Event Management System

Team Name: Group 10

**Students**

Ch Saiteja CB.EN.U4CSE16413

Rahul N CB.EN.U4CSE16441

Vishal N CB.EN.U4CSE16463

Hariganesh S CB.EN.U4CSE16417

Chenthur V CB.EN.U4CSE16412

Table of Contents

Abstract

The project focuses on creating a software application that caters to an important aspect of Event Management. The software developed by our team is a mobile application that provides a platform for people in close proximities to connect to each other, through a variety of events.

Preview for the Project

**Introduction**

The software application enables the user to perform two essential functions:

* Post events in order to augments his/her popularity in the locality
* View events in close proximity that cater to his/her interest.

The software application also provides a platform for publicity as popular event organizers are authenticated and enabled to post their events at a national scale.

**Motivation and Need**

An event management system that caters to the above mentioned purpose is necessary as this software application can be an opportunity to many people who are devoid of a platform to showcase their capability and passion.

The purpose behind designing the application is to:

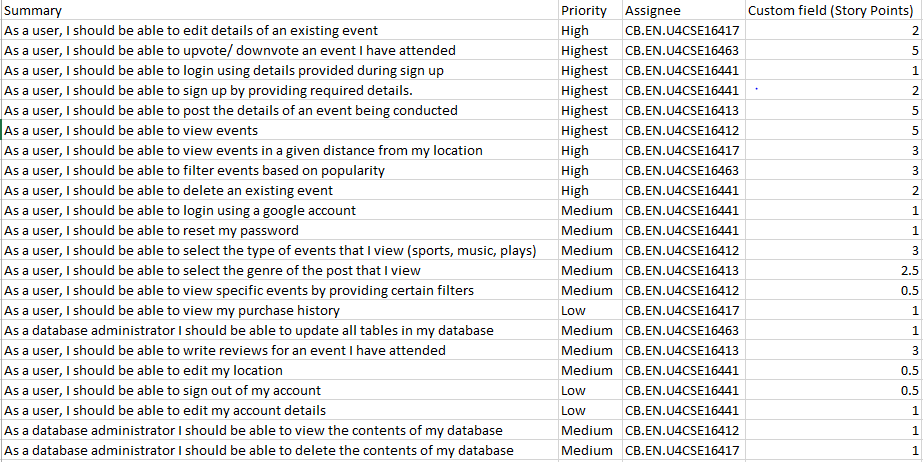
* Encourage social interactions among people in nearby localities
* Provide a platform for people to gain recognition
* Foster enthusiasm among people who wish to explore and learn new stuff
* Help people connect to popular event festivals

**Tools Used**

* Front End: React Native (for Development)
* Front End: Sonar Cloud, Jest (for Testing)
* Back End: Node.js (Express.js), MongoDB (for Development)
* Back End: Sonar Cloud, Mocha, Super Test (for Testing)

Product Backlog

**Report of Product Backlog (from JIRA)**



**Epic: Data Base Administration**

* As a database administrator I should be able to view the contents of my database
* As a database administrator I should be able to delete the contents of my database
* As a database administrator I should be able to update all tables in my database

**Epic: Filtering Events**

* As a user, I should be able to select the type of events that I view
* As a user, I should be able to select the genre of the post that I view
* As a user, I should be able to view events in a given distance from my location
* As a user, I should be able to filter events based on popularity

**Epic: Login/Sign up Page**

* As a user, I should be able to reset my password
* As a user, I should be able to sign up by providing required details
* As a user, I should be able to login using details provided during sign up
* As a user, I should be able to login using a google account

**Epic: Posting an Event**

* As a user, I should be able to delete an existing event
* As a user, I should be able to edit details of an existing event
* As a user, I should be able to post the details of an event being conducted
* As a user, I should be able to sign out of my account

**Epic: Provide options to view and edit user profile**

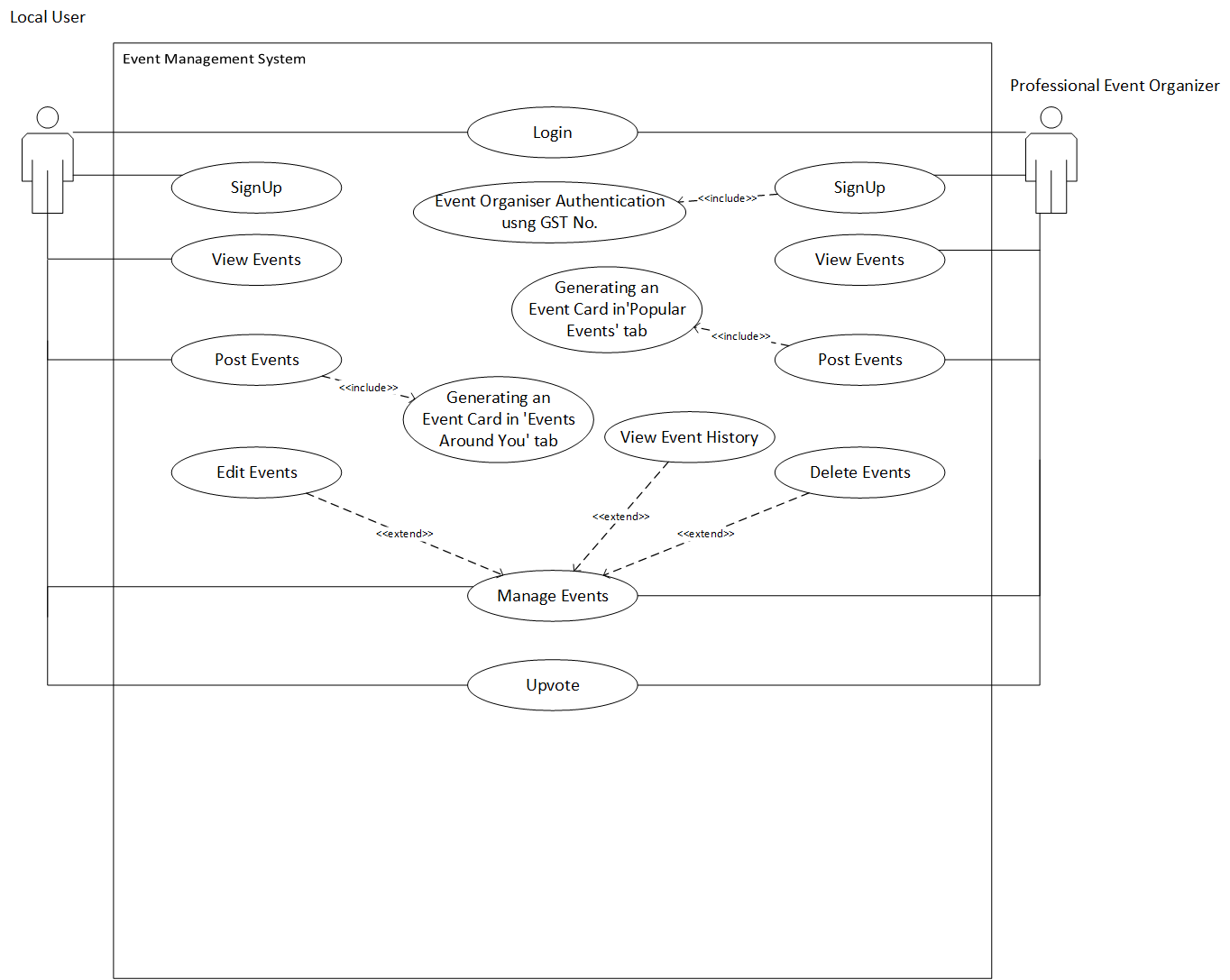
* As a user, I should be able to edit my location
* As a user, I should be able to view my purchase history
* As a user, I should be able to edit my account details
* As a user, I should be able to upvote/downvote an event I have attended

