress Diagrams

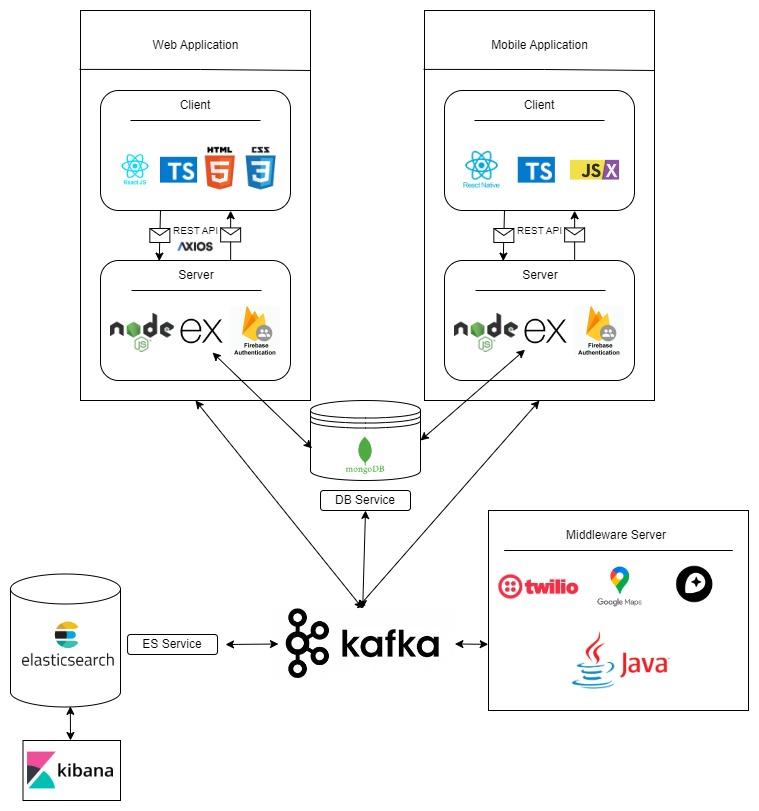


Fig x – Architecture Diagram

**System Architecture**

Our system architecture is designed to ensure seamless interaction between web and mobile platforms, utilizing a robust suite of technologies to deliver secure, efficient, and responsive applications. Central to our approach is the use of modern frameworks and tools across both front and backend implementations, supported by advanced data handling and authentication systems. This architecture not only enhances user experience but also maintains system integrity and scalability as demands grow. Below, we outline the specific technologies and configurations employed in each segment of our system:

**Web Application Architecture**

The web application architecture is built using React and Node.js, leveraging Firebase Auth for authentication and MongoDB for data storage. Here's how each component functions within the system:

* **React:** Handle the user interface (UI) and user interactions. React's efficient update and rendering system make it ideal for responsive web applications.
* **Node.js:** Manages server-side logic and handles API requests and responses. It serves as the central communication point with various databases, ensuring efficient data management and scalability.
* **Express.js:** Utilized as the web application framework running on the Node.js server, Express.js simplifies the routing and middleware architecture, enhancing API development and server management for mobile applications.
* **Firebase Auth:** Manages user authentication and session management, ensuring secure access to the web application.

**Mobile Application Architecture**

The mobile application leverages the synergy of React Native and a dedicated Node.js backend, incorporating Firebase Auth for secure authentication and MongoDB for robust data storage. Here is an overview of each component's role within the mobile ecosystem:

* **React Native:** Takes charge of the mobile user interface (UI) and user interactions. Its use of native components ensures that the mobile application delivers a responsive and fluid experience akin to native applications.
* **Node.js:** Manages server-side logic distinctively for the mobile context. It processes API requests and responses specific to the mobile application, acting as a liaison for mobile-specific data flows and services.
* **Express.js:** As the web application framework running on the Node.js server, Express.js is crucial for handling the RESTful API endpoints that the mobile app utilizes to communicate. It is tailored to manage mobile session states and optimize API responses for low-latency communication essential in mobile environments.
* Firebase Auth: Handles secure user authentication for the mobile application. It ensures that user sessions are managed securely, facilitating reliable access to mobile-specific features and services.

**Data Storage and Search Technologies**

Before detailing the middleware server, it is important to outline the key data storage and search technologies employed:

* **MongoDB:** Serves as the primary database, storing application data in a NoSQL format which allows for flexible, schema-less data structures, making it ideal for applications that require quick iterations and adaptations.
* **Elasticsearch:** Provides powerful search and analytics capabilities with near real-time indexing, greatly enhancing the application’s ability to perform complex searches, data visualization, and real-time analytics.

