# CHAPTER 5

**IMPLEMENTATION AND TESTING**



## Implementation

The Coach Me Personal Mobile Application is developed with a micro-services architecture in mind. Therefor the front-end is decoupled from the back end and database. Express with is a Node.js REST API framework is used to serve HTTP request from the client while an Object Relation Model (ORM) called Sequelize alongside with MySQL which is a relational database.

### REST API

const http = require("http");

const app = require("./app");

const ip = require("ip");

const clc = require("cli-color");

const port = process.env.PORT || 3000;

const server = http.createServer(app);

var printServerInfo =

  clc.cyan.bold("[server]") +

  " Sever deployed on: " +

  clc.magentaBright("http://" + ip.address() + ":" + port);

server.listen(port, () => console.log(clc.blueBright(printServerInfo)));

REST API Code Segment

### Middleware

var accessLogStream = rfs.createStream("access.log", {

  interval: "1d", // rotate daily

  path: path.join(\_\_dirname, "logs"),

});

app.use(morgan("combined", { stream: accessLogStream }));

app.use(morgan("dev"));

app.use("/images", express.static("res/images"));

app.use("/videos", express.static("res/videos"));

app.use("/thumbnails", express.static("res/thumbnails"));

// Parsing bodies

app.use(bodyParser.urlencoded({ extended: false }));

app.use(bodyParser.json());

app.use((req, res, next) => {

  res.header("Access-Control-Allow-Origin", "\*");

  res.header(

    "Access-Control-Allow-Headers",

    "Origin, X-Requested-With, Content-Type, Accept, Authorization"

  );

  if (req.method === "OPTIONS") {

    res.header("Access-Control-Allow-Methods", "PUT, POST, PATCH, DELETE, GET");

    return res.status(200).json({});

  }

  next();

});

