UTA-Event-Mapper

Iteration One

Team 10

Mason Berry Jeffrey Carver Omar Kahn

CSE 3311-001

This document outlines the project and will be used for meeting written deliverable requirements for canvas turn-ins, developing slides for presenting to class, and overall development of the web tool.

Project plan

Constantly ongoing items
Website layout and overall aesthetic
Seeking feedback on currently implemented items

Implemented / In progress features
These core features are critical for further development of additional features and are integral for the user experience.

Account creation
Log in / Log out
Viewable and navigable map

Iteration 2 feature goals
These core features are critical for further development of additional features and are integral for the user experience.

Create event
View event
Event list

Planned features
If there is additional time before the next iteration, these next features will be worked on starting from highest priority. If not they will be pushed to further iterations.
These features are not as integral to the user experience but may significantly improve it.

Event RSVPs
Bookmarked/RSVPd events
Notifications
Friends list
Categorize events
Event filters

Competitors

- Google Maps

A global scale map project maintained by Google, Google Maps shows access routes and locations and provides a wide array of additional functionality related to maps.

How our project differs:
Smaller scale (restricted to UTA)
Serves a different function; will map events that are ongoing or upcoming and then remove them from the map

- https://events.uta.edu/

College hosted site that shows upcoming campus events. Events provide links to locations that show on a map

How our project differs:
Will not be limited to who can publish events.
Will show the events side by side on a page with the map
Events can be found through the map if you are looking for events based on location rather than time

Risks

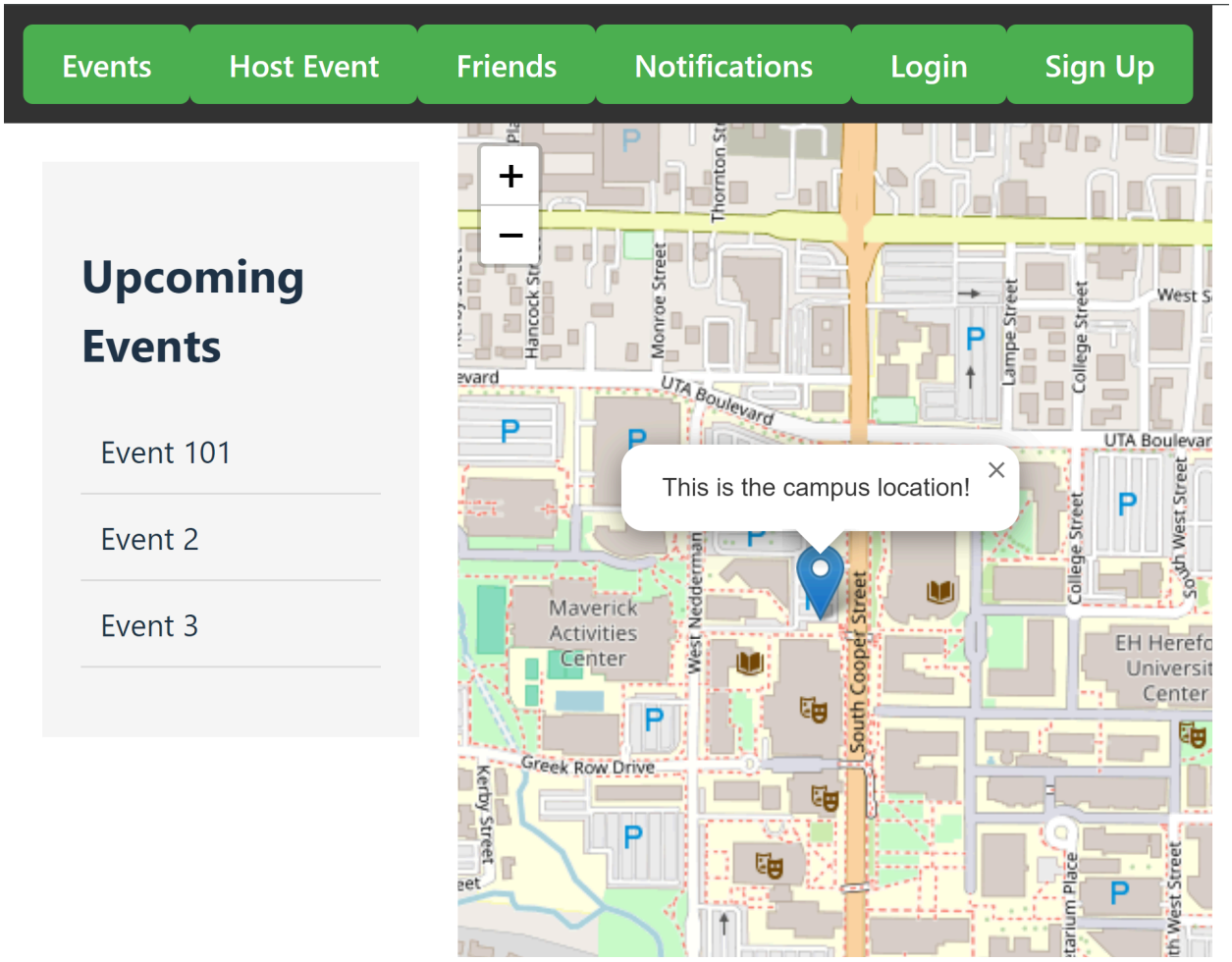
1.	Lack of experience	50%	30	15
2.	Version control mistakes	50%	20	10
3.	Absent Teammates	25%	20	5
4.	Map API incompatible w/ vision	30%	10	3
5.	Database incompatible w/ vision	15%	20	3
6.	Web hosting issues	50%	4	2
7.	Finding Users	25%	4	1