**Middleware Server Architecture**

To enhance performance and scalability, a middleware server written in Java or a similar robust language is incorporated, serving as an advanced processing layer that interacts with various databases and services:

* **Middleware Server:** Acts as a robust conduit between the web and mobile backends and the databases. It manages more complex queries and operations that are beyond the scope of regular backend functions.
* **Database Communication:** The middleware server uses MongoDB for sophisticated data manipulations and retrievals, efficiently managing large-scale data operations.

Elasticsearch Integration: The middleware server also integrates with Elasticsearch to harness its advanced search capabilities, providing quick data retrieval and real-time analytics.

Additional Functionalities: Can also integrate other third-party services and APIs, providing a flexible and scalable environment to meet growing application demands.

**Conclusion**

This comprehensive system architecture supports robust scalability, advanced security, and high efficiency, making it well-suited for demanding environments. By utilizing React and React Native for frontend development and Node.js for backend operations, the architecture ensures a unified development approach and optimal resource utilization across both web and mobile platforms. The addition of MongoDB and Elasticsearch enhances the system's data handling and search capabilities, while the middleware server ensures complex operations are managed effectively, keeping the application scalable and responsive as user demands increase.

**Client-side technologies:**

React and React Native are two powerful frameworks developed by Facebook for building user interfaces. While React is primarily used for web development, React Native extends its capabilities to mobile app development. Both frameworks share many similarities, such as their language and syntax, but they also have distinct features tailored to their respective platforms. In this comparison, we'll explore the key differences between React for web and React Native for mobile, focusing on various aspects such as performance, testing, user interface, scalability, and community support.

|  |  |  |
| --- | --- | --- |
| **Feature** | **React (Web)** | **React Native (Mobile)** |
| **Language and Syntax** | JavaScript and JSX for creating UI components with the ease of HTML-like syntax embedded in JavaScript. | JavaScript and JSX, translating to native platform components for a seamless mobile experience. |
| **Performance** | Uses Virtual DOM for efficient updates. Calculates differences and re-renders only the changed parts on the real DOM. | Employs Virtual DOM and native views for a performance-optimized mobile experience. |
| **Testing** | Jest for testing framework. Enzyme and React Testing Library for testing components in isolation. | Jest for testing framework. React Native Testing Library for mobile-specific testing, including native behaviours. |
| **User Interface** | Component-based, declarative views making the code predictable and debuggable. | Component-based, declarative views with native platform UI elements integration. |
| **Web Support** | Cross-browser compatibility and SEO-friendly features. | Cross-platform compatibility with access to native device features like camera and storage. |
| **Scalability** | Designed for large-scale web applications, it can handle complex architectures and traffic volumes. | Suitable for large-scale mobile applications with the ability to integrate with native modules for enhanced performance. |
| **Community** | Large, active developer community with a rich ecosystem of libraries and tools. | Rapidly growing community with increasing contributions and an evolving ecosystem. |
| **Accessibility** | ARIA (Accessible Rich Internet Applications) standards support for web accessibility. | ARIA support plus native mobile accessibility features like VoiceOver and Talkback for an inclusive user experience. |

During our research about the client-side possible tools we found out that for our use case of building an application both for web and mobile, react and react native a good solution [4].

**Server-side technologies:**

Our application architecture is designed to effectively serve both web and mobile clients through a robust backend that scales and adapts to the needs of both platforms.

**Node.js:**Node.js is a versatile platform for building a variety of server-side and networking applications. Renowned for its non-blocking, event-driven architecture, it’s particularly adept at managing multiple simultaneous connections with high throughput, which makes it a suitable choice for high-load applications. Node.js brings the familiarity of JavaScript to the server-side, allowing developers to use a single programming language across both front-end and back-end.

**Express.js:**Built on top of Node.js, Express.js is a minimalist and flexible web application framework that provides a robust set of features to develop web and mobile applications efficiently. It simplifies the server creation process with its middleware framework, making it easy to perform tasks like parsing request bodies, managing cookies, and implementing sessions. Express.js is designed to be a lightweight framework that gives you the tools to create a server quickly while also being robust enough to manage web application routes, handle requests, and serve responses. Its ability to create a REST API makes it a powerful tool for backend development, facilitating seamless front-end to back-end integration.

**MongoDB:**MongoDB is an open-source NoSQL database that leverages a flexible document model for storing data. It’s well-suited for applications that need quick iterations and can manage diverse sets of data types. MongoDB’s schema-less nature allows it to handle large volumes of disparate data, which can be particularly useful for applications that require agility in data management and design.

**Elasticsearch:**  
Elasticsearch is a powerful open-source search and analytics engine that allows applications to perform complex searches, real-time data analysis, and visualizations. It enhances our application by enabling fast and scalable search capabilities, crucial for handling large datasets and providing real-time responses to user queries.

