



Team 4 Mayday

Foundation of Software Engineering Section B

Simrata Gandhi
Keerthana Thangaraju
Jason Wang
Angel Fu

AGENDA

- Vision
- Functionalities
- Software Architecture
- Software Engineering Practices
- Quality Attributes
- Challenges
- Takeaways

VISION

We want to help people to connect with each other when there are no connections to contact others due to natural disasters or other emergencies by building the Mayday emergency chatroom.

CONNECT THE UNCONNECTED

FUNCTIONALITIES

- First-aid Advice

- Join Community
- Chat Publicly
- Chat Privately
- Share Status
- Post Announcement
- Search Information
- Measure Performance
- Administer User Profile

PERFORMANCE

Measure Performance

Max Requests: 10

Test Duration: 30

Test Stop

Progress

+ POST 87

+ GET 95

Home

Welcome afu !

Through our emergency website Mayday, you will be able to share your status, connect with people through chat and view their status.

You can choose to share the status below

☒ OK
I am okay,
I don't really need help at the moment.

☐ HELP
I need help,
but it's not a life threatening emergency.

☐ EMERGENCY
I need help now,
as this is a life threatening emergency!

ADMIN

All Users (5)

User Id	Password	Privilege	Account Active	Action
keert	hello	ADMIN	Active	Save
SSNAdmin	admin	ADMIN	Active	Save
coord	coord	CO-ORDINATOR	Active	Save
monitor	monitor	MONITOR	Active	Save
afu	hello	CITIZEN	Active	Save

Private

Private Chat

Send a Private Message

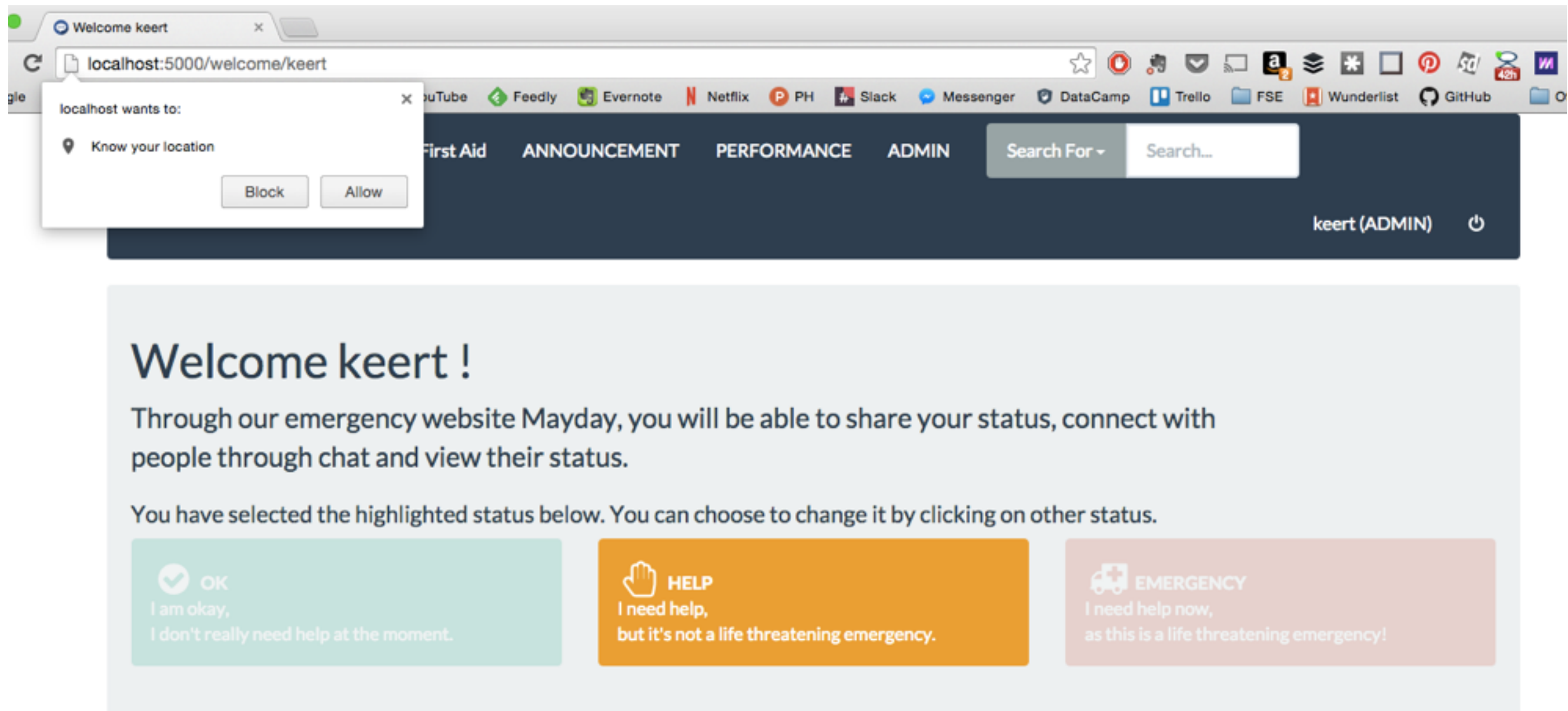
Click a name on the left to start chatting....