Mitigations

1. Identify deficiencies and spend time self learning using tutorials or AI instruction
2. Group member with most Github experience directs version control management
3. Ongoing endeavor to include and communicate with all members
4. Research, explore, compare map solutions early on, have backup plan
5. Explore web hosting options early on before implementation required
6. Find backup plan for database service

Specification and Design

General Layout (screenshot of work in progress preview)

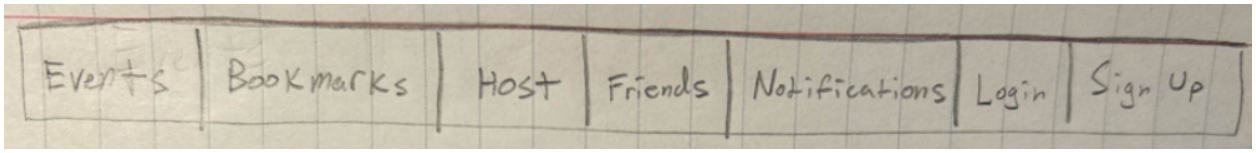


Left section changes through all available UIs. Right section always shows map

Navigation bar

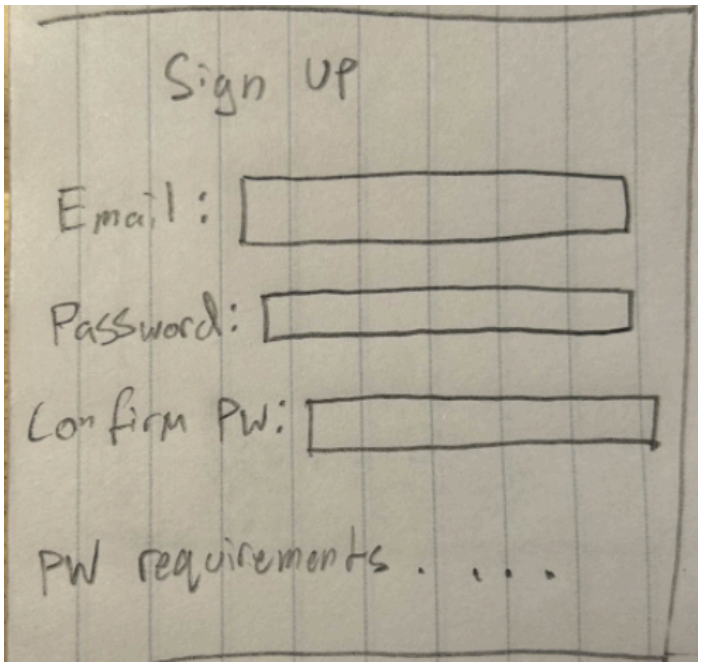
Button bar at the top of the web page displays navigation buttons. These buttons will change the left window to the appropriate UI

