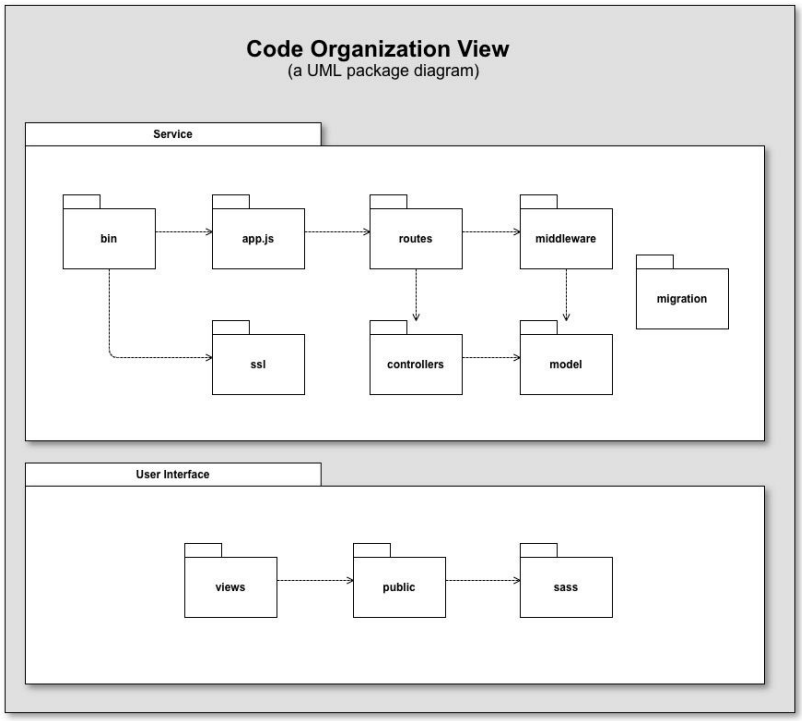
# Emergency Social Network Architecture Haiku by Team SA-1

*The purpose of the ESN system is to help citizens communicate with others in case of an emergency. They could chat publicly on a public wall or privately with other people. Citizens can also set their own status and search information. Besides, coordinators can post a public announcement and the administrator can manage user profiles. Finally, one surprising feature will be provided.*

## Technical Constraints

- Node.js is our Web App Server. Express is the Web framework exposing the API from the backend.
- Users connect to the app server via their browsers. Memory and performance are limited by hardware.
- No native app, only web stack (HTML5, CSS, JS) on mobile browser (iPhone & Android) and desktop browser (Chrome, Safari)
- System has a RESTful API - works with and without UI
- System supports real-time dynamic updates.
- Application




## High-Level Functional Requirements

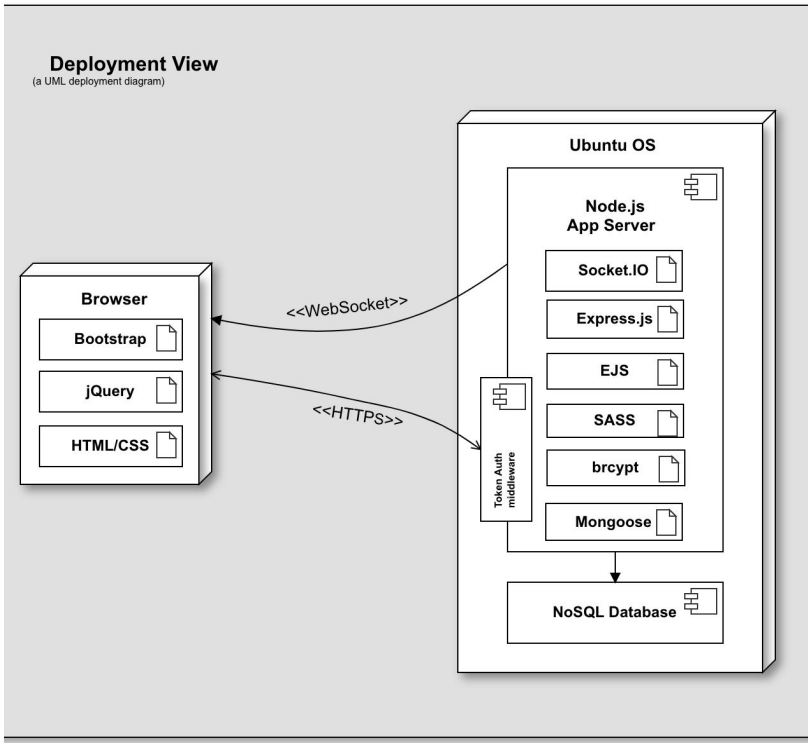
- Users can Login-Logout, Share status, Chat, Search info
- Coordinator can post an public announcement; Admin can manage user profiles.
- … (Surprise Case Here)

## Top 3 Non-Functional Requirements

Usability > Performance > Extensibility

## Architectural Decisions with Rationale

- Client-Server as main architectural style
- Server-side JS (node.js) for small footprint and performance.
- Lightweight MVC on the server side using Express.js framework.
- RESTful API provides core functionality and reduces coupling between UI and back-end
- Web-sockets allow event-based fast dynamic updates.
- Lightweight NoSQL DB with small footprint.
- Secure based connections through HTTPS.
- Authorization with JWT, stored in browser cookie.
- Single Application Page adopted. Bootstrap and jQuery to implement Responsive UI.




## Design Decisions with Rationale

- Encapsulate data and behavior in data models for better modularization
- JWT instead of Session to follow RESTfulness principle.
- Singleton design pattern for mongoose connection with DB. Any DB access will use this connection.
- Singleton design pattern for API request.
- Instance of MongoMemoryServer for Unit and Integration test.

## Responsibilities of Main Components

- **models:** encapsulate data and behavior for entities of the system, main models are user.js. Are the only components that interact with the DB.
- **controllers**: separated from routes; integrate with models and views together. Capture http requests data, control the flow and respond back with a http response to the client.
- **views**: contain EJS templates, taking advantage of Bootstrap, jQuery, CSS (SASS). Render data passed by controller.
- **middlewares**: validate JWT web token and privilege access by role.