Survivable Social Network on a Chip Team Pittsburgh TeamA1

Iteration 1.

…

Technical Constraints

- **Hardware**: Sever and database run on a Beaglebone Black. Wireless coverage of WiFi network is limited. Security is not guaranteed. Memory and performance limited by hardware.
- **Client Side Software**: no native app, only web stack (HTML5, CSS, JS) on mobile browser
- **Database Side Software:** not working well with multi-threads

High-Level Functional Requirements

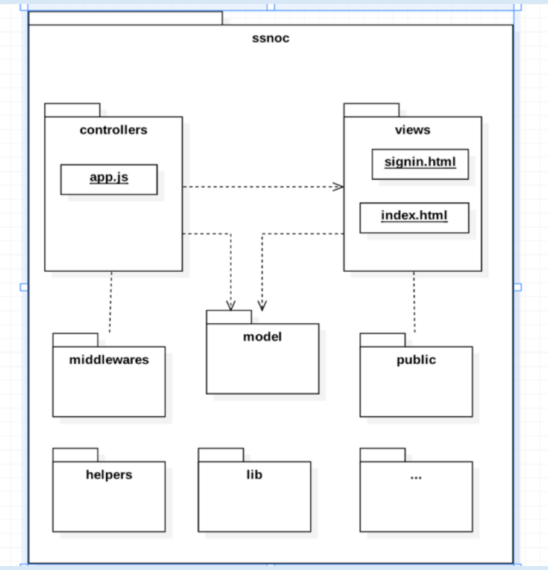
- An user can register for SSNoC network to join the community.
- Users can communicate with other user in survivable social network.
- Information about user and user message will be saved into system database.

Top 3 Non-Functional Requirements

Reliability > Robustness > Maintainability

Code Organization View

(a UML package diagram)



Architectural Decisions with Rationale

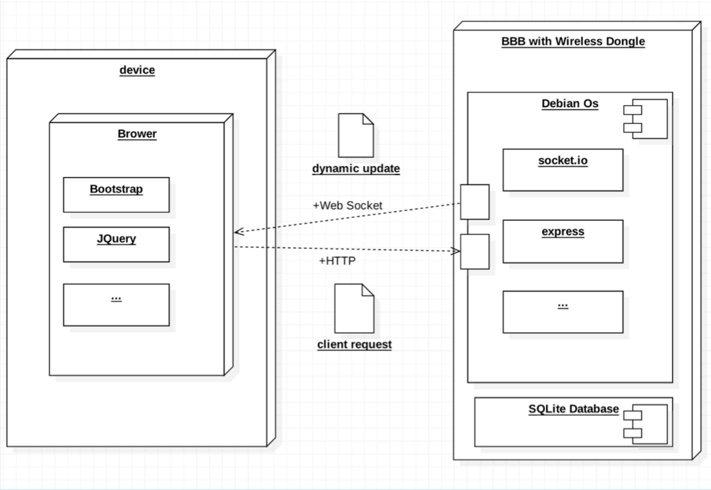
- Client-Server as main architectural style
- Server-side JS (node.js) for low footprint and reasonable performance (event-based, non-blocking asynchronous I/O, easily configurable pipe-and-filter for processing incoming requests via middleware)
- Lightweight MVC on the server side via the **express**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

Design Decisions with Rationale

- Development **Strategy** pattern to encapsulate data and behavior in models when testing and modularizing the system.
- Encapsulate the subsystem with applying a simply User Interface.
- **Singleton** pattern has been applied in the system design as only one model is used in the system
- Remove the coupling between different subsystems by apply **Bridge** patterns
- **Composite** patterns has been used when designed User Interface.

Deployment View

(a UML deployment diagram)



Responsibilities of Main Components

- **socket.io:** dynamic updates from server to client, clients communicates with server with methods of socket.io.
- **Bootstrap**: responsive design, clean, scalable UI layout
- **jQuery**: light-weight JavaScript library, free and open source, easy to deal with HTML event.
- **SQLite**: light-weight DB, ACID database manage system, designed for embedded system, saves the information and message of users.
- **Express**: simple and flexible framework based on Node.js, provides powerful API for web application.