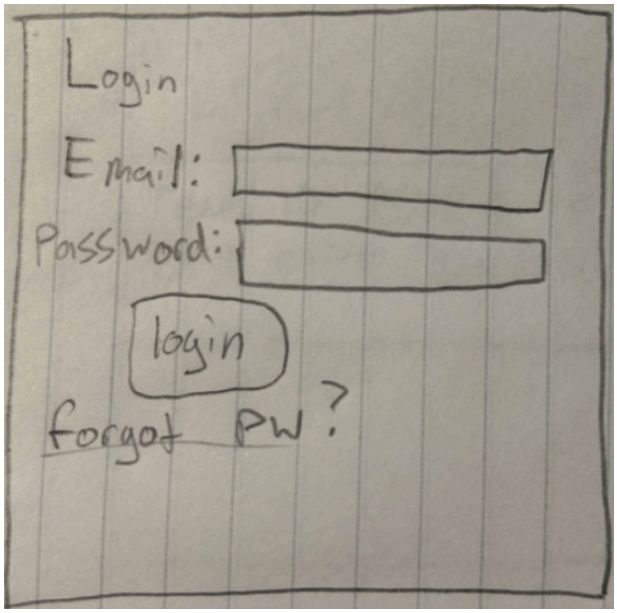
Event List button - Change window to the event list UI
Host Event button - Change window to the Host event UI
Friends button - Change window to the Friends List UI
Notifications button - Change window to the notifications UI
Log In / Log Out button - Relevant button is displayed depending on current logged in/logged out status. Displays the Log In UI
Sign Up button - Only shows if currently Logged out. Displays the Sign up UI



Sign Up / Log in / Log out:

