Android\_Code

//Android Tags Main ~Brandi’s Section Start (Tags)

/\*\*

\*    Name: MainActivity

\*    Creator: Brandi Werner

\*    Component: This is the main class. When the application is open this is the

\*           class that first pulls up.

\*    Created: 10/05/2017        Last Edited:    11/12/2017

\*    Reason: Automatically created

\*    Uses: Pretty much collects data and calls other classes

\*/

public class MainActivity extends AppCompatActivity {

  private Intent intent;

  DatabaseHelper myDb;    // Instance of the Database

  EditText editEventName;        // The text input1 on the screen

  EditText editEventDesc;        // The text input1 on the screen

  Button btnAddData;        // Button on screen to add data

  boolean CS;            // Whether CS checkbox is checked

  boolean CE;            // Whether CE checkbox is checked

  @Override

  protected void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

      setContentView(R.layout.*activity\_main*);

      myDb = new DatabaseHelper(this, null, null, 1);

      editEventName = (EditText) findViewById(R.id.*E\_Name\_pt*);

      editEventDesc = (EditText) findViewById(R.id.E\_Desc\_pt);

      btnAddData = (Button)findViewById(R.id.button\_add\_data);

    AddData();    // Calling add data

}

/\*\*

\*Description: Class for the check boxes that determines which boxes have been checked

\*             and sets a variable for them if they have been checked.

\*/

public void onCheckboxClicked(View view){

   boolean checked = ((CheckBox) view).isChecked();

   //Check which checkbox was clicked

   switch(view.getId()){

       case R.id.CS\_check:

           if (checked) // Set a variable to true that way I can find out which class needs to be displayed in the next view

               CS = true;

           else

               CS = false;

           break;

       case R.id.CE\_check:

           if(checked) //Set a variable to true that way I can find out which class needs to be displayed in the next view

               CE = true;

           else // Not much to do

               CE = false;

           break;

   }

}

/\*\*

\*Description: This function is for when the button on the homescreen is clicked.

\*      When the button is clicked it will take the user to the tabSearch view.

\*/

public void sendMessage(View view){

   intent = new Intent(MainActivity.this, tabView.class);

   startActivity(intent);

}

/\*\*

\*Description: This function is to take the information the user put on the screen

\*          into the database. When the add data button is clicked on the screen

\*         it runs the add data function in the database handler class.

\*/

public void AddData(){

      btnAddData.setOnClickListener(

              new View.OnClickListener() {

                  @Override

                  public void onClick(View v) {

                      /\*\* Call insert Data Method which returns true or false telling if the insert succeeded or failed \*\*/

                      boolean isInserted =  myDb.insertData(editEventName.getText().toString(), editEventDesc.getText().toString(), CS, CE);

          // Displays a toast message if add data class worked or failed

                      if(isInserted == true)

                          Toast.makeText(MainActivity.this, "Data Inserted", Toast.LENGTH\_LONG).show();

                      else

                          Toast.makeText(MainActivity.this, "Data not Inserted", Toast.LENGTH\_LONG).show();

              // Resetting the edit text box to empty string

                      editEventName.setText("");

                      editEventDesc.setOnClickListener("");

                  }

              }

      );

  }

}

//Activity\_Main.xml ~Brandi Werner and Tsung-Han Hsieh (Tags)

<!--

\*    Name: MainActivity

\*    Creator: Brandi Werner and Tsung-Han Hsieh

\*    Component: This is the main class’s view. When the application is

\*           what is shown up on the screen.

\*    Created: 10/05/2017        Last Edited:    11/12/2017

\*    Reason: Automatically created

\*    Uses: Manipulate how the home screen looks.

