San Francisco State University School Project

CSC667/867 Spring 2019

EventUp

Team 4

<https://github.com/csc667-02-sp19/csc667-sp19-Team04>

Cory Lewis (Team Lead, [clewis9@mail.sfsu.edu](mailto:clewis9@mail.sfsu.edu))

Mitul (GitHub Master)

Chintan Sanjay Puri

Alex Wolski

Vincent Santos

Milestone 3

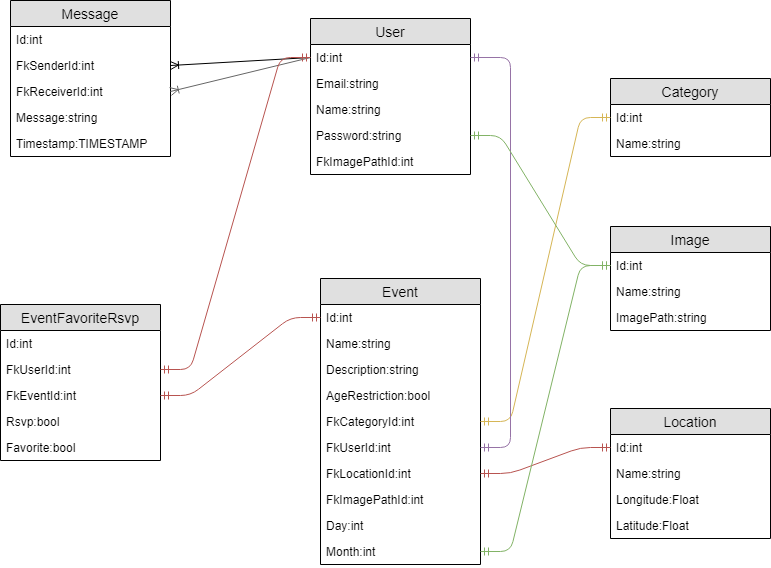
04/05/19

**1. Definitions of operations for database entities**

* Register: POST request that passes user information and creates a User in the database
* Login: POST request that creates a ‘session’ between the user and the server via

their User information and token.

* Profile Update: PATCH request that updates some User information
* Event Post: POST request that creates an Event in the database
* Grab Event(s): GET request that returns an Event or list of Events
* Event Update: PATCH request that updates some Event information
* RSVP: POST request that allows the user to signal attendance for an event ahead of time by creating an entry in the Flag table
* Get Attendees: GET request that returns all the Users who RSVP’d for an event, to inform the event host. Attendees are retrieved through associated Flag entries
* Get Reservations: GET request that returns all the current Events a user has RSVP’d for. Reservations are retrieved through associated Flag entries
* Message Send: POST request which creates a new entry in the Message table with the two associated users
* Event Expired: DELETE request that removes the expired Event from the database
* User Delete: DELETE request that deletes a User from the database



Relational Database Diagram. Most tables have a 1 to 1 relationship. The User to Message table has a 1 to many relationship.

**2. Define routes for MVC structure (express Rest API routes)**

**post auth/signup**

Purpose:

The user sends their name, e-mail, and password in the html body.

Then the API will create an account with that information.

Response:

If the signup succeeds,

res.send("Succeeded");

If the signup fails,

res.status(400).send("Account with that email already exists")

**post auth/signin**

Purpose:

The user sends their Email and password in the html body.

Then the server returns an authentication token that grants access.

Response:

If the login succeeds,

res.send(authToken);

If the login fails,

res.status(401).send("invalid Email/password");

**get api/events**

Purpose:

Get a list of events.

Response:

Returns a json object containing all the events to be displayed.

Json structure:

Int Id;

String Name;

String ImageName;

String ImageURL;

Bool AgeRestriction;

Int Month;

Int Day;

**get api/events/:id**

Purpose:

Get more information on a specific event.

Response:

Returns a json object containing the information.

Json structure:

Int Id;

String Name;

String Description;

String Category;

String ImageName;

String ImageUrl;

String Address;

Bool AgeRestriction;

String Date;

Int Day;

Int Month;

Float Longitude;

Float Latitude;

**get api/events/:categoryid**

Purpose:

Get a list of events that are a certain category.

Response:

Returns an array of json objects containing all the events for that category.

Json structure:

Int Id;

String ImageName;

String Name;

String ImageUrl;

Bool AgeRestriction;

Int Month;

Int Day;

**post api/rsvp/:id**

Purpose:

Get the events the user is going to attend.

Response:

Returns a json object containing all of the events.