All the messages sent will be private and only visible to the selected user.

In case of unread chat messages, you will be notified.

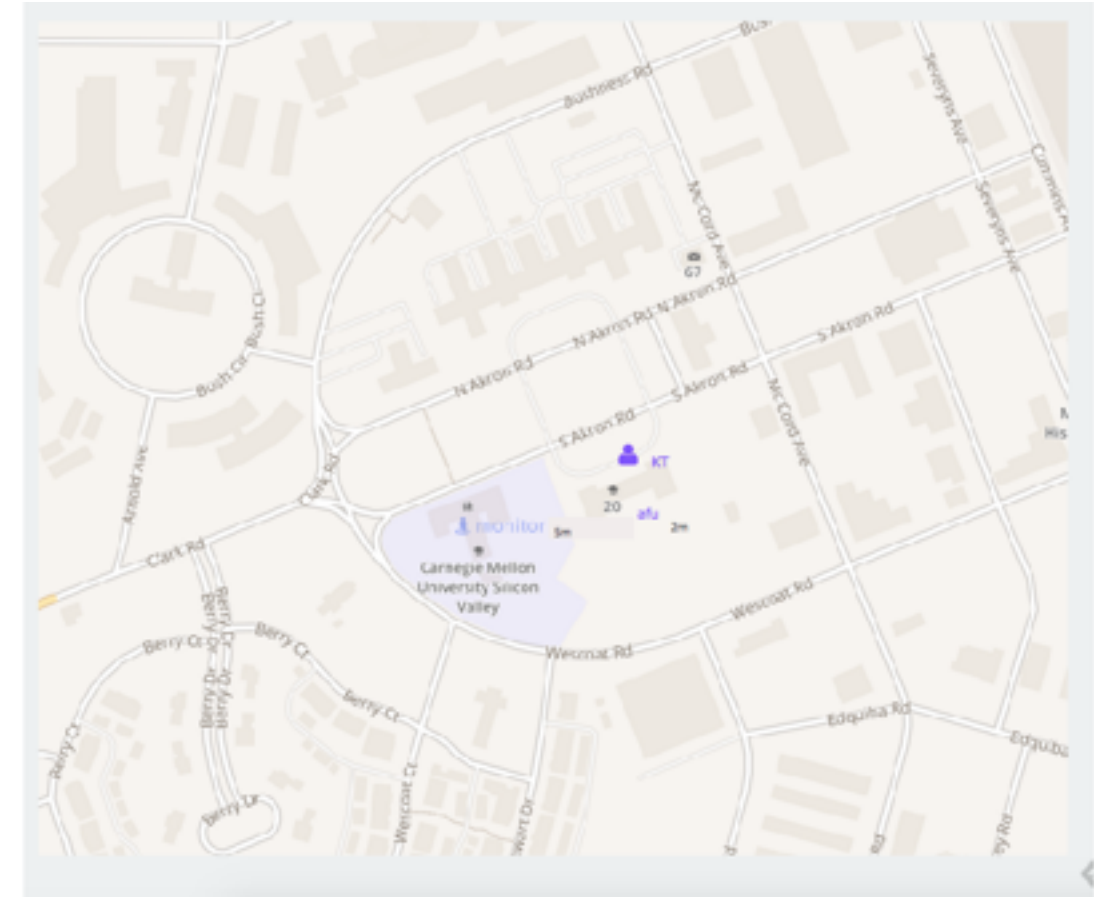
FIRST-AID ADVICE

Goal: Allow users to ask for help from nearby online users and look up basic first-aid advice in the chatroom