**Epic: Viewing Events**

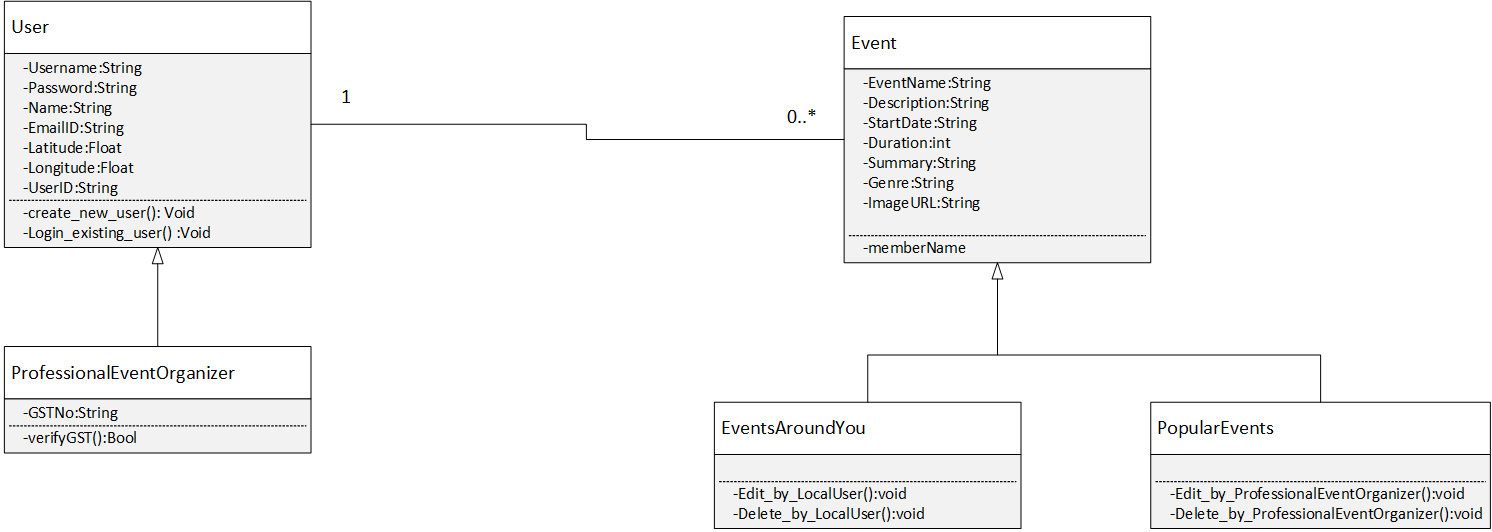
* As a user, I should be able to write reviews for an event I have attended
* As a user, I should be able to view specific events by providing certain filters
* As a user, I should be able to view events

Diagrams

**Use Case Diagram**

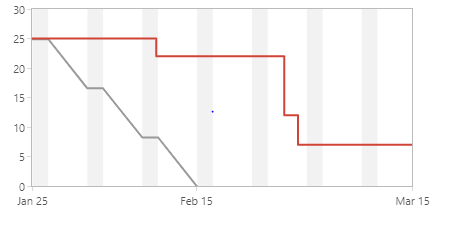
****

**Class Diagram**

****

Scrum Activities

**Sprint 1 Backlog**



|  |  |
| --- | --- |
| [CEEMS-25] [As a user, I should be able to upvote/ downvote an event I have attended](http://jira.amritanet.edu:8080/browse/CEEMS-25) Created: 22/Jan/19  Updated: 28/Feb/19  Resolved: 28/Feb/19 | |
| **Status:** | Done |
| **Project:** | [CSE E EVENT MANAGEMENT SYS](http://jira.amritanet.edu:8080/secure/BrowseProject.jspa?id=10162) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Story | **Priority:** | Highest |
| **Reporter:** | [CB.EN.U4CSE16413](http://jira.amritanet.edu:8080/secure/ViewProfile.jspa?name=CB.EN.U4CSE16413) | **Assignee:** | [CB.EN.U4CSE16463](http://jira.amritanet.edu:8080/secure/ViewProfile.jspa?name=CB.EN.U4CSE16463) |
| **Resolution:** | Done | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | CEEMS Sprint 1 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This feature acts as a review system, enabling users to share their review on what they felt about the event. |

|  |  |
| --- | --- |
| [CEEMS-22] [As a user, I should be able to view events](http://jira.amritanet.edu:8080/browse/CEEMS-22) Created: 22/Jan/19  Updated: 27/Feb/19  Resolved: 27/Feb/19 | |
| **Status:** | Done |
| **Project:** | [CSE E EVENT MANAGEMENT SYS](http://jira.amritanet.edu:8080/secure/BrowseProject.jspa?id=10162) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Story | **Priority:** | Highest |
| **Reporter:** | [CB.EN.U4CSE16413](http://jira.amritanet.edu:8080/secure/ViewProfile.jspa?name=CB.EN.U4CSE16413) | **Assignee:** | [CB.EN.U4CSE16412](http://jira.amritanet.edu:8080/secure/ViewProfile.jspa?name=CB.EN.U4CSE16412) |
| **Resolution:** | Done | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | CEEMS Sprint 1 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The system empowers users to view several events available on the application.  These events can be viewed based on filters. |

|  |  |
| --- | --- |
| [CEEMS-13] [As a user, I should be able to post the details of an event being conducted](http://jira.amritanet.edu:8080/browse/CEEMS-13) Created: 22/Jan/19  Updated: 27/Feb/19  Resolved: 27/Feb/19 | |
| **Status:** | Done |
| **Project:** | [CSE E EVENT MANAGEMENT SYS](http://jira.amritanet.edu:8080/secure/BrowseProject.jspa?id=10162) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Story | **Priority:** | Highest |
| **Reporter:** | [CB.EN.U4CSE16413](http://jira.amritanet.edu:8080/secure/ViewProfile.jspa?name=CB.EN.U4CSE16413) | **Assignee:** | [CB.EN.U4CSE16413](http://jira.amritanet.edu:8080/secure/ViewProfile.jspa?name=CB.EN.U4CSE16413) |
| **Resolution:** | Done | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | CEEMS Sprint 1 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The system enables users to post relevant details of an event they wish to conduct.   * Location * Category * Event Agenda * Event Fee |

|  |  |
| --- | --- |
| [CEEMS-5] [As a user, I should be able to sign up by providing required details.](http://jira.amritanet.edu:8080/browse/CEEMS-5) Created: 22/Jan/19  Updated: 10/Feb/19  Resolved: 10/Feb/19 | |
| **Status:** | Done |
| **Project:** | [CSE E EVENT MANAGEMENT SYS](http://jira.amritanet.edu:8080/secure/BrowseProject.jspa?id=10162) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Story | **Priority:** | Highest |
| **Reporter:** | [CB.EN.U4CSE16413](http://jira.amritanet.edu:8080/secure/ViewProfile.jspa?name=CB.EN.U4CSE16413) | **Assignee:** | [CB.EN.U4CSE16441](http://jira.amritanet.edu:8080/secure/ViewProfile.jspa?name=CB.EN.U4CSE16441) |
| **Resolution:** | Done | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

|  |  |
| --- | --- |
| **Sprint:** | CEEMS Sprint 1 |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| This user story enables users to authenticate themselves in the simplest manner using personal details |

|  |  |
| --- | --- |
| [CEEMS-4] [As a user, I should be able to login using details provided during sign up](http://jira.amritanet.edu:8080/browse/CEEMS-4) Created: 22/Jan/19  Updated: 10/Feb/19  Resolved: 10/Feb/19 | |
| **Status:** | Done |
| **Project:** | [CSE E EVENT MANAGEMENT SYS](http://jira.amritanet.edu:8080/secure/BrowseProject.jspa?id=10162) |
| **Component/s:** | None |
| **Affects Version/s:** | None |
| **Fix Version/s:** | None |

