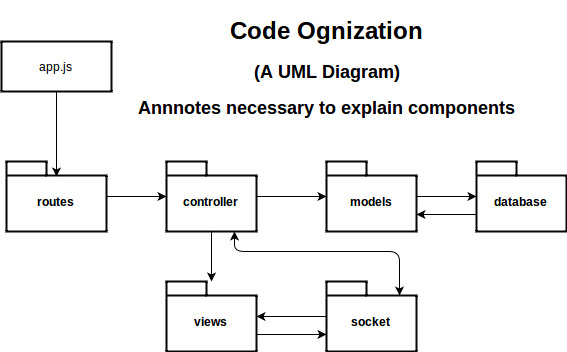
# Survivable Social Network on a Chip Team plusOne

*The system is a lightweight application designed to provide citizens with the opportunity to communicate with each other - from friends to relatives, independent of the communications infrastructure. It uses Heroku server to ensure that connections are made in natural disasters (where communication may fail) in order to be able to evacuate personnel in time.***Technical Constraints**

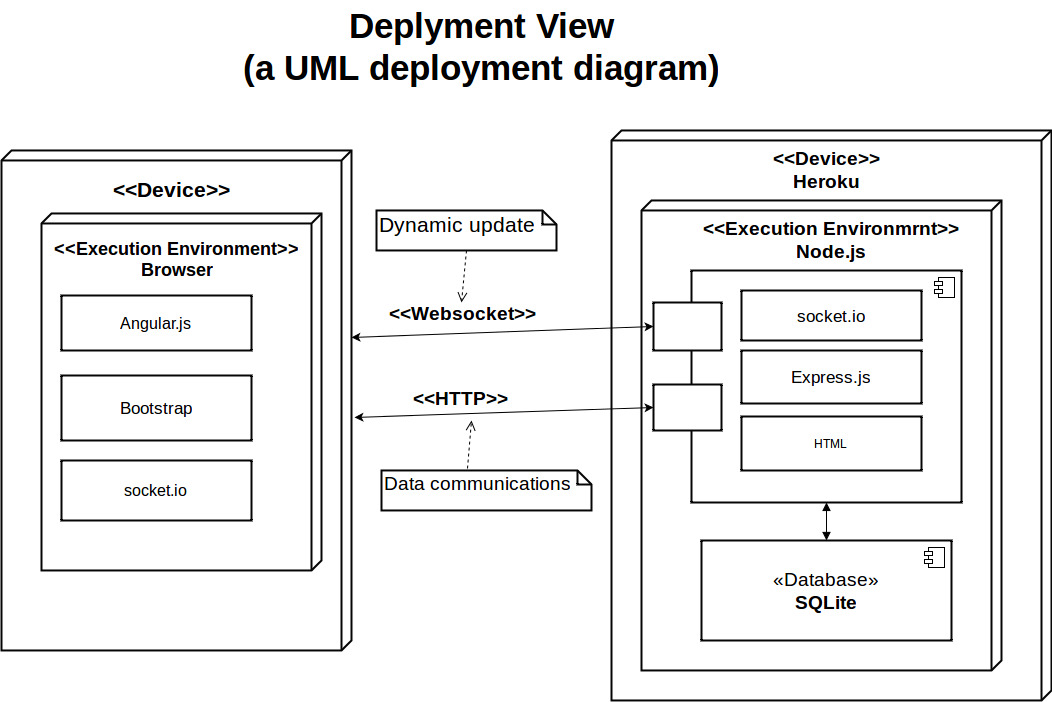
* App server runs on a Heroku platform.
* Clients connect to the app server via their mobile phone browsers. Memory and performance limited by hardware.
* No native app, only web stack (HTML5, CSS, JS) on mobile browser 
* System has a RESTful API and supports real-time dynamic updates

## High-Level Functional Requirements

* Citizen is able to login, logout and sign-up
* Citizen is able to share his/her status
* Citizen is able to post a public message
* Citizen is able to search information
* Citizen is able to share his/her location
* Privilege difference: normal user and administrator

## Top 3 Non-Functional Requirements

* Usability: For all kinds of citizens, proved to be usable with feedback.
* Cross-platform compatibility: Must be well-performed with high consistency on different mobile platforms.
* Reliability: Must be stable in emergency situation.



## Architectural Decisions with Rationale *(some examples below)*

* Model-View-Controller 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.
* Socket.io allow event-based fast dynamic updates.
* SQLite, the lightweight database provides data storage and access.

## Design Decisions with Rationale

* **Singleton** use Socket.io (observer pattern )and AngularJS throughout the application
* **Adapter** design pattern to substitute a test database for the production database during testing
* **Facade** encapsulate complicate implementation details from client interface.

## Responsibilities of Main Components

* **AngularJS:** dynamically control, modify and acquire client-side code and
* **Bootstrap**: responsive design UI
* **HTML**: UI views
* **MongoDB**: [NoSQL](https://en.wikipedia.org/wiki/NoSQL) database uses [JSON](https://en.wikipedia.org/wiki/JSON)-like documents with [schemas](https://en.wikipedia.org/wiki/Database_schema).
* **SQLite**: Quick, easy, lightweight SQL DB
* **Socket.io:** event-based fast dynamic updates from back-end to front-end.