FIRST-AID ADVICE CONT'D

Search For ▾ Search...					keert (ADMIN) ⏻
Online Users (2)					
User Id	Status	Timestamp	Location	Request Help	
afu	🟢 OK	🕒 2015-12-06 16:40:27	📍 2 meters	❤️	
keert	🛑 HELP	🕒 2015-12-06 16:48:33	📍 37.469597500000006,-122.14208160000001		



Mapbox

Allows maps to be rendered in real time on web and mobile devices



Plots markers on the map

FIRST-AID ADVICE CONT'D

Online Users (2)

User Id	Status	Timestamp	Location	Request Help
afu	OK	2015-12-06 16:39:38	0 meters	
keert	HELP	2015-12-06 16:39:42	37.469603299999996,-122.1420532	

Offline Users (3)

User Id	Status	Timestamp	Location
monitor	Not Available	2015-12-03 21:48:19	9802 meters
SSNAdmin	OK		12784840 meters
coord	Not Available		12784840 meters



Help request from user: keert Accept

Welcome afu !

Through our emergency website Mayday, you will be able to share your status, connect with people through chat and view their status.

You have selected the highlighted status below. You can choose to change it by clicking on other status.

 **OK**
I am okay,
I don't really need help at the moment.

 **HELP**
I need help,
but it's not a life threatening emergency.

User requests for help from another user

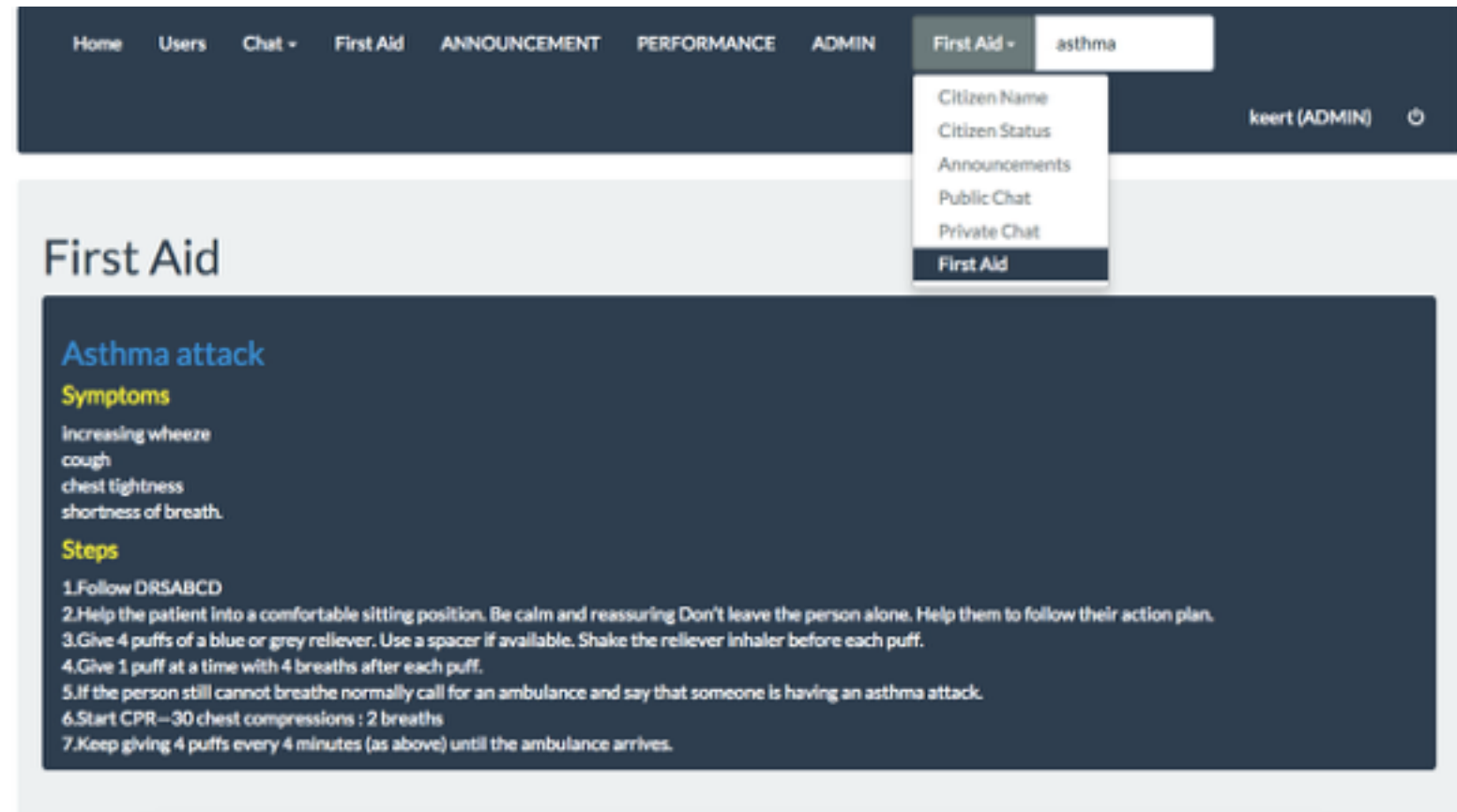
afu: has accepted your request! The user is on the way.