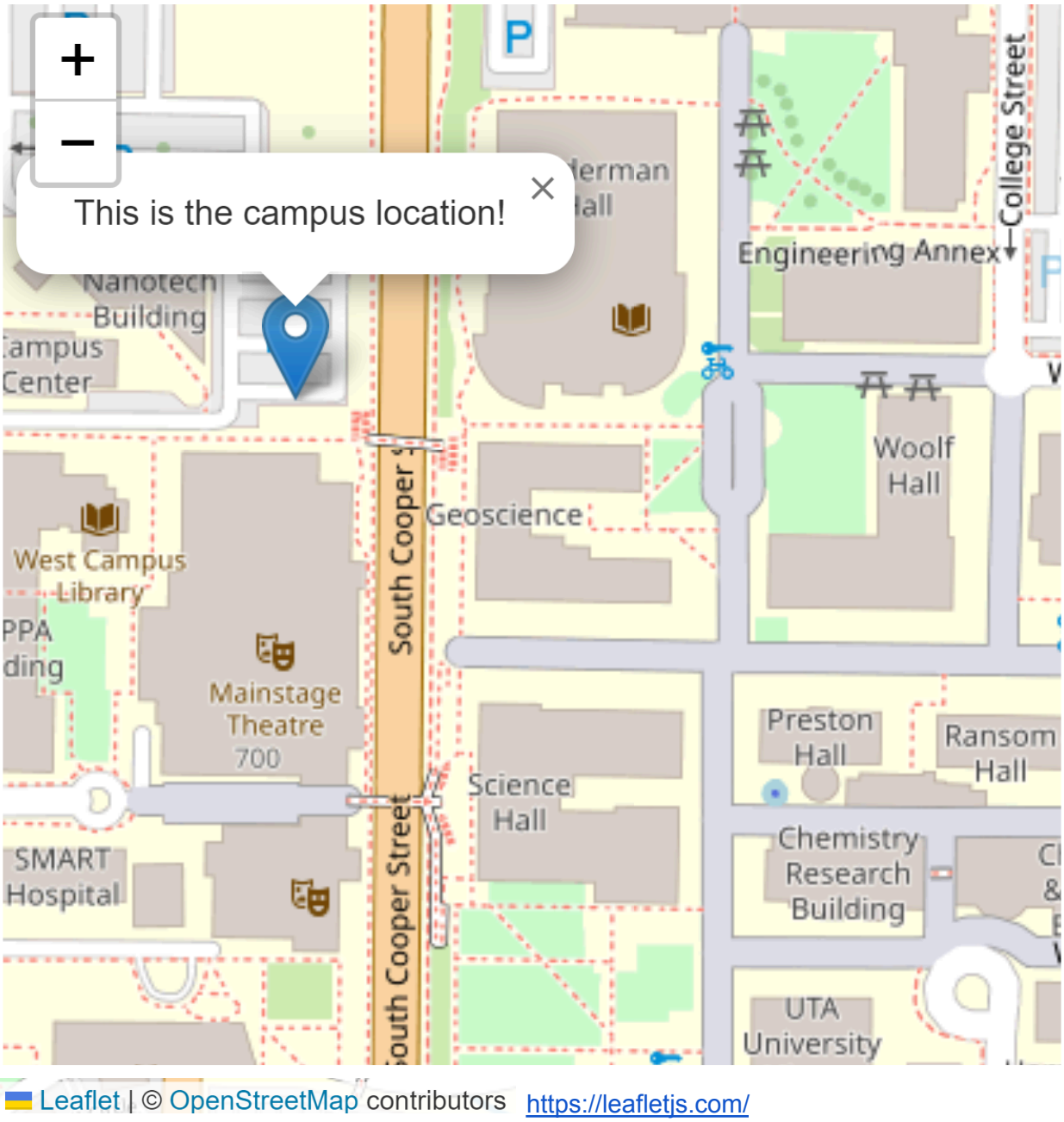
- User can press the buttons for these options in the nav bar and the left side of window will navigate to that UI
- User can enter Email and PW in sign up or Log In (Storage and security provided by firebase API library functions)
- Upon sign up, an automated email is sent to the provided email with a link that must be clicked to authenticate the User Email address
- Once signed in the Sign up button is hidden and the Log In button changes to Log out button
- Upon logging in, out or pressing sign up, a toast message is sent in a small window displaying the success of the action





Map

- Map includes a + and - button to zoom. Zoom can also be done with scroll wheel or other standard inputs
- User can click and drag to scroll the map
- Mousing over an event pin on the map will show event name and time in a small window near the map pin
- Clicking on a map pin will show the event details in the left section of web page and highlight the map Pin on the map.
- The small window with event name and time will stay open near the map pin. A small X will be available in this window in order to de-select the event



View Event

-displays event information in side view (Host name and email, time, additional location info, contact info, organization/social media links if available)