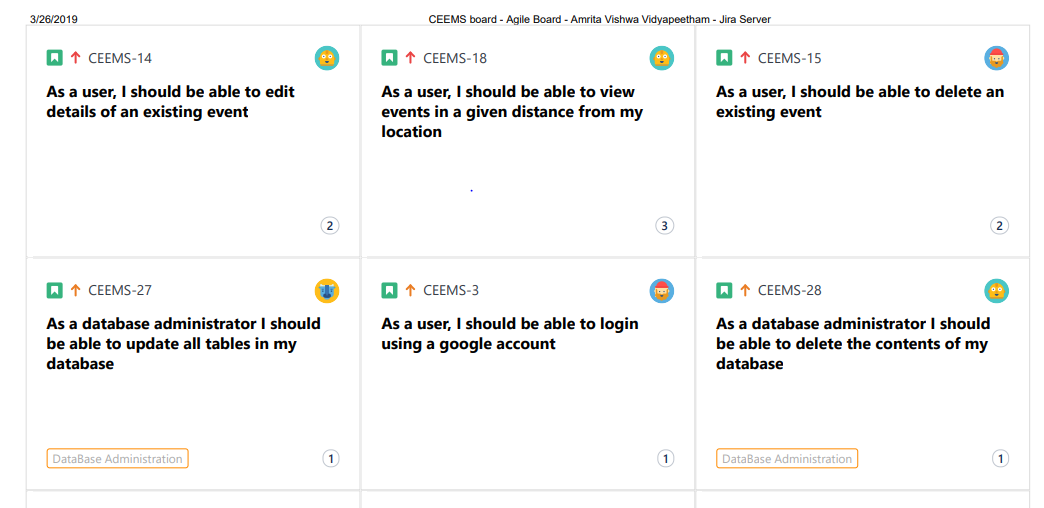
|  |  |  |  |
| --- | --- | --- | --- |
| **Type:** | Story | **Priority:** | Highest |
| **Reporter:** | [CB.EN.U4CSE16413](http://jira.amritanet.edu:8080/secure/ViewProfile.jspa?name=CB.EN.U4CSE16413) | **Assignee:** | [CB.EN.U4CSE16441](http://jira.amritanet.edu:8080/secure/ViewProfile.jspa?name=CB.EN.U4CSE16441) |
| **Resolution:** | Done | **Votes:** | 0 |
| **Labels:** | None | | |
| **Remaining Estimate:** | Not Specified | | |
| **Time Spent:** | Not Specified | | |
| **Original Estimate:** | Not Specified | | |

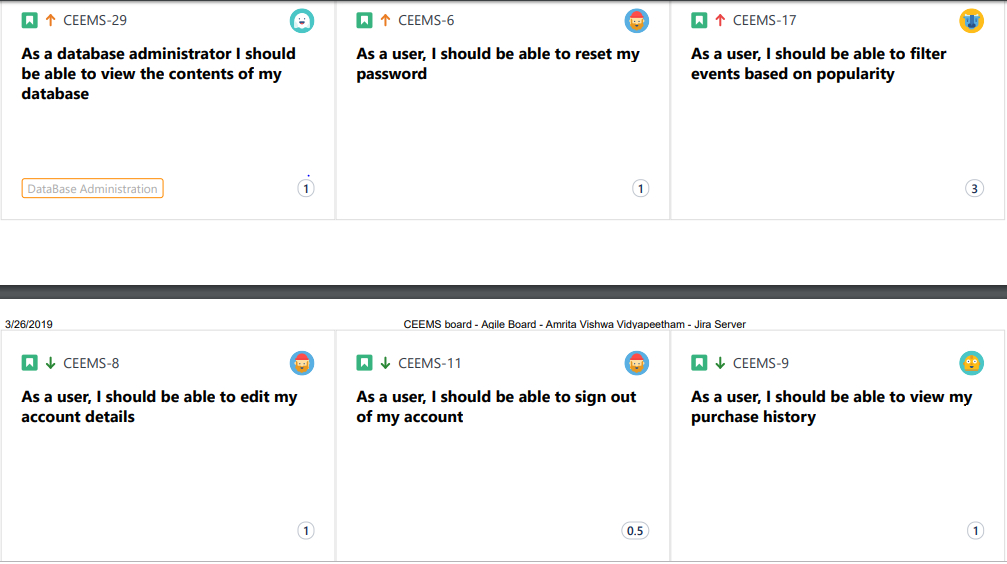
|  |  |
| --- | --- |
| **Sprint:** | CEEMS Sprint 1 |

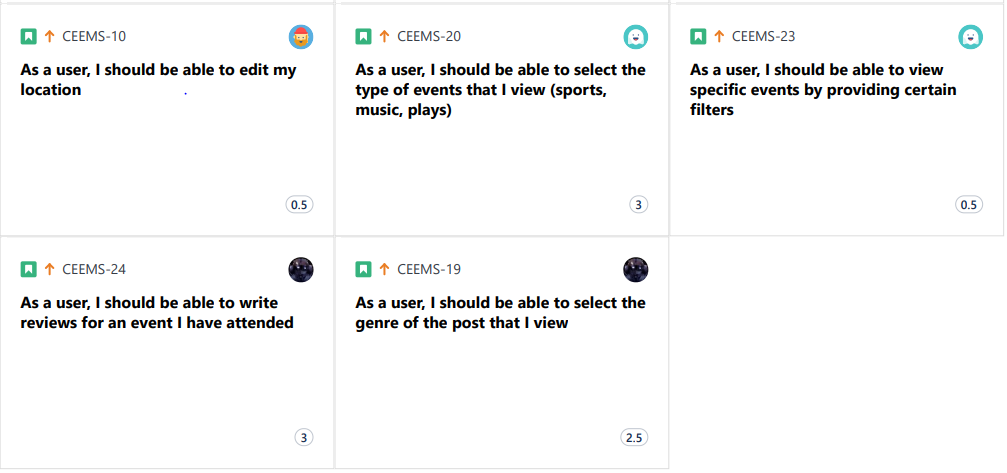
|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| The system also allows users to use be authenticated within our application even if they are devoid of a google account |

**Sprint 2 Backlog**







Implementation (Sample Code)

Backend

1. let express = require("express");
2. let app = express();
3. let port = 3000;
5. let bodyParser = require('body-parser');
6. app.use(bodyParser.json({
7. limit: "50mb"
8. }));
9. app.use(bodyParser.urlencoded({
10. limit: "50mb",
11. extended: **true**,
12. parameterLimit: 50000
13. }));

16. let mongoose = require("mongoose");
17. mongoose.Promise = global.Promise;
18. mongoose.connect("mongodb://localhost:27017/EMS-DB", {
19. useNewUrlParser: **true**
20. });
22. mongoose.connection.on('connected', **function** () {
23. console.log("Mongoose default connection open to mongodb://localhost:27017/EMS-DB");
24. });
26. let nameSchema = **new** mongoose.Schema({
27. name: String,
28. emailID: String,
29. gstNumber: String,
30. userName: String,
31. passwordHash: String
32. });
34. let eventSchema = **new** mongoose.Schema({
35. title: String,
36. event\_summary: String,
37. duration: Number,
38. description: String,
39. price: Number,
40. genre: String,
41. start\_date: Date,
42. latitude: Number,
43. longitude: Number,
44. userID: String,
45. image\_url: String
46. });
48. let ImageSchema = **new** mongoose.Schema({
49. category: String,
50. image\_url: String
51. });
53. let GSTSchema = **new** mongoose.Schema({
54. GSTnumber: String
55. });
57. let User = mongoose.model("User", nameSchema);
58. let GSTid = mongoose.model("GSTid", GSTSchema);
60. let localEvents = mongoose.model('local-events', eventSchema);
61. let globalEvents = mongoose.model('global-events', eventSchema);
63. let Images = mongoose.model('images', ImageSchema);

66. **function** distance(lat1, lon1, lat2, lon2) {
67. **if** ((lat1 == lat2) && (lon1 == lon2)) {
68. **return** 0;
69. } **else** {
70. let radlat1 = Math.PI \* lat1 / 180;
71. let radlat2 = Math.PI \* lat2 / 180;
72. let theta = lon1 - lon2;
73. let radtheta = Math.PI \* theta / 180;
74. let dist = Math.sin(radlat1) \* Math.sin(radlat2) + Math.cos(radlat1) \* Math.cos(radlat2) \* Math.cos(radtheta);
75. **if** (dist > 1) {
76. dist = 1;
77. }
78. dist = Math.acos(dist);
79. dist = dist \* 180 / Math.PI;
80. dist = dist \* 60 \* 1.1515;
82. dist = dist \* 1.609344;
84. **return** dist;
85. }
86. }
88. app.post("/adduser", (req, res) => {
90. console.log('add user');
91. //console.log(req.body);

94. GSTid.find({
95. gstNumber: req.body.gstNumber
96. }, (err, docs) => {
97. **if** (docs.length > 0) {
99. User.find({
100. userName: req.body.userName
101. }).lean().exec((err, docs) => {
102. **if** (docs.length == 0) {
103. let myData = **new** User(req.body);
105. myData.save()
106. .then(item => {
107. let respObj = {
108. respMsg: "user\_saved",
109. userID: item.\_id
110. };
111. res.json(respObj);
112. })
113. .**catch**(err => {
114. let respObj = {
115. respMsg: "unable\_to\_save"
116. };
117. res.json(respObj);
118. });
119. } **else** {
120. let respObj = {
121. respMsg: "username\_exists"
122. };
123. res.json(respObj);
124. }
125. });
127. } **else** {
128. let respObj = {
129. respMsg: "unknown\_gst\_number"
130. };
131. res.send(respObj);
132. }
133. });
134. });

Front-End Implementation.

