ExpoSure

MOBILE
DEVELOPMENT
PROJECT PROPOSAL

Your Companion for Expos and Events. Having the power to catalogue exhibition products while making them at the finger tips of visitors who can browse and save their favorite products on their mobiles

TABLE OF CONTENTS

INTRODUCTION

ExpoSure is an app for businesses, event organizers and visitors alike. Events are listed together with details such as location and dates giving businesses the ability of displaying their products to visitors. Visitors can save a product to favorites before hand so that they get to have a shortlist of their coveted products to see during the event. This way they will know the booth number associated with their favorite products saving time and focusing on what matters. Moreover, they can use their cameras to quickly add products to their favorite list on location. Organizers can coordinate with businesses making event guides that help visitors find booths they desire.

FUNCTIONALITY

THE APP IS DIVIDED INTO THREE WORK FLOWS.

ORGANIZERS

The Organizer role is to create events. They can view a list of events. Also, organizers have the ability to create businesses and add them to events.

BUSINESSES

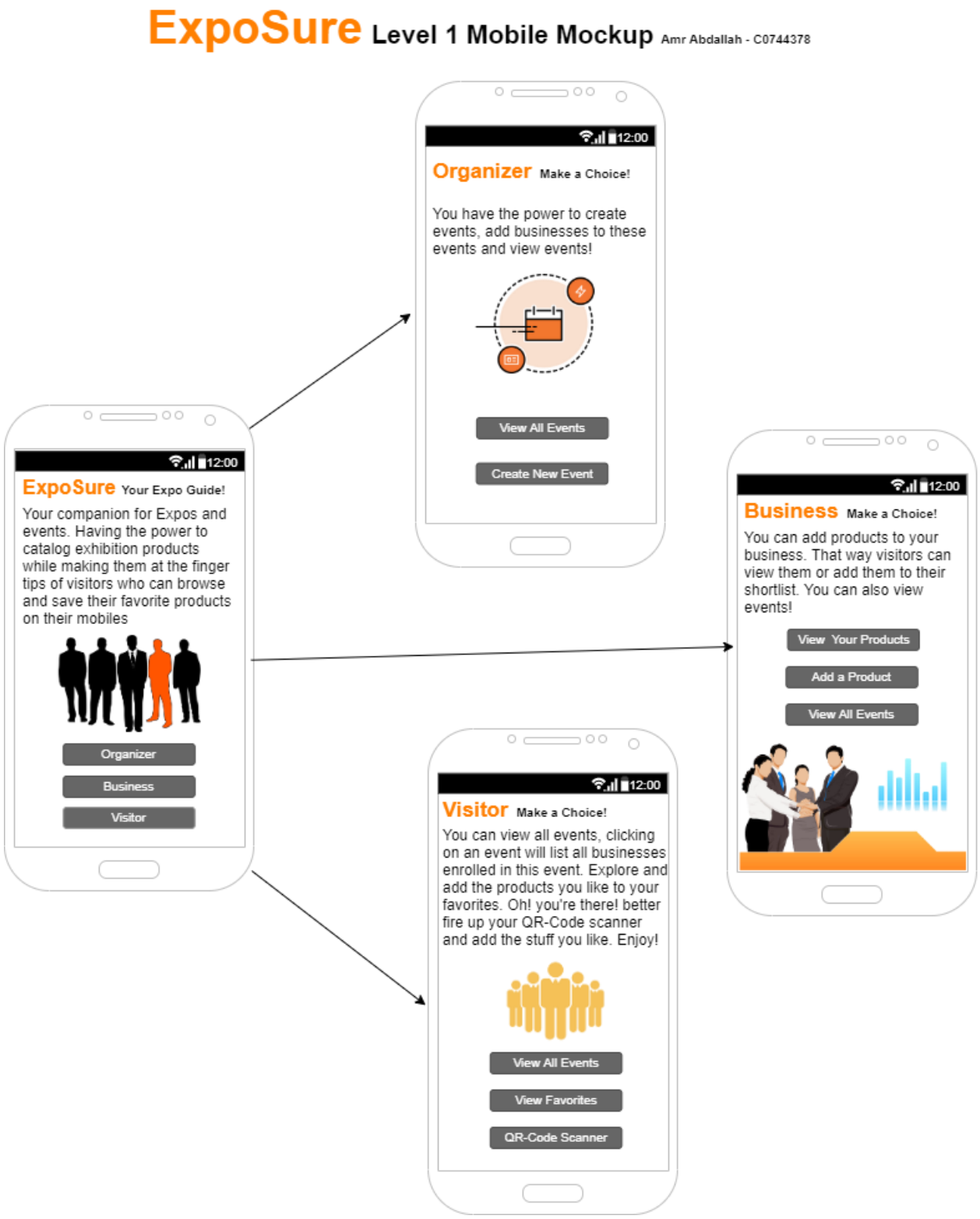
Businesses can create products but cannot create events. Each product can be identified by a QR-Code. They can view a list of all their products.

VISITORS

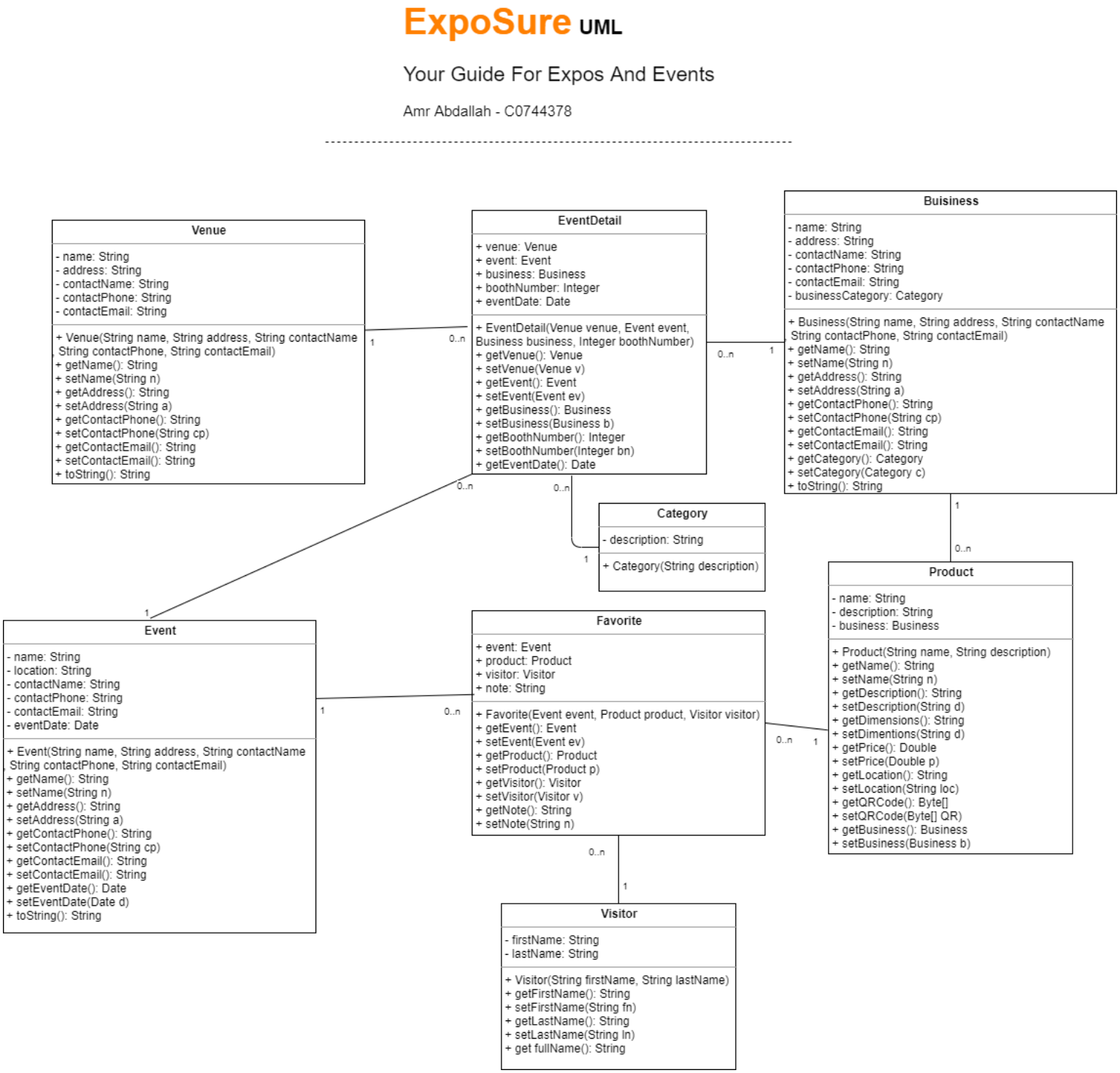
Visitors are the main focus of this app; they are capable of viewing events in a list. Each event informs the visitor of the venue's location and event date. They can browse for businesses enrolled in events and furthermore, browse the products belonging to each business. They can do this beforehand. This way they can plan their visit carefully ahead of time. Visitors can also, use their phone cameras to scan a QR-Code sticker provided by the business owners to add their chosen products to favorites.

