**Project Report**

**On**

**EventOws**

**(Event-Shows Android App)**

**PURPOSE**

* To design and develop an Event-Show Locator (EventOws) Android mobile app for Android platforms which will provide the smallest events and shows available in your locality.
* The option will be there to check the nearest events and shows available to you at your locality. This app basically divides the definition of Events and Shows in two parts: -
* **Events which means basically a small event organized for around 2 to 3 hours…which will mostly describe it as private Event.**
* **Shows which basically means a large event organized for a large period of time which will mostly describe it as a public event.**
* There would be an option provided in this application where you can add your own Event and Shows based on the conditions which will display it to your added Events/Shows and which will be also be added to the main Event data table.
* Login and Register, to register you as a user in the application.

**Scope**

* The system being designed is economically with respect to the users and Organizer’s point of view.
* The goal is to extract useful information from an unstructured data using the concept of information retrieval, filtering and secure random algorithms.
* To develop an Event Search application that can help the user to find nearby events around him.
* My basic approach attempts to develop a smart phone-based application using Android which can be used to make this process easier, secure and less error prone. More efficient details will be achieved through this system to the user to provide access to information related to events and shows with their each and every particular detail information on the go.

**EventOws Application Architecture**

**The overall system design consists of following modules:**

* **Register Module**: In this module, the new user registers himself in this app by creating a new account which will be used for verification and authentication of the user. This information of the user is stored in the firebase and later used for displaying it in the User Profile.
* **User Module:** In this module, the registered user now logins with his new account and the app will accept his account only if he has verified his mail sent on his registered email ID during the registration. The details will be checked and authenticated and allow him further to access the app. At the failure of 5 attempts the user login is failed and the login button is disabled.
* **Events** **Module:** In this module, the data stored in the Realtime database of the firebase is displayed here in the Events Section. They can be sorted according to the type of Event which can be from newest to oldest and from oldest to newest and can be searched according to Age, Location and Title. **The data entered is manually by me.**
* **Shows Module**: Similarly, for shows, the data is fetched from the data base and can be sorted according to the needs. **The data entered is manually by me.**
* **Add Event Module:** This module is a feature where the registered user can add an Event on his own. The user shall be notified that the any Event in this app shall last for Maximum 4 hours long. It consists of data models Title, Description, Location, Date, Time, Organizer, Guest appearance and contact number. **This data is entered by the user operating the application.**
* **Add Show Module**: Similarly for Events, the registered user can add his own show which can maximum last for 8 to 9 hours long. Consist of the same data models of that of Shows. **This data is entered by the user operating the application.**
* **Database Module:** **The** system uses **Firebase** as its database because of their simplicity and flexibility. This module stores every single information about user details, Events, Shows, Added Events, Added Shows and model their data on specified operations. These operations can be storing user details, Event Information, Show Information, your Event Information, your Show Information data or can be authentication credentials.

**Event-Ows App Architecture Diagram**



**Your Added Shows**

**Your Added Events**

Shows

Events

**Username and Password**

Register

**When you want to add a show, you fill the details and click on add button which will add your event in Events Section and display it under My events where you can edit that Event.**

**When you want to add an event, you fill the details and click on add button which will add your event in Events Section and display it under My events where you can edit that Event.**

**Shows registered in the database are now displayed here including the title, location, date, time.**

**Register your Full Name Password and your age for your user profile.**

**Login Process After verification from your registered email to app main page.**

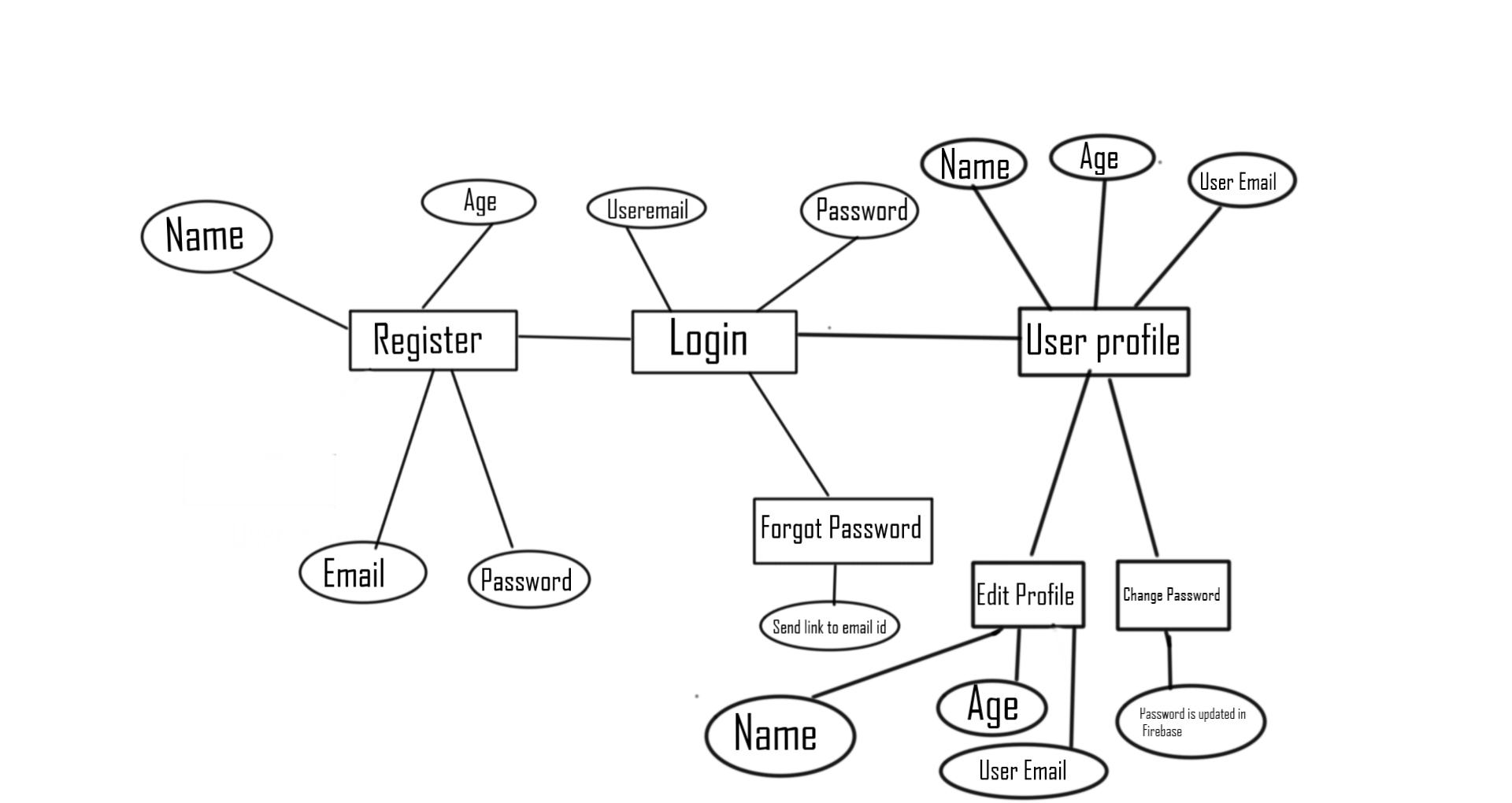
**Events registered in the database are now displayed here including the title, location, date, time.**

Firebase

**Application Functionalities: -**

I have divided the Design into three types: -

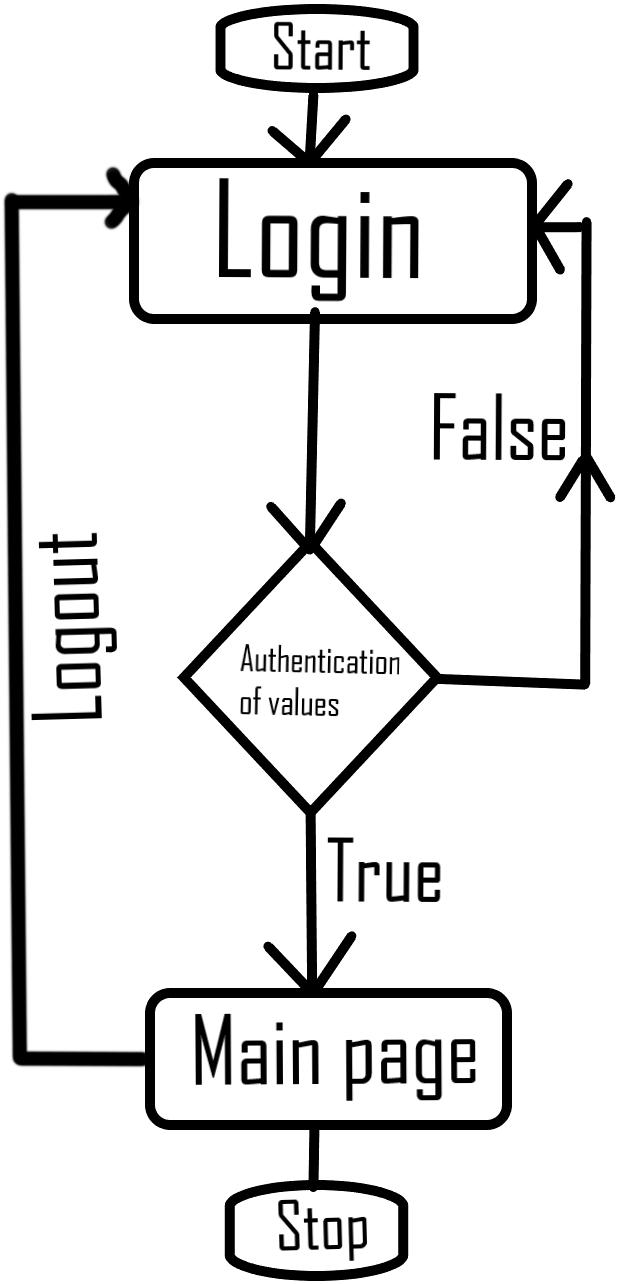
* Register, Login Authentication and User Profile
* Event and Shows (View an event or show and Book ticket option for checkout)
* Add an Event or Show and view it in My Events or My Shows.