Middleware Code Segment

### Log In and Sign Up

#### **Log In User Interfaces**

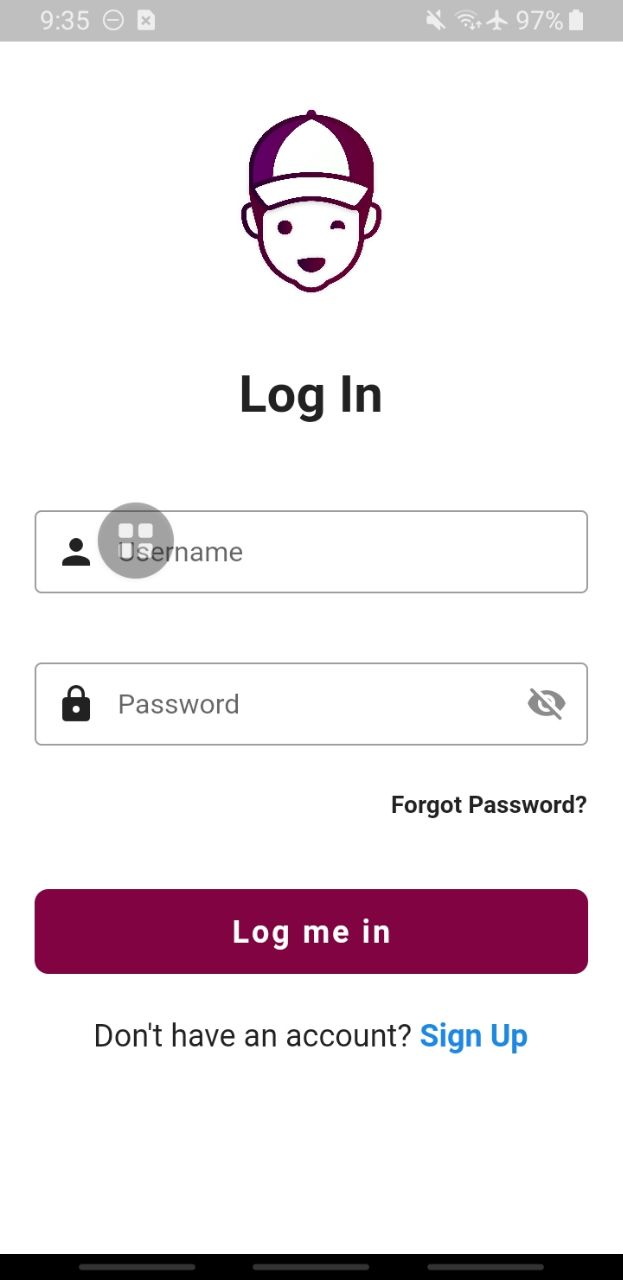


Figure 5.1: Log In User Interface

\_login(username, password) async {

    setState(() {

      showProgress = true;

    });

    var url = restAPIUri.apiUri + '/login';

    try {

      final response = await http

          .post(Uri.parse(url), headers: <String, String>{'Content-Type': 'application/json; charset=UTF-8'}, body: jsonEncode(<String, String>{'username': username, 'password': password}))

          .timeout(const Duration(seconds: 3));

      final prefs = await SharedPreferences.getInstance();

      var parse = jsonDecode(response.body);

      print(response.body);

      if (parse['route'] != null) {

        setState(() {

          // set the progress indicator to true so it would not be visible

          showProgress = false;

          var loginModel = LoginModel.fromJson(parse);

          prefs.setString('user\_id', loginModel.userId!);

          prefs.setString('user\_type', loginModel.userType!);

          loginModel.saveToSharedPref().then((value) => Navigator.pushNamed(context, loginModel.route!, arguments: prefs.getString('user\_id')));

        });

      } else {

        print(parse['route']);

        throw Exception('Wrong username or password');

      }

    }

    on Exception catch (e) {

      print(e);

      ScaffoldMessenger.of(context).showSnackBar(

        new SnackBar(

          content: Text((e.toString() == 'XMLHttpRequest error.') ? 'Oops something wrong...' : 'Wrong username or password'),

          action: SnackBarAction(

            label: 'Dismiss',

            onPressed: () {

              // Some code to undo the change.

            },

          ),

        ),

      );

      setState(() {

        showProgress = false;

      });

    }

  }

Log In Code Segment

#### **Sign Up User Interfaces**

Sign Up User Interface

\_signUp(userType, name, email, username, password) async {

    var url = 'http://192.168.239.1:3000/${userType}/add';

    final response = await http.post(

      Uri.parse(url),

      headers: <String, String>{

        'Content-Type': 'application/json; charset=UTF-8',

      },

      body: jsonEncode(<String, String>{

        'userType': userType,

        'name': name,

        'email': email,

        'username': username,

        'password': password,

      }),

    );

    SharedPreferences? prefs = await SharedPreferences.getInstance();

    print(response);

    var parse = jsonDecode(response.body);

    print(parse);

    if (parse['route'] != null) {

      setState(() {

        Navigator.pushNamed(context, parse['route']);

      });

    } else {

      throw Exception('Wrong username or password');

    }

    await prefs.setString('token', parse['token']);

    await prefs.setString('route', parse['route']);

    await prefs.setString('user\_id', parse['user\_id']);

    await prefs.setString('user\_type', parse['user\_type']);

  }

Sign Up Code Segment

### Coaching

#### **Dashboard User Interfaces**

User Interface

var res = await http.get(Uri.parse(url + 'coach/' + userId), headers: {"Accept": "application/json"});

    var resCoachProfile = await http.get(Uri.parse(restAPIUri.apiUri + '/coach/' + userId), headers: {"Accept": "application/json"});

    var parsedCoachProfile = json.decode(resCoachProfile.body);

    coachModel = CoachModel.fromJson(parsedCoachProfile);

    var resBody = json.decode(res.body);

    print(resBody.length);

    Future.delayed(Duration(milliseconds: 0), () {

      setState(() {

        data = resBody;

        isLoading = false;