-->

<?xml version="1.0" encoding="utf-8"?>

<!-- Linear-layout that holds two linear layouts horizontally -->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

  xmlns:app="http://schemas.android.com/apk/res-auto"

  xmlns:tools="http://schemas.android.com/tools"

  android:id="@+id/LinearLayout\_Base"

  android:layout\_width="match\_parent"

  android:layout\_height="match\_parent"

  android:orientation="vertical"

  android:weightSum="1"

  tools:context="com.brandi.tabs.MainActivity">

  <!-- Linear layout that holds the Database items -->

  <LinearLayout

      android:id="@+id/Layout1"

      android:layout\_width="match\_parent"

      android:layout\_height="match\_parent"

      android:layout\_alignParentLeft="true"

      android:layout\_alignParentStart="true"

      android:layout\_alignParentTop="true"

      android:layout\_weight=".4"

      android:orientation="vertical">

    <!-- Text that says “Event Name” -->

      <TextView

          android:id="@+id/E\_Name\_Text"

          android:layout\_width="wrap\_content"

          android:layout\_height="wrap\_content"

          android:text="Event Name"

          android:textColor="@android:color/black" />

    <!-- Text that takes in the Event Name -->

      <EditText

          android:id="@+id/E\_Name\_pt"

          android:layout\_width="wrap\_content"

          android:layout\_height="wrap\_content"

          android:ems="10"

          android:inputType="textPersonName"

          android:text="Name" />

    <!-- Text that says “Event Description” -->

      <TextView

          android:id="@+id/E\_Desc\_Text"

          android:layout\_width="wrap\_content"

          android:layout\_height="wrap\_content"

          android:text="Event Description" />

    <!-- Text that takes in the Event Description -->

      <EditText

          android:id="@+id/E\_Desc\_pt"

          android:layout\_width="wrap\_content"

          android:layout\_height="wrap\_content"

          android:ems="10"

          android:inputType="textPersonName"

          android:text="Name" />

    <!-- Text that says “Tags” -->

      <TextView

          android:id="@+id/Tags\_text"

          android:layout\_width="wrap\_content"

          android:layout\_height="wrap\_content"

          android:text="Tags" />

    <!-- Check Box for cs -->

      <CheckBox

          android:id="@+id/CS\_check"

          android:layout\_width="wrap\_content"

          android:layout\_height="wrap\_content"

          android:layout\_centerHorizontal="true"

          android:layout\_centerVertical="true"

          android:text="Computer Science" />

    <!-- Check Box for ce -->

      <CheckBox

          android:id="@+id/CE\_check"

          android:layout\_width="wrap\_content"

          android:layout\_height="wrap\_content"

          android:layout\_above="@+id/CS\_check"

          android:layout\_alignLeft="@+id/CS\_check"

          android:layout\_alignStart="@+id/CS\_check"

          android:text="Computer Engineering" />

    <!-- Button connected to the database -->

      <Button

          android:id="@+id/button\_add\_data"

          android:layout\_width="wrap\_content"

          android:layout\_height="wrap\_content"

          android:layout\_alignParentBottom="true"

          android:layout\_alignParentLeft="true"

          android:layout\_alignParentStart="true"

          android:layout\_marginBottom="34dp"

          android:layout\_marginLeft="134dp"

          android:layout\_marginStart="134dp"

          android:text="Add Data" />

  </LinearLayout>

  <!-- Linear layout that items that get the user to the tag view -->

  <LinearLayout

      android:id="@+id/Layout2"

      android:layout\_width="match\_parent"

      android:layout\_height="match\_parent"

      android:layout\_alignLeft="@+id/Layout1"

      android:layout\_toEndOf="@+id/Layout1"

      android:layout\_toRightOf="@+id/Layout1"

      android:layout\_weight=".3"

      android:orientation="vertical">

<!-- Text that says “Search by Tags” -->

      <TextView

          android:id="@+id/textView3"

          android:layout\_width="wrap\_content"

          android:layout\_height="wrap\_content"

          android:layout\_alignLeft="@+id/button"

          android:layout\_alignStart="@+id/button"

          android:layout\_centerVertical="true"

          android:text="Search by Tags"

          app:layout\_constraintBottom\_toBottomOf="parent"

          app:layout\_constraintLeft\_toLeftOf="parent"

          app:layout\_constraintRight\_toRightOf="parent" />

    <!-- Button that takes to tags view -->

      <Button

          android:id="@+id/button"

          android:layout\_width="wrap\_content"

          android:layout\_height="wrap\_content"

          android:layout\_alignParentBottom="true"

          android:layout\_centerHorizontal="true"

          android:layout\_marginBottom="25dp"

          android:onClick="sendMessage"

          android:text="Button"

          tools:layout\_editor\_absoluteX="0dp"

          tools:layout\_editor\_absoluteY="0dp" />

  </LinearLayout>

</LinearLayout>

// Database Helper Class ~Brandi Werner (Tags)

/\*\*

\*    Name: DataBase Helper

\*    Creator: Brandi Werner

\*    Component: This is the class that sets up the database and allows for editing

\*            and updating of the database.