The Json object has the same structure as the events request.

**Get api/eventmap**

Purpose:

Get the locations of all of the events near SFSU

Response:

Returns an array of Json objects with all the locations.

Json structure:

Int Longitude;

Int Latitude;

Int EventId;

**Get api/profile/:id**

Purpose:

Get the user's information.

Response:

Returns a Json object with the information.

Json structure:

String Name;

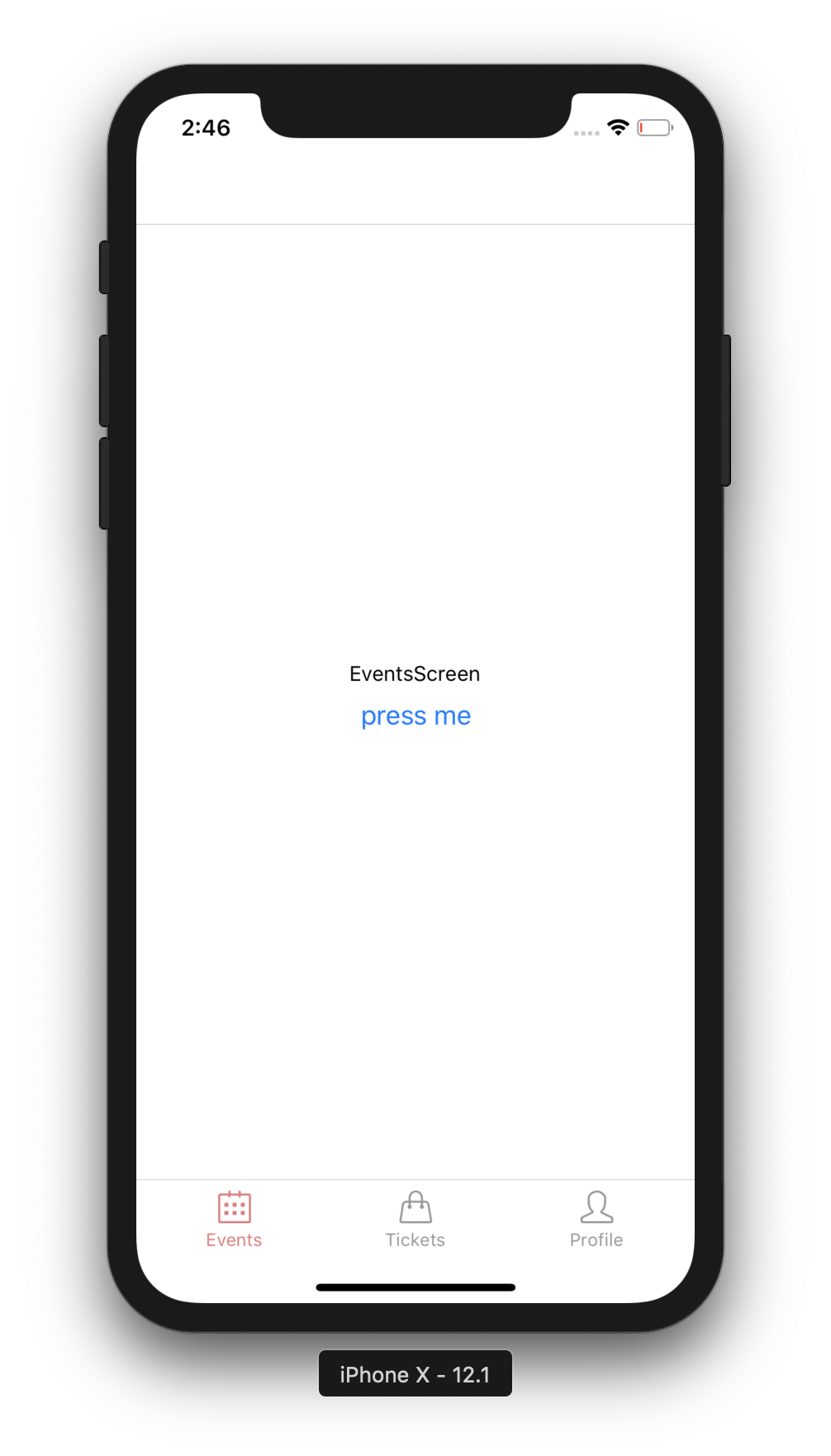
String Email;

String ImagePath;

**3. Implemented Wireframes**

* The only views not created are ‘Favorites’, and ‘MAP’.

Events Screen and Event Details Screen





Profile Screen, and RSVP’d events screen