**Firebase Authentication:**Firebase Authentication provides a complete identity solution, supporting a variety of authentication methods, including social media accounts, phone numbers, and email/password. It integrates with Firebase's real-time database, which synchronizes data across all clients in real-time. This combination allows for the implementation of secure, authenticated user access to your application's data. Firebase's real-time database uses JSON documents and offers offline data synchronization and retrieval, enhancing the user experience even when connectivity is intermittent.

In summary, the combination of Node.js, Express.js, MongoDB, and Firebase Authentication forms a comprehensive server-side stack that facilitates the development of modern, scalable, and real-time web applications. Each component plays a crucial role in the system: Node.js serves as the runtime environment, Express.js adds structure and features for web application development, MongoDB provides a flexible data storage solution, and Firebase Authentication handles secure user management and real-time data synchronization.

**3) Workflow**

**3.2 Interviews**

To gather accurate and actionable data for the development of our response and resilience team management application, engaging directly with experienced individuals in the field was essential. Interviews with key stakeholders from various settlements provided invaluable insights into the unique challenges and requirements faced by first response and residential resilience teams. Through these discussions, we were able to understand the complexities of their operations and identify specific features and functionalities that our application must include to effectively support their critical work. **Here's a summary of the interviews and the key insights we gained from them:**

**1) Muli Spiegel** - Architect, Director of Engineering and Planning Department, the settlements in the Golan.  
Background: Muli oversees logistics for his settlement's first response team, which is well-organized into specialized units.  
Key Insights: The need for a system that effectively delineates and connects various operational teams, both geographically and functionally, while facilitating proper communication among them.

**2) Roee Tavor** - Chairman of the Residential Resilience Team in Kanaf settlement.  
Background: Kanaf operates with a residential resilience team collaborating closely with the first response team.  
Key Insights: The need for a user-friendly interface that integrates all aspects of routine and emergency operations, facilitates communication among teams, and works even with limited internet connectivity and GPS availability.

**3) Hagit Geva & Itamar Cohen** - Chairman & Vice Chairman of the Residential Resilience Team in Givat Yoav settlement.  
Background: Givat Yoav has several resilience teams, including logistics, health, education, spokesperson, welfare, and first response.  
Key Insights: The importance of a strict hierarchy with appropriate permissions, a simplified mobile app for general team members, and a more complex web application for leaders. The need for a feature to mark significant locations on a map and assign tasks across teams.

**Based on these interviews, we learned the following:**

1. GPS connections cannot be relied upon during emergencies, so the application should function without GPS.

2. Push-to-talk functionality is not required, as the teams already have walkie-talkies for communication.

3. User experience (UX) and ease of use are crucial for the successful adoption of the application by the teams.

4. The application should cater to different roles and permissions within the team hierarchy.

5. Marking significant locations on a map and assigning tasks across teams are essential features.

6. The application should work seamlessly during both routine operations and emergencies, with a simplified interface for emergencies.

**3.3 FR & NFR Requirements:**

**Functional Requirements Document:**

|  |  |
| --- | --- |
| **No.** | **Requirement** |
| 1 | The System shall provide communication Tools. |
| 1.1 | The System shall support alerts |
| 1.2 | The System shall support alarms |
| 1.3 | The System shall support chat. |
| 2 | The System shall support voice commands. |
| 3 | The System shall include a map. |
| 3.1 | The System shall support marking on a map. |
| 3.2 | The System shall support defining map sectors. |
| 3.3 | The System shall support offline maps. |
| 4 | The System shall Monitor Performance. |
| 4.1 | The System shall record action logs. |
| 5. | The System shall Analyze Performance. |
| 5.1 | The System shall identify trends and bottlenecks. |
| 6. | The System shall ensure Web Compatibility. |
| 7 | The System shall ensure Mobile Compatibility. |
| 8 | The System shall include Management Interface. (BO) |
| 9 | The System shall support different user roles. |
| 9.1 | The System shall support adding new roles. |
| 9.2 | The System shall support setting roles. |
| 10 | The System shall include a Wellness check. |
| 11 | The System shall support Events. |
| 11.1 | The System shall support adding events. |
| 11.2 | The System shall support removing events. |
| 11.3 | The System shall support joining events. |
| 12 | The System shall support authentication. |
| 13 | The System shall support task management. |
| 13.1 | The System shall allow task creation. |
| 13.2 | The System shall allow task assignment. |
| 13.3 | The System shall allow task tracking. |
| 14 | The System shall provide data synchronization. |
| 14.1 | The System shall synchronize data across devices in real-time. |
| 15 | The System shall support customizable dashboards. |
| 15.1 | The System shall allow users to customize views. |
| 16 | The System shall provide secure data storage. |
| 16.1 | The System shall encrypt sensitive data. |

