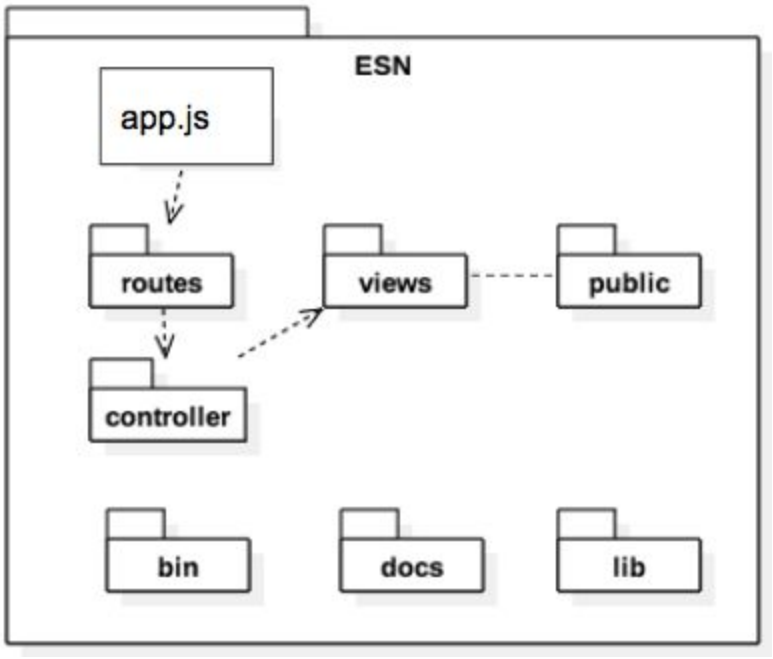
# Emergency Social Network (ESN)

Team SA-3 DySci

*Build a Social Network to support civilians effectively during emergency situations like earthquake, tsunami, tornado, wildfire, etc.*

## Technical Constraints

- **Server Side Software**: relational databases are not designed for scalability. They are designed to run on a single server to maintain data integrity. On cloud platform, developers have limited control over hosting infrastructure.
- **Client Side Software**: no native app, only web stack (HTML5, CSS, JS) on mobile browser.




## High-Level Functional Requirements

- Citizen can join community and share status in ESN
- Users can chat publicly & privately
- Users can search for information stored in the system
- Coordinators can post announcement
- Administrators can change user profile

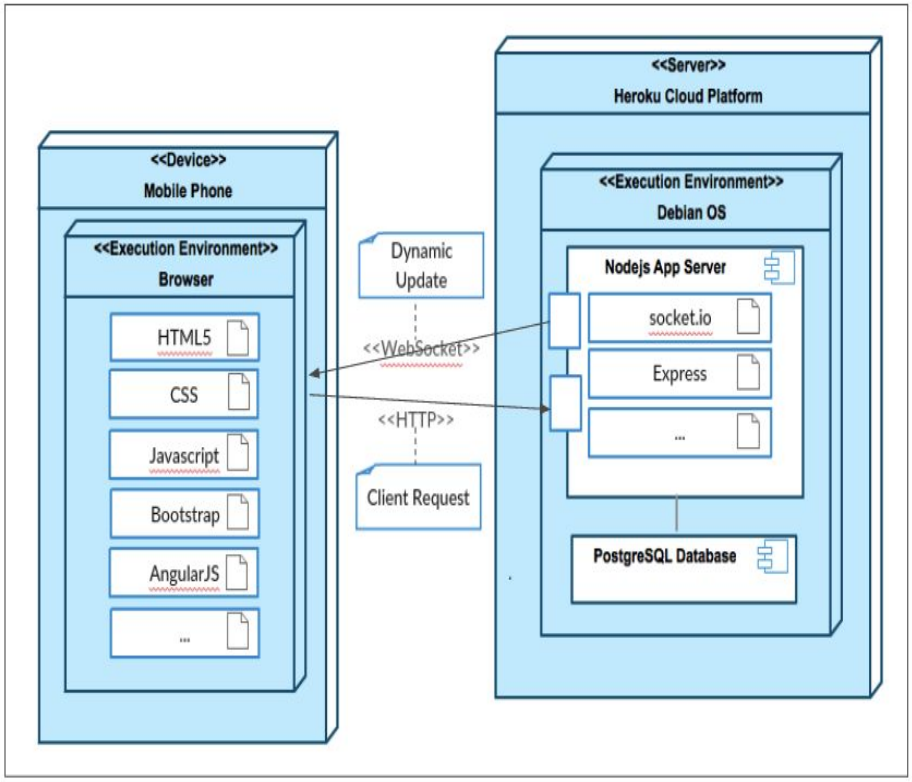
## Top 3 Non-Functional Requirements

Works Out-of-the-Box > Usability > Performance

We provide our software as a service. Every user can use it immediately after installation, which is highly important in emergency situations. Our application is easy to use, so even new user can use it without a problem. Performance is also a concern, as high latency can result in missing the best time of life saving.

## Architectural Decisions with Rationale

- Client-Server as main architectural style
- Server-side JS (node.js) for low footprint and reasonable performance (event-based, non-blocking asynchronous I/O, easily configurable pipe-and-filter for processing incoming requests via middleware)
- Lightweight MVC on the server side via the **express** framework
- RESTful API for core functionality to reduce coupling between UI and back-end
- Event-based fast dynamic updates via web-sockets




## Design Decisions with Rationale

- Encapsulate data and behavior in models for easy testing and better modularization
- **Observer** design pattern is used to make sure the change of status and going online/offline of one person can be viewed by others. Also, it ensures chat messages can be viewed by users in real time.
- **Singleton** design pattern is used by database interface, which provides a global point of access to database connection and therefore ensures data consistency.

## Responsibilities of Main Components

- **Node.js:** server side Javascript that provides a rich library of various modules which simplifies web development.
- **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**: free front-end framework that supports responsive design, clean, scalable UI layout. It is compatible with all modern browsers (e.g. Chrome, Firefox, IE, Safari).
- **AngularJS**: declarative paradigm is used for creating patterns, therefore lightweight, easier to read and iterate.
- **Heroku**: PaaS that allows to migrate and start your apps quickly. Automatically handles load balancing, database cluster, deployment system and log aggregation. Works well with git.
- **PostgreSQL**: Open source relational database that uses standard SQL as query language. Provided by Heroku as a service.