Online Users (2)

User Id	Status	Timestamp	Location	Request Help
afu	OK	2015-12-06 16:40:27	3 meters	
keert	HELP	2015-12-06 16:43:58	37.4695933,-122.14205539999999	

The other user agrees to help

FIRST-AID ADVICE CONT'D

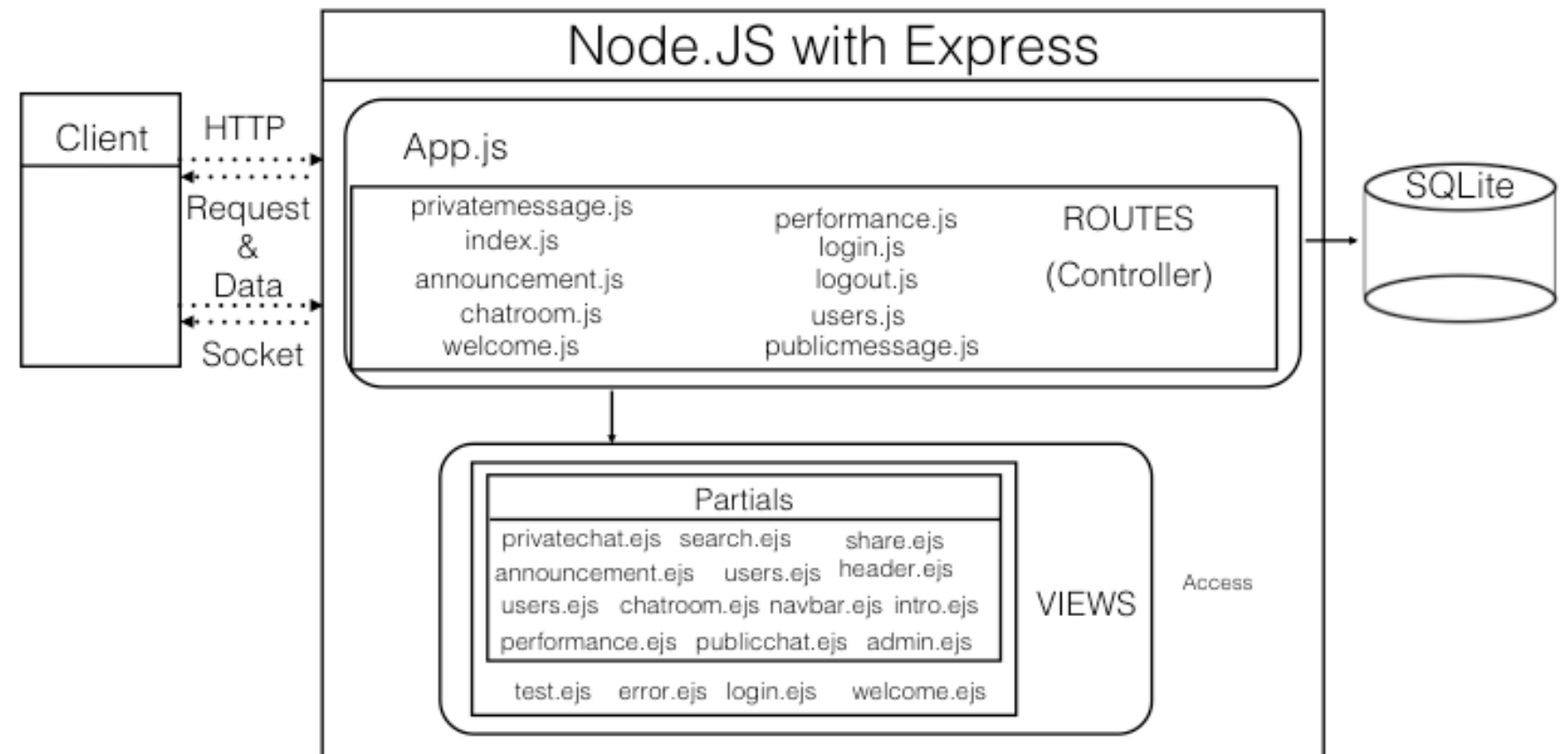