**Non-functional Requirements:**

|  |  |  |
| --- | --- | --- |
| **No.** | **Requirement** | **Type** |
| 1 | The Application should be quick and easy to use. | Performance |
| 1.1 | The interface shall require no more than 2 clicks or taps from the home screen to initiate any form of communication. | Usability |
| 1.2 | The application shall load its main content and be ready for user interaction within 3 seconds of launch. | Performance |
| 1.3 | The system must support up to 10,000 concurrent users without degradation of performance. | Scalability |
| 2 | The System shall ensure all chats, recordings and locations are encrypted. | Security (cyber and physical) |
| 3 | The System shall use mapping API from one of the providers:  Google Maps, Leaflet, Mapbox, OpenLayers. | Compatibility |
| 3.1 | Marking on a map should be by selecting the option after right clicking (pc) or clicking (on mobile). | Usability |
| 3.2 | The System shall provide a 'Free Draw' feature enabling users to define map sectors by manually drawing the sector boundaries directly on the map interface. | Usability and Flexibility |
| 4 | The System shall monitor the performance of the follows:  Time took between important actions, Number of actions until important action. | Usability and Performance |
| 5 | The System should analyze application by recording data and showing it on a graph on the manager side | Usability and Performance |
| 6 | The System should record and store the bi events on Google Analytics 4. | Auditability and Control |
| 7 | The Application will allow the leaders of each circle to manage the different roles by names and permissions. | Usability and Flexibility |
| 8 | Alerts should be one of the following: Real Alarm, Test Alarm, Recorded Voice, Wellness Alarm | Usability and Flexibility |
| 9 | The Wellness Check will be by alarm and confirmed by each user and will calculate the time it took for everyone to confirm. | Performance and Response time |
| 12 | The Application will allow managing events and tasks by setting their name, description, participants and date. | Usability and Modifiability |
| 13 | The Web Application will be consisted of the next pages:  1. Login Page 2. Sign Up Page 3. Contact Us 4. Request to join Page 5.Users Page 6. Map Page 7. Chat Page 8. Alert Page 9. Logs Page 10. BO Page 11. Files Page 12. Events Page 13. Tasks Page | Usability and Scalability |
| 14 | The Mobile Application will be consisted of the next pages: 1. Login Page 2. Sign Up Page 3. Contact Us 5. Users Page 6. Map Page 7. Chat Page 8. Alert Page 9. Tasks status page | Usability and Scalability |
| 15 | The Application is going to be developed both for web and for mobile. | Platform compatibility and Interoperability |
| 15.1 | Web Tech Stack: 1. ReactJS (Front Library) 2. CSS / TailwindCSS 3. TS (Superset of JS) 4. NodeJS (JS Runtime) 5. ExpressJS (Server) 6. MongoDB / FirebaseDB (DB) 7. Firebase Authenticator (Auth) 8. Axios (HTTP Calls) 9. Jest (testing) 10. Render or AWS (hosting) | Compatibility |
| 15.2 | Mobile Tech Stack: 1. ReactJS (Front Library) 2. CSS / TailwindCSS 3. TS (Superset of JS) 4. NodeJS (JS Runtime) 5. ExpressJS (Server) 6. MongoDB / FirebaseDB (DB) 7. Firebase Authenticator (Auth) 8. Axios (HTTP Calls) 9. Jest (testing) 10. android / app store (hosting) | Compatibility |

**3.4 Use Case Diagram**

Unregistered User: This is an anonymous individual who is not recognized by our database. They have not registered or authenticated with our system.

Registered User: This user is recognized and authenticated by our system, known through their verified login credentials.