MOCKUP

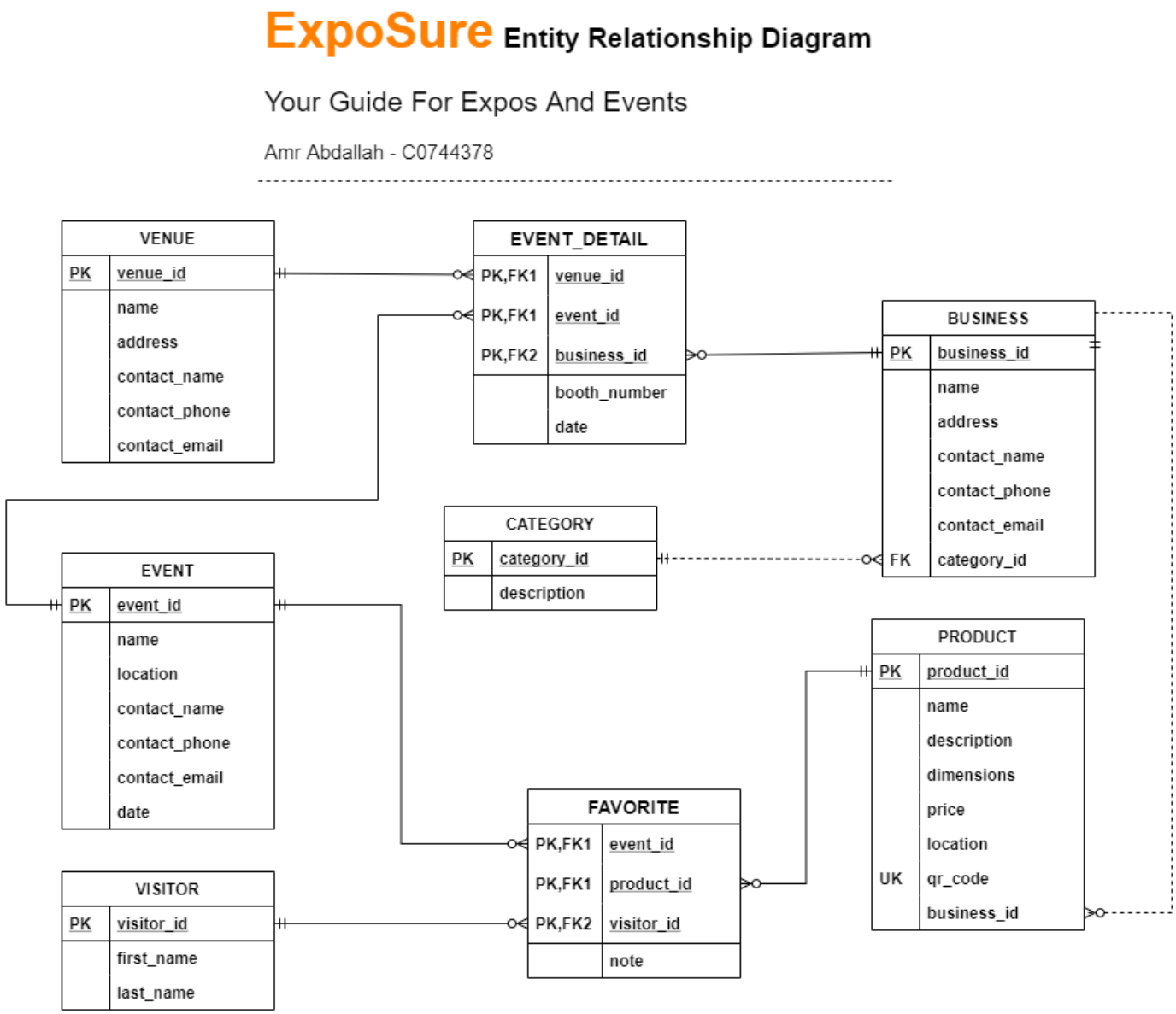
LEVEL 1



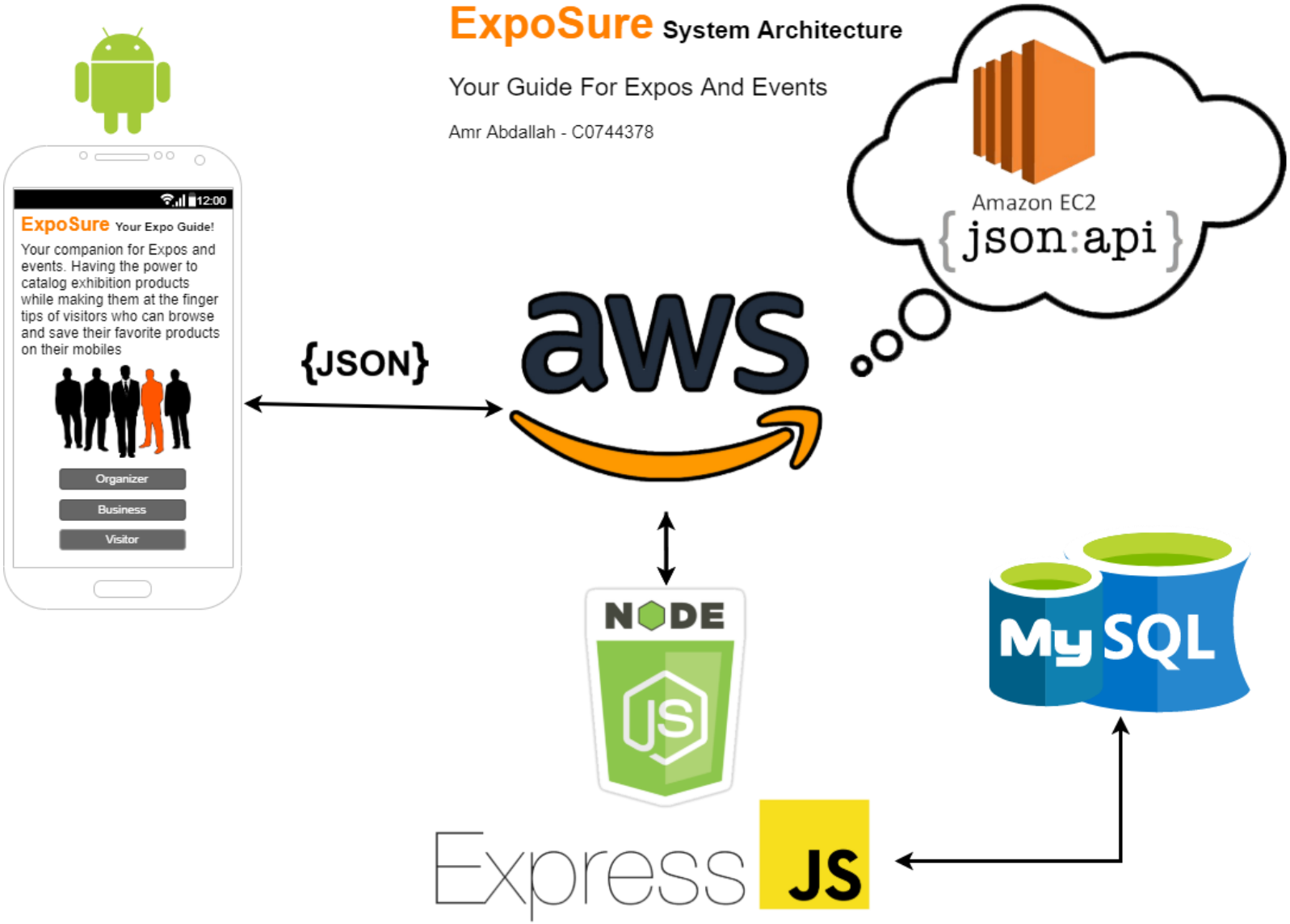
CLASS UML DIAGRAM



ENTITY RELATIONSHIP DIAGRAM



SYSTEM ARCHITECTURE



BACK END CODE SNIPPETS

GENERAL VIEW

Response(12ms) - ExpoSure_Node [SSH: Android_ec2-User_AWS] - Visual Studio Code

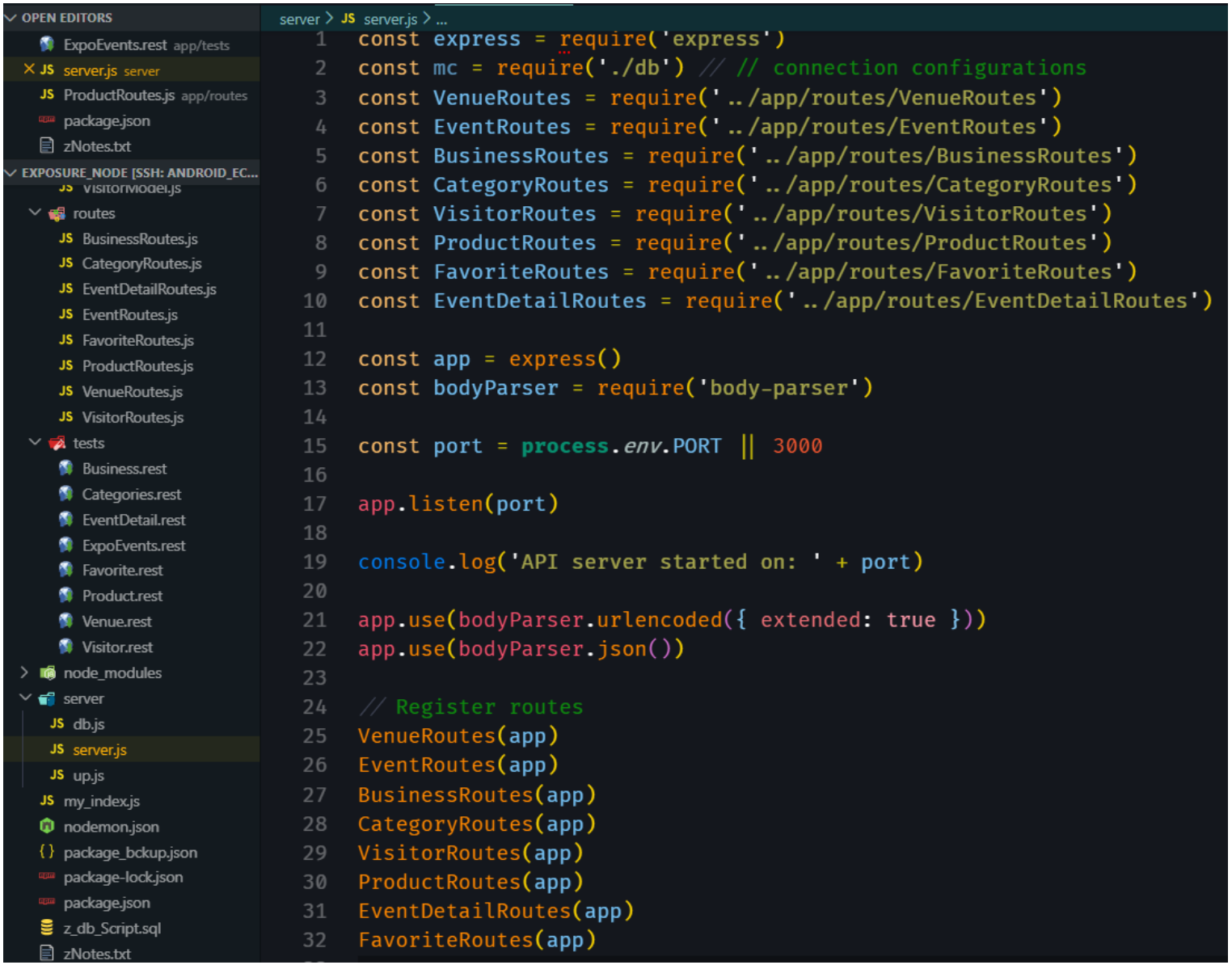
app > tests > ExpoEvents.rest

```
# https://marketplace.visualstudio.com/items?itemName=humao.
# Change url from vscode Settings.json
# [ctrl+.] → type json → click Settings.json → Very Botto
0 references
@event_Id = 5
6 references
@contentType = application/json

### Get ALL Events
Send Request
GET {{url}}/Expo_Events HTTP/1.1
Accept: {{contentType}}

### Get Event By ID
Send Request
GET {{url}}/Expo_Events/13 HTTP/1.1
Accept: {{contentType}}

### Get Event By Name
Send Request
POST {{url}}/expo_events_by_name
Content-Type: : {{contentType}}

{
    "name": "data"
}

### Create Event
Send Request
POST {{url}}/expo_events
Content-Type: {{contentType}}

{
    "name": "Ibrahim",
    "location": "Bla St.",
    "contact_name": "Bla Contact",
    "contact_phone": "516-455-5688",
    "contact_email": "bla@bla.com",
    "date": "2020/12/02 13:00",
```