      });

    });

Code Segment

#### **My Classes User Interfaces**

User Interface

var resClasses = await http.get(Uri.parse(restAPIUri.apiUri + "/class/coach/" + userId), headers: {"Accept": "application/json"});

    var parsedClasses = jsonDecode(resClasses.body);

    classesModel = List<ClassesModel>.from(parsedClasses.map((model) => ClassesModel.fromJson(model)));

    Future.delayed(Duration(milliseconds: 1000), () {

      setState(() {

        isLoading = false;

      });

    });

Code Segment

#### **Performance User Interfaces**

User Interface

 String url = restAPIUri.apiUri;

    var res = await http.get(Uri.parse(url + "/coach/teach/" + coachId), headers: {"Accept": "application/json"});

    var parsed = json.decode(res.body);

    performanceModel = PerformanceModel.fromJson(parsed);

    Future.delayed(Duration(milliseconds: 0), () {

      setState(() {

        isLoading = false;

      });

    });

Code Segment

#### **Sessions User Interfaces**

User Interface

var resSessions = await http.get(Uri.parse(restAPIUri.apiUri + "/booking/coach\_room/" + userId), headers: {"Accept": "application/json"});

    var parsedSessions = jsonDecode(resSessions.body);

    coachSessionModel = List<CoachSessionModel>.from(parsedSessions.map((model) => CoachSessionModel.fromJson(model)));

    coachSessionModel = bubbleSort(coachSessionModel);

    inspect(coachSessionModel);

    Future.delayed(Duration(milliseconds: 1000), () {

      setState(() {

        isLoading = false;

      });

    });

List<CoachSessionModel> bubbleSort(List<CoachSessionModel> coachSessionModel) {

    DateTime now = new DateTime.now();

    for (var i = 0; i < coachSessionModel.length; i++) {

      for (var j = 0; j < coachSessionModel.length - i - 1; j++) {

        if (coachSessionModel[i].studentBookings!.length < coachSessionModel[i + 1].studentBookings!.length) {

          var temp = coachSessionModel[i];

          coachSessionModel[i] = coachSessionModel[i + 1];

          coachSessionModel[i + 1] = temp;

          if (AppConstants.dateParser(coachSessionModel[i].startDate!).isBefore(now)) {

            pastCoachSessionModel.add(coachSessionModel[i]);

            coachSessionModel.remove(coachSessionModel[i]);

          }

        }

      }

    }

    return coachSessionModel;

  }

Code Segment

#### **Account User Interfaces**

User Interface

var res = await http.get(Uri.parse(url + userId), headers: {"Accept": "application/json"});

    var parsed = json.decode(res.body);

    coachModel = CoachModel.fromJson(parsed);

    Future.delayed(Duration(milliseconds: 1000), () {

      setState(() {

        isLoading = false;

      });

    });

Code Segment

### Learning

#### **Featured User Interfaces**

User Interface

var resClasses = await http.get(Uri.parse(url + "/class/"), headers: {"Accept": "application/json"});

    var resCategory = await http.get(Uri.parse(url + "/category/"), headers: {"Accept": "application/json"});

    var resCoaches = await http.get(Uri.parse(url + "/coach/"), headers: {"Accept": "application/json"});

    var resSearch = await http.get(Uri.parse(url + "/search/"), headers: {"Accept": "application/json"});

    var parsedClasses = jsonDecode(resClasses.body);

    var parsedCategory = jsonDecode(resCategory.body);

    var parsedCoaches = jsonDecode(resCoaches.body);

    var parsedSearch = jsonDecode(resSearch.body);

    classesModel = List<ClassesModel>.from(parsedClasses.map((model) => ClassesModel.fromJson(model)));

    categoriesModel = List<CategoryModel>.from(parsedCategory.map((model) => CategoryModel.fromJson(model)));

    coachesModel = List<CoachModel>.from(parsedCoaches.map((model) => CoachModel.fromJson(model)));

    searchDataModel = List<SearchDataModel>.from(parsedSearch.map((model) => SearchDataModel.fromJson(model)));

    inspect(parsedClasses);