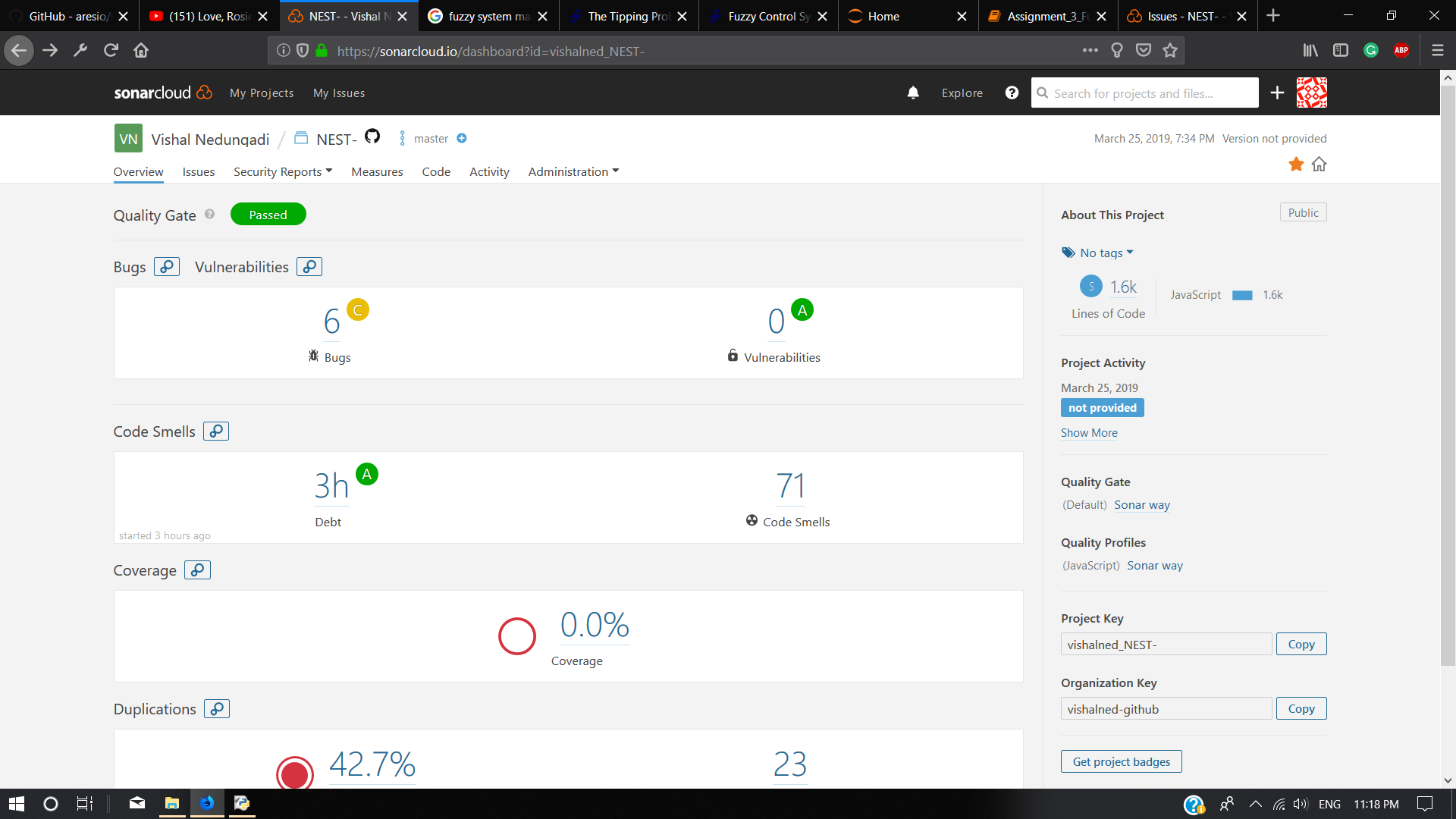
React Native lets you build mobile apps using only JavaScript.