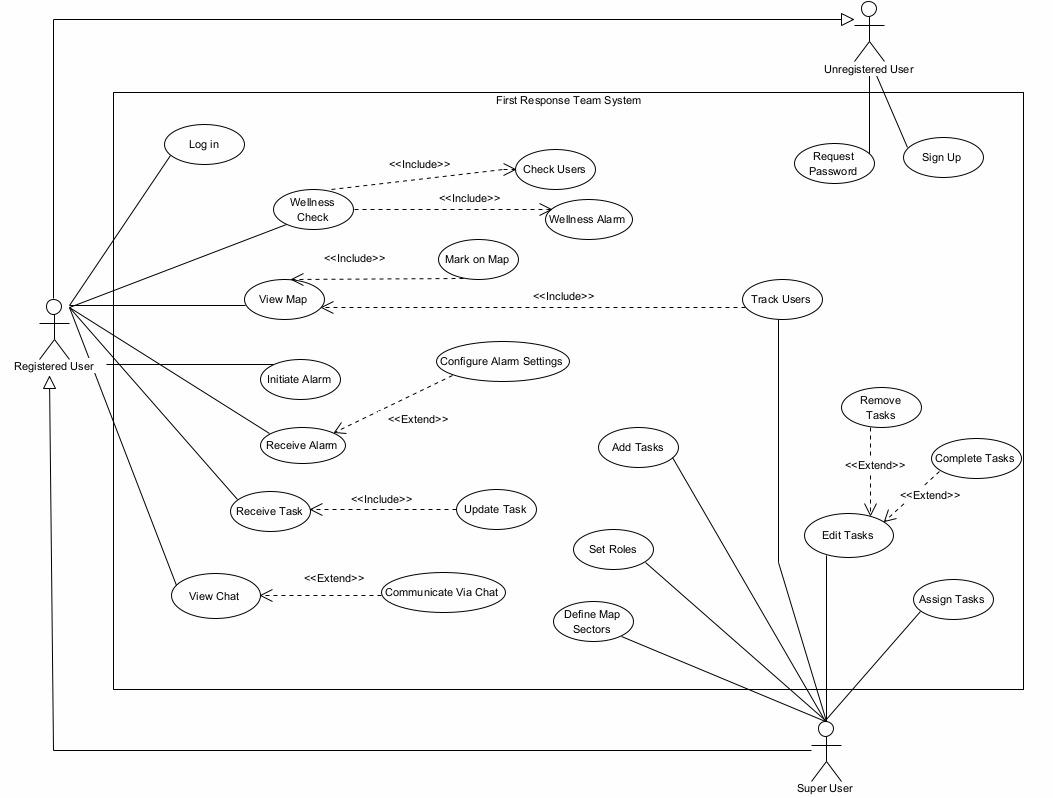
Super User: A user with administrative privileges who manages at least one registered user, overseeing their activities and permissions within the system.