****

*Register, Login Authentication and User Profile (ER-DIAGRAM)*

It describes as follows: -

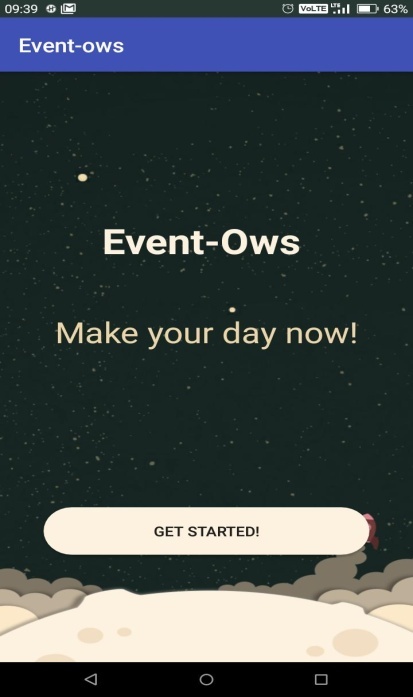
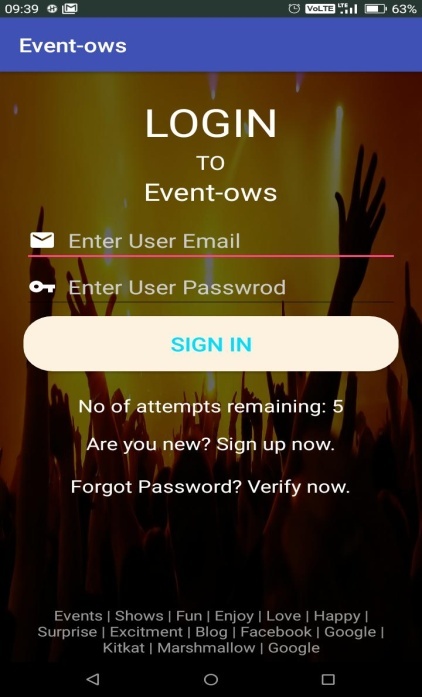
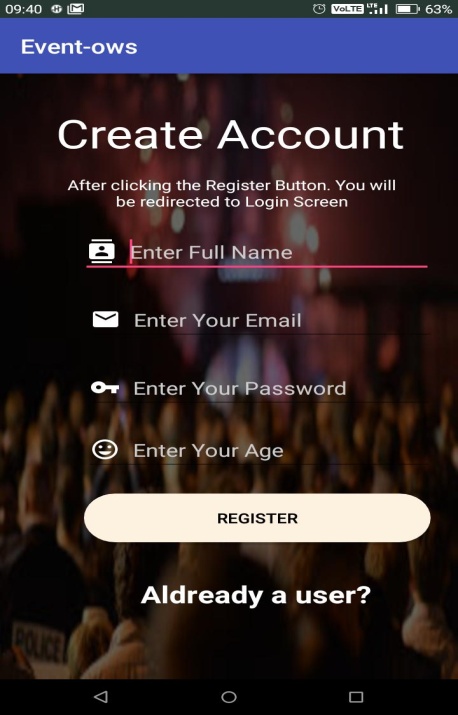
* Starting from Register, A new register always register himself in the app by putting his: -
  + 1. Name used for User Profile
    2. Valid Email for Verification and Authentication (Also for User Profile)
    3. A Password for Verification and Security
    4. Age for User Profile.
* After successful Registration he is reverted to the Login Page where he has to verify himself and this is done by Firebase Authentication.
* Firebase sends an email verification link to the email id which the user had registered during Registration. This needs to be verified otherwise your account is not activated and you are not permitted to login.
* After successful verification and correct password, you are signed in.
* The forgot password feature helps you to change and keep the new password in case you forget the password. For this you need to be a registered user and enter the registered email id.
* A password reset mail will be sent to your email id which on clicking, you set the new password, and you can successfully login your account.



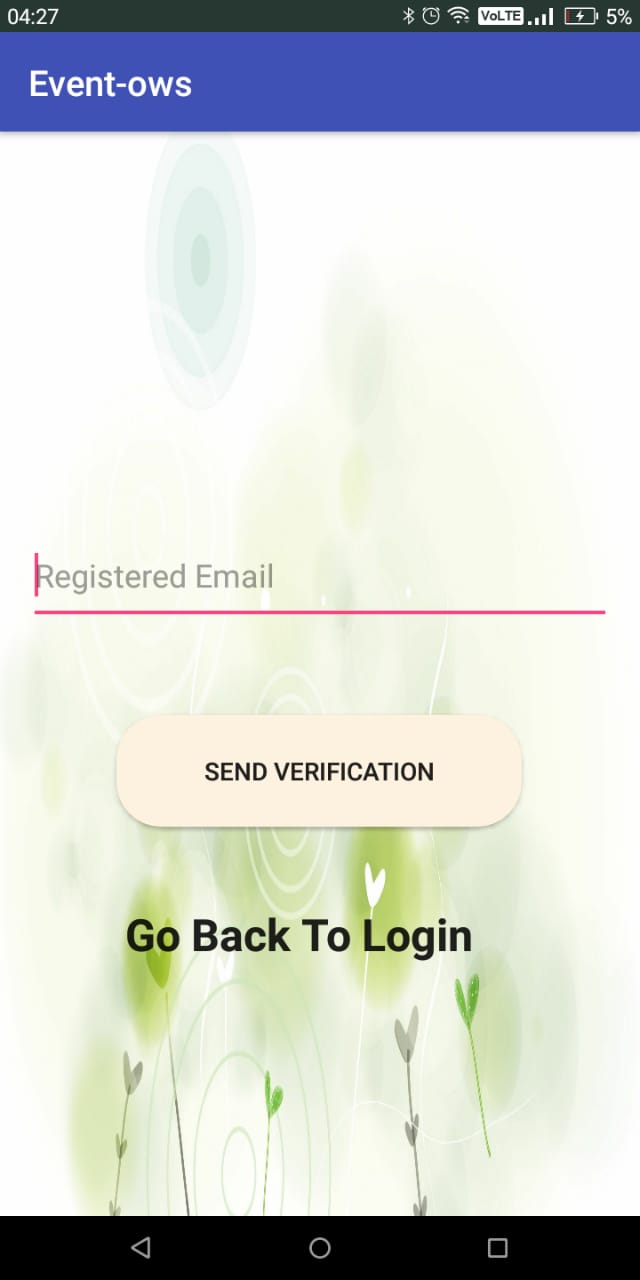
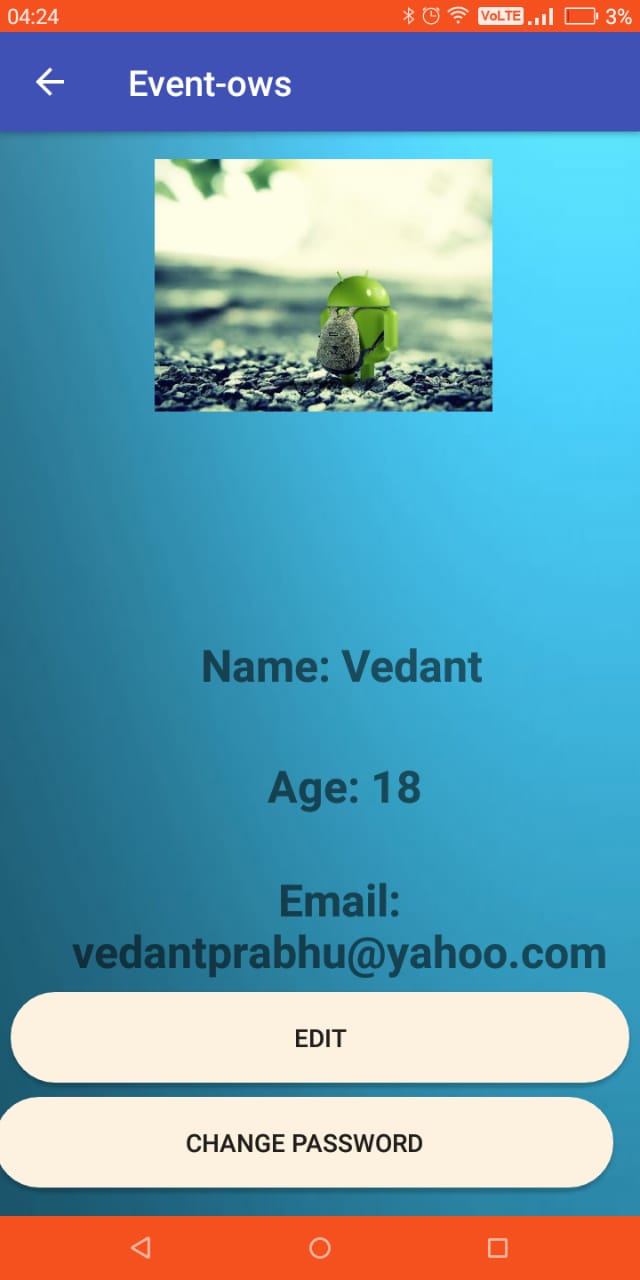
Flow chart of Login Authentication

* For user Profile, the Firebase reads the data of the user from Realtime Database which displays: -
  + 1. Name- Entered during Registration.
    2. Age-Entered during Registration.
    3. Email- Id Entered during Registration.
* On clicking edit profile, you can edit the above preferences and on Saving the changes get updated.
* On clicking change password, you can change and set and keep a new password.
* Now when you add yourself a new Profile Picture on the application it works as following: -
* The picture you add is stored in the Firebase Realtime Storage where your user UID is created.
* Under this User UID, inside the Images folder your profile picture will be updated as per you change your profile picture. It is done to avoid Confusion.