Response(12ms)

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 7361
ETag: W/"1cc1-UGLECvjLgsh9l69C6P7HKBs4Aks"
Date: Tue, 14 Apr 2020 03:22:06 GMT
Connection: close

[
  {
    "event_id": 1,
    "name": "Data Connectors Toronto Cybersecurity Conference",
    "location": "6 Prairieview Junction",
    "contact_name": "Brantley March",
    "contact_phone": "504(666)608-7746",
    "contact_email": "bmarch0@google.cn",
    "photo": "https://images.unsplash.com/photo-1523580494863-6f3031224c94?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=1050&q=80",
    "date": "2019-10-18T14:00:00.000Z"
  },
  {
    "event_id": 2,
    "name": "Digital Talent Acquisition Summit (DTA)",
    "location": "98 Bellgrove Hill",
    "contact_name": "Constancy Newall",
    "contact_phone": "62(904)847-1968",
    "contact_email": "cnewall1@storify.com",
    "photo": "https://images.unsplash.com/photo-1556761175-4b46a572b786?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=967&q=80",
    "date": "2020-12-12T14:08:00.000Z"
  },
  {
    "event_id": 3,
    "name": "The World Leading AI Summit",
    "location": "61820 Jackson Circle",
    "contact_name": "Papagena Stacey",
    "contact_phone": "7(519)212-4580",
    "contact_email": "pstacey2@ucoz.ru",
    "photo": "https://images.unsplash.com/photo-1560298803-1d998f6b5249?ixlib=rb
```

SETTING UP EXPRESS SERVER



ROUTE CODE SNIPPET EXAMPLE

pp > routes > JS EventRoutes.js > <unknown> > module.exports

```
1   'use strict';
2   module.exports = (app) => {
3       var expoEvent = require('../controller/EventController');
4
5       // expoEvent Routes
6       app.route('/expo_events')
7           .get(expoEvent.list_all_events)
8           .post(expoEvent.create_an_event);
9
10      app.route('/expo_events/:eventId')
11          .get(expoEvent.read_an_event)
12          .put(expoEvent.update_an_event)
13          .delete(expoEvent.delete_an_event);
14          any
15      app.route('/expo_events_by_name')
16          .post(expoEvent.get_expo_events_by_name);
17  };
18
```

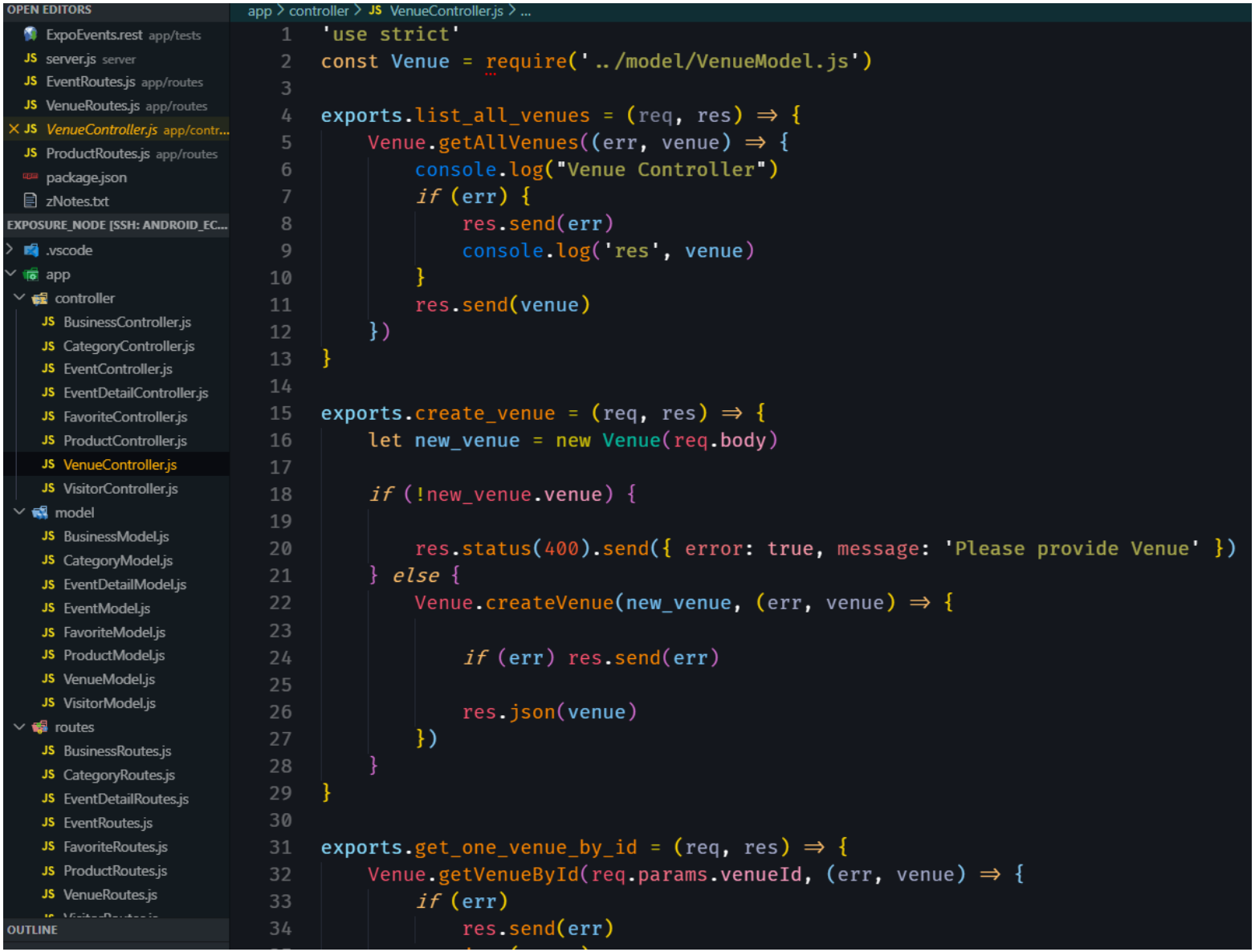
MODEL CODE SNIPPET

BusinessModel.js - ExpoSure_Node [SSH: Android_ec2-User_AWS] - Visual Studio Code

app > model > BusinessModel.js > ...