Fig 1: Use Case Diagram

**3.5 Activity Diagram**

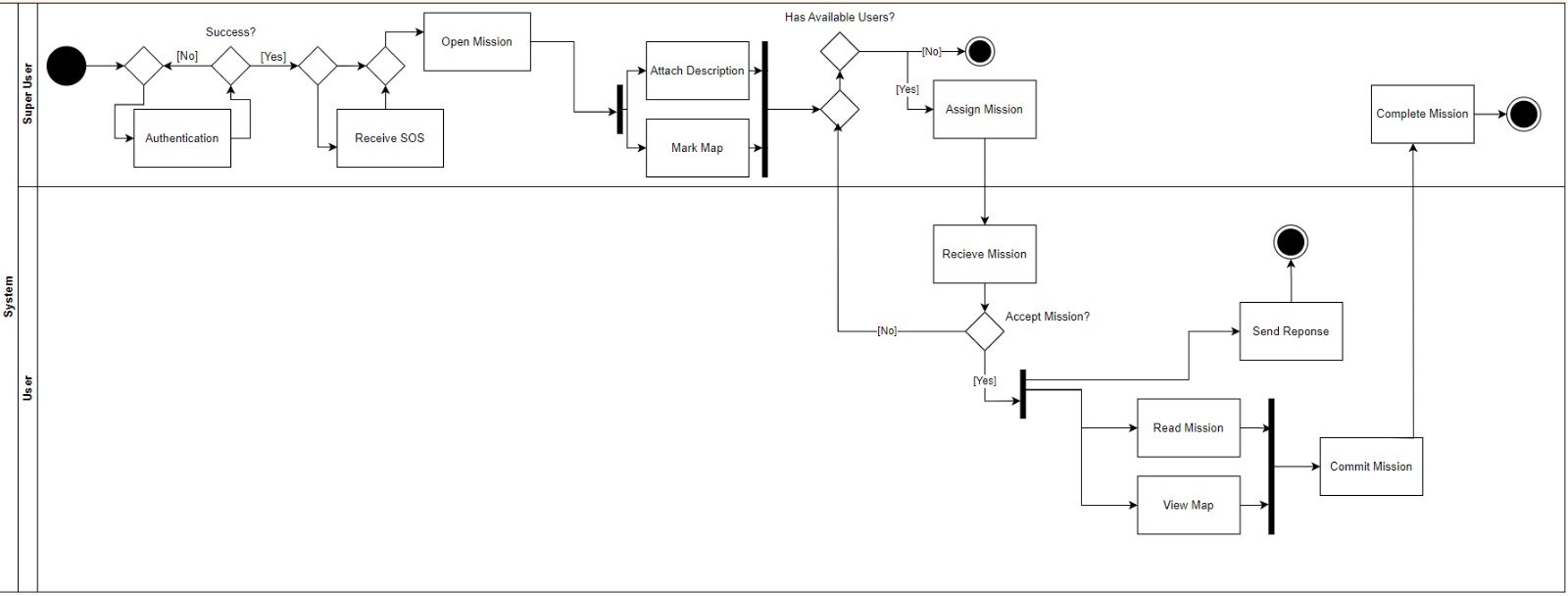