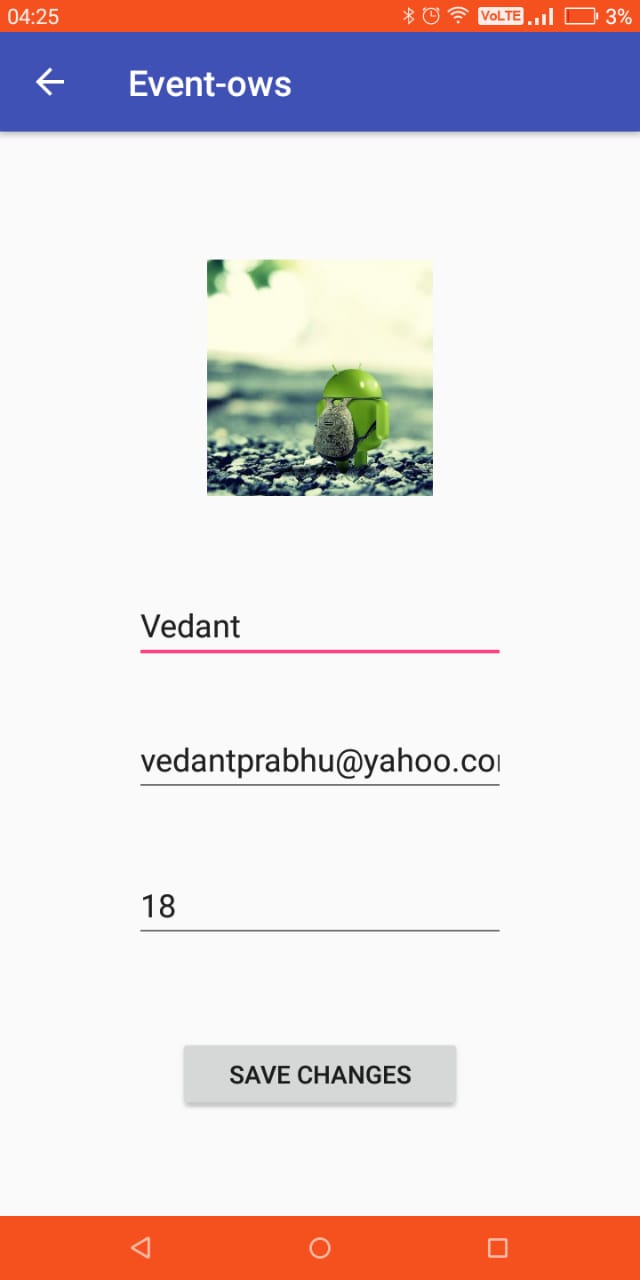
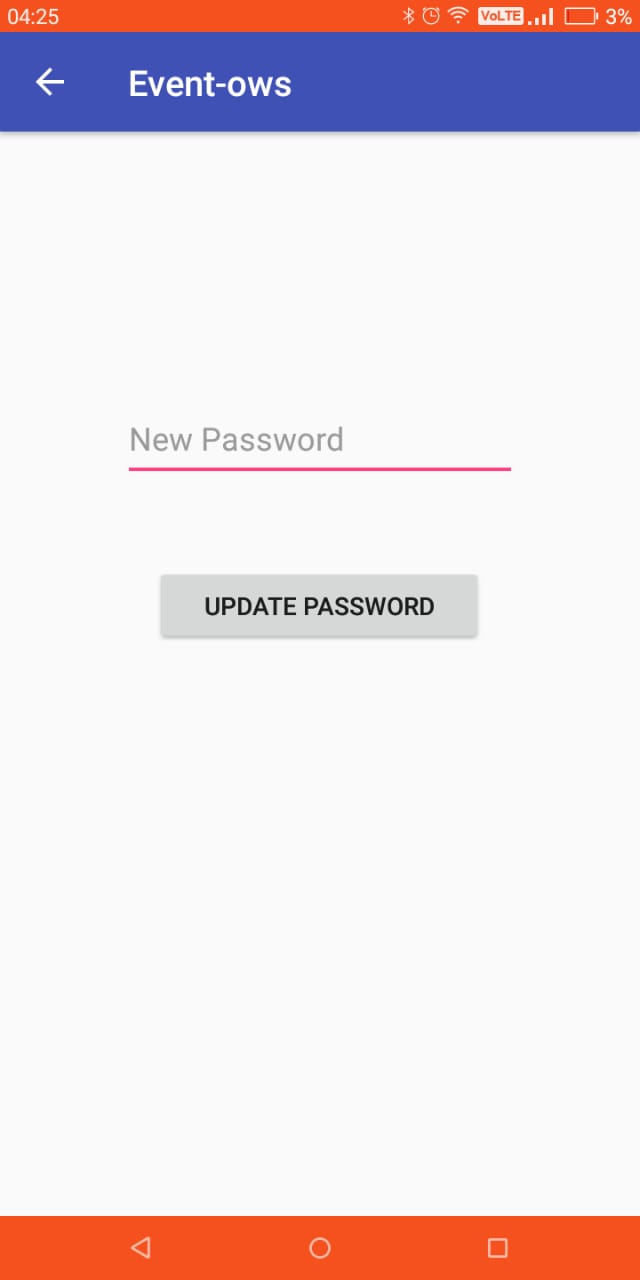
App Popup Screen Login Screen Register Screen

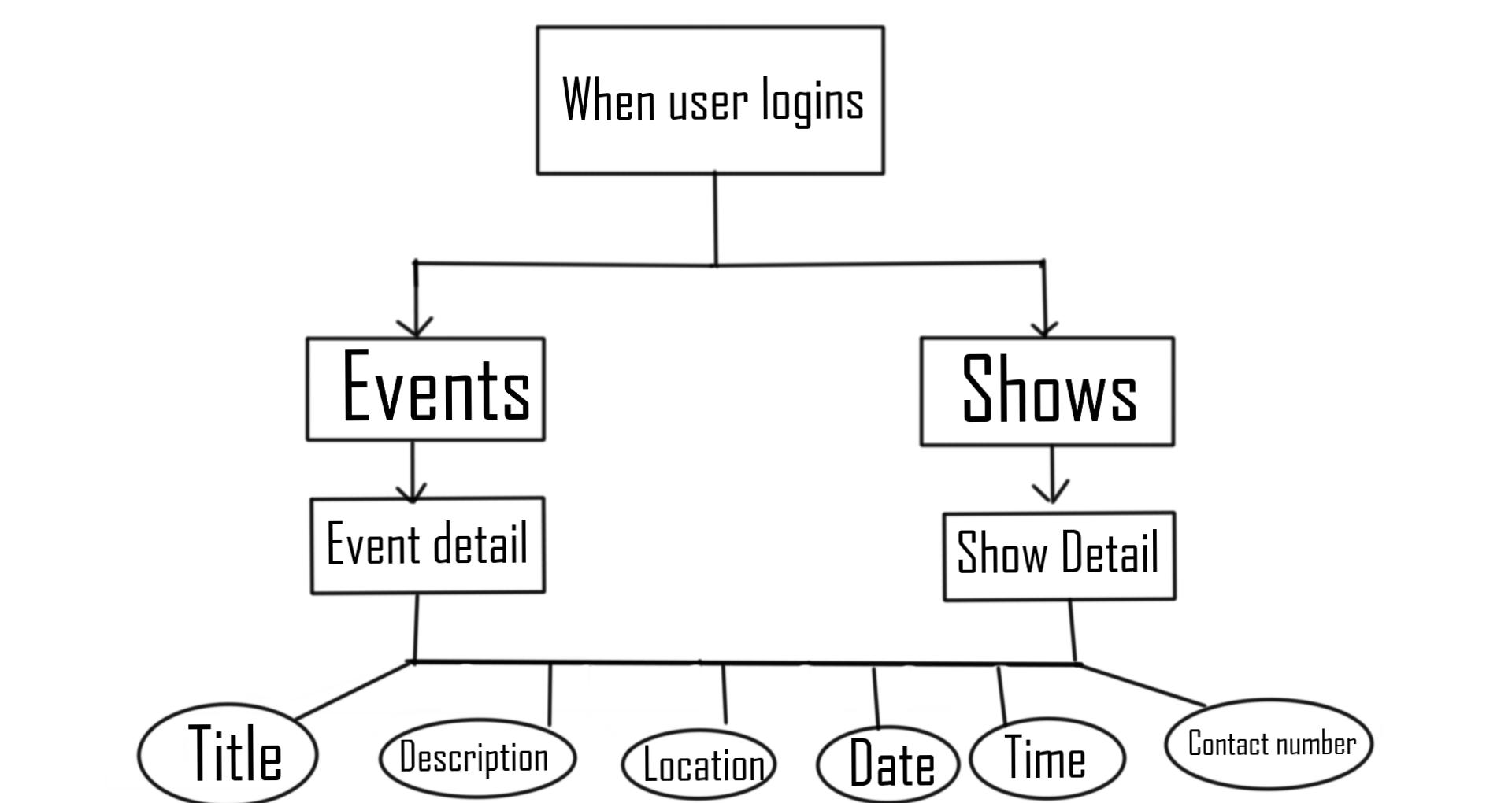
Forgot Password Screen User Profile Screen

Edit Profile Screen Change Password Screen

**Event and Shows**

*******ER DIAGRAM REPRESENTING EVENTS AND SHOWS***

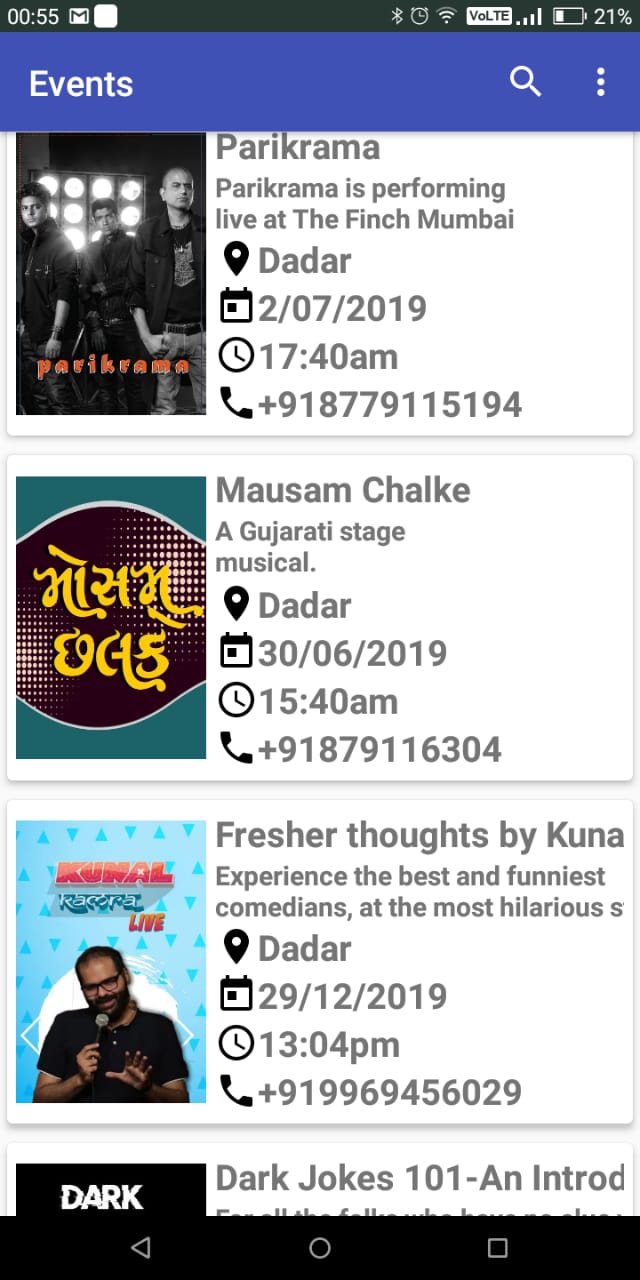
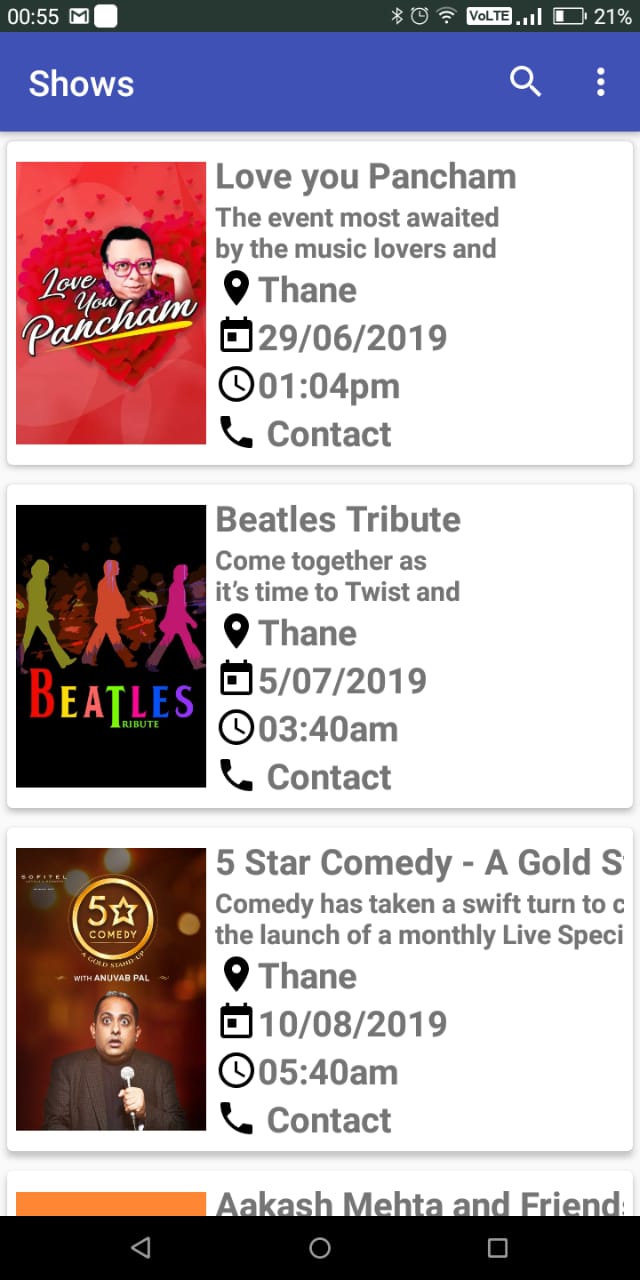
It describes as follows: -