1. **import** React, { Component } from "react";
2. **import** {
3. Platform,
4. StyleSheet,
5. Text,
6. View,
7. TextInput,
8. TouchableOpacity,
9. Alert,
10. AsyncStorage
11. } from "react-native";
13. //import Signupform from './components/signup'
14. // import {createStackNavigator} from 'react-navigation';
16. **import** { Container, Header, Content, Button } from "native-base";
17. **import** App from "../../../App";
18. **import** DefaultPreference from "react-native-default-preference";
19. **import** NavigationService from "./../../../NavigationService";
20. **import** Loader from "../loader";
21. **var** hash = require("hash.js");
23. **export** **default** **class** LoginForm **extends** Component {
24. constructor() {
25. **super**();
26. **this**.state = {
27. username: "",
28. password: "",
29. isloading: **false**
30. };
31. **this**.getFromFrom = **this**.getFromFrom.bind(**this**);
32. **this**.onChangeUser = **this**.onChangeUser.bind(**this**);
33. **this**.onChangePwd = **this**.onChangePwd.bind(**this**);
34. }
36. componentWillMount() {
37. //console.warn("going")
38. DefaultPreference.get("userID").then(**function**(value) {
39. //console.warn(value);
40. **if** (value != **null**) {
41. NavigationService.navigate("Dashboard");
42. }
43. });
44. }
46. onChange1(e) {
47. // console.warn(e)
48. **this**.setState({
49. [e.target.name]: e
50. });
51. }
53. onChangeUser(e) {
54. **this**.setState({ username: e });
55. }
56. onChangePwd(e) {
57. **this**.setState({
58. password: hash
59. .sha256()
60. .update(e)
61. .digest("hex")
62. });
63. }
65. sendRequest(data) {
66. **this**.setState({ isloading: **true** });
67. fetch("http://192.168.43.209:3000/validateuser", {
68. method: "POST",
69. headers: {
70. Accept: "application/json",
71. "Content-Type": "application/json"
72. },
73. body: JSON.stringify(data)
74. })
75. .then(res => **this**.handleResponse(res))
76. .**catch**(err => console.warn(err));
77. }
79. handleResponse(response) {
80. **this**.setState({ isloading: **false** });
82. **var** bodyText = response["\_bodyText"];
83. **var** jsObj = JSON.parse(bodyText);
84. **var** response\_msg = jsObj.respMsg;
85. // console.warn(response\_msg)
86. **if** (response\_msg == "valid\_user") {
87. //User Saved
88. **var** user\_id = jsObj.userID;
89. // console.warn(user\_id)
91. //SharedPreferences.setItem('userID', user\_id)
92. // Works on both iOS and Android
94. DefaultPreference.set("userID", user\_id).then(**function**() {
95. console.warn("done");
96. });
97. Alert.alert("Login Succesful!!", " ", [{ text: "OK" }], {
98. cancelable: **false**
99. });
101. **this**.props.navigation.navigate("Dashboard");
102. } **else** **if** (response\_msg == "wrong\_password") {
103. //pls try again, server messed up
104. Alert.alert(
105. "Entered Wrong Username/Password",
106. "Please Try Again@",
107. [{ text: "OK" }],
108. { cancelable: **false** }
109. );
110. } **else** **if** (response\_msg == "unknown\_user") {
111. //pls try again, server messed up
112. Alert.alert(
113. "Unkown User",
114. "You are not a registerd User ,Please Signup",
115. [{ text: "OK" }],
116. { cancelable: **false** }
117. );
118. }
119. //SharedPreferences.setItem('userID', 'John13231231232132312321')
120. }
122. getFromFrom() {
123. **const** post\_data = {
124. userName: **this**.state.username,
125. passwordHash: **this**.state.password
126. };
127. console.warn(post\_data);
128. **this**.sendRequest(post\_data);
129. }
131. render() {
132. **return** (
133. <View style={styles.container}>
134. <Loader loading={**this**.state.isloading} />
135. <View style={styles.login}>
136. <Text style={styles.header}>Login Form</Text>
137. <TextInput
138. name="username"
139. style={styles.textinput}
140. placeholder="Username"
141. placeholderTextColor="#000"
142. onChangeText={text => **this**.setState({ username: text })}
143. />
144. <TextInput
145. name="password"
146. style={styles.textinput}
147. placeholder="Password"
148. placeholderTextColor="#000"
149. secureTextEntry={**true**}
150. onChangeText={text =>
151. **this**.setState({
152. password: hash
153. .sha256()
154. .update(text)
155. .digest("hex")
156. })
157. }
158. />
160. {/\* <TouchableOpacity style={styles.button}>
161. <Text style = {styles.btntext}>Login in</Text>
162. </TouchableOpacity> \*/}
163. <Button style={styles.button} onPress={**this**.getFromFrom}>
164. <Text style={styles.btntext}>Login</Text>
165. </Button>
167. <Button
168. style={styles.button}
169. onPress={() => **this**.props.navigation.navigate("Signup")}
170. >
171. <Text style={styles.btntext}>Dont have an account? Sign up</Text>
172. </Button>
173. </View>
174. </View>
175. );
176. }
177. }
179. **const** styles = StyleSheet.create({
180. container: {
181. flex: 1,
182. justifyContent: "center",
183. // alignItems: 'center',
184. backgroundColor: "#36485f",
185. paddingLeft: 60,
186. paddingRight: 60
187. },
189. login: {
190. alignSelf: "stretch"
191. },
192. header: {
193. fontSize: 24,
194. color: "#ffffff",
195. paddingBottom: 10,
196. marginBottom: 40,
197. borderBottomColor: "#199187",
198. borderBottomWidth: 1
199. },
200. textinput: {
201. alignSelf: "stretch",
202. height: 40,
203. marginBottom: 30,
204. color: "#ffffff",
205. borderBottomColor: "#f8f8f8",
206. borderBottomWidth: 1
207. },
208. button: {
209. alignSelf: "stretch",
210. alignItems: "center",
211. padding: 20,
212. backgroundColor: "#59cbbd",
213. marginTop: 30,
214. justifyContent: "center"
215. },
216. btntext: {
217. color: "#fff",
219. fontWeight: "bold"
220. }
221. });
222. **import** React, { Component } from "react";
223. **import** {
224. Platform,
225. StyleSheet,
226. Text,
227. View,
228. TextInput,
229. FlatList
230. } from "react-native";
231. **import** { NavigationEvents } from "react-navigation";
232. **import** ViewEditCard from "../Dashboard/ViewEditEvent/ViewEditCard.js";
233. //import getEvents from '../Dashboard/ViewEvents/getEvents.js'
234. **import** DefaultPreference from "react-native-default-preference";
236. **export** **default** **class** User\_Events **extends** Component {
237. constructor(props) {
238. **super**(props);
239. **this**.state = { events: [], refreshing: **true** };
240. **this**.fetchEvents = **this**.fetchEvents.bind(**this**);
241. }
243. async componentDidMount() {
244. console.warn(**this**.props);
245. **const** uid = await DefaultPreference.get("userID");
246. //console.warn(uid);
247. **this**.setState({ userID: uid });
248. navigator.geolocation.getCurrentPosition(
249. position => {
250. **this**.setState({
251. latitude: position.coords.latitude,
252. longitude: position.coords.longitude,
254. error: **null**
255. });
256. },
257. error => console.warn(error.message),
258. {
259. enableHighAccuracy: **false**,
260. timeout: 5000,
261. maximumAge: 10000
262. }
263. );
265. **this**.fetchEvents();
266. }
268. fetchEvents() {
269. **const** params = {
270. userID: **this**.state.userID,
271. type: "self"
272. };
273. **this**.getEvents(params);
274. }
276. getEvents(params) {
277. **if** (params["type"] == "global") {
278. url = "http://192.168.43.209:3000/getglobalevents";
279. } **else** **if** (params["type"] == "local") {
280. url = "http://192.168.43.209:3000/getlocalevents";
281. } **else** **if** (params["type"] == "self") {
