Survivable Social Network on a Chip MayDay

Team SB-4

The purpose of the project is to help people to connect with each other online when they are in situations where they don't have methods such as phone and internet to communication with outside.

Technical Constraints

- Run the server on the Beaglebone with limited Power and Memory
- Use Node.js, Express, EJS and Socket.io to build the application
- The server and the client must be able to interact
- The RESTful API must function and should be accessible through various devices
- Use Sequelize for building models
- Use Static Analysis, Mocha and Shippable for testing

High-Level Functional Requirements

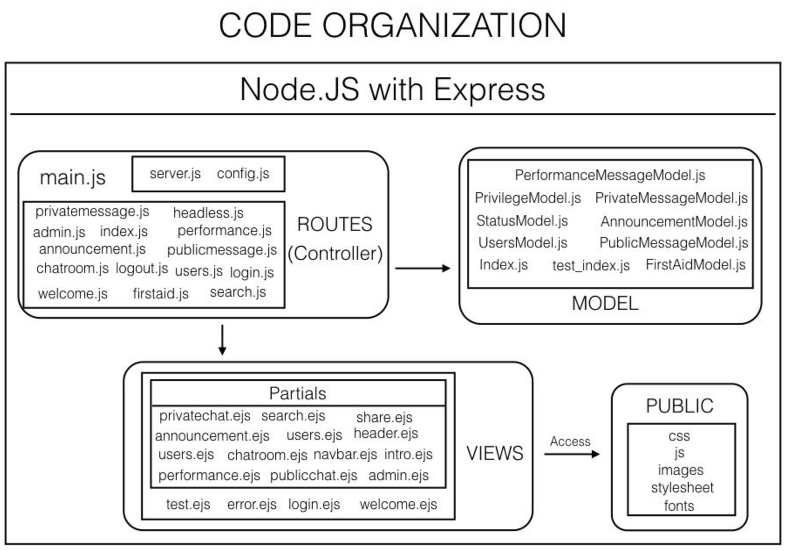
- Join Community - Log in and out of the chat room
- Chat publicly and privately
- Share Status
- Post Announcement (Coordinator)
- Search Information
- Manager user profile (Administrator)
- First-aid Advice

Top 3 Non-Functional Requirements

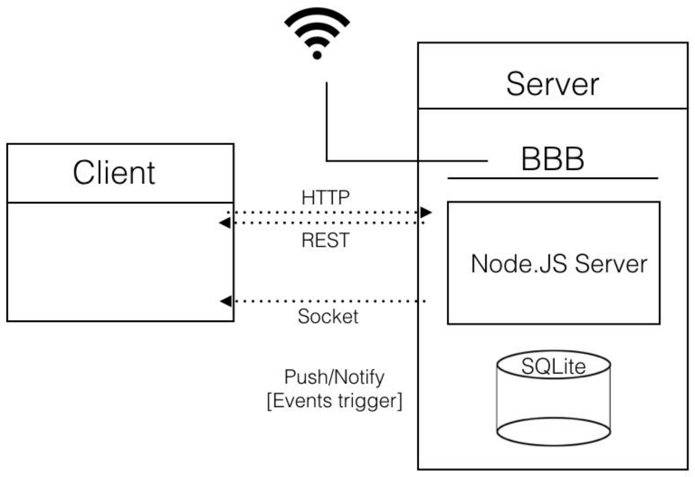
- **NFR1: Performance**
  - Users will be able to send and receive messages in real time.
- **NFR2: Usability**
  - User interface should be simple and friendly so that users must be able to understand where and how to access the network, log in and use the chat room.
- **NFR3: Efficiency**
  - We have limited storage so we want to be able to run the server efficiently.

Architectural Styles/Patterns with Rationale

- Client-server
  - It enables communications for multiple clients to connect to server (server.js).
- MVC
  - The current code organization includes models - views - controllers (routes).
- Event-based
  - Socket.io and AJAX calls enable events.
- Repository


CODE ORGANIZATION
Node.JS with Express
main.js
server.js config.js
privatemessage.js headless.js
admin.js index.js performance.js
announcement.js publicmessage.js
chatroom.js logout.js users.js login.js
welcome.js firstaid.js search.js
ROUTES (Controller)
PerformanceMessageModel.js
PrivilegeModel.js PrivateMessageModel.js
StatusModel.js AnnouncementModel.js
UsersModel.js PublicMessageModel.js
Index.js test_index.js FirstAidModel.js
MODEL
Partials
privatechat.ejs search.ejs share.ejs
announcement.ejs users.ejs header.ejs
users.ejs chatroom.ejs navbar.ejs intro.ejs
performance.ejs publicchat.ejs admin.ejs
test.ejs error.ejs login.ejs welcome.ejs
VIEWS
Access
PUBLIC
css
js
images
stylesheet
fonts

Deployment View



    - The mayday2.db file contains all the data and is shared by all the components.
- RESTful
    - REST API returns JSON objects and includes the mapping of CRUD operations to HTTP methods
- Pipe and Filter
    - Node.js uses pipe and filter internally, and we use Node.js for our application.

Design Patterns with Rationale

Observer

- When a new user joins the community, sends a message or posts an announcement, all other users will be notified.
- Files invovled: announcement.ejs, announcement.js, publicmessage.ejs, publicmessage.js, privatemessage.ejs, privatemessage.js, chat.js, login.ejs and login.js.

Facade

- When a user signs up or logs in, he or she will only see the sign-up or log-in form, and the system will validate all the requirements and store all the information.
- Files involved: login.ejs and login.js

Other Design/Architectural Decisions

Bootstrap

- We also use Bootstrap in our UI design to ensure that the chatroom is user friendly, and users will be able to navigate seamlessly.

Responsibilities of Main Components

NodeJS Server

- Serves RESTful API request for rendering pages and sends back JSON objects.
- Serves HTTP requests and gives responses by rendering the views.
- Complies with the BBB port.