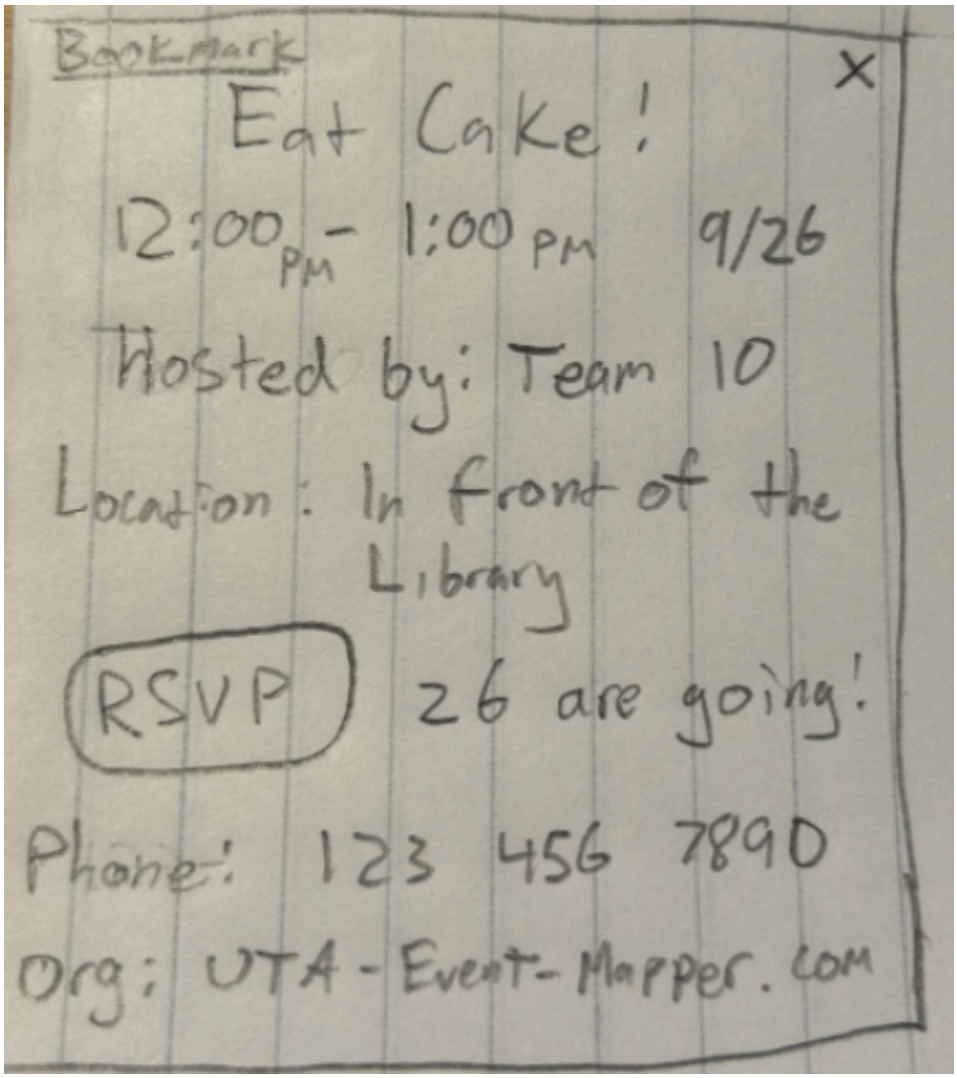
-If RSVP enabled on event, a button is available to RSVP and a number of how many RSVPs is shown

-Shows and friends you have that have RSVPd to the event

- A button to bookmark event will add it to the users bookmarked events list (if already bookmarked, this button changes to remove bookmark button)