* When a user has been authenticated successfully, he is logged into the app.
* In the home fragment, the user will observe two buttons namely, Events and Shows.
* On clicking Event/Shows, a list of Event/Shows are displayed which have the entities: -

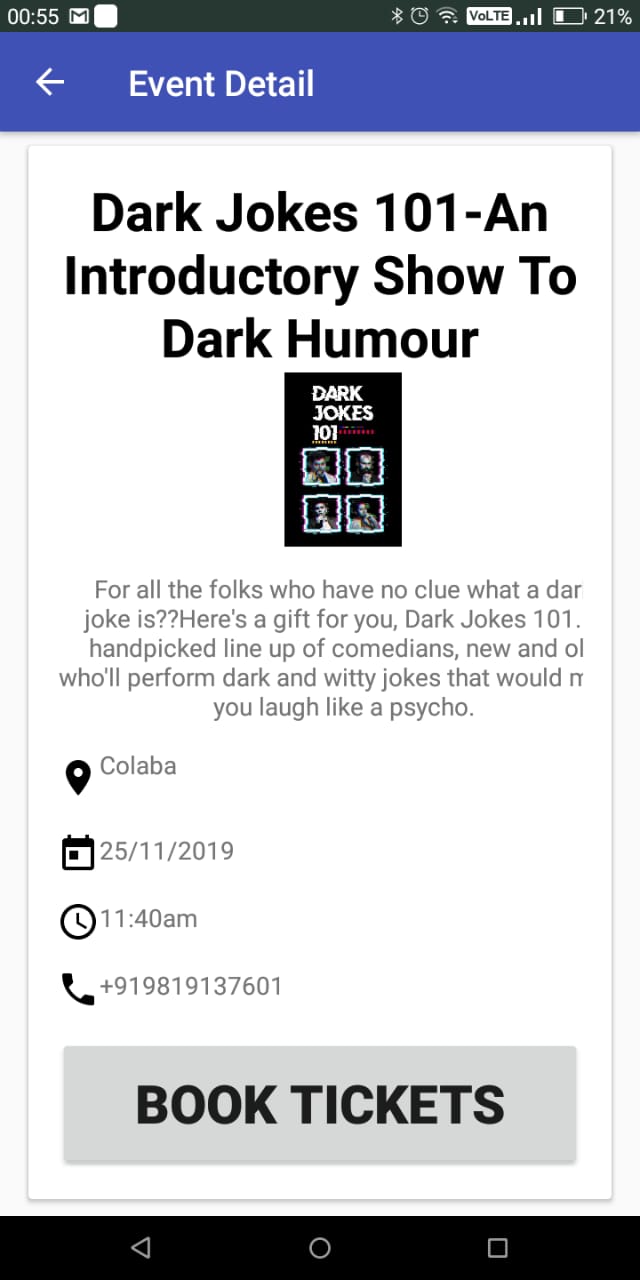
1. Title
2. Description
3. Location
4. Date
5. Time

* Clicking on an event/show further, a new screen is opened where it describes about the description more of that particular event Clicking on the Book Tickets button.
* Payment gateway has not been implemented therefore it will pop up a message saying tickets Booked Successfully and it will revert to the Home Fragment.

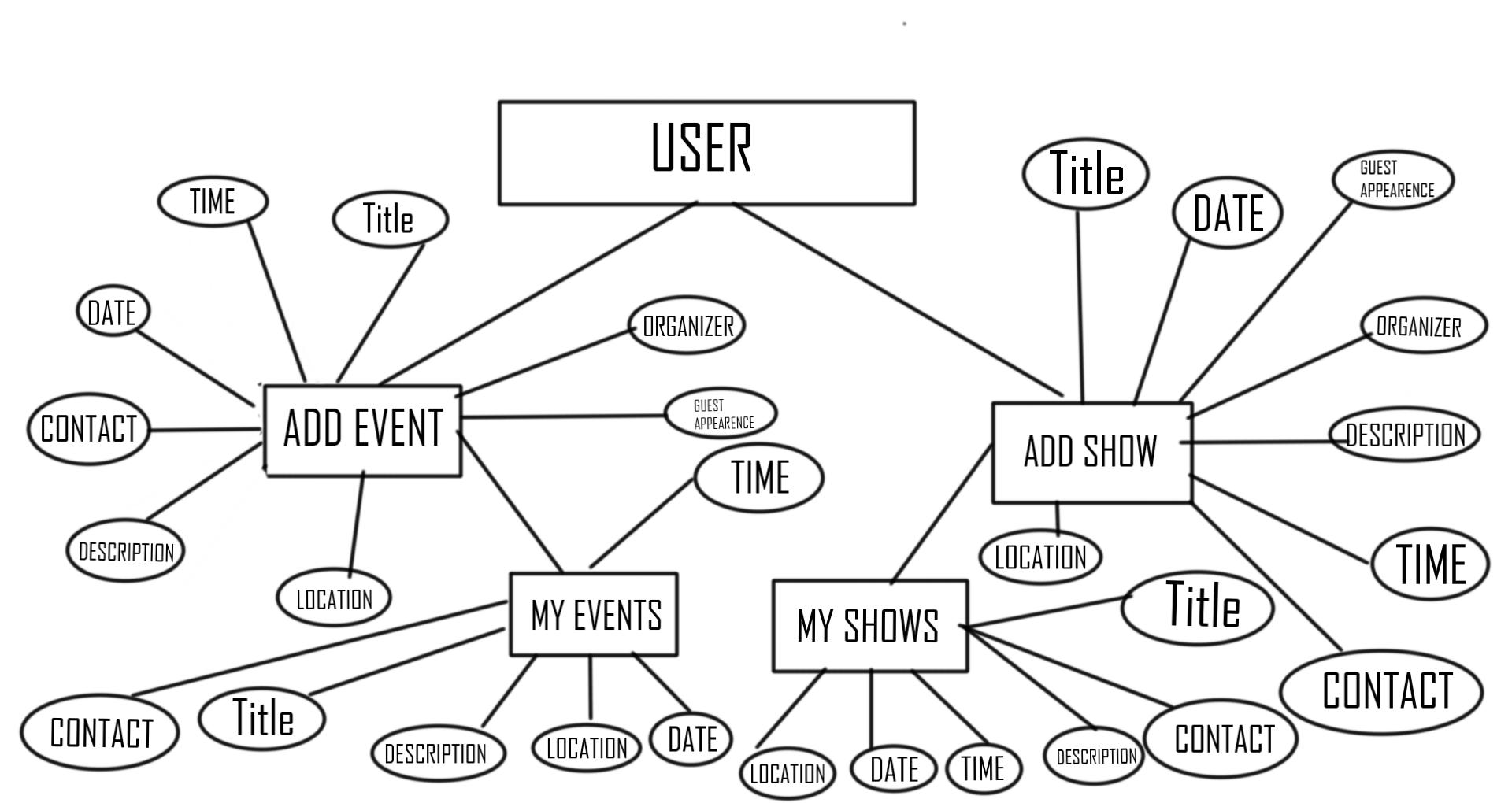
Main Page Events Activity Shows Activity