    Future.delayed(Duration(milliseconds: 0), () {

      setState(() {

        isLoading = false;

      });

    });

Code Segment

#### **Search User Interfaces**

User Interface

class DataSearch extends SearchDelegate<String> {

  @override

  List<Widget> buildActions(BuildContext context) {

    return [

      IconButton(

          onPressed: () {

            query = "";

          },

          icon: Icon(Icons.clear))

    ];

  }

  @override

  Widget buildLeading(BuildContext context) {

    return IconButton(

      onPressed: () {

        close(context, '');

      },

      icon: AnimatedIcon( icon: AnimatedIcons.menu\_arrow, progress: transitionAnimation),

    );

  }

  Widget forIcon(String icon) {

    log(icon);

    switch (icon) {

      case 'class':

        return Icon(Icons.class\_);

      case 'student':

        return Icon(Icons.person);

      case 'coach':

        return Icon(Icons.directions\_run\_rounded);

      default:

        return Icon(Icons.access\_time\_filled\_sharp);

    }

  }

}

Code Segment

#### **My Classes User Interfaces**

User Interface

Future<String> getClassesModelData(userId) async {

    var resClasses = await http.get(Uri.parse(url + "/class/student/" + userId), headers: {"Accept": "application/json"});

    var parsedClasses = jsonDecode(resClasses.body);

    classesModel = List<ClassesModel>.from(parsedClasses.map((model) => ClassesModel.fromJson(model)));

    var parsedEnroll;

    try {

      parsedEnroll = parsedClasses![0]["students"][0]['enroll'];

      enrolledModel = EnrolledModel.fromJson(parsedEnroll);

      inspect(enrolledModel);

    } on RangeError catch (e) {

      parsedEnroll = [];

    }

    Future.delayed(Duration(milliseconds: 1000), () {

      setState(() {

        isLoading = false;

      });

    });

    return "Success!";

  }

Code Segment

#### **Booked Sessions User Interfaces**

User Interface

var resClasses = await http.get(Uri.parse(url + "/booking/student\_book/" + userId), headers: {"Accept": "application/json"});

    var parsedBookings = jsonDecode(resClasses.body);

    studentBookingModel = List<StudentBookingModel>.from(parsedBookings.map((model) => StudentBookingModel.fromJson(model)));

    Future.delayed(Duration(milliseconds: 0), () {

      setState(() {

        isLoading = false;

      });

    });

Code Segment

#### **Account User Interfaces**

User Interface

var prefs = await SharedPreferences.getInstance();

    var studentId = prefs.getString('user\_id') ?? '';

    var res = await http.get(Uri.parse(restAPIUri.apiUri + '/student/' + studentId), headers: {"Accept": "application/json"});

    log(res.body);

    var parsed = json.decode(res.body);

    studentModel = StudentModel.fromJson(parsed);

    setState(() {

      isLoading = false;

    });

Code Segment

#### **Recuring Payments User Interfaces**

User Interface

final String url = restAPIUri.apiUri;

    final res = await http.post(Uri.parse(url + '/enroll/add'),

        headers: <String, String>{'Content-Type': 'application/json; charset=UTF-8'},

        body: jsonEncode(

          <String, String>{

            'price': totalPrice,

            'classId': classId,

            'studentId': studentId,

            'provider': 'credit card',

            'coachId': coachId,

            'product': className,

          },

        ));

    var parsed = json.decode(res.body);

    print(parsed["id"]);

    if (parsed["id"] != null) {

      paymentStatus = true;

    }

exports.reccuringPaymentCheck = () => {

  const scheduledJobFunction = cron.schedule("0 0 \* \* \*", () => {

    // console.log("checkInterval() :>> ");

    Enroll.findAll()