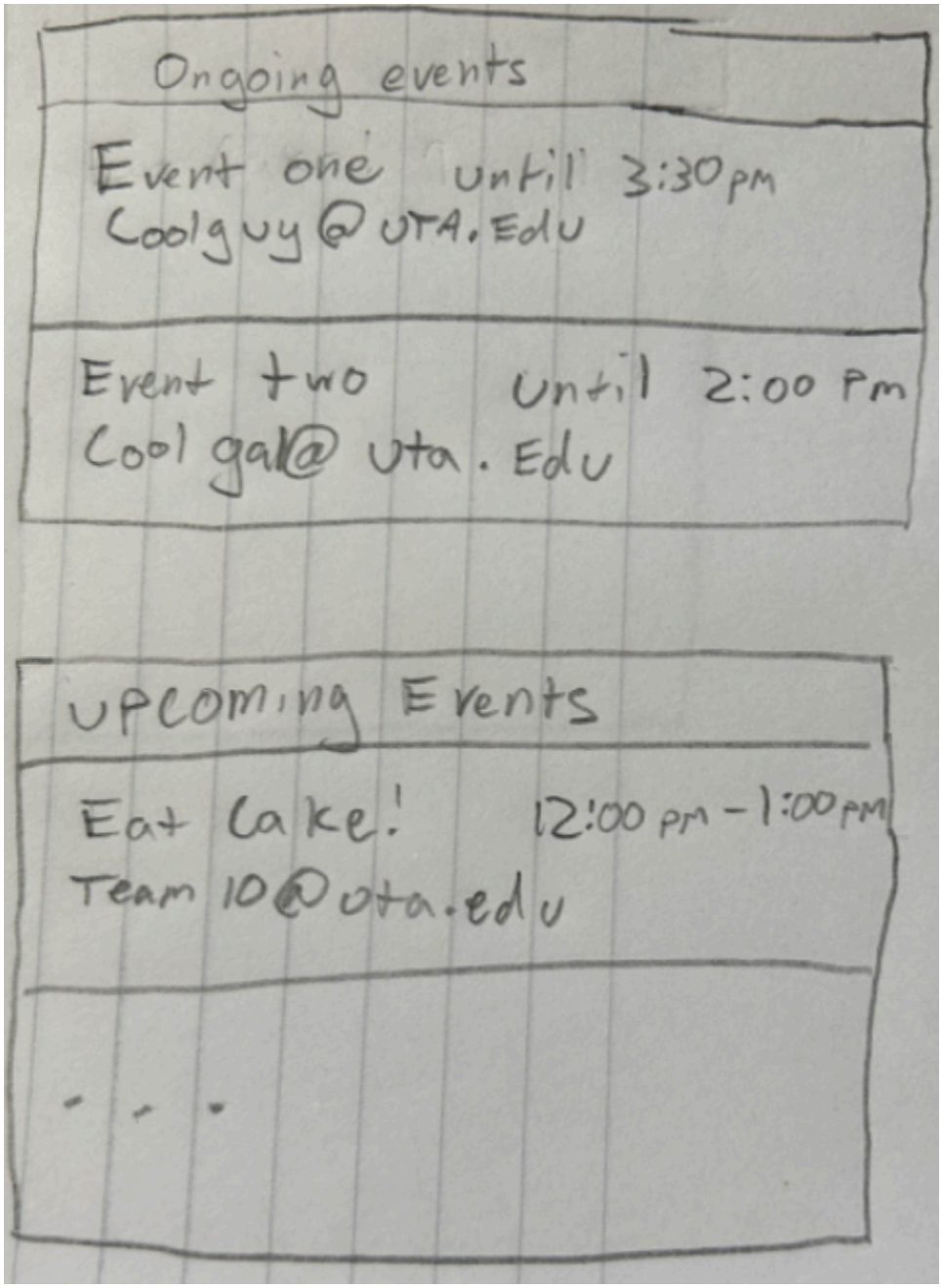
-an X can be clicked at top of view to return to event list

-If viewing an Event you hosted, an edit button and a cancel button are available



Event List

-Top sections shows ongoing events
-Bottom section shows upcoming events, soonest near top
-List is scrollable
-mousing over an event in the list will highlight map pin
-clicking an event will view the event



Host Event

-In host UI event name, location info, time and description are required to fill in to press create event button