\*    Created: 11/05/2017        Last Edited:    11/12/2017

\*    Reason: We needed a local data to test out the functionality of the app.

\*    Uses: Allows for creation and editing of the database.

\*/

public class DatabaseHelper extends SQLiteOpenHelper {

  // When updating the structure on the database you will need to change the version

  private static final int DATABASE\_VERSION = 1;

  public static final String DATABASE\_NAME = "events.db";

  public static final String TABLE\_NAME = "events\_table";

  // The different columns of the database

  public static final String COL\_1 = "Event\_ID";

  public static final String COL\_2 = "Event\_Name";

  public static final String COL\_3 = "Event\_Description";

  public static final String COL\_4 = "CS\_Tag";

  public static final String COL\_5 = "CE\_Tag";

  /\*\*

   \* Constructor

   \*/

  public DatabaseHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {

      super(context, DATABASE\_NAME, factory, DATABASE\_VERSION);

  }

  /\*\*

   \* Creates the Database

   \* Runs the very first time we run this

   \*/

  @Override

  public void onCreate(SQLiteDatabase db) {

      /\*\* Takes the querey inside this method as an argument \*/

      String query = "CREATE TABLE" + TABLE\_NAME + "(" + COL\_1 + " INTEGER PRIMARY KEY AUTOINCREMENT, " + COL\_2 + " TEXT " + COL\_3 + " TEXT " + COL\_4 + " INTEGER " + COL\_5 + " INTEGER " + ");" ;       db.execSQL("CREATE TABLE" + TABLE\_NAME + "(Event\_ID INTEGER PRIMARY KEY AUTOINCREMENT, Event\_Name TEXT, Event\_Description TEXT, CS\_Tag INTEGER, CE\_Tag INTEGER)");

      db.execSQL(query);

  }

  /\*\*

   \* Upgrades the Database when the version is called. Pretty much says drop the old table and create a new one.

   \* @param db

   \* @param oldVersion

   \* @param newVersion

   \*/

  @Override

 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

      db.execSQL("DROP TABLE IF EXISTS" + TABLE\_NAME);

      onCreate(db);

  }

  /\*\*

   \* Insert data into the database

   \* @param Event\_Name is Column 2

   \* @param Event\_Desc is Column 3

   \* @return true or false depending on if the insert succeeded or not

   \*/

  public boolean insertData(String Event\_Name, String Event\_Desc, Boolean CS\_Tag, Boolean CE\_Tag) {

      SQLiteDatabase db = this.getReadableDatabase(); //DataBase instance

      // Allows to set different values for different columns and insert in one statement

      ContentValues contentValues = new ContentValues();

      contentValues.put(COL\_2, Event\_Name); // Arg1: Column name that want to input data, Arg2: Data to be inputed

      contentValues.put(COL\_3, Event\_Desc);

      contentValues.put(COL\_4, CS\_Tag);

      contentValues.put(COL\_5, CE\_Tag);

      long result = db.insert(TABLE\_NAME, null, contentValues); // Arg1: Table name, Arg2: null , Arg3: Content variables

      db.close();    // Closing database

    // Returns false if data was not inserted and true if it was successful in inserting

      if(result == -1)

          return false;

      else

          return true;

  }

  // Delete a product from the database

  public void deleteColumn(String Event\_Name){

      SQLiteDatabase db = getWritableDatabase();

    // SQL command to delete a column

      db.execSQL("Delete FROM " + TABLE\_NAME + " WHERE " + COL\_2 + "=\"" + Event\_Name + "\";");

  }

  // Print out the database as a string

  public String databaseToString(){

      String dbString = "";

      SQLiteDatabase db = getWritableDatabase();

      String query = "SELECT \* FROM " + TABLE\_NAME + " WHERE 1";    // Query to add data

      // Cursor points to a location in your results

      Cursor c = db.rawQuery(query, null);

      // Move to first row in your results

      c.moveToFirst();

    // While loop that adds all of the data to the string so it can print out

      while(!c.isAfterLast()){

          if(c.getString(c.getColumnIndex("Event\_Name"))!=null){

              dbString += c.getString(c.getColumnIndex("Event\_Name"));

              dbString += "\n";

          }

      }

      db.close();    // Closing database

      return dbString;

  }

}

//Tags View Class ~Brandi Werner (Tags)

/\*\*

\*    Name: Tags View

\*    Creator: Brandi Werner

\*    Component: This class is for the content that will show up on the tabs view.