Event Details Activity Toast showing tickets booked

**Adding Event and Shows**

****

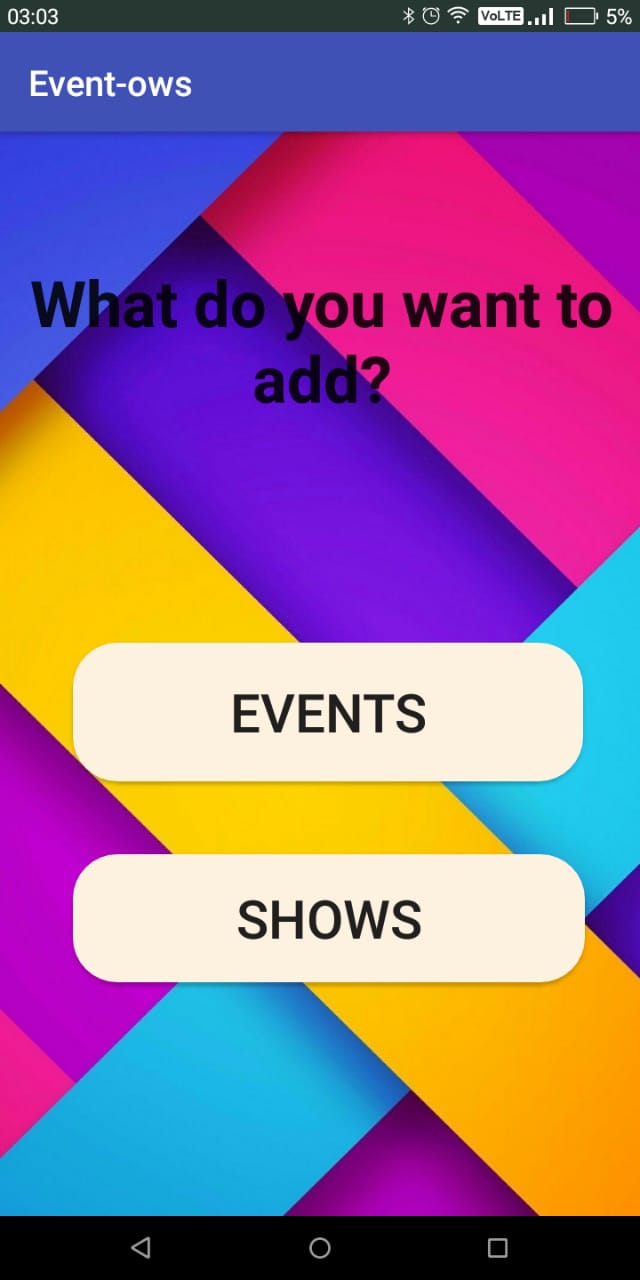
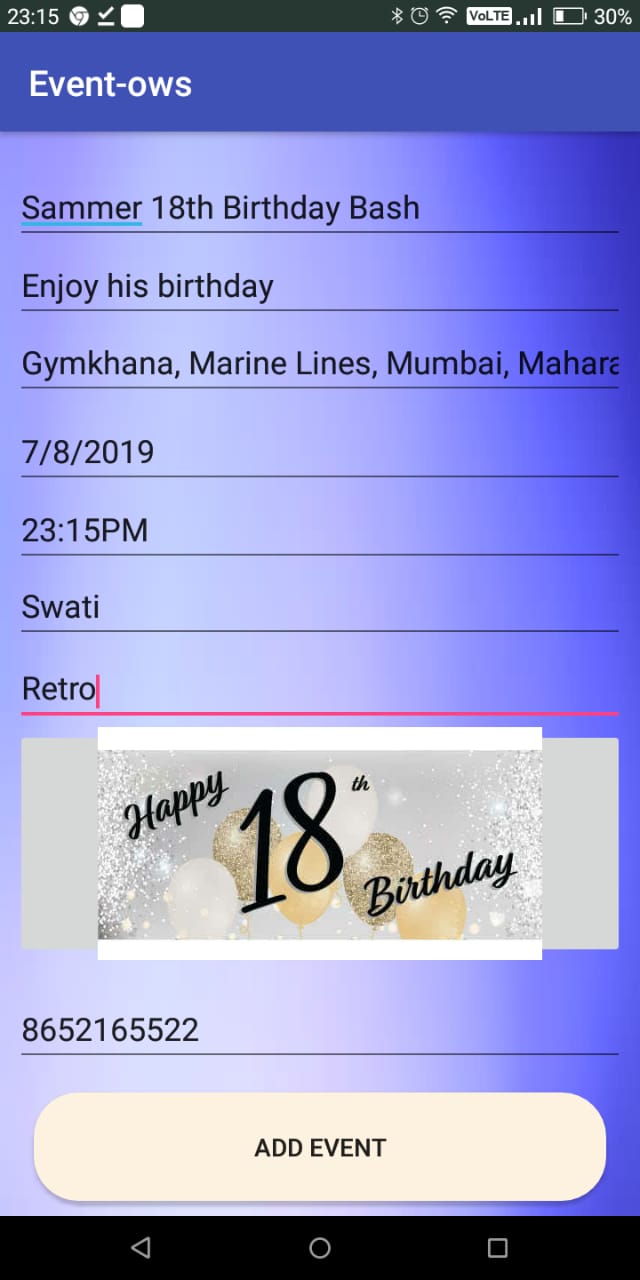
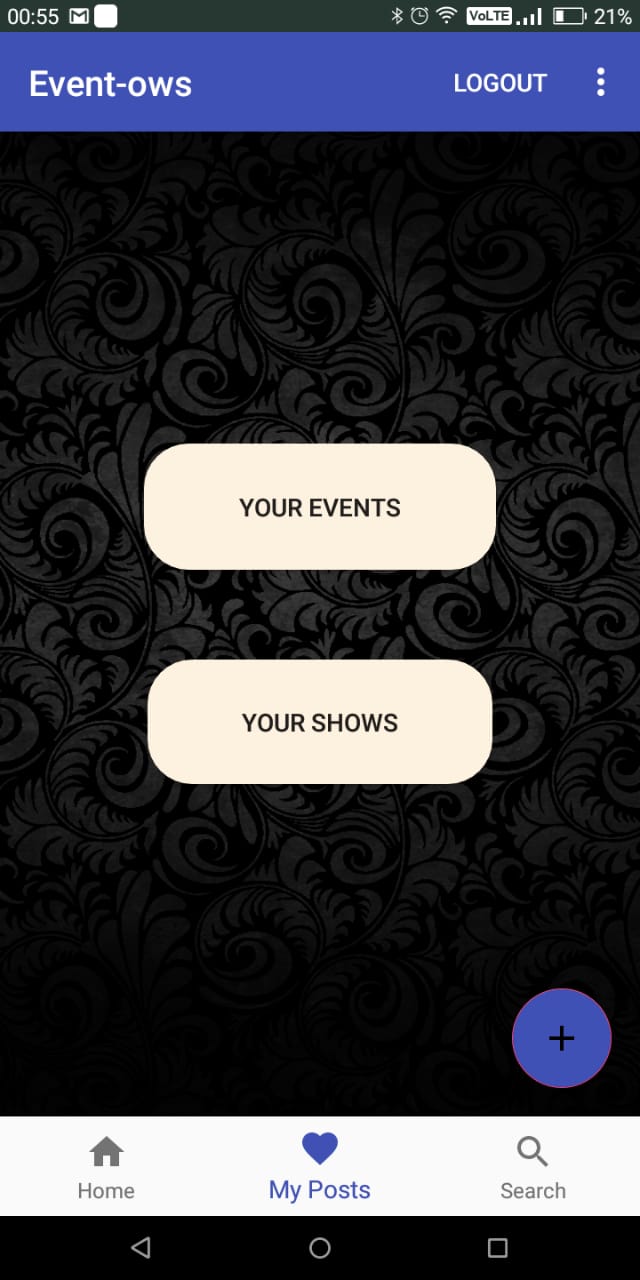
***ER DIAGRAM REPRESENTING ADDING EVENTS AND SHOWS***

* I have implemented a functionality in the application where you can add an Event/Show.
* When you click on the floating plus button a new screen will open which will show two buttons namely Add Event and Add Shows.
* On clicking Any of the two buttons a screen will open which will display the following entities: -

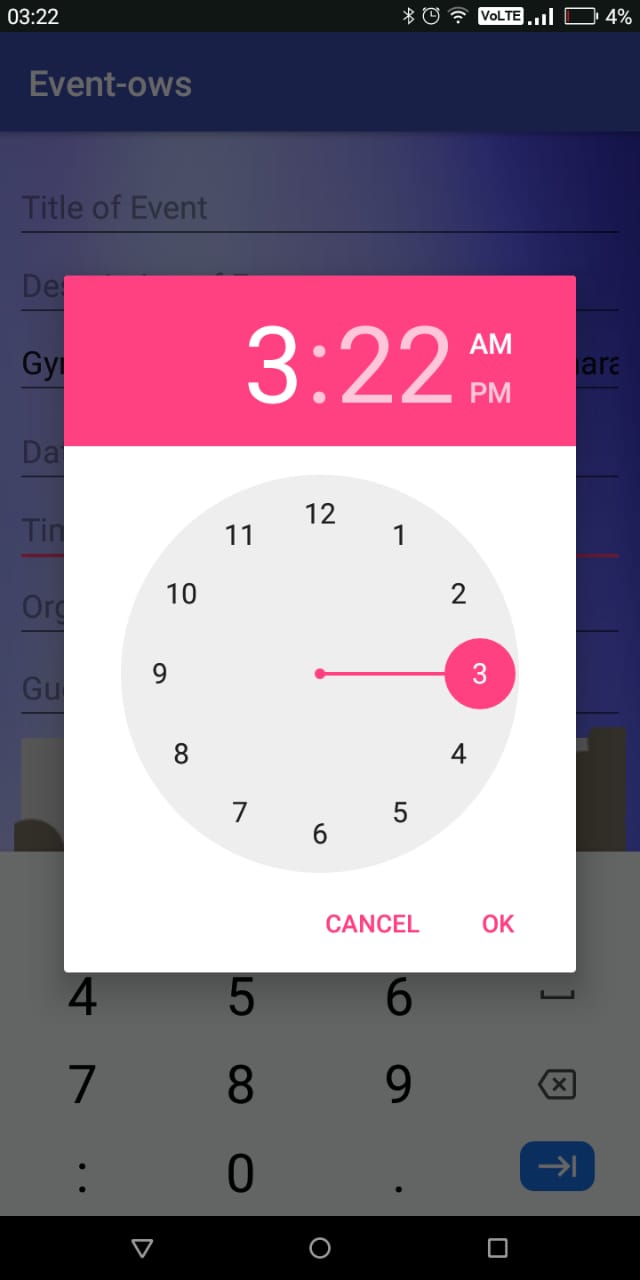
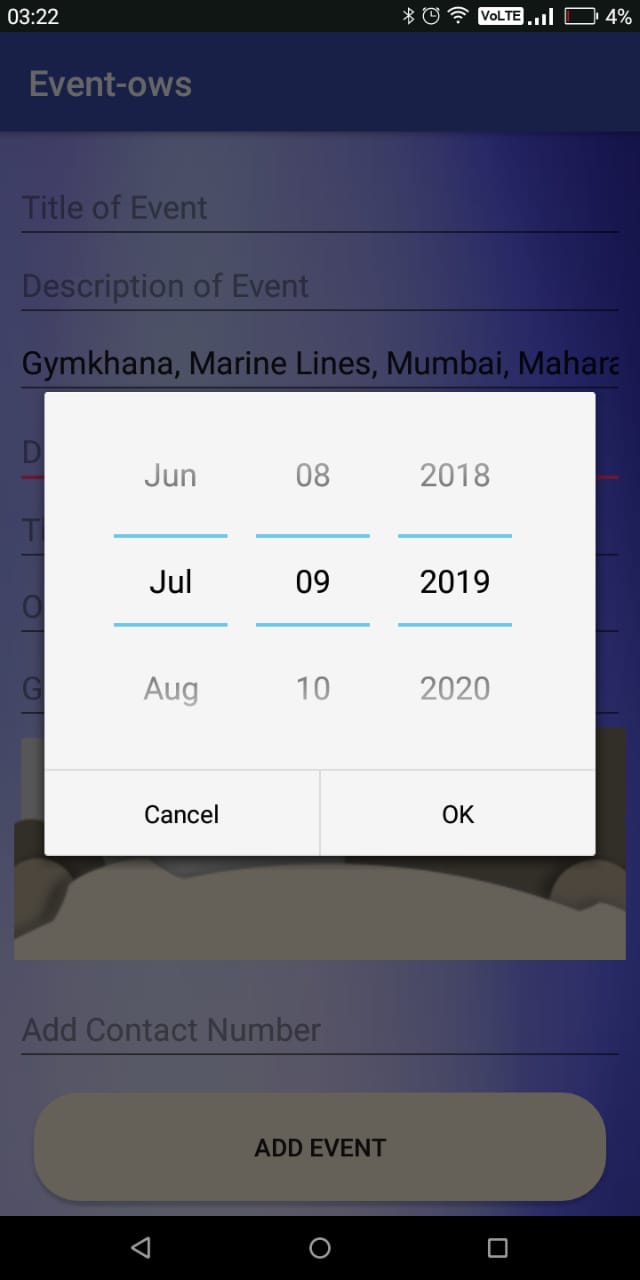
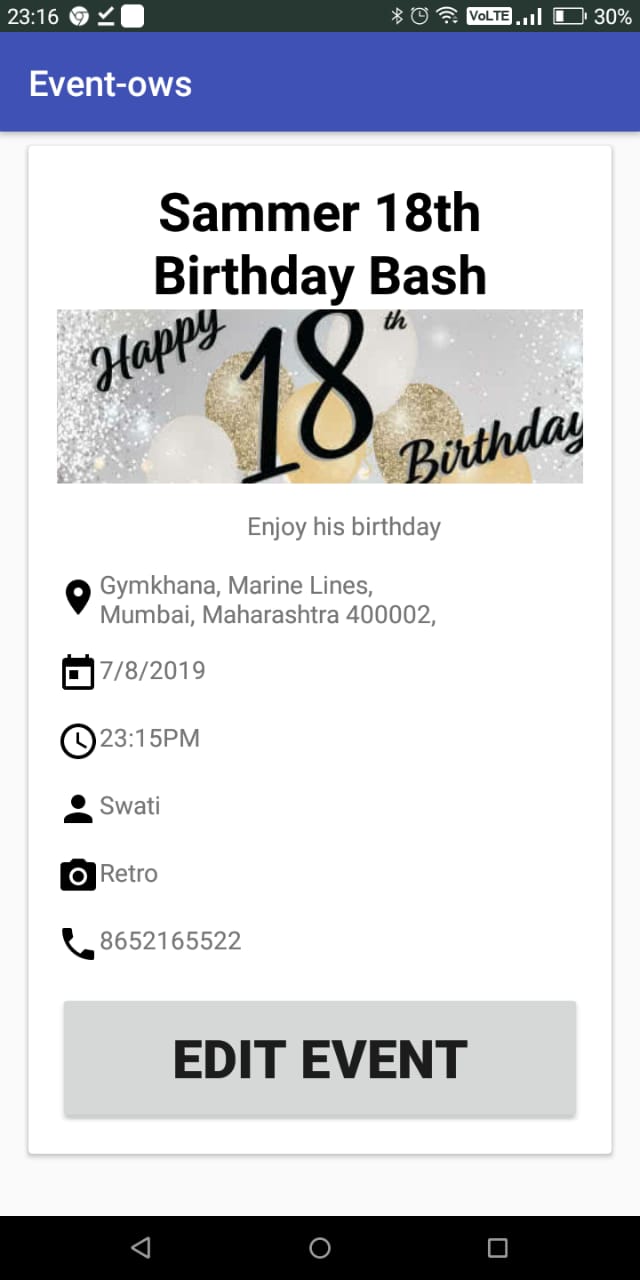
1. Title
2. Description
3. Location
4. Date
5. Time
6. Organizer
7. Guest Appearance
8. Contact Number