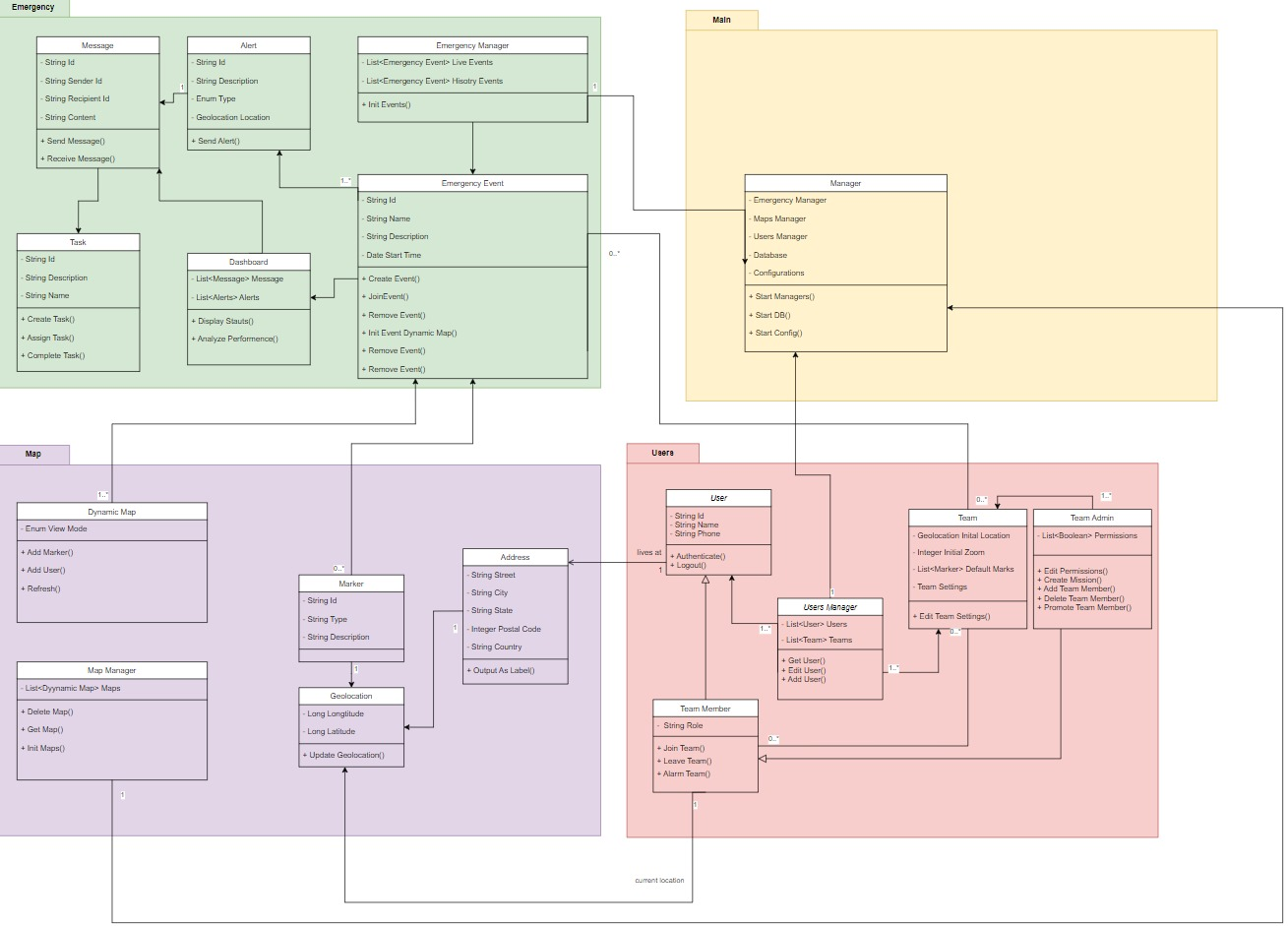
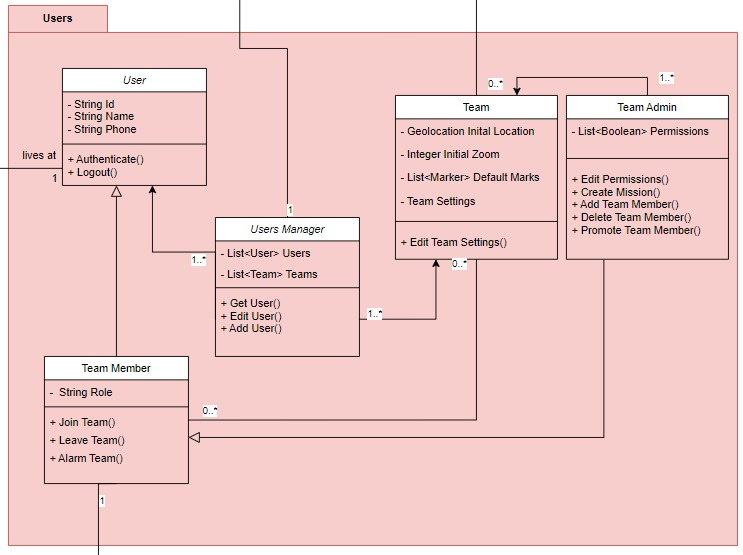
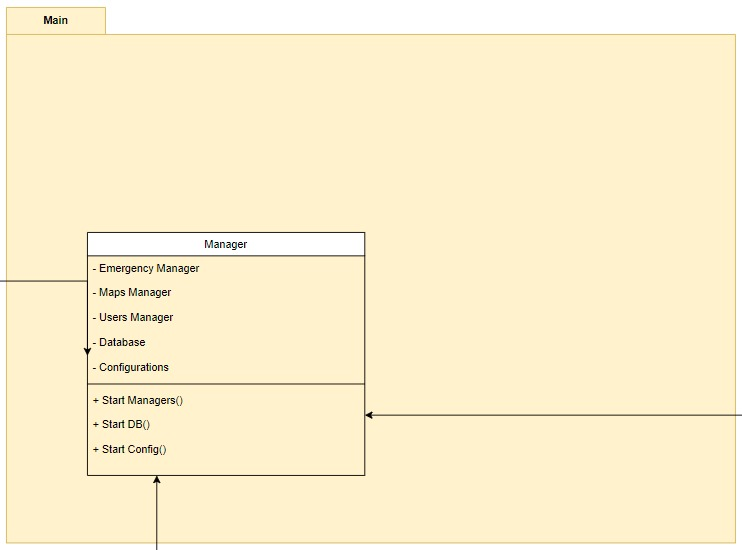