      .then((value) => {

        value.forEach(function (enroll) {

          checkInterval(enroll.classId, enroll.id);

        });

      })

      .catch((err) => console.log(err));

  });

  scheduledJobFunction.start();

};

Code Segment

### Scheduling

#### **Coach Sessions User Interfaces**

User Interface

postNewSession() async {

    var prefs = await SharedPreferences.getInstance();

    final response = await http.post(

      Uri.parse(restAPIUri.apiUri + '/booking/coach\_room'),

      headers: <String, String>{

        'Access-Control-Allow-Headers': 'Access-Control-Allow-Origin: \*',

        'Content-Type': 'application/json; charset=UTF-8',

      },

      body: jsonEncode(<String, String>{

        "coachId": prefs.getString('user\_id')!,

        "classId": classModel.id!,

        "startDate": startDateTime.toString(),

        "endDate": endDateTime.toString(),

      }),

    );

    Navigator.pop(context);

  }

Code Segment

#### **Student Bookings User Interfaces**

User Interface

postNewBooking() async {

    var prefs = await SharedPreferences.getInstance();

    final response = await http.post(

      Uri.parse(restAPIUri.apiUri + '/booking/student\_book'),

      headers: <String, String>{

        'Access-Control-Allow-Headers': 'Access-Control-Allow-Origin: \*',

        'Content-Type': 'application/json; charset=UTF-8',

      },

      body: jsonEncode(<String, String>{

        "coachSessionId": widget.coachSessionModel[selectedIndexList[0]].id!,

        "title": title,

        "description": description,

        "studentId": prefs.getString('user\_id')!,

      }),

    );

    Navigator.of(context).pop();

    pushNewScreenWithRouteSettings(

      context,

      settings: RouteSettings(name: '/class\_details'),

      screen: StudentHomePage(initalBottomNavBarIndex: 3),

      withNavBar: true,

      pageTransitionAnimation: PageTransitionAnimation.cupertino,

    );

  }

Code Segment

### Administrator

#### **Dashboard User Interfaces**

User Interface

Code Segment

#### **Account deletion User Interfaces**

User Interface

Code Segment

#### **Reporting User Interfaces**

User Interface

Code Segment

## Testing

<text>

### Test Cases

<text>

Table .: List of test cases

|  |  |  |
| --- | --- | --- |
| No. | Test Cases | Description |
| TEST\_100 | | |
|  | TEST\_100\_001 | Students sign up by completing sign up form. |
|  | TEST\_100\_002 | Student presses Sign Up button. |
|  | TEST\_100\_003 | System verify Sign Up form. |
|  | TEST\_100\_004 | System verifies username and password |
|  | TEST\_100\_005 | System shows message(s) when Sign Up form is incomplete. |
|  | TEST\_100\_006 | System redirect student homepage if verification is complete. |
|  | TEST\_100\_007 | Coach sign up by completing sign up form |
|  | TEST\_100\_008 | Coach presses Sign Up button |
|  | TEST\_100\_009 | System verify Sign Up form |
|  | TEST\_100\_010 | System verifies username and password |
|  | TEST\_100\_011 | System shows message(s) when Sign Up form is incomplete |
|  | TEST\_100\_012 | System redirect coach homepage if verification is complete. |
|  | TEST\_100\_013 | Users enter username and password to log in. |
|  | TEST\_100\_014 | User presses Log In button |
|  | TEST\_100\_015 | System verifies username and password |
|  | TEST\_100\_016 | System will redirect user to appropriate homepage |
| TEST\_200 | | |
|  |  |  |
|  |  |  |

### Test Case Result of Create Medical Form

<text>

<a pie chart indicates % of pass and fail for Create Medical Form >

<text>

### Test Case Result of Manage Users Details

<text>

<a pie chart indicates % of pass and fail for Manage Users Details >

<text>

### Overall Result

<text>

<a pie chart indicates overall % of pass and fail>

<text>

## Chapter Summary

<text>