* All these fields are required to be filled otherwise the app will throw a toast saying “To fill all details”.
* On clicking add event will store all your details and will display it in My Posts Fragment.
* Under My posts you will observe two buttons which says My Events and My shows.
* Clicking on My events will display your Event with the image. Similarly, it happens for Shows as well.
* Further you can edit the particular Event/Show by clicking on the edit Event button
* Clicking save will update your changes.
* If you have not added any event or show and try to click on My Event or My Show the app will close down since there is no value hence the value is null.

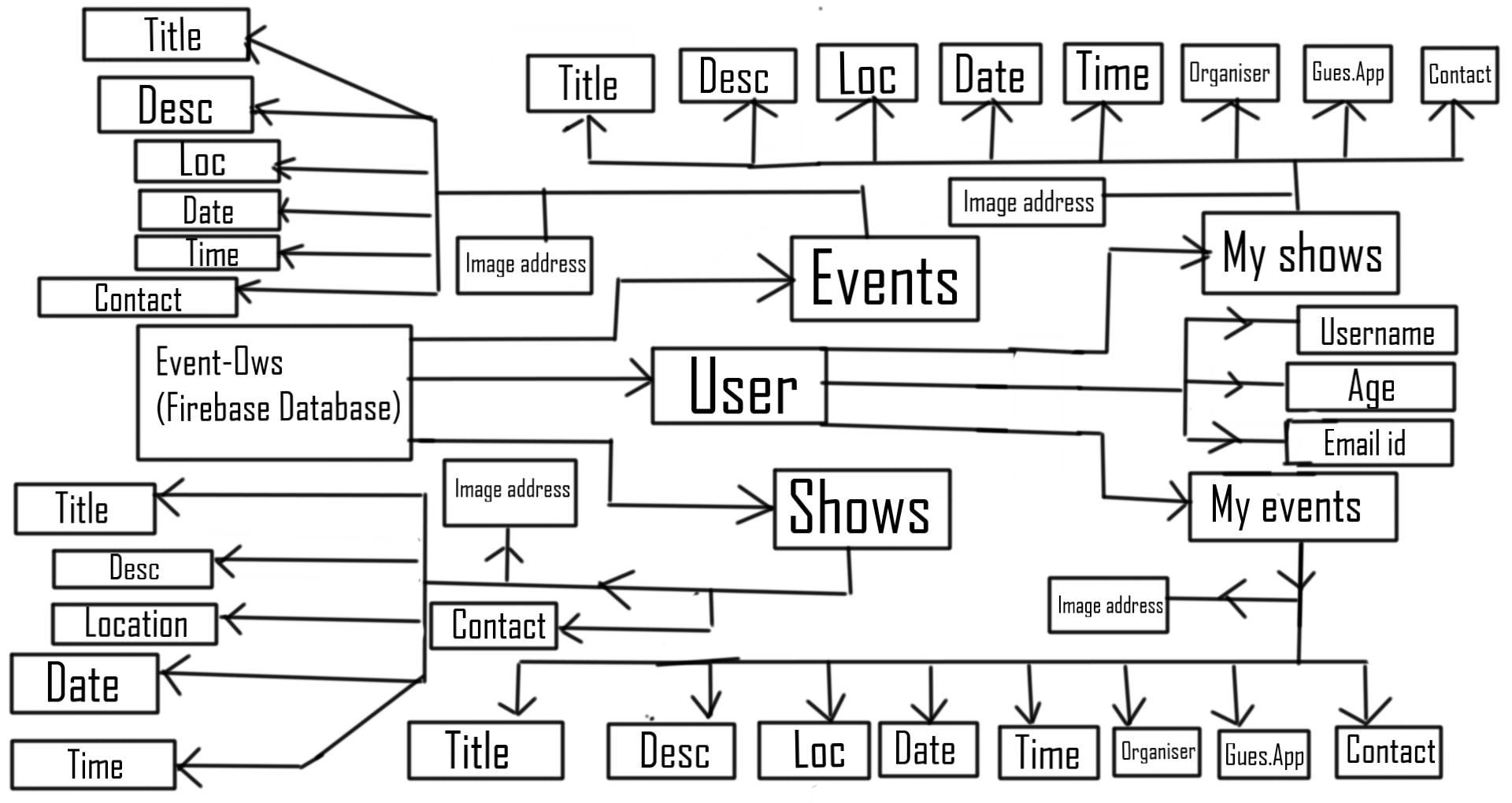
On clicking the “+” Event Details to be filled Viewing your Event