282. url = "http://192.168.43.209:3000/getuserevents";
283. } **else** {
284. }
286. fetch(url, {
287. method: "POST",
288. headers: {
289. Accept: "application/json",
290. "Content-Type": "application/json"
291. },
292. body: JSON.stringify(params)
293. })
294. .then(res =>
295. **this**.setState({
296. events: JSON.parse(res["\_bodyText"]).event\_list,
297. refreshing: **false**
298. })
299. )
300. .then(console.log(**this**.state.events))
301. .**catch**(err => console.warn(err));
302. }
304. handleRefresh() {
305. **this**.setState(
306. {
307. refreshing: **true**
308. },
309. () => **this**.fetchEvents()
310. );
311. }
313. render() {
314. **return** (
315. <FlatList
316. data={**this**.state.events}
317. renderItem={({ item }) => (
318. <ViewEditCard
319. article={item}
320. nav={**this**.props.navigation}
321. parentFlatlist={**this**}
322. />
323. )}
324. keyExtractor={item => item.\_id}
325. refreshing={**this**.state.refreshing}
326. onRefresh={**this**.handleRefresh.bind(**this**)}
327. />
328. );
329. }
330. }
331. **import** React, { Component } from "react";
332. **import** {
333. Image,
334. View,
335. StyleSheet,
336. Dimensions,
337. ImageBackground,
338. TouchableNativeFeedback,
339. Alert,
340. ActivityIndicator
341. } from "react-native";
342. **import** {
343. Container,
344. Header,
345. Content,
346. Card,
347. CardItem,
348. Thumbnail,
349. Text,
350. Button,
351. Icon,
352. Left,
353. Body,
354. ActionSheet
355. } from "native-base";
356. **import** NavigationService from "./../../../../NavigationService";
357. **import** Swipeout from "react-native-swipeout";
358. **import** Loader from "../../loader";
359. require("datejs");
360. **export** **default** **class** ViewEditCard **extends** Component {
361. constructor() {
362. **super**();
363. **this**.state = { count: 0, loading: **false** };
364. **this**.deleteEvent = **this**.deleteEvent.bind(**this**);
365. }
367. \_incrementCount() {
368. **this**.setState = {
369. count: ~**this**.state.count
370. };
371. }
373. deleteEvent(\_id, userID) {
374. console.warn(\_id);
376. **const** params = {
377. \_id: \_id,
378. userID: userID
379. };
380. **this**.setState({ loading: **true** });
382. fetch("http://192.168.43.209:3000/deleteevent", {
383. method: "POST",
384. headers: {
385. Accept: "application/json",
386. "Content-Type": "application/json"
387. },
388. body: JSON.stringify(params)
389. })
390. .then(res => console.warn(res))
392. .**catch**(err => console.warn(err));
393. }
395. render() {
396. /\*
397. 'userID':'5c75a2ec12ce5d19d055ea2e',
398. 'title':this.state.title,
399. 'event\_summary':this.state.summary,
400. 'duration':this.state.days,
401. 'description':this.state.price,
402. 'genre': this.state.genre,
403. 'start\_date':this.state.chosenDate,
404. 'image\_base64':this.state.photo\_data,
405. 'latitude':this.state.latitude,
406. 'longitude':this.state.longitude
407. \*/
408. **const** nav = **this**.props.nav;
409. **const** art = **this**.props.article;
410. **const** price = "₹" + art.price;
411. **const** date = art.start\_date;
412. **const** art\_key = art.\_id;
413. **const** uid = art.userID;
414. **const** summary = art.event\_summary;
415. **const** genre = art.genre;
416. **const** title = art.title;
417. **const** distance = art.distance;
418. **const** img\_url = art.image\_url;
419. **const** prop = **this**.props.parentFlatlist;
420. **const** swipeSettings = {
421. autoClose: **true**,
422. onClose: (secId, rowId, direction) => {},
423. onOpen: (secId, rowId, direction) => {},
424. right: [
425. {
426. onPress: () => {
427. NavigationService.navigate("EditEventsPage", { pre\_data: art });
428. prop.handleRefresh();
429. },
430. text: "Edit  ",
431. type: "primary"
432. },
433. {
434. onPress: () => {
435. Alert.alert(
436. "Confirmation",
437. "Are you sure you want to delete this post?",
438. [
439. {
440. text: "NO",
441. onPress: () => console.log("Cancel Pressed"),
442. style: "cancel"
443. },
444. {
445. text: "YES",
446. onPress: () => {
447. **this**.deleteEvent(art\_key, uid);
448. prop.handleRefresh();
449. }
450. }
451. ],
452. { cancelable: **false** }
453. );
454. },
455. text: "Delete",
456. type: "delete"
457. }
458. ],
459. rowId: art\_key,
460. sectionId: 1
461. };
462. **return** (
463. <Swipeout {...swipeSettings}>
464. <TouchableNativeFeedback
465. useForeground
466. onPress={() => console.warn(art)}
467. >
468. <Card style={styles.container}>
469. <CardItem>
470. <Left>
471. <Body>
472. <Text>{title}</Text>
473. <Text note>
474. {Date.parse(date.slice(0, 10)).toString("D")}
475. </Text>
476. </Body>
477. </Left>
478. </CardItem>
479. <CardItem>
480. <View>
481. <ImageBackground
482. style={styles.mainimage}
483. source={{ uri: img\_url }}
484. >
485. <Text
486. style={{
487. bottom: -90,
488. textAlign: "bottom",
489. textAlign: "center",
490. fontSize: 25
491. }}
492. >
493. {title}
494. </Text>
495. </ImageBackground>
496. </View>
497. </CardItem>
498. <CardItem>
499. <Body>
500. <Text>{summary}</Text>
501. </Body>
502. </CardItem>
503. <CardItem>
504. <Left>
505. <Body>
506. <Text style={{ color: "#EB4958" }}>
507. <Image
508. style={styles.image1}
509. source={require("../../../images/images/chef.png")}
510. />
511. {genre}
512. </Text>
513. <Text
514. style={{
515. marginLeft: 7,
516. marginBottom: 5,
517. marginTop: 10,
518. color: "#228B22"
519. }}
520. >
521. Distance: {distance} km
522. </Text>
523. </Body>
524. </Left>
525. <View style={{ flexDirection: "row" }}>
526. <Text>{price}</Text>
527. </View>
528. </CardItem>
529. </Card>
530. </TouchableNativeFeedback>
531. </Swipeout>
532. );
533. }
534. }
535. **var** styles = StyleSheet.create({
536. description: {
537. marginBottom: 20,
538. fontSize: 18,
539. textAlign: "center",
540. color: "#656565"
541. },
542. container: {
543. padding: 30,
544. marginTop: 65,
545. alignItems: "center"
546. },
547. flowRight: {
548. flexDirection: "row",
549. alignItems: "center",
550. alignSelf: "stretch"
551. },
552. buttonText: {
553. fontSize: 18,
554. color: "white",
555. alignSelf: "center"
556. },
557. button: {
558. height: 36,
559. flex: 1,
560. flexDirection: "row",
561. backgroundColor: "#48BBEC",
562. borderColor: "#48BBEC",
563. borderWidth: 1,
564. borderRadius: 8,
565. marginBottom: 10,
566. alignSelf: "stretch",
567. justifyContent: "center"
568. },
569. searchInput: {
570. height: 36,
571. padding: 4,
572. marginRight: 5,
573. flex: 4,
574. fontSize: 18,
575. borderWidth: 1,
576. borderColor: "#48BBEC",
577. borderRadius: 8,
578. color: "#48BBEC"
579. },
580. mainimage: {
581. width: Dimensions.get("window").width - 37,
582. height: 150,
583. alignSelf: "center",
584. textAlign: "center"
585. },
586. container: {
587. marginTop: 5,
588. borderRadius: 5
589. },
590. image: {
591. width: 20,
592. height: 20,
593. resizeMode: "contain"
594. },
595. image1: {
596. width: 20,
597. height: 20
598. }
599. });
600. **return** (
601. <Modal
602. transparent={**true**}
603. animationType={"none"}
604. visible={loading}
605. onRequestClose={() => {
606. console.log("close modal");
607. }}
608. >
609. <View style={styles.modalBackground}>
610. <View style={styles.activityIndicatorWrapper}>
611. <ActivityIndicator animating={loading} />
612. </View>
613. </View>
614. </Modal>
615. );