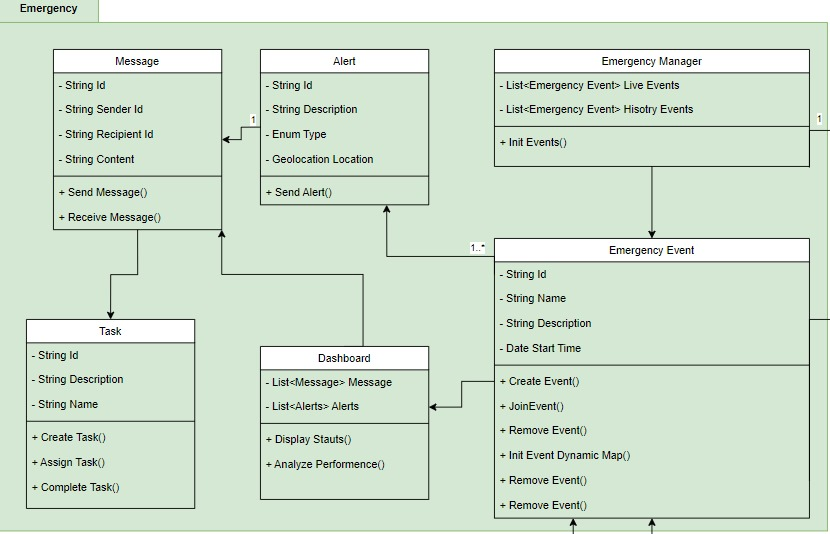
Fig 2: Activity Diagram בחירום

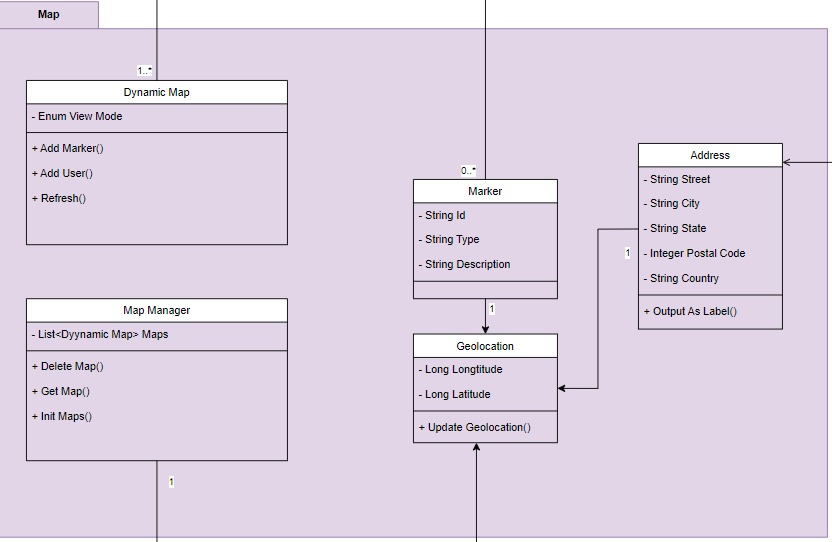
Class Maybe we should remove











**Expected Achievements**

* **Enhanced Coordination:** The application will facilitate real-time coordination among different teams, improving response efficiency during emergencies.
* **Reliable Functionality Offline:** Key features will be accessible without constant internet connectivity, ensuring functionality under varied conditions.
* **Dynamic Role-Based Access:** Customizable interfaces and functionalities based on user roles and permissions, ensuring that each team member has the tools and information necessary for their responsibilities.
* **Mapping and Task Management:** Features for marking significant locations and assigning tasks, enhancing operational awareness and efficiency.
* **User-Friendly Design**: A focus on intuitive navigation and accessibility to encourage adoption and minimize training requirements.

**Challenges**

* **Data Synchronization:** Ensuring seamless sync across devices in fluctuating network conditions.
* **Interface Simplicity vs. Feature Richness:** Balancing a simple user interface with the complex functionalities needed by different teams.
* **Scalability:** Preparing the system to handle increasing amounts of data and users without performance loss.

**Success Criteria**

* **User Engagement and Retention:** Targeting an engagement rate of 85% among all users, with feedback loops to continuously refine the application.
* **Operational Impact:** Aiming for a 30% reduction in response times, demonstrating improvement in coordination during emergencies.
* **Stakeholder Satisfaction:** Seeking a 90% satisfaction rate from team leaders and members on the utility and usability of the application.

**Testing Plan:**

**Introduction**

This detailed testing plan is designed to validate the functionality, usability, performance, and security of the application developed for first response and emergency resilience teams.

**Scope**

Comprehensive testing of all web and mobile interfaces, backend functionalities, offline capabilities, and user interactions under simulated real-world conditions.

**Objectives**

• Ensure all features function according to specification.

• Validate the user interface for ease of use and accessibility.

• Confirm application performance under normal and peak loads.

• Safeguard sensitive data through rigorous security testing.

**Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Area** | **Test Name** | **Description** | **Procedure** | **Expected Result** |
| **Login Functionality** | Successful User Login | Verify that a user can log in with correct credentials. | Enter username: admin and password: 12345. Press "Login". | User is successfully logged in. |
|  | Incorrect Password Login | Check the system's response to an incorrect password. | Enter username: admin and password: wrongpassword. Press "Login". | An error message "Incorrect username or password" is displayed. |
|  | Re-login Attempt | Ensure the system correctly handles attempts to log in by a user who is already logged in. | Login as admin with password 12345. attempt to log in again in another browser tab. | An error message "User is already logged in" appears. |
| **User Registration** | Complete User Registration | Test successful registration process. | Enter all required details in the signup form and submit. | A success message "Sign up successfully" is displayed. |
|  | Incomplete User Registration | Verify the system's handling of incomplete registration attempts. | Omit the last name in the signup form and submit. | An error message "Need to fill the last name field" is displayed. |
| **Task Assignment and Mapping** | Task Creation and Assignment | Test the functionality for creating and assigning tasks. | Access the task creation form, fill in details, and assign it to a team member. | A message "Task assigned successfully" is displayed. |
| **Emergency Mode Operation** | Offline Functionality Check | Confirm that critical features are accessible offline. | Simulate a network outage and attempt to use critical features. | Core functionalities remain operational. |

**REFERENCES**

**סידור לפי שם משפחה של כותב ראשון**[1] McWilliams, M. (2020). Creating an effective emergency response team. Professional Safety, 65(6), 66-70.‏

[2] King, R. V., North, C. S., Larkin, G. L., Downs, D. L., Klein, K. R., Fowler, R. L., ... & Pepe, P. E. (2010). Attributes of effective disaster responders: focus group discussions with key emergency response leaders. Disaster medicine and public health preparedness, 4(4), 332-338.

[3] Brown, O., Power, N., & Conchie, S. M. (2021). Communication and coordination across event phases: A multi‐team system emergency response. Journal of Occupational and Organizational Psychology, 94(3), 591-615.

[4] Boduch, A. (2017). React and react native. Packt Publishing Ltd.

[5] FEMA (Federal Emergency Management Agency). (2022). First Responders.

[6] FEMA (Federal Emergency Management Agency). (2022). Community Emergency Response Team.

[7] <https://www.avuka-app.com/squad/about>

[8] <https://www.avia-app.org>

[9] https://zello.com/product/frontline-communications/