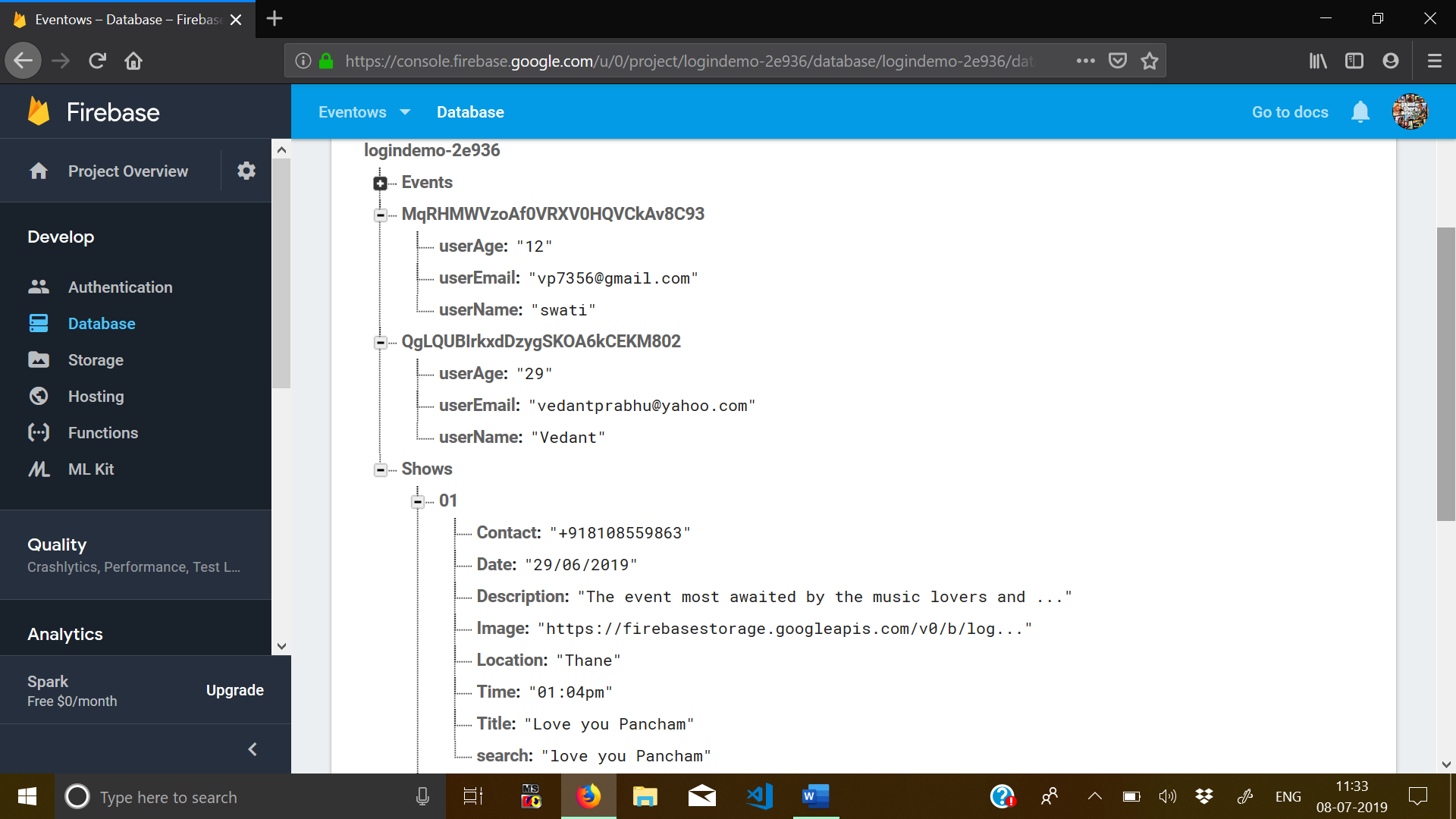
  

Event is viewed Date Functionality Time Functionality

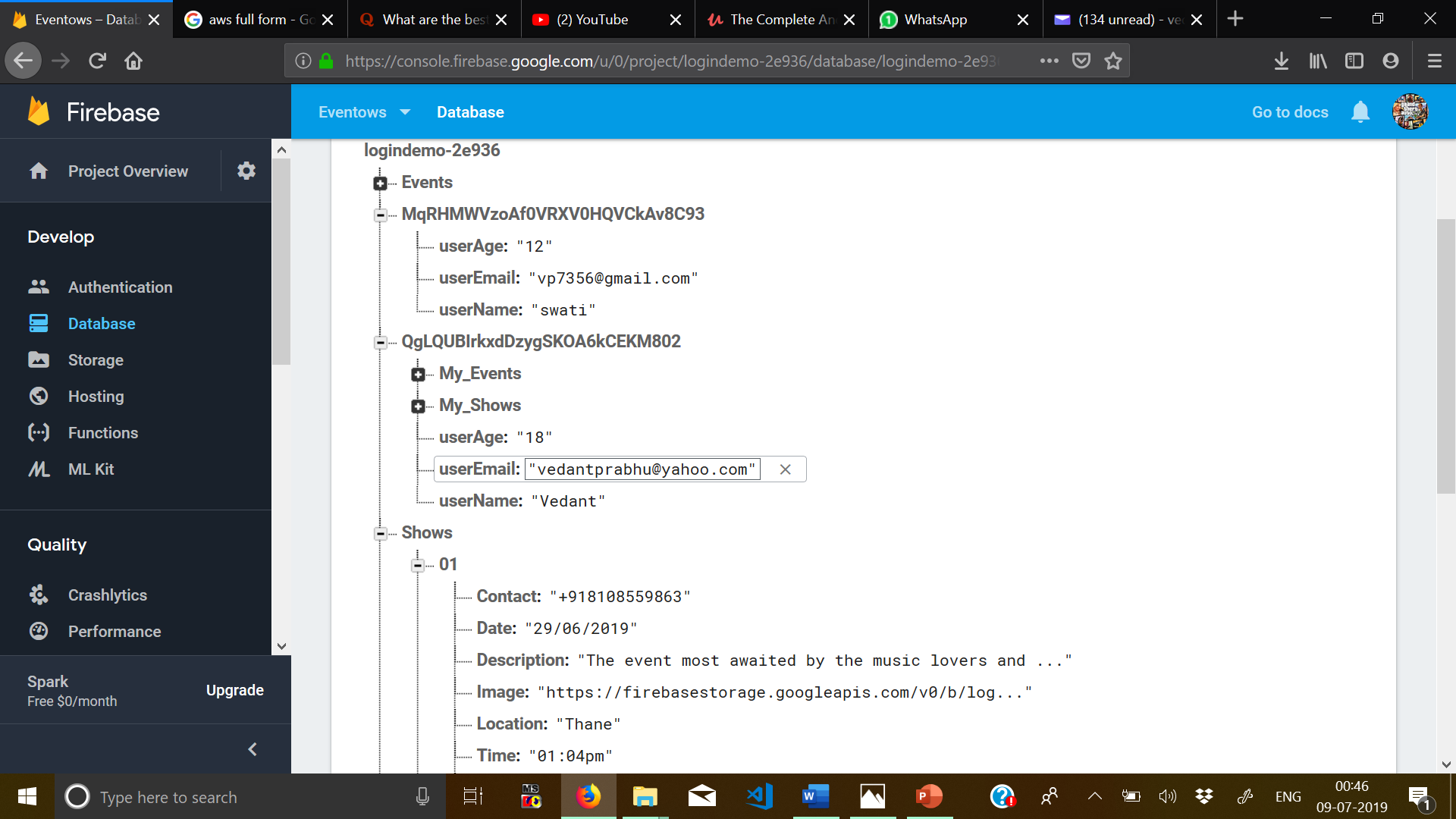


**Database Design for EventOws**





Realtime Database before adding Event/Show



Realtime Database after Adding Event/Show or both.

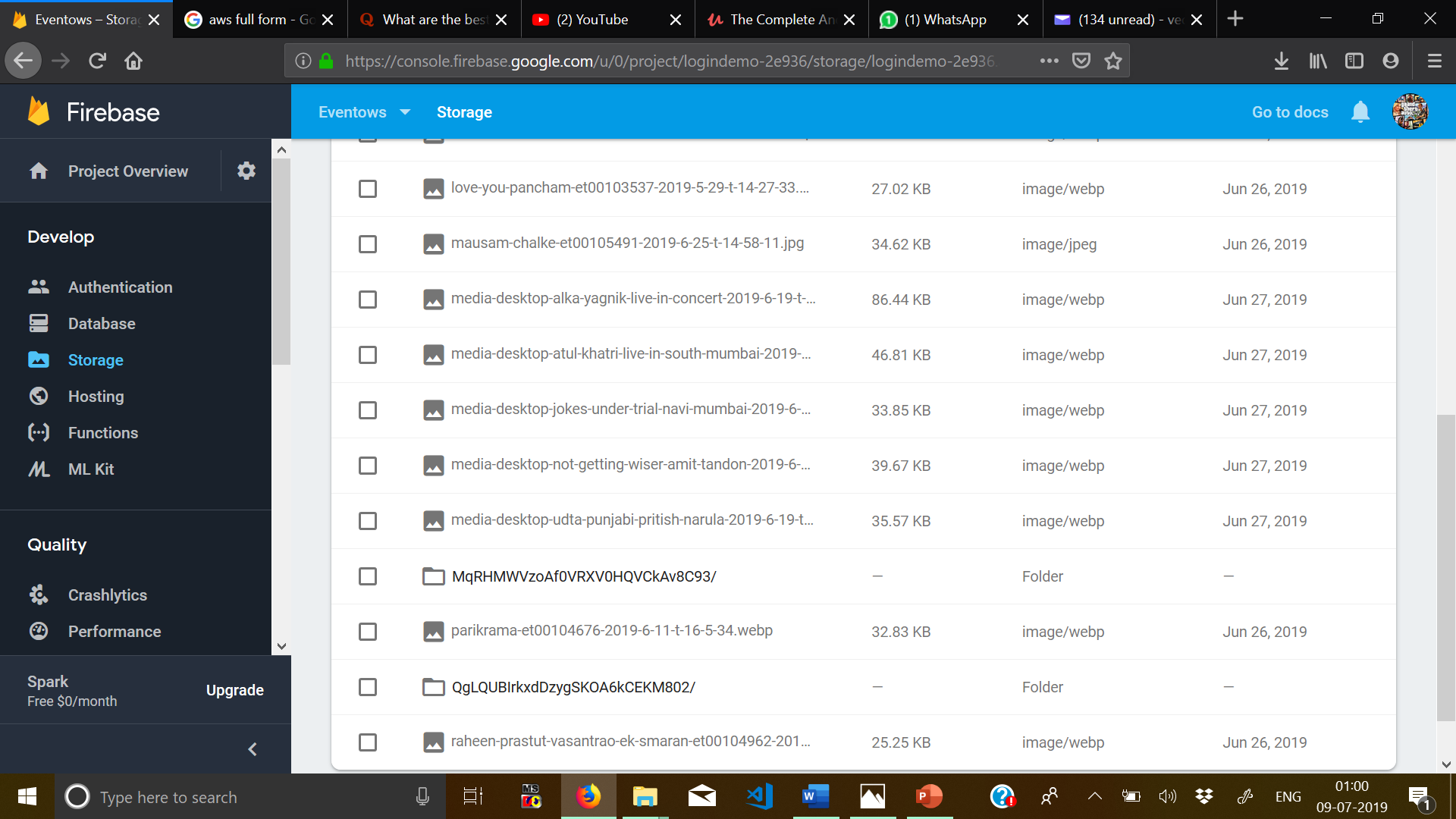
* The database is divided into 3 main tables: -

1. Events Table: -Will consist of the events stored in the database
2. Shows Table: -Will consist of the shows stored in the database.
3. User UID table: - The name of the table is the special unique ID that is generated after Successful Registration. It will consist of user details which is used for User Profile. This table is further divided into: -
4. My Events: -This table is created only when you Add a new Event.
5. My Shows: - This table is created only when you Add a new Show.

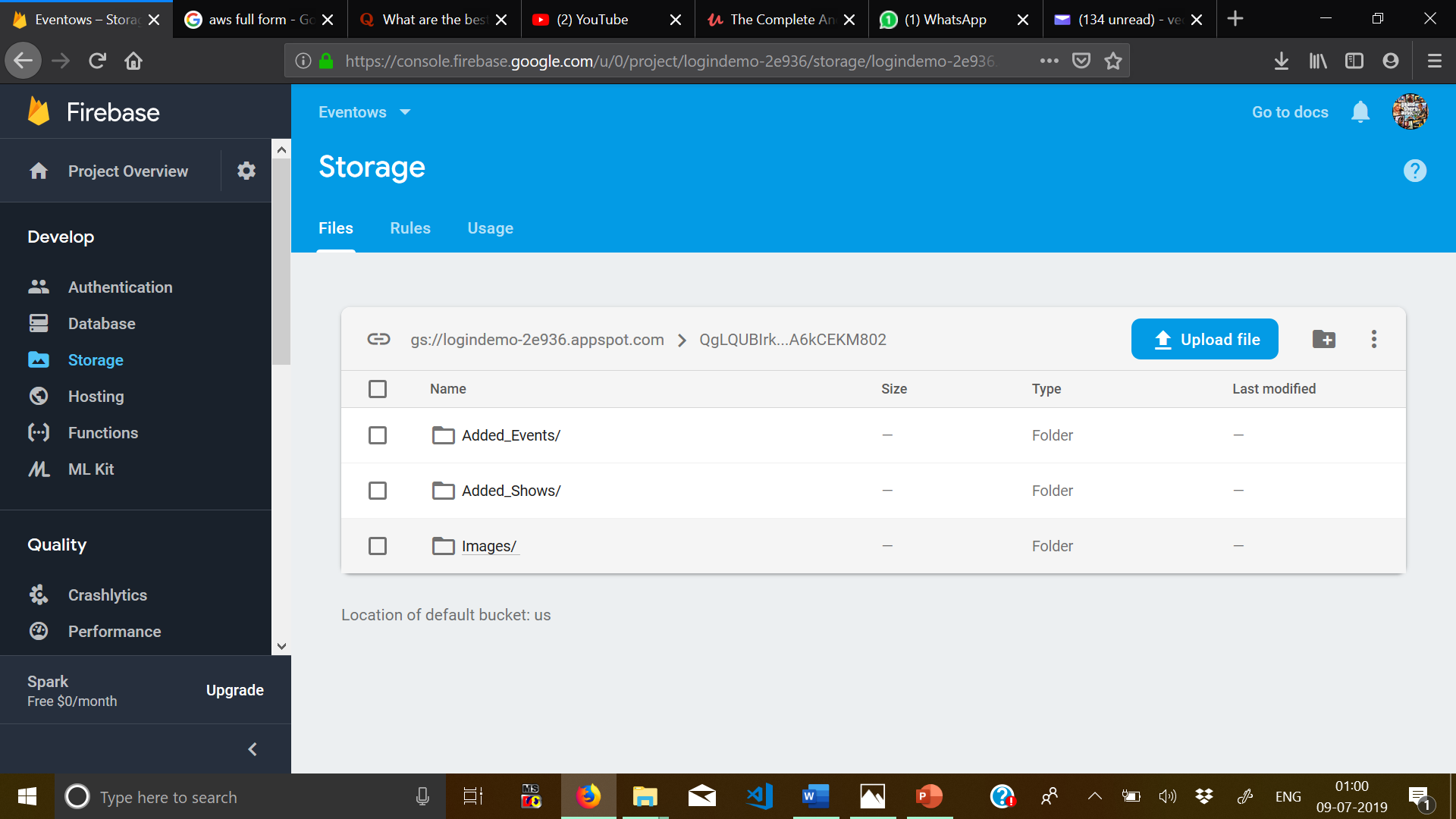
**Firebase Storage Design for EventOws**

