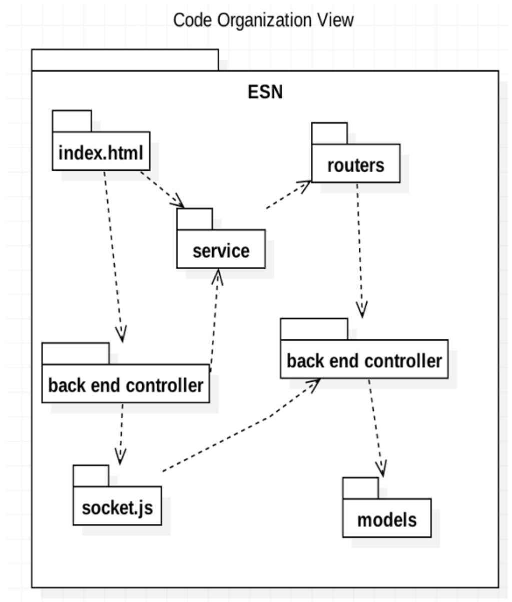
Emergency Social Networking Team: Facepalm SE

The Emergency Social Networking(ESN) serves as a communication center during emergent situation. However, the system is different from other existing social networks since it is specifically designed to support consumers who are under situations such as earthquake, tsunami, tornado, wildfire, etc.

Technical Constraints

- **App** server runs on Heroku. Clients can connect to the server via their mobile phones or computer under emergent situation.
- **No** native app, only web stack (HTML, CSS, JS) on mobile browser (initially Google Chrome and Safari are supported).
- **System** support real-time dynamic updates.



High Level Function

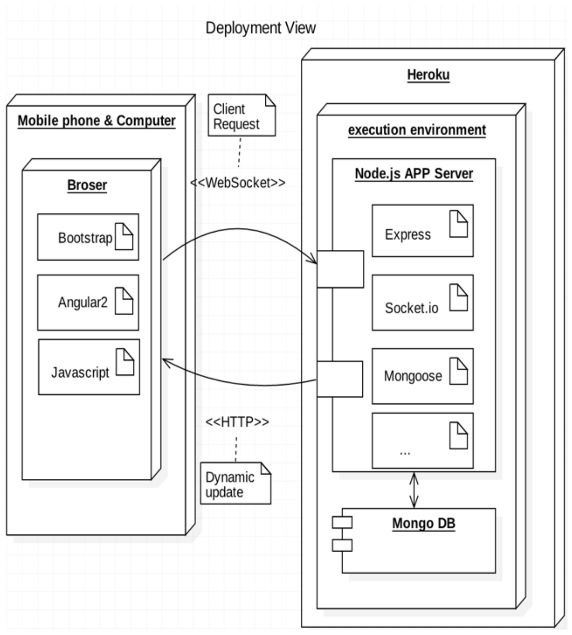
- **A** user affected by a disaster can be able to communicate critical information with others, post information on public area, select the status to other users, search information from chat messages and public area.
- **A** coordinator has additional authority to post announcement and broadcast messages to every user.
- **A**n administrator, apart from all things a coordinator and a user can do, has the additional authority to edit other users' profiles

Top 3 Non-Function Requirements

- **Easiness**: a user can as quickly as possible to find the function he needs
- **Robustness**: the app should be stable and guarantee clients can stably keep contact with others even under emergent situation
- **Reliability**: the app is mostly used during emergency so it should be easily maintained and can work well in time of emergency

Architectural Decisions with Rationale

- **Client-Server** as the main architectural style
- **Server-side** JS (node.js) for low footprint and reasonable performance
- **Lightweight** MVC on the server side via the **express** framework
- **Event-based** fast dynamic updates via web-sockets



Design Decisions with Rationale

- **Project** is to be split in two separate modules to enable Low-Coupling
- **Frontend** consumes that information through a service interface and implements visual components and view logic functionality.
- **Beckend** service will implement DAO pattern to access and retrieve information from database and expose functionality and information through a RESTful API.

Responsibilities of Main Components

- **socket.io:** dynamic updates from server to client, clients' views are automatically updated when new messages are post or when new new users login
- **Bootstrap**: responsive design, clean, scalable UI layout
- **MongoDB (mLab)**: lightweight No-SQL database
- **AngularJS:** for client side encapsulation of application behaviours