Users can browse and search first-aid advice in the chatroom

SOFTWARE ARCHITECTURE

Before

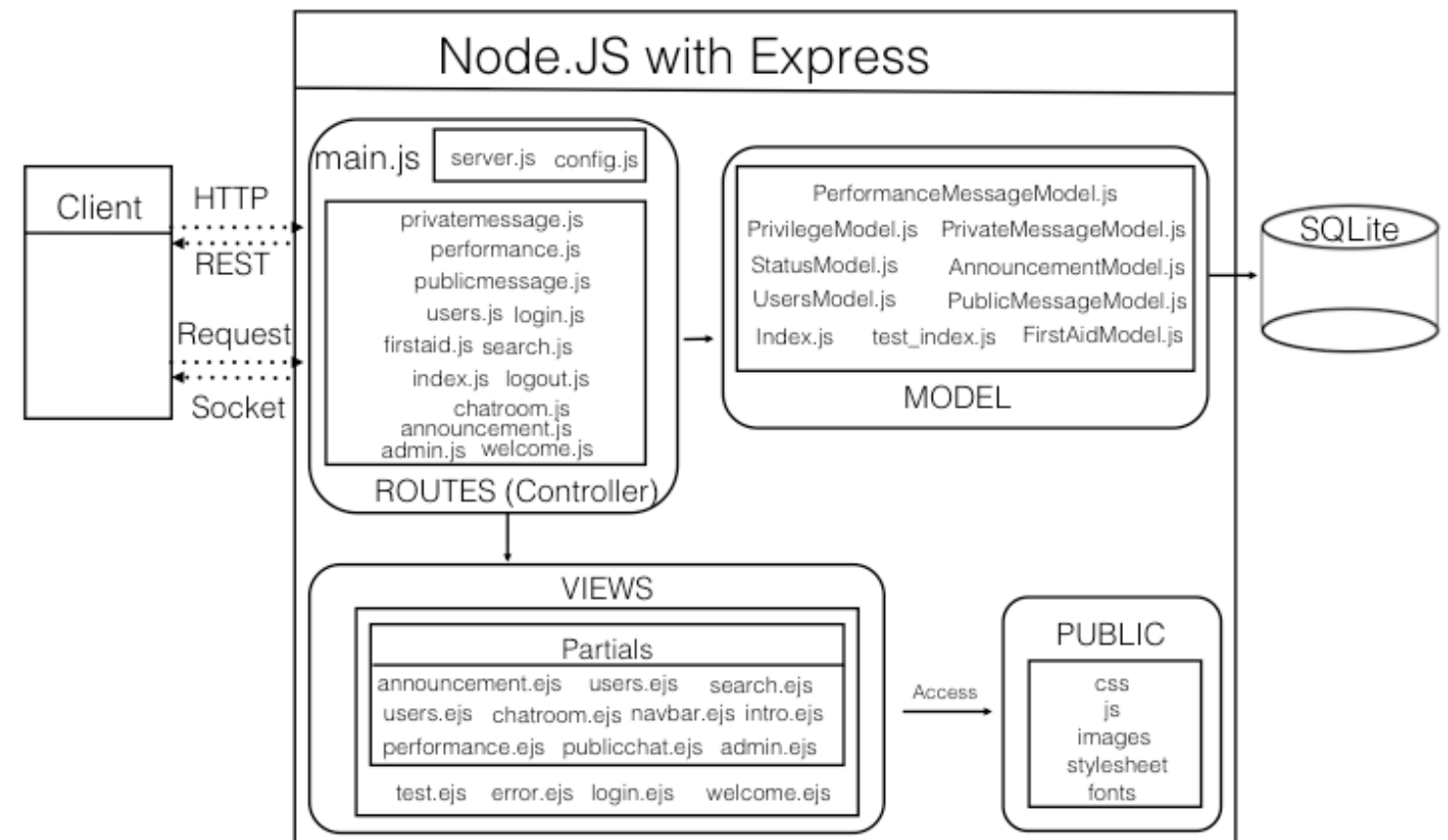
- Client-server
- Event-based
- Repository
- Restful
- Socket used for communication from client and server
- No models
- Unstructured code