\*    Created: 10/22/2017        Last Edited:    11/12/2017

\*    Reason: To have the user search the events through tags.

\*    Uses: View that has check boxes that help the user search for events.

\*/

public class tabView extends AppCompatActivity {

  @Override

  protected void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

      setContentView(R.layout.activity\_tab\_view);

  }

  /\*\*

   \*Description: Class for the check boxes that determines which boxes have been checked

   \*             and sets a variable for them if they have been checked.

   \*/

  public void onCheckboxClicked(View view){

      //Is the view now checked?

      boolean checked = ((CheckBox) view).isChecked();

      //Check which checkbox was clicked

      switch(view.getId()){

          case R.id.CheckCompSci:

              if (checked) {

                  // Set a variable to true that way I can find out which class needs to be displayed in the next view

              }else {

                  // Not much to do

              }

              break;

          case R.id.CheckCompEng:

              if(checked) {

                  //Set a variable to true that way I can find out which class needs to be displayed in the next view

              }else {

                  // Not much to do

              }

              break;

      }

  }

}

Activity\_Tag\_View ~Brandi Werner and Tsung-Han Hsieh (Tags)

<!--

\*    Name: Tag activity View

\*    Creator: Brandi Werner and Tsung-Han Hsieh

\*    Component: This is the tag class’s view. When a user is selecting tags to look

\*           for an event this is the screen shown.

\*    Created: 10/05/2017        Last Edited:    11/12/2017

\*    Reason: Created so the tag could show up on the screen

\*    Uses: Make the screen look good.

-->

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

  xmlns:app="http://schemas.android.com/apk/res-auto"

  xmlns:tools="http://schemas.android.com/tools"

  android:layout\_width="match\_parent"

  android:layout\_height="match\_parent"

  tools:context="com.brandi.tabs.tabView">

  <!-- Tells the user that they on the new screen -->

  <TextView

      android:layout\_width="wrap\_content"

      android:layout\_height="wrap\_content"

      android:text="We are here!"/>

  <!-- Check boxes -->

  <CheckBox

      android:id="@+id/CheckCompSci"

      android:layout\_width="wrap\_content"

      android:layout\_height="wrap\_content"

      android:text="Computer Science"

      android:onClick="onCheckboxClicked"

      tools:layout\_editor\_absoluteY="109dp"

      tools:layout\_editor\_absoluteX="85dp" />

  <CheckBox

      android:id="@+id/CheckCompEng"

      android:layout\_width="wrap\_content"

      android:layout\_height="wrap\_content"

      android:text="Computer Engineering"

      android:onClick="onCheckboxClicked"

      tools:layout\_editor\_absoluteY="62dp"

      tools:layout\_editor\_absoluteX="85dp" />

</android.support.constraint.ConstraintLayout>

//discoevents.java (Calendar)

/\*\*

\*    Author: Daniel Jimenez

\*    This is part of the android section of the code. Works with the android

\*         calendar from the xml file to read variables in for the specific date

\*        clicked and saves them to pull up specific database values. ALso

\*        creates pop ups based on calendar clicks

\*    Revised: 11-14-17

\*    Component will eventually pull from database, but is to be implemented in the

\*        future

\*/

public class DiscoEvents extends AppCompatActivity {  
  
   private TextView mTextMessage;  
  
   private BottomNavigationView.OnNavigationItemSelectedListener mOnNavigationItemSelectedListener  
           = new BottomNavigationView.OnNavigationItemSelectedListener() {

       @Override // code below sets bottom navigation tool names  
       public boolean onNavigationItemSelected(@NonNull MenuItem item) {  
           switch (item.getItemId()) {  
               case R.id.navigation\_home:  
                   mTextMessage.setText("Calendar");  
                   return true;  
               case R.id.navigation\_dashboard:  
                   mTextMessage.setText("Organizations");  
                   return true;  
               case R.id.navigation\_notifications:  
                   mTextMessage.setText(R.string.title\_notifications);  
                   return true;  
           }  
           return false;  
       }  
  
