FSE Iteration 1 Team Justice League (SB5)

An application that allows citizens to be able to join an online community and in a disaster situation be able to communicate to others in a public chat. With more functionality pending in the coming iterations.

Technical Constraints

- **JS Objects**: Node.js utilizes JS Objects for all things. MongoDB stores said objects as JSON
- **HTTP Codes**: Sending correct HTTP codes that comply to expected practices

High-Level Functional Requirements

- **Join Community**: citizens can register and login as a user and look at a directory of all other users
- **Chat Publicly**: users can chat with all other users, where messages are dynamically updated

Top 3 Non-Functional Requirements

- **Security**: ensure that password is never sent unencrypted between client and server Password is also decrypted and then reencrypted using another algorithm.
- **Session**: maintain user session, logged in users remain logged in
- **Socket.io**: allow dynamic updates of messages

Architectural Decisions with Rationale

- Client-Server as main architectual style
- Node.js Server: event-based, nonblocking asynchronous I/O
- Lightweight MVC on server side via **express** framework
- RESTful API for core functionality to reduce coupling between UI and backend
- Event-based fast dynamic updates via web-sockets for chat

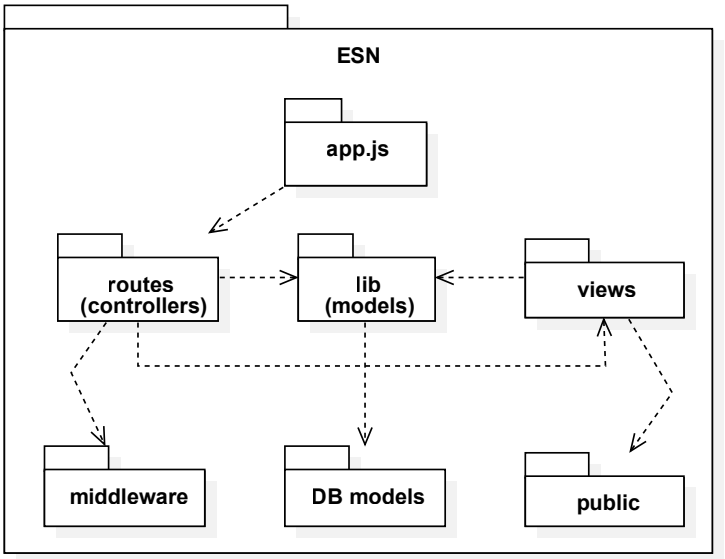
Design Decisions with Rationale

- Object modeling in a NoSQL database to provide relational modeling
- Encaspsulate data and behvaior in classes for easy testing and better modularization
- Use **Adapter** design pattern to substitute a test database for production database during testing

Responsibilities of Main Components

- Custom **Bootstrap**: responsive design, clean, scalable UI layout with a custom theme
- **pug**: template engine for rendering UI views
- **MongoDB**: a document oriented NoSQL DB
- **Mongoose**: provide relational modeling for DB
- **socket.io**: dynamic updates from server to client, clients' views are automatically updated when new messages are posted

Code Organization View



- **app.js**: main file of app, like a master controller
- **routes**: controller files that call models (in lib) and views
- **lib**: models (classes) of users, messages, and database adapter
- **views**: front end pug files that are rendered to user
- **middleware**: mongoose, crypto-js, expree-session, express, pug, socket.io
- **DB models**: mongoose models for MongoDb
- **public**: frontend resources: css, js, and images

Deployment View