```
'user strict'
var sql = require('../../server/db')

//Business object constructor
var Business = function (business) {
    this.business = business
}

Business.createBusiness = (newBusiness, result) => {
    let sqlQuery = "INSERT INTO Businesses(name,address,contact_name,contact_phone,contact_email
    sql.query(sqlQuery,
        [newBusiness.business.name,
        newBusiness.business.address,
        newBusiness.business.contact_name,
        newBusiness.business.contact_phone,
        newBusiness.business.contact_email,
        newBusiness.business.category_id], (err, res) => {
            if (err) {
                console.log("error: ", err)
                result(err, null)
            }
            else {
                console.log(res.insertId)
                result(null, res.insertId)
            }
        })
}

Business.getBusinessById = (businessId, result) => {
    sql.query("Select * from Businesses where business_id = ? ", businessId, (err, res) => {
        if (err) {
            console.log("error: ", err)
```

CONTROLLER SNIPPET



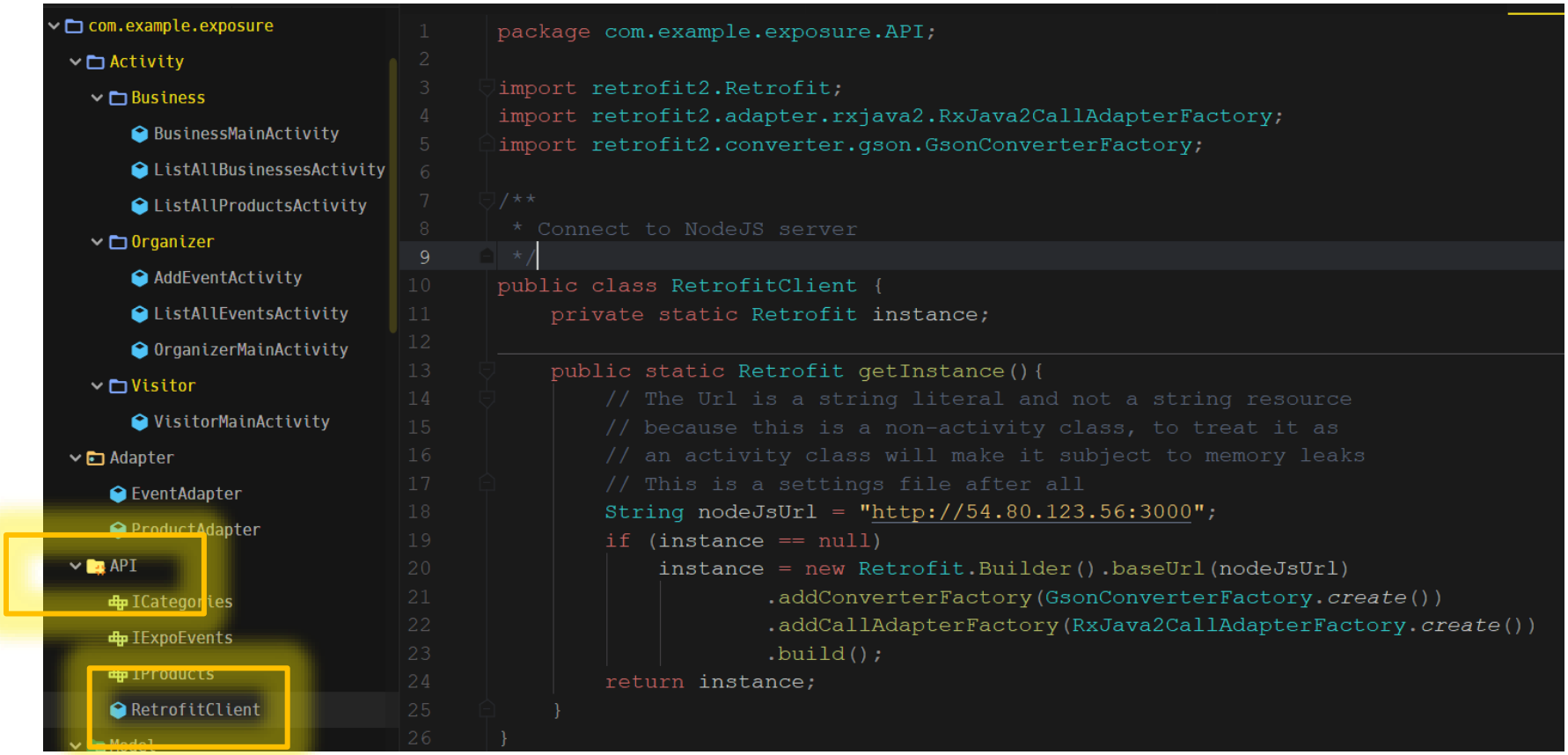
VIEW SNIPPETS (ANDROID APPLICATION)

GENERAL VIEW