Software Engineering Tools Used

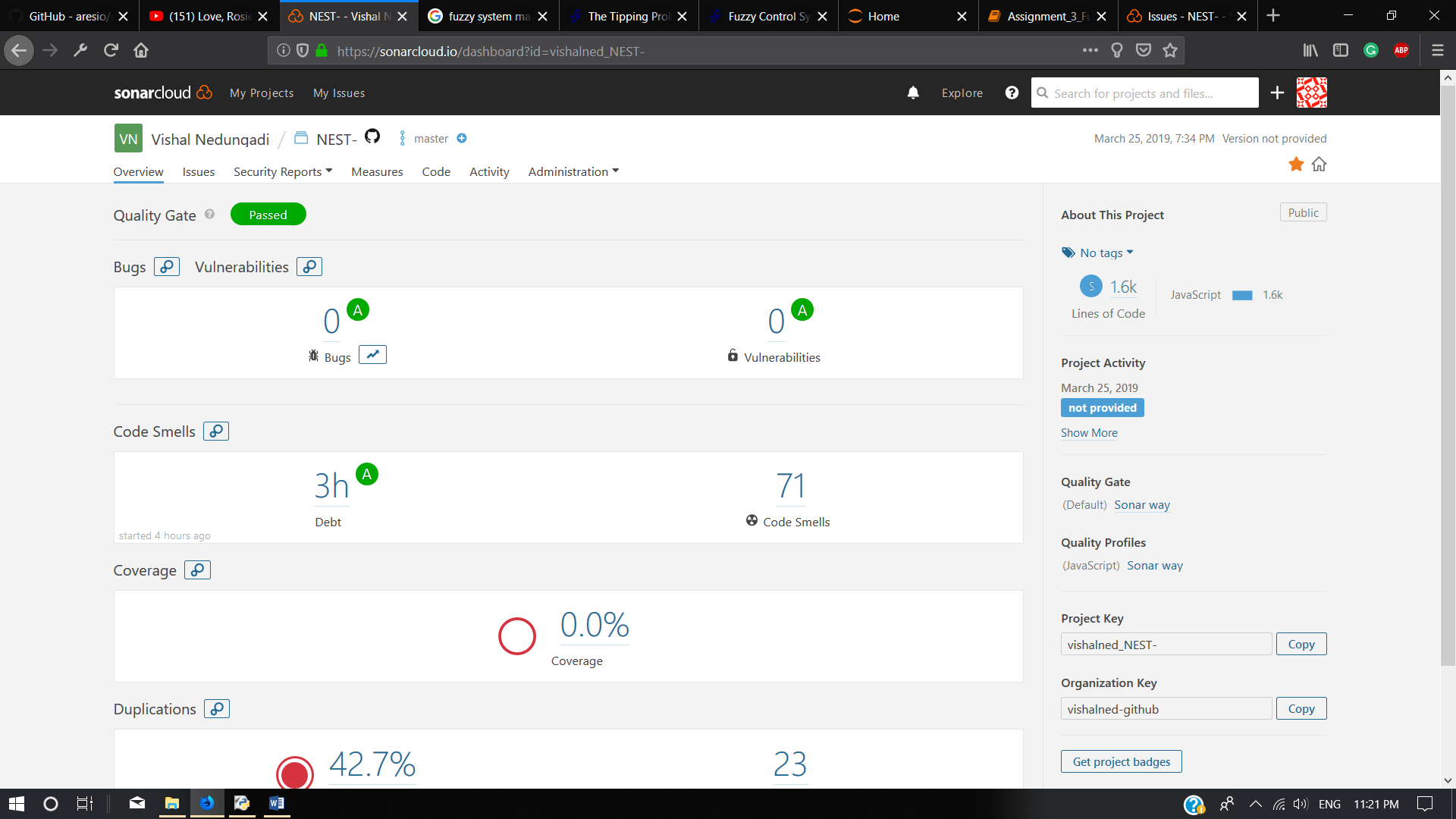
**Frontend Static Code Analysis**

Initial Analysis

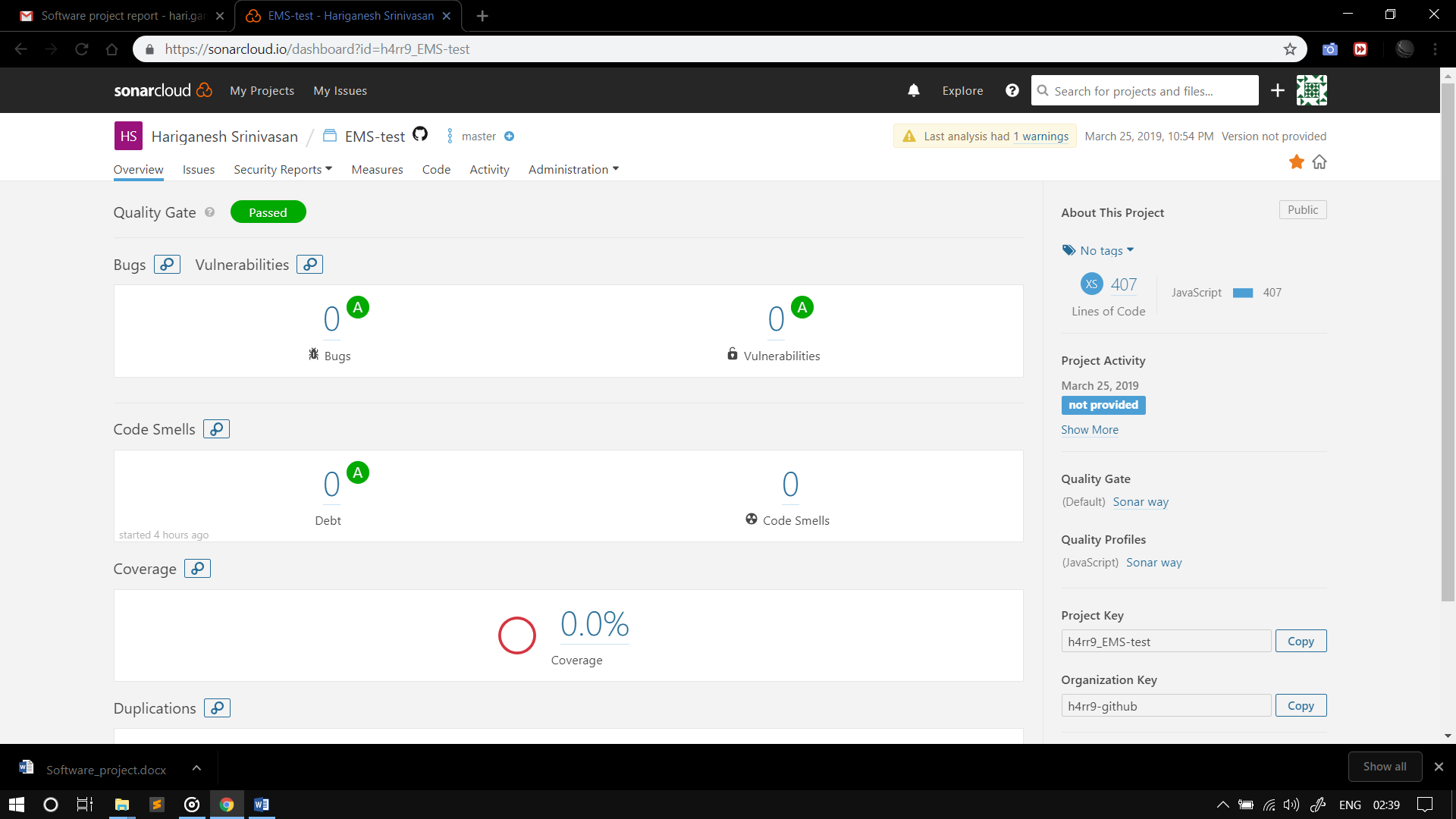


The code was modified and the bugs were fixed.

Final Code Analysis



**Backend Static Code Analysis**

****

No Issues were found, therefore no rectification required.

**Backend Unit Testing**

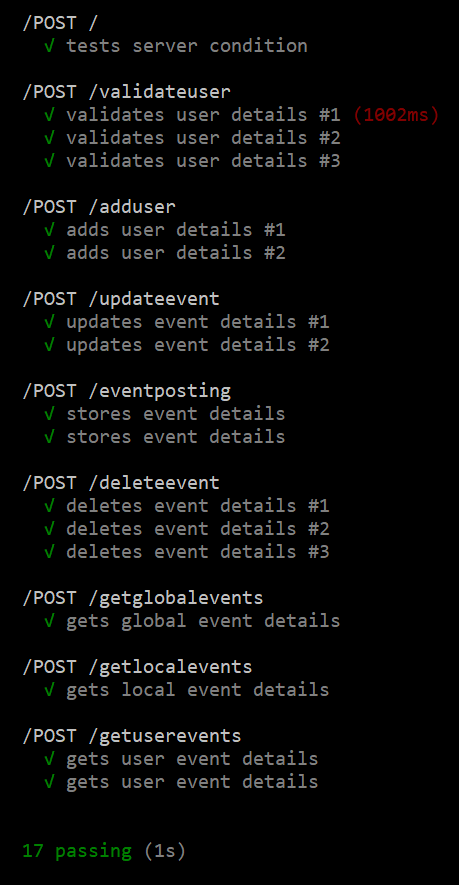
Mocha and supertest were used to perform unit testing on the server code.

1. **var** request = require('supertest');
2. **var** app = require('./app')
4. process.env.NODE\_ENV = 'test';

7. // init route
8. describe('/POST /', **function** (){
9. it('tests server condition', **function** (done){
10. request(app).get('/')
11. .expect(200, done);
12. });
13. });