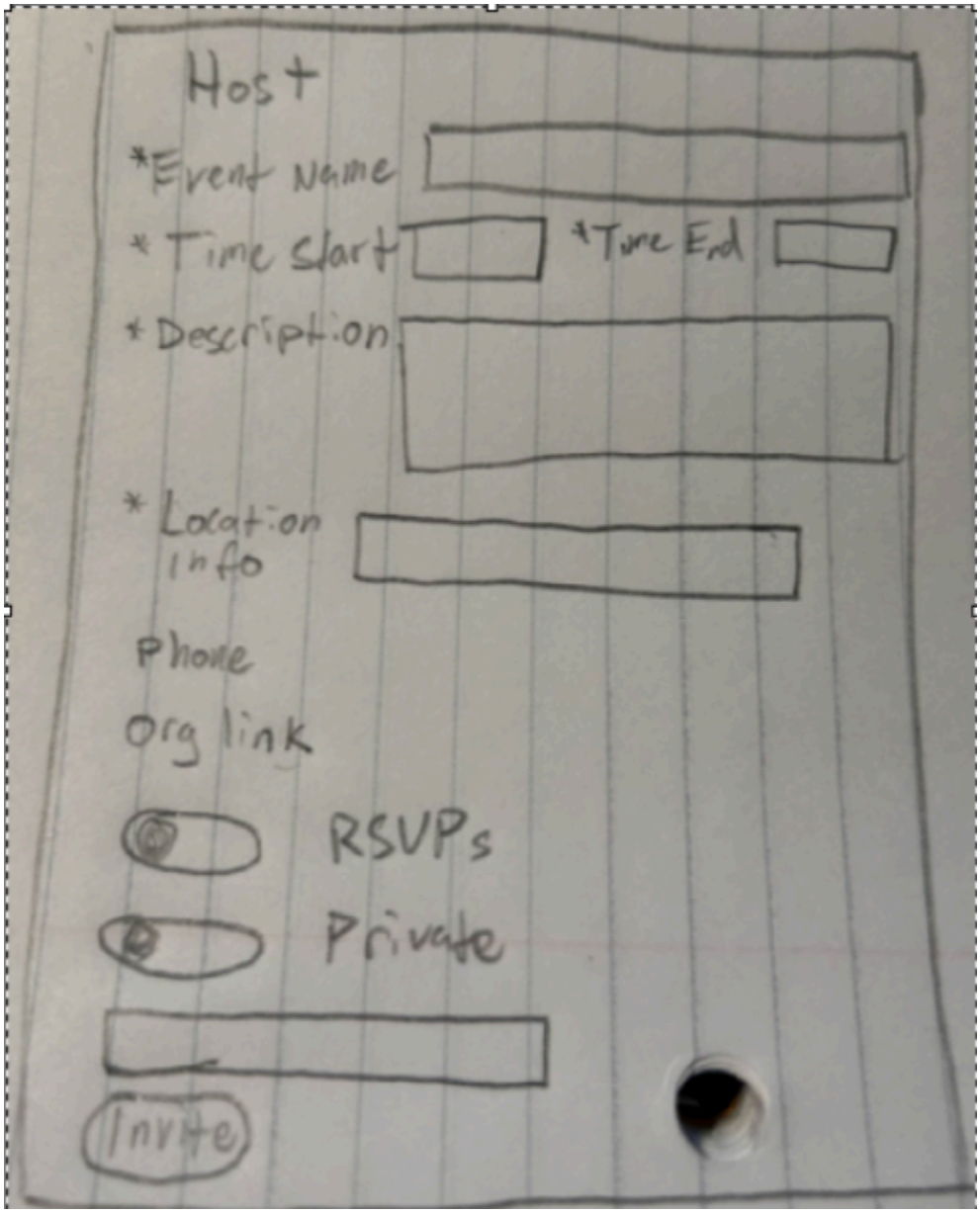
-optional fields to provide additional contact info or organization or social media links

-toggle button to make event private

-If private selected, a text box is available to send invites through Email which will provide a link to the website. (If Email owner not signed up yet they will be prompted in the Email to create an account to view the private event. Event invite will be visible in notifications after account creation)