It describes as follows: -

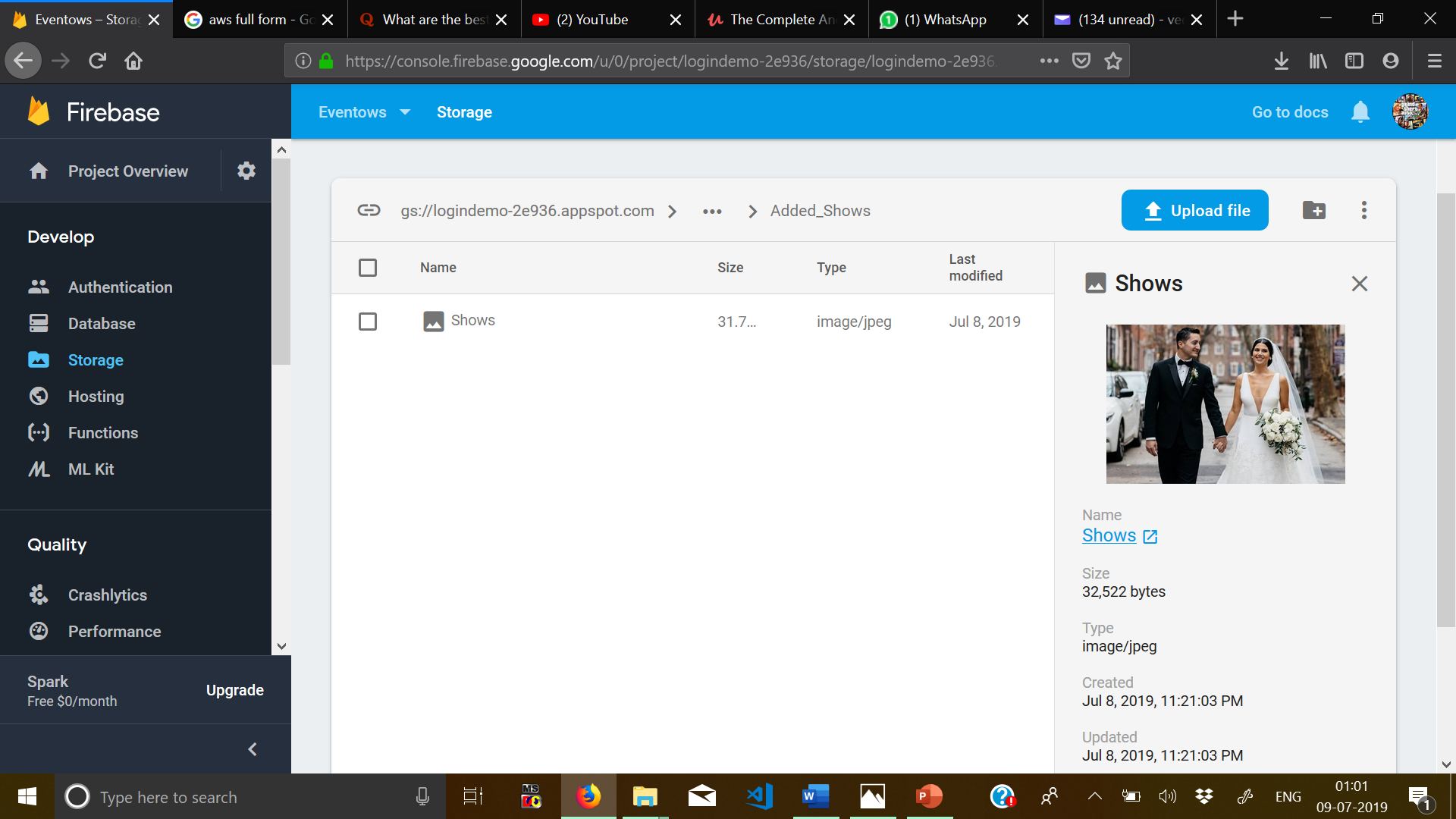
* Firebase Storage acts as a cloud storage to store images, pdfs or word documents.
* For my application I have implemented Firebase Storage because it includes profile pictures to update or add for the registered user
* Secondly, when we add an event/show, we can also add picture to it. To store that picture and later update it, I have used Firebase Storage.
* As you can see pictures of Events are put directly to the storage.
* Under a particular User UID folder there are three sub folders namely Images, Added Events, Added shows.
* Images will have your added Profile Picture
* Added\_Events will contain the image you uploaded for your Event.
* Added\_Shows will contain the image you uploaded for your Show.



The circled with red represents the folder of the current user representing as User UID.

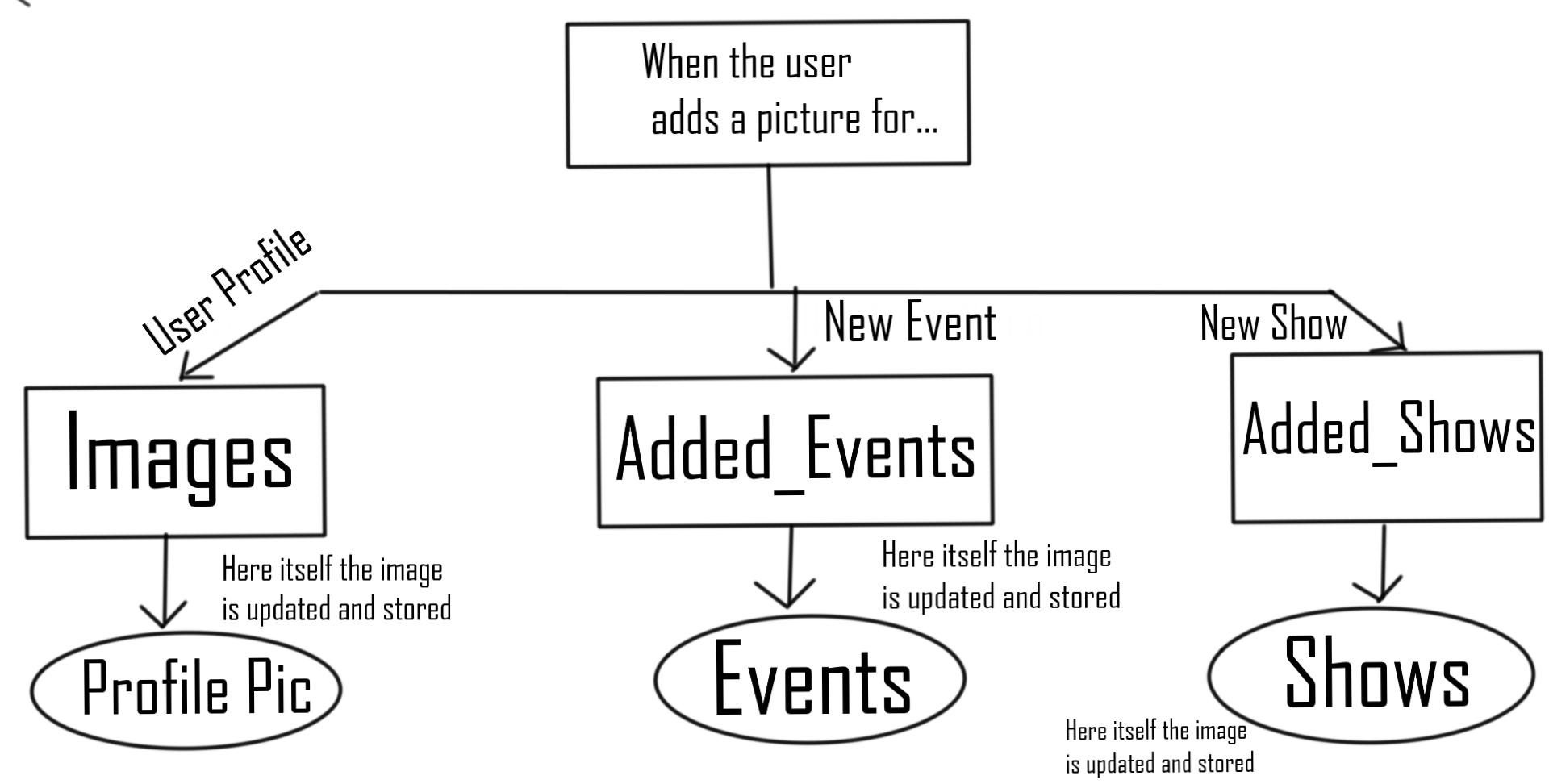


Inside the user UID folder consists of three folders namely Added\_Events, Added\_Shows and Images



For Example, under My\_Shows it shows the added Show picture which is updated and replaced according to you needs.

**The below diagram describes how the Firebase Storage works.**

****

**Application Hosting for EventOws**

* **For my application I have decided to use the Amazon AWS for cloud storage.**
* **As a temporary use I am currently using Firebase Storage for storing the data.**
* **The reason for using Firebase is because it was easy to learn and understand.**
* **The reason for choosing AWS as my hosting is because: -**

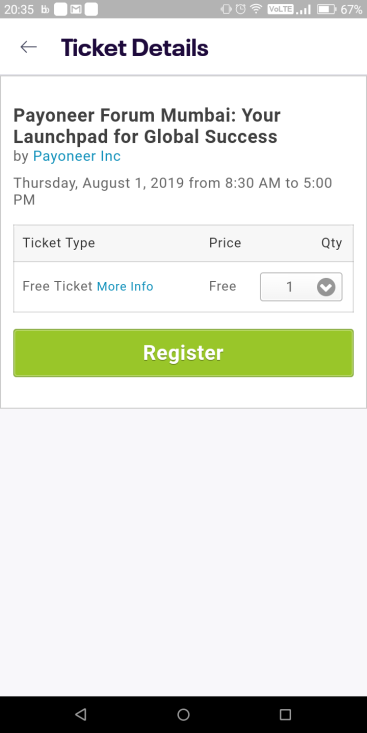
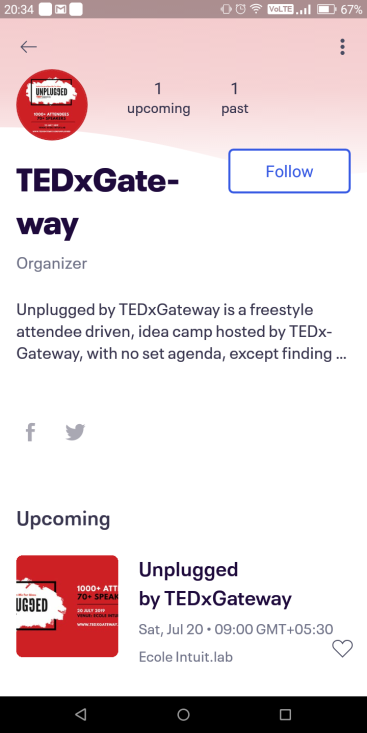
1. **It has good security**
2. **It is easy to use.**



**Future of The App**

* In this app more upgrades that can be introduced are: -

1. More interactive User-Interface to be implemented in future.
2. Adding the Maps API and tracking features to enable identifying locations more precisely.
3. Adding Comments and Reviews to every particular Event or Show to look it more social and attractive.
4. Website shall soon be developed.
5. Payment Integration (Brain-Tree) shall be implemented for fast and safe checkout.

**** **** 