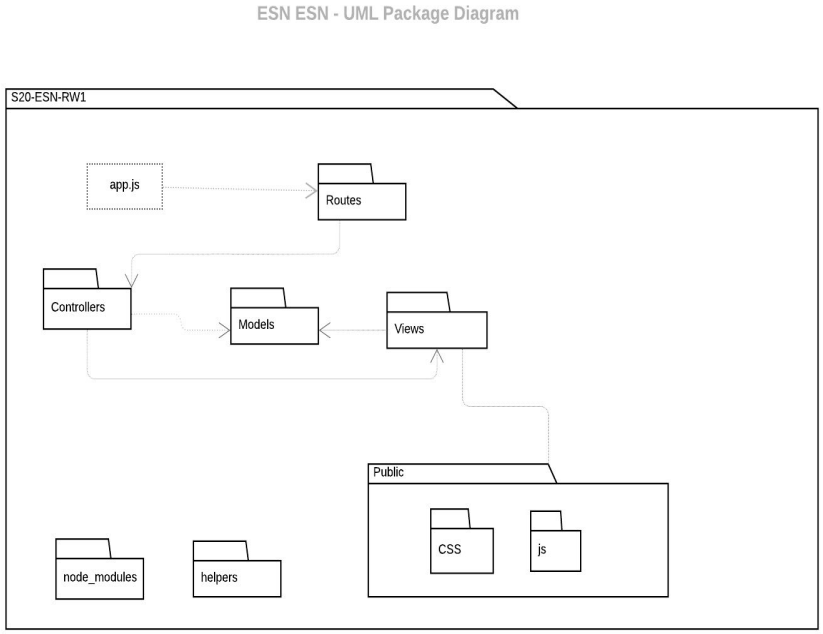
Emergency Social Network

Team S20-ESN-RW1

To save lives by issuing early warnings about impending calamities and have emergency incidents attended to quickly

Technical Constraints

- App server runs locally
- Clients connect to the app server via their browsers.
- For UI development, HTML5, CSS, JS must be used and the UI should be responsive(PC and Mobile views)
- For back-end, Node.js shall be used with Express.js to handle http requests routing
- System has a RESTful API - should function with and without UI
- System supports real-time dynamic updates



High-Level Functional Requirements

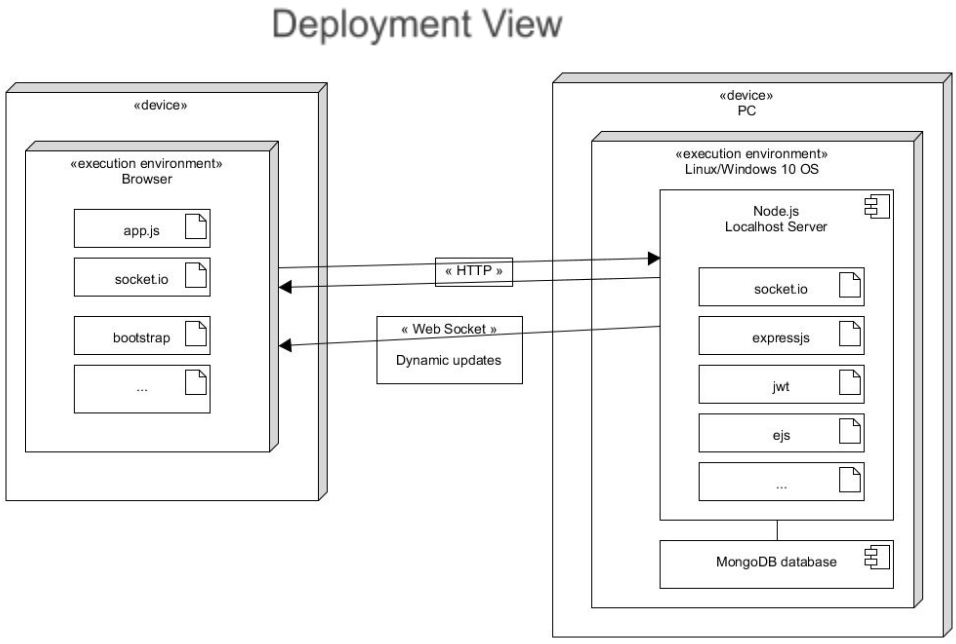
- Citizens shall be able to register into the system
- A citizen shall be able to login and leave the system
- A citizen shall be able join a community using a username and password with additional optional information
- A citizen shall see themselves listed in the directory alongside other citizens
- A citizen shall be able to post a message on a public wall (can be seen by everyone in the community)
- A citizen shall be able to post a message on a private wall
- A citizen will be able to update the status on the profile
- A citizen will be able to post an announcement
- A citizen will be able to search other citizens based on different criteria such as name and status
- A citizen will be able to search private or public chats based on content
- A citizen will be able to search announcements based on content

Top 3 Non-Functional Requirements

Security> Performance

Architectural Decisions with Rationale

- Client-Server as main architectural style as the server will act as a central point that coordinates and controls clients' communication
- Server-side JS (node.js) for small footprint and performance
- Lightweight MVC on the server-side using **express** framework
- RESTful API provides core functionality and reduces coupling between UI and back-end
- Web-sockets allow real-time communications or updates
- Lightweight SQL DB with small footprint



Design Decisions with Rationale

Include other important design decisions.

- Encapsulate data and behavior in models for easy testing and better modularization
- **Observer** design pattern to keep the connected citizens in the community in sync with new updates (realtime behavior)

Responsibilities of Main Components

- **Models:** encapsulate data and behavior for entities of the system. The main models are:
  - Chat

    - User
    - Status
    - Announcement
- **Controllers**: ChatController, UserController, StatusController, AnnouncementController,SearchController
- **Views**: UserRegistrationPage, ChatPage, AnnouncementPage,SearchPage