SOFTWARE ARCHITECTURE CONT'D

After so much pain

- Client-server
- Event-based
- Repository
- RESTful
- MVC
- Socket communication only from server



MVC WITH EXPRESS, EJS, SEQUELIZE

Express (Routing)

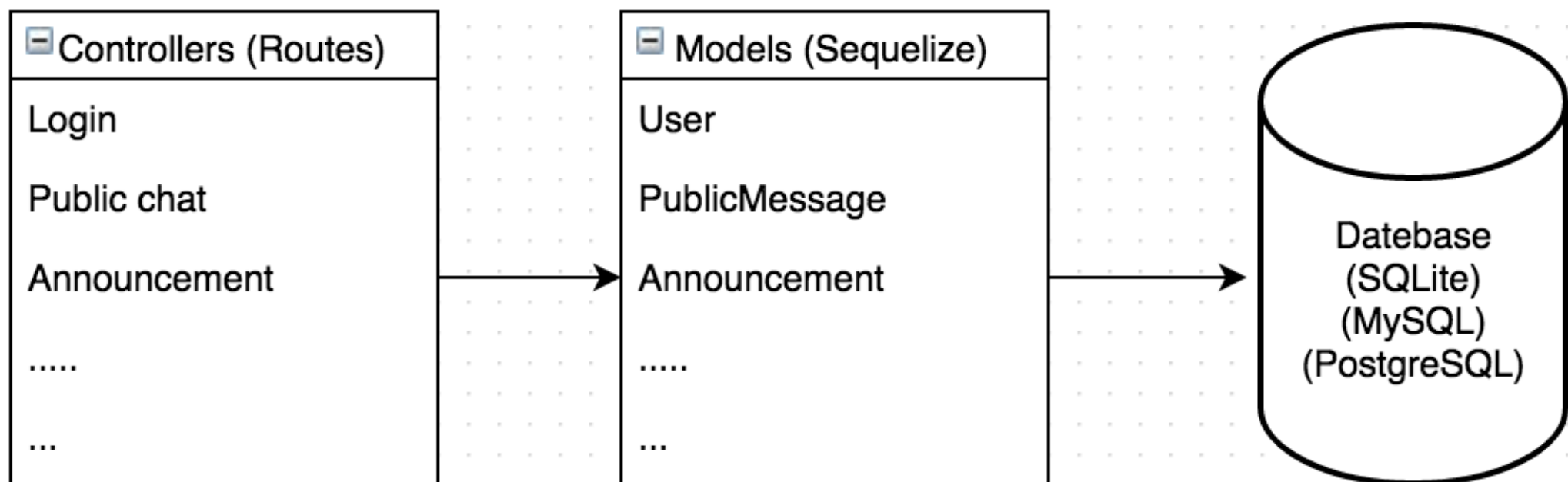
It provides easy routing framework to manage HTTP requests to create RESTful APIs for our application.

EJS (Views)

It allows us to create partial views and we only need to change the content in the layout using EJS.

Sequelize (Models)

It helps model Data Access Objects for SQLite tables.