16. // validate user route
17. describe('/POST /validateuser', **function** (){
18. let data\_1 = {
19. "name" : "Johnny",
20. "emailID" :"Johnny@gmail.com",
21. "userName" : "Johnny",
22. "gstNumber" :"AB12345XY",
23. "passwordHash" : "randomText"
24. };
26. let data\_2 = {
27. "name" : "Johnny",
28. "emailID" :"Johnny@gmail.com",
29. "userName" : "Johnx",
30. "gstNumber" :"AB12345XY",
31. "passwordHash" : "randomText"
32. };
34. let data\_3 = {
35. "name" : "Johnny",
36. "emailID" :"Johnny@gmail.com",
37. "userName" : "Johnny",
38. "gstNumber" :"AB12345XY",
39. "passwordHash" : "randomTexts"
40. };
42. let ret\_1 = {
43. "respMsg":"valid\_user",
44. "userID":"5c98e0665ba62521ec48e42f"
45. };
47. let ret\_2 = {
48. "respMsg":"unknown\_user"
49. }
51. let ret\_3 = {
52. "respMsg": "wrong\_password"
53. }
55. it('validates user details #1', **function** (done){
56. request(app).post('/validateuser')
57. .send(data\_1)
58. .set('Accept', 'application/json')
59. .expect(ret\_1, done);
60. });
62. it('validates user details #2', **function** (done){
63. request(app).post('/validateuser')
64. .send(data\_2)
65. .set('Accept', 'application/json')
66. .expect(ret\_2, done);
67. });
69. it('validates user details #3', **function** (done){
70. request(app).post('/validateuser')
71. .send(data\_3)
72. .set('Accept', 'application/json')
73. .expect(ret\_3, done);
74. });
75. });
77. // add user route
78. describe('/POST /adduser', **function** (){
79. let data\_1 = {
80. "name" : "Johnny",
81. "emailID" :"Johnny@gmail.com",
82. "userName" : "Johnny",
83. "gstNumber" :"AB12345XY",
84. "passwordHash" : "randomText"
85. };
87. let data\_2 = {
88. "name" : "Johnny",
89. "emailID" :"Johnny@gmail.com",
90. "userName" : "Johnny",
91. "gstNumber" :"AB12345XC",
92. "passwordHash" : "randomText"
93. };
95. let ret\_1 = {
96. "respMsg":"username\_exists"
97. };
99. let ret\_2 = {
100. "respMsg":"unknown\_gst\_number"
101. };
103. it('adds user details #1', **function** (done){
104. request(app).post('/adduser')
105. .send(data\_1)
106. .set('Accept', 'application/json')
107. .expect(ret\_1, done);
108. });
110. it('adds user details #2', **function** (done){
111. request(app).post('/adduser')
112. .send(data\_2)
113. .set('Accept', 'application/json')
114. .expect(ret\_2, done);
115. });
116. });

Similarly, other testcases have been written



The Application is passing the test cases described for all the routes that have been defined.

**Front-End Unit Testing**

The **react-test-renderer** and **Jest** libraries were used to unit test the components of the react-native app.

These libraries were installed in the JavaScript development environment of the app using yarn.

In react-native, the screens are rendered as components. Jest uses snapshots and render trees of components to test the application.

**Front-End Unit Testing Code**

The configuration for Jest:

1. **const** mockGeolocation = {
2. getCurrentPosition: jest.fn()
3. .mockImplementationOnce((success) => Promise.resolve(success({
4. coords: {
5. latitude: 51.1,
6. longitude: 45.3
7. }
8. })))
9. };
10. global.navigator.geolocation = mockGeolocation;
11. **import** React from 'react';
12. **import** navigate from './NavigationService.js';
13. **import** renderer from 'react-test-renderer';
15. describe('navigatePostEvents', **function**() {
16. it('renders a Post event component ', **function**() {
17. **const** tree = renderer.create(
18. <navigate />
19. ).toJSON();
20. expect(tree).toMatchSnapshot();
21. });
22. });
23. module.exports = {
24. // // Automatically clear mock calls and instances between every test
25. // clearMocks: true,
27. // // An array of glob patterns indicating a set of files for which coverage information should be collected
28. // collectCoverageFrom: ['src/\*\*/\*.{js,jsx,mjs}'],
29. preset:"react-native",
30. // The directory where Jest should output its coverage files
31. coverageDirectory: 'coverage',
32. reporters: [
33. "default",
34. ["./node\_modules/jest-html-reporter", {
35. "pageTitle": "Test Report"
36. }]
37. ],
39. // An array of file extensions your modules use
40. moduleFileExtensions: ['js', 'json', 'jsx','map'],
42. // The paths to modules that run some code to configure or set up the testing environment before each test
43. setupFiles: [
44. "<rootDir>/\_\_mocks\_\_/setup.js"
45. ],
46. // The test environment that will be used for testing
47. testEnvironment: 'jsdom',
49. // The glob patterns Jest uses to detect test files
50. testMatch: [ '\*\*/?(\*.)+(spec|test).js?(x)'],
52. // An array of regexp pattern strings that are matched against all test paths, matched tests are skipped
53. testPathIgnorePatterns: ['\\\\node\_modules\\\\'],


57. transformIgnorePatterns: ['/node\_modules/(?!react-navigation|react-native|expo|native-base-shoutem-theme|react-native-implementation|react-native|react-native-gesture-handler|react-navigation-stack)' ],
59. // Indicates whether each individual test should be reported during the run
60. verbose: **false**,
61. };
63. // Jest Snapshot v1, https://goo.gl/fbAQLP
65. exports[`Global\_Events component renders a SignupCompont  1`] = `
66. **<View**
67. style={
68. Object {
69. "flex": 1,
70. }
71. }
72. **>**
73. **<View**
74. style={
75. Object {
76. "backgroundColor": "skyblue",
77. "height": 60,
78. }
79. }
80. **/>**
81. **<RCTScrollView**
82. data={Array []}
83. disableVirtualization={false}
84. getItem={[Function]}
85. getItemCount={[Function]}
86. horizontal={false}
87. initialNumToRender={10}
88. keyExtractor={[Function]}
89. maxToRenderPerBatch={10}
90. numColumns={1}
91. onContentSizeChange={[Function]}
92. onEndReachedThreshold={2}
93. onLayout={[Function]}
94. onMomentumScrollEnd={[Function]}
95. onRefresh={[Function]}
96. onScroll={[Function]}
97. onScrollBeginDrag={[Function]}
98. onScrollEndDrag={[Function]}
99. refreshControl={
100. **<RefreshControlMock**
101. onRefresh={[Function]}
102. refreshing={true}
103. **/>**
104. }
105. refreshing={true}
106. renderItem={[Function]}
107. scrollEventThrottle={50}
108. stickyHeaderIndices={Array []}
109. updateCellsBatchingPeriod={50}
110. viewabilityConfigCallbackPairs={Array []}
111. windowSize={21}
112. **>**
113. **<RCTRefreshControl** **/>**
114. **<View** **/>**
115. **</RCTScrollView>**
116. **</View>**
117. `;
118. **import** React from 'react';
119. **import** Signupform from './index.js';
121. **import** renderer from 'react-test-renderer';
123. describe('Signup component', **function**() {
124. it('renders a SignupCompont ', **function**() {
125. **const** tree = renderer.create(
126. <Signupform />
127. ).toJSON();
128. expect(tree).toMatchSnapshot();
129. });
130. });

Similarly, Other components were also tested

Here is the detailed report:

**Jest Unit Testing Report**

Start: 2019-03-26 01:03:42

10 tests -- 10 passed / 0 failed / 0 pending

passed in 0.068s

*Successfully renders a User\_Events component*

**User\_Events component**

**E:\test\simplelogin\app\components\Tabs\e\_posted.test.js**

9.112s

passed in 0.139s

*Successfully renders a DashboardPage component*

**DashboardPage component**

**E:\test\simplelogin\app\components\Dashboard\index.test.js**

9.108s

passed in 0.797s

*Successfully renders a Signup component*

**Signup component**

**E:\test\simplelogin\app\components\Signup\index.test.js**

9.419s

passed in 0.00731s

**Login component** *Successfully renders a Login component*

3.017s

**E:\test\simplelogin\app\components\login\login.test.js**

passed in 2.535s

*Successfully renders a Loader component*

**Loader component**

**E:\test\simplelogin\app\components\loader.test.js**

6.825s

passed in 0.013s

*Successfully renders a Global\_Events component*

**Global\_Events component**

**E:\test\simplelogin\app\components\Tabs\e\_global.test.js**

9.204s

passed in 0.014s

*Successfully renders a Global\_Events component*

**Global\_Events component**

**E:\test\simplelogin\app\components\Tabs\e\_local.test.js**

9.309s

passed in 0.003s

*Successfully renders a getEvents component*

**getEvents component**

4.027s

**E:\test\simplelogin\app\components\Dashboard\ViewEvents\getEvents.test.js**

E:\test\simplelogin\NavigationService.test.js 4.344s

passed in 0.004s

**navigatePostEvents** *Successfully renders a navigatePostEvents Component*

passed in 0.007s

*Successfully renders a getLocalEvents component*

**getLocalEvents component**

4.015s

**E:\test\simplelogin\app\components\Dashboard\ViewEvents\getLocalEvents.test.js**