   };

   @Override  
   protected void onCreate(Bundle savedInstanceState) {  
       super.onCreate(savedInstanceState);  
       setContentView(R.layout.activity\_disco\_events);  
  
       //mTextMessage = (TextView) findViewById(R.id.message);  
       BottomNavigationView navigation = (BottomNavigationView) findViewById(R.id.navigation);  
       navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener);

       CalendarView calendarView=(CalendarView) findViewById(R.id.calendarView);  
       calendarView.setDate(System.currentTimeMillis(), false, true); // statement sets calendar //using the system time and date

// statement below runs each time a calendar day is clicked, and retrieves int values for year, //month, and day  
       calendarView.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {  
  
           @RequiresApi(api = Build.VERSION\_CODES.O)  
           @Override  
           public void onSelectedDayChange(CalendarView view, int year, int month,

                                           int dayOfMonth) {  
               Toast.makeText(getApplicationContext(), ""+dayOfMonth, Toast.LENGTH\_SHORT).show(); // test print message to screen for validation  
  
  
               month++; //idk why but because it reads first months as 0  
               startActivity(new Intent(DiscoEvents.this, Pop.class)); // calls class to create a popup   
  
  
           }  
       });  
  
   }  
  
  
}

//pop.java (Calendar)

/\*\*

\*    Author: Daniel Jimenez

\*    This code works with the discoevent java file to create a pop up

\*    Revised: 11-14-17

\*    Component will eventually pull from database, but is to be implemented in the

\*        future

\*/

public class Pop extends Activity {

  @Override

  protected void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

      setContentView(R.layout.popwindow); // links relevant xml file for popup

      DisplayMetrics dm = new DisplayMetrics(); // creates variable to receive screen display size

      getWindowManager().getDefaultDisplay().getMetrics(dm); // initializes to phone screen size

      int width = dm.widthPixels; //moves width into width int var

      int height = dm.heightPixels; // moves height into height int var

      getWindow().setLayout((int) (width\*0.8),(int) (height\*0.8) ); // creates window 80% size of screen

      //TextView evLog = (TextView) findViewById(R.id.eventLog);

      //evLog.setMovementMethod(new ScrollingMovementMethod());

  }

}

<!--themes.xml

Author: Daniel Jimenez

Revised: 11-14-17

XML file for java code, has specific style used for pop up

Linked to main code and pop.class

-->

<resources>

  <!-- Base application theme. -->

  <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

      <item name="colorPrimary">@color/colorPrimary</item>

      <item name="colorPrimaryDark">@color/colorPrimaryDark</item>

      <item name="colorAccent">@color/colorAccent</item>

  </style>

<!-- popclass theme to have transparent edges and allow for the popup to close on outside edges. -->

  <style name="AppTheme.CustomTheme">

      <item name="android:windowIsTranslucent">true</item>

      <item name="android:windowCloseOnTouchOutside">true</item>

  </style>

</resources>

//Main.xml (Calendar) ~Daniel Jimenez

<!--main.xml

Author: Daniel Jimenez

Revised:11-14-17

Main XML file used for main java code. Builds the general GUI using xml elements such as a calendar, and a container called a frame, as well as the bottom nav bar.

Linked to main code and pop.class -->

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

  xmlns:app="http://schemas.android.com/apk/res-auto"

  xmlns:tools="http://schemas.android.com/tools"

  android:id="@+id/container"

  android:layout\_width="match\_parent"

  android:layout\_height="match\_parent"

  android:orientation="vertical"

  tools:context="com.example.discoevents.DiscoEvents">

  <FrameLayout

      android:id="@+id/content"

      android:layout\_width="match\_parent"

      android:layout\_height="0dp"

      android:layout\_weight="1">

  </FrameLayout>

  <CalendarView

      android:id="@+id/calendarView"

      android:layout\_width="match\_parent"

      android:layout\_height="match\_parent"

      android:layout\_marginBottom="10dp"

      android:layout\_marginTop="10dp" />

  <android.support.design.widget.BottomNavigationView

      android:id="@+id/navigation"

      android:layout\_width="match\_parent"