```
com.example.exposure
  Activity
    Business
      BusinessMainActivity
      ListAllBusinessesActivity
      ListAllProductsActivity
    Organizer
      AddEventActivity
      ListAllEventsActivity
      OrganizerMainActivity
    Visitor
      VisitorMainActivity
  Adapter
    EventAdapter
    ProductAdapter
  API
    ICategories
    IExpoEvents
    IProducts
    RetrofitClient
  Model
    Business
    Category
```

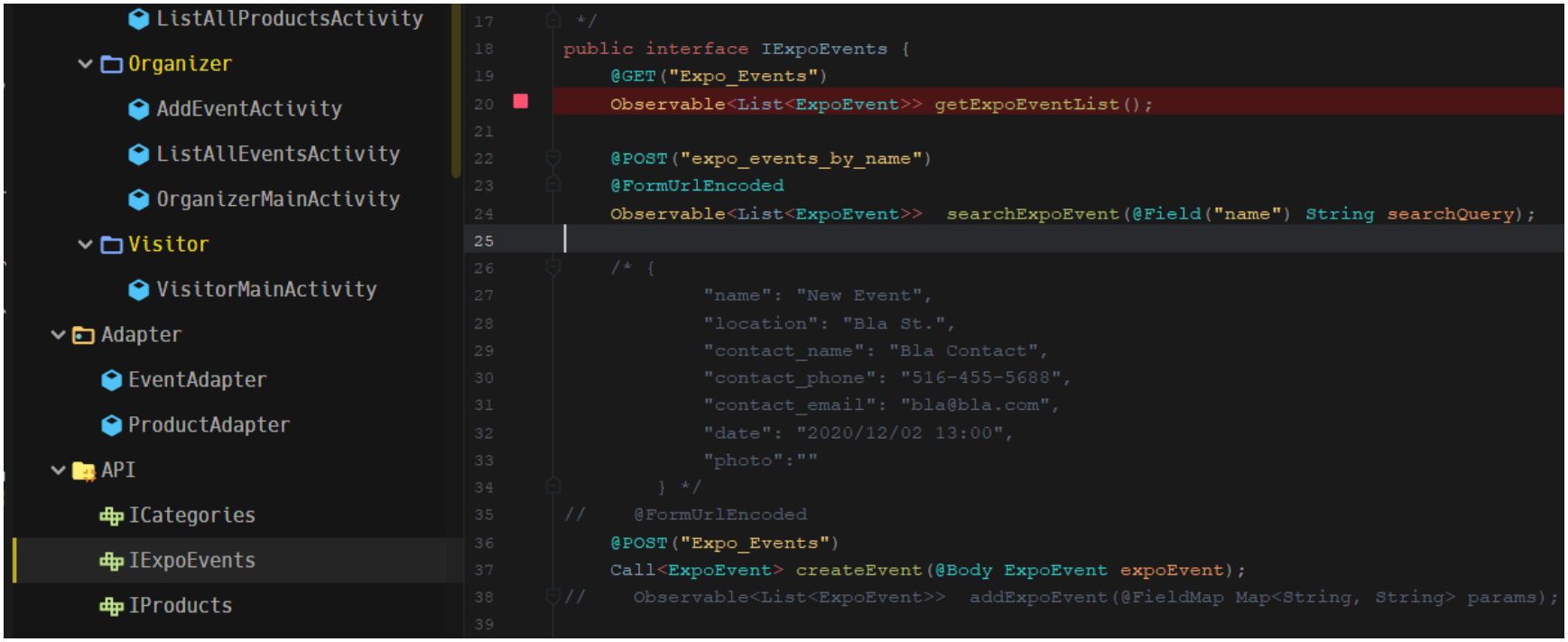
```java
22      private Button btnVisitor;
23
24      @Override
25      protected void onCreate(Bundle savedInstanceState) {
26          super.onCreate(savedInstanceState);
27          setContentView(R.layout.activity_main);
28
29          // Connecting buttons code with interface
30          btnOrganizer = findViewById(R.id.btnOrganizer);
31          btnBusiness =  findViewById(R.id.btnBusiness);
32          btnVisitor =  findViewById(R.id.btnVisitor);
33
34          // setting up button OnClick listener
35          btnOrganizer.setOnClickListener((v) → { openOrganizerMainActivity(); });
41
42          // setting up button OnClick listener
43          btnBusiness.setOnClickListener((v) → { openBusinessMainActivity(); });
49
50          // setting up button OnClick listener
51          btnVisitor.setOnClickListener((v) → { openVisitorMainActivity(); });
57      }
58
59      // Organizer Program Flow is complete - Read and write to api
60      public void openOrganizerMainActivity (){
61          // Declaring intent to inflate target activity
62          Intent intent = new Intent( packageContext: this, OrganizerMainActivity.class);
63          // fire up intent
64          startActivity(intent);
```

REST CLIENT API



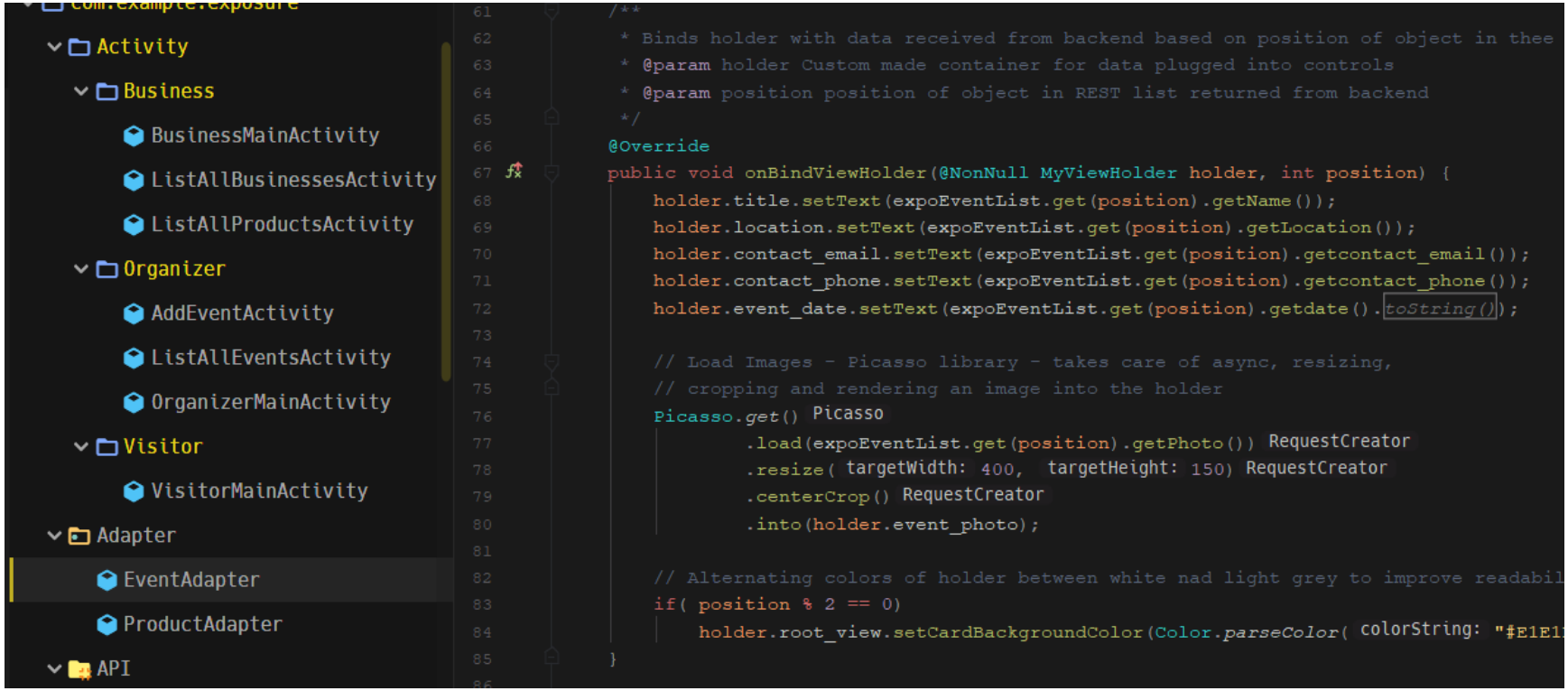

```java
package com.example.exposure.API;

import retrofit2.Retrofit;
import retrofit2.adapter.rxjava2.RxJava2CallAdapterFactory;
import retrofit2.converter.gson.GsonConverterFactory;

/**
 * Connect to NodeJS server
 */
public class RetrofitClient {
    private static Retrofit instance;

    public static Retrofit getInstance(){
        // The Url is a string literal and not a string resource
        // because this is a non-activity class, to treat it as
        // an activity class will make it subject to memory leaks
        // This is a settings file after all
        String nodeJsUrl = "http://54.80.123.56:3000";
        if (instance == null)
            instance = new Retrofit.Builder().baseUrl(nodeJsUrl)
                    .addConverterFactory(GsonConverterFactory.create())
                    .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
                    .build();
        return instance;
    }
}
```

API INTERFACE SNIPPET




```
17  */
18  public interface IExpoEvents {
19      @GET("Expo_Events")
20      Observable<List<ExpoEvent>> getExpoEventList();
21
22      @POST("expo_events_by_name")
23      @FormUrlEncoded
24      Observable<List<ExpoEvent>>  searchExpoEvent(@Field("name") String searchQuery);
25
26      /* {
27              "name": "New Event",
28              "location": "Bla St.",
29              "contact_name": "Bla Contact",
30              "contact_phone": "516-455-5688",
31              "contact_email": "bla@bla.com",
32              "date": "2020/12/02 13:00",
33              "photo":""
34          } */
35  //    @FormUrlEncoded
36      @POST("Expo_Events")
37      Call<ExpoEvent> createEvent(@Body ExpoEvent expoEvent);
38  //    Observable<List<ExpoEvent>>  addExpoEvent(@FieldMap Map<String, String> params);
39
```

ADAPTER SNIPPET



ANDROID MODEL SNIPPET

```java
@SerializedName("contact_email")
@Expose
private  String contact_email;

@SerializedName("photo")
@Expose
private  String photo;

@SerializedName("date")
@Expose
private String date;
```

```java
/**
 * Events Constructor with id
 * @param event_id Event Id
 * @param name Event Name
 * @param location  Event Location
 * @param contact_name Event contact name
 * @param contact_phone Event contact phone
 * @param contact_email Event contact email
 * @param date Event date
 * @param photo Event photo
 */
public ExpoEvent(String event_id, String name, String location, String contact_name,
                 String contact_phone, String contact_email, String date, String photo) {
    this.event_id = event_id;
    this.name = name;
    this.location = location;
    this.contact_name = contact_name;
    this.contact_phone = contact_phone;
    this.contact_email = contact_email;
    this.date = date;
    this.photo = photo;
}
```

ExpoEvent › getevent_id()

ACTIVITY SNIPPET