SOFTWARE ENGINEERING PRACTICES

PAIR PROGRAMMING

- 2 members per team working on each module
- Improved efficiency, communication among team members

CONTINUOUS INTEGRATION VERSION CONTROL

- *Branching* and *merging* on Github for every iteration to track changes
- Helped tracking issues on time

HYBRID: AGILE + KANBAN

- Emphasis on value and also utilize time effectively to reduce waste

TESTING & COVERAGE (Mocha, Grunt and Shippable)

- Integration tests and automated script for testing http get & post requests
- Help to detect bugs early

OO ANALYSIS & DESIGN

- Manage requirements

SOFTWARE ENGINEERING PRACTICES CONT'D

Practices Missed Initially

- In process unit testing: Unable to perform due to absence of Models in code
- Refactoring: Lack of modularization and unorganized code structure



REFACTORING

After we analyzed
our problems and
re-prioritized....

Created models



Conducted in-process unit testing



Modified socket connections



Implemented REST API fully



Re-organized EJS files



Removed unused code and
Added comments for explanation



REFACTORING CONT'D

```
14 router.post('/', function(req, res){
15   var annon = req.body.annon;
16   var uid = req.body.uid;
17   db.run("INSERT into ANNOUNCEMENTS(annon,user_id) VALUES ('" + annon + "','" + uid + "')", function(err, val) {
18     // console.log("insert" + err + ":" + val);
19     if (err)
20       res.status(500).send({
21         error: 'Announcement could not be stored!'
22       });
23     else {
24       res.json({
25         success: true,
26         message: 'Announcement stored!',
27         annon: annon,
28         uid: uid
29       });
30     }
31   });
32 });
33
34 // router.post('/', function(req, res){
35 //   console.log("post request received");
36 //   //console.log(req.body.annon);
37 //   //console.log(req.body.uid);
38 //   var annon = req.body.annon;
39 //   var uid = req.body.uid;
40 //   var result = [];
41 //   db.run("INSERT into ANNOUNCEMENTS(annon,user_id) VALUES ('" + annon + "','" + uid + "')");
42 // }
43
44
45 router.post('/getHistory', function(req, res){
46   console.log('getHistory request received');
47   var lastLogin = req.body.lastLoginTime;
48   var token = req.get('x-user-token')
49   var result = [];
50   console.log(lastLogin);
51   console.log(token);
52   db.serialize(function() {
53     db.each("SELECT * FROM ANNOUNCEMENTS ;", function(err, row) {
54       result.push(row);
55     }, function() {
56       console.log(result);
57       res.send(result);
58     });
59   });
60 });
```

BEFORE

```
/*jslint node: true */
'use strict';

var express = require('express');
var router = express.Router();
var bodyParser = require('body-parser');
var models = require('../models');

module.exports = function(io) {
  /*insert the announcement into database when an administrator announces something*/
  router.post('/publishAnnon', function(req, res) {
    models.ANNOUNCEMENTS.create(req.body)
      .then(function(announcements) {
        io.emit('showAnnouncements', announcements);
        res.status(201).json({
          success: true,
          message: 'Announcement stored!',
        });
      }).catch(function() {
        res.status(500).send({
          error: 'Announcement could not be stored!'
        });
      });
  });

  /*fetch all historic announcements once a user enters announcements tag*/
  router.get('/getHistory', function(req, res) {
    models.ANNOUNCEMENTS.findAll()
      .then(function(announcements) {
        res.status(201).json({
          success: true,
          message: 'Get historic announcements',
          announcements: announcements,
        });
      }).catch(function() {
        res.status(500).send({
          error: 'Unable to fetch historic announcements!'
        });
      });
  });

  return router;
};
```

AFTER

QUALITY ATTRIBUTES

USABILITY

- Implemented the MVC architectural pattern
- Allowed us to create synchronized views and provide intuitive usability and responsiveness

SECURITY

- Chatroom allows users to log in with credentials
- Validated the token on the backend to ensure security

REUSABILITY

- After refactoring our code and creating models with Sequelize, we are able to reuse some of the code to reduce the redundancy

CHALLENGES

- A gap between team members' level of software engineering skill
- Struggles during the transitions between the previous iteration and the next one
- Not enough time spent on retrospective

TAKEAWAYS

- Helped out each other by answering questions and holding code review sessions
- Slowed down and re-evaluated our code and the team
- Ensured to reduce technical debt during each iteration
- Prioritized functionalities before design due to time constraints