      android:layout\_height="wrap\_content"

      android:layout\_gravity="bottom"

      android:background="?android:attr/windowBackground"

      app:menu="@menu/navigation" />

</LinearLayout>

Site\_Code

The HTML.erb files are not included in this test code

Events\_controller.rb

# Events controller

# Kaytlin Lafleur

# This holds the controller for the Events table within the site

# Created 11/3/17, Most recent update 11/13/17

# Controller for events table, with functionalities added

#each def is an action that happens to an event in the events table

class EventsController < ApplicationController

#index: this action shows the users list of events on the Org main page

def index

    @events = Event.all

end

 #show: this action is the view page of a single event

 def show

     @event = Event.find(params[:id])

 end

 #new: this action is the form to create a new event

 def new

     @event = Event.new

 end

 #create: this action is what saves the new event to the table

 def create

     @event = Event.new(event\_params)

     if(@event.save)

         redirect\_to @event

     else

         render 'new'

    end

 end

 #edit: this action is the form to edit an already made event

 def edit

     @event = Event.find(params[:id])

 end

 #update: this action actually saves the changed event in the table

 def update

     @event = Event.find(params[id])

     if(@event.update(event\_params))

         redirect\_to @event

     else

         render 'edit'

     end

 end

 #destroy: this action deletes an event from the table

 def destroy

     @event = Event.find(params[:id])

     @event.destroy

     redirect\_to index\_path

 End

#this action is a parameter for the other actions

 private def event\_params

     params.require(:event).permit(:title, :body, :time, :date, :location)

 end

end

Static\_pages\_controller.rb

# Static\_Pages Controller

# Kaytlin Lafleur

# This holds the controller for the Static\_pages

# Created 11/12/17, Most recent update 11/13/17

# Controller for the static\_pages

#each def is a static page, ie the home page

class StaticPagesController < ApplicationController

 #this is the definition for the home page

 def home

 end

 #this is the definition for the help page

 def help

 end

end

NOTE: on controllers

There are also a users\_controller.rb and application\_controller.rb file, but they are currently empty.

User.rb

# User model

# Kaytlin Lafleur

# This is the model for a singular user

# Created 11/12/17, Most recent update 11/13/17

# The user model contains validations to check that correct data is entered.

class User < ApplicationRecord

    before\_save {self.email = email.downcase}

    validates :organization, presence: true, length: {maximum: 50}

    VALID\_EMAIL\_REGEX = /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z\d\-]+)\*\.[a-z]+\z/i

    validates :email, presence: true, length: {maximum: 255},

              format: {with: VALID\_EMAIL\_REGEX },

              uniqueness: {case\_sensitive: false}

    validates :password, presence: true, length: {maximum: 225}

validates :about, presence: true, length: {maximum: 2000}

validates :organization, presence: true, length: {maximum: 50}

end

NOTE: on models

There is also event.rb and application\_record.rb, which are currently empty

Routes.rb

# Routes file

# Kaytlin Lafleur

# This holds the routes associated with certain actions and pages

# Created 11/3/17, Most recent update 11/13/17

# each get is a ‘route’

Rails.application.routes.draw do

 get  'home', to: 'static\_pages#home'     # routes to the static page home

 get 'static\_pages/help'            # routes to the static page help

 get 'events/new'                # routes to new event page

 get 'events/index', as: 'index'        # routes to events index

 get '/signup', to: 'users#new'            #routes to the signup page

 resources :events                #defines events resource

end

Schema.rb

# Schema File

# Kaytlin Lafleur

# This is the database schema for the tables, which include Users and Events

# Created 11/3/17, Most recent update 11/13/17

# the schema holds the creation of tables and what they contain.

ActiveRecord::Schema.define(version: 20171113233942) do

 #this table holds the events

create\_table "events", force: :cascade do |t|

   t.string "title"

   t.text "body"

   t.datetime "created\_at", null: false

   t.datetime "updated\_at", null: false

   t.string "time"

   t.string "date"

   t.string "location"

   T.string “organization”

 End

 #this table holds the users

 create\_table "users", force: :cascade do |t|

   t.string "organization"

   t.string "email"

   t.string “password”

   t.text “about”

   t.string “department”

   t.datetime "created\_at", null: false

   t.datetime "updated\_at", null: false

 end

end