```
150         // plugs into adapter and searches for event name
151         private void startSearch(String query) {
152             materialSearchBar.clearFocus();
153             compositeDisposable.add(myAPI.searchExpoEvent(query)
154                     .subscribeOn(Schedulers.io())
155                     .observeOn(AndroidSchedulers.mainThread())
156                     .subscribe((Consumer) (expoEvents) -> {
159                             adapter = new EventAdapter(expoEvents);
160                             recycler_event_search.setAdapter(adapter);
161                     }, (Consumer) (throwable) -> {
165                             String msg = "Not Found";
166                             Toast.makeText( context: ListAllEventsActivity.this, msg, Toast.LENGTH_SHORT).show();
167                     }));
169
170         }
171
172         // plugs into adapter and gets ExpoEvents
173         private void  getAllEvents(){
174
175             compositeDisposable.add(myAPI.getExpoEventList()
176                     .subscribeOn(Schedulers.io())
177                     .observeOn(AndroidSchedulers.mainThread())
178                     .subscribe((Consumer) (expoEvents) -> {
181                             adapter = new EventAdapter(expoEvents);
182                             recycler_event_search.setAdapter(adapter);
183                     }, (Consumer) (throwable) -> {
187                             String msg = "Not Found";
188                             Toast.makeText( context: ListAllEventsActivity.this, msg, Toast.LENGTH_SHORT).show();
189                     }));
191         }
192
193     }
194
```

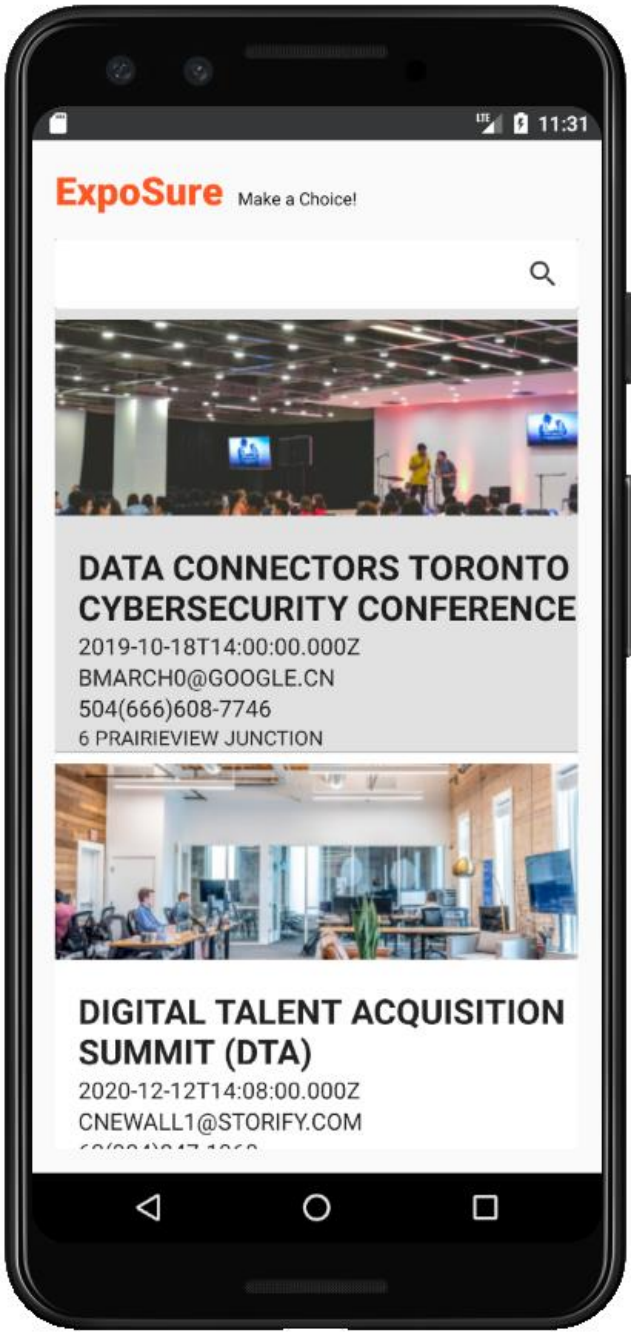
DATABASE SCRIPT SAMPLE

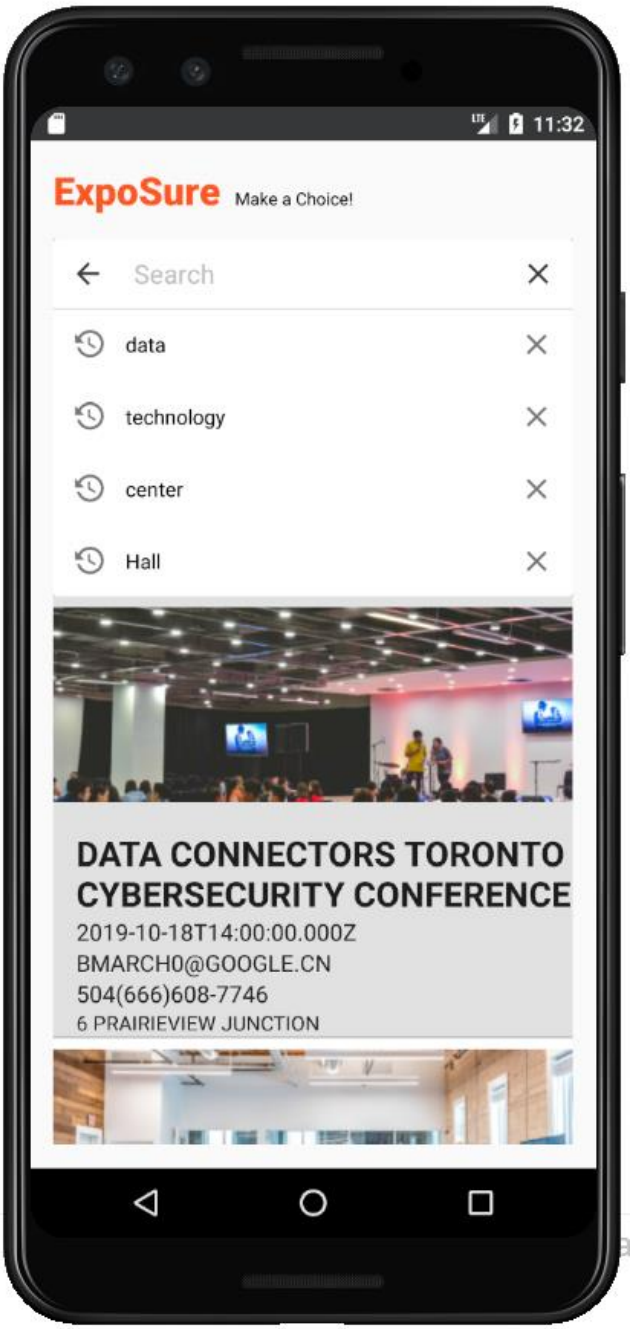
```sql
-- -----------------------------------------------------
-- Table `ExpoSure`.`Favorites`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ExpoSure`.`Favorites` ;

CREATE TABLE IF NOT EXISTS `ExpoSure`.`Favorites` (
  `Products_product_id` INT NOT NULL,
  `Visitors_visitor_id` INT NOT NULL,
  `Expo_Events_event_id` INT NOT NULL,
  `note` VARCHAR(200) NULL DEFAULT NULL,
  PRIMARY KEY (`Products_product_id`, `Visitors_visitor_id`, `Expo_Events_event_id`),
  INDEX `fk_Favorites_Products1_idx` (`Products_product_id` ASC) VISIBLE,
  INDEX `fk_Favorites_Visitors1_idx` (`Visitors_visitor_id` ASC) VISIBLE,
  INDEX `fk_Favorites_Expo_Events1_idx` (`Expo_Events_event_id` ASC) VISIBLE,
  CONSTRAINT `fk_Favorites_Products1`
    FOREIGN KEY (`Products_product_id`)
    REFERENCES `ExpoSure`.`Products` (`product_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Favorites_Visitors1`
    FOREIGN KEY (`Visitors_visitor_id`)
    REFERENCES `ExpoSure`.`Visitors` (`visitor_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Favorites_Expo_Events1`
    FOREIGN KEY (`Expo_Events_event_id`)
    REFERENCES `ExpoSure`.`Expo_Events` (`event_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

APPLICATION SCREENSHOTS

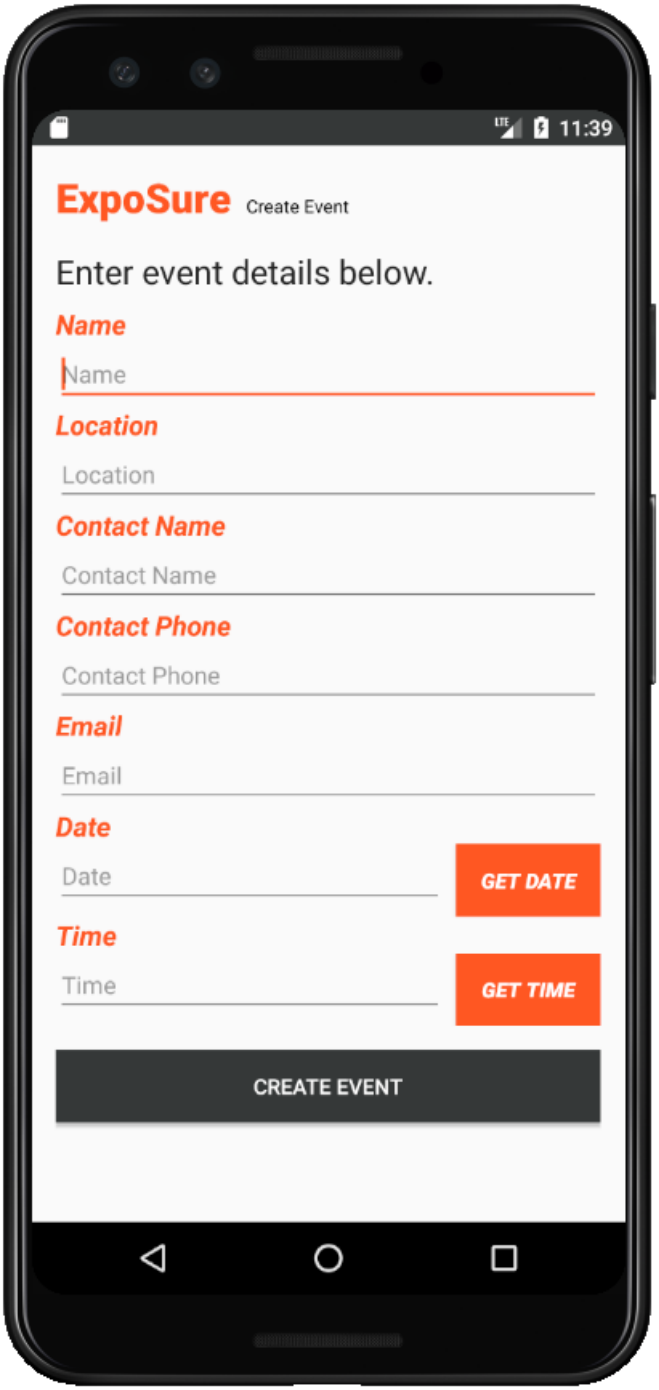
LIST ALL EVENTS

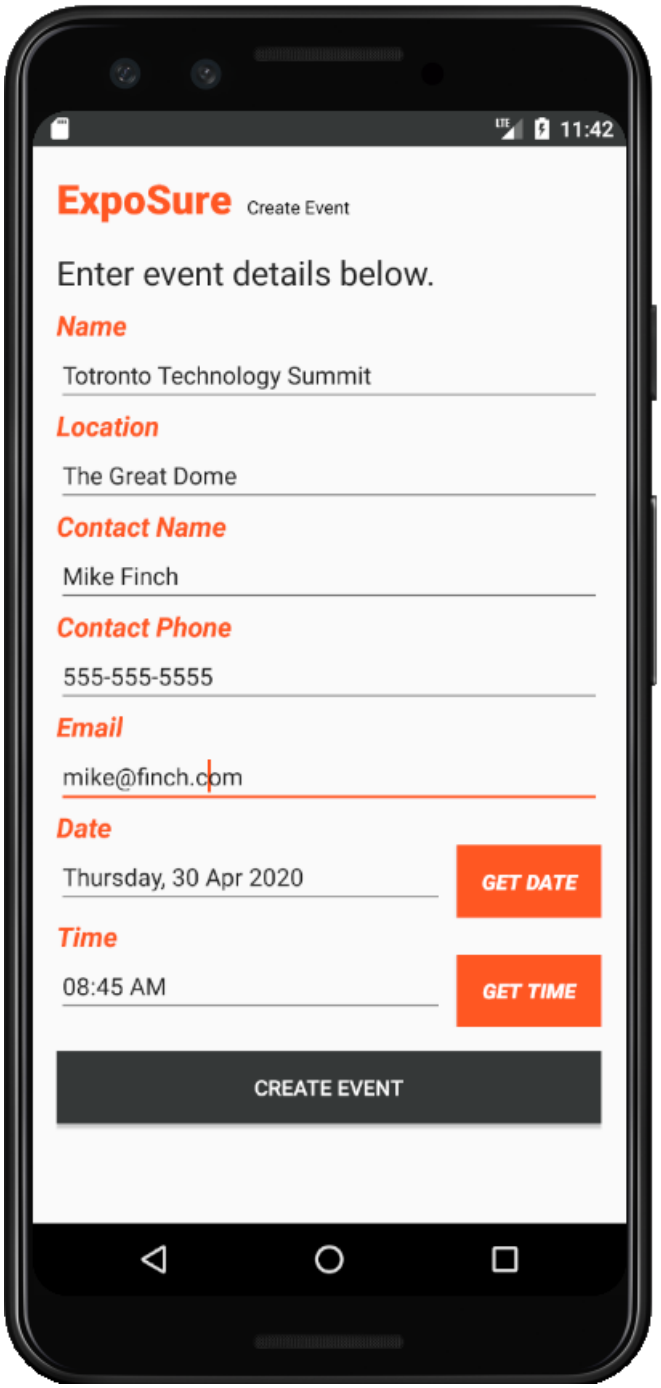
All the events saved to the database will be retrieved on this screen. The user can use the search bar to filter events by name.





CREATING NEW EVENTS





Typo is intentional to show search capability.