-toggle button to make event RSVP cable

- If Private is toggled on then RSVP is forced on



Friends List

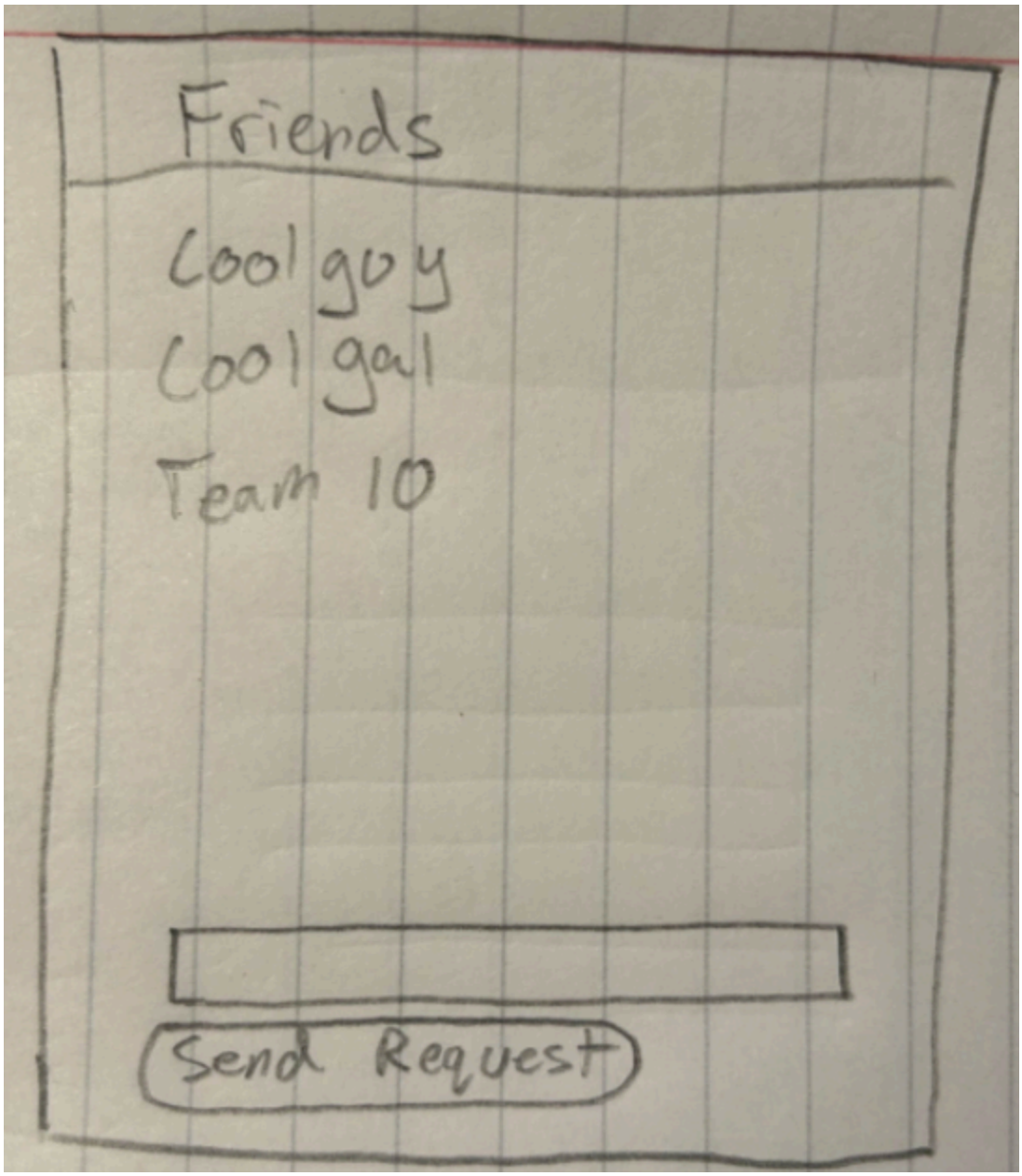
-Displays users by email address

-Contains a send request button with a text entry field that accepts any email address

-upon sending a request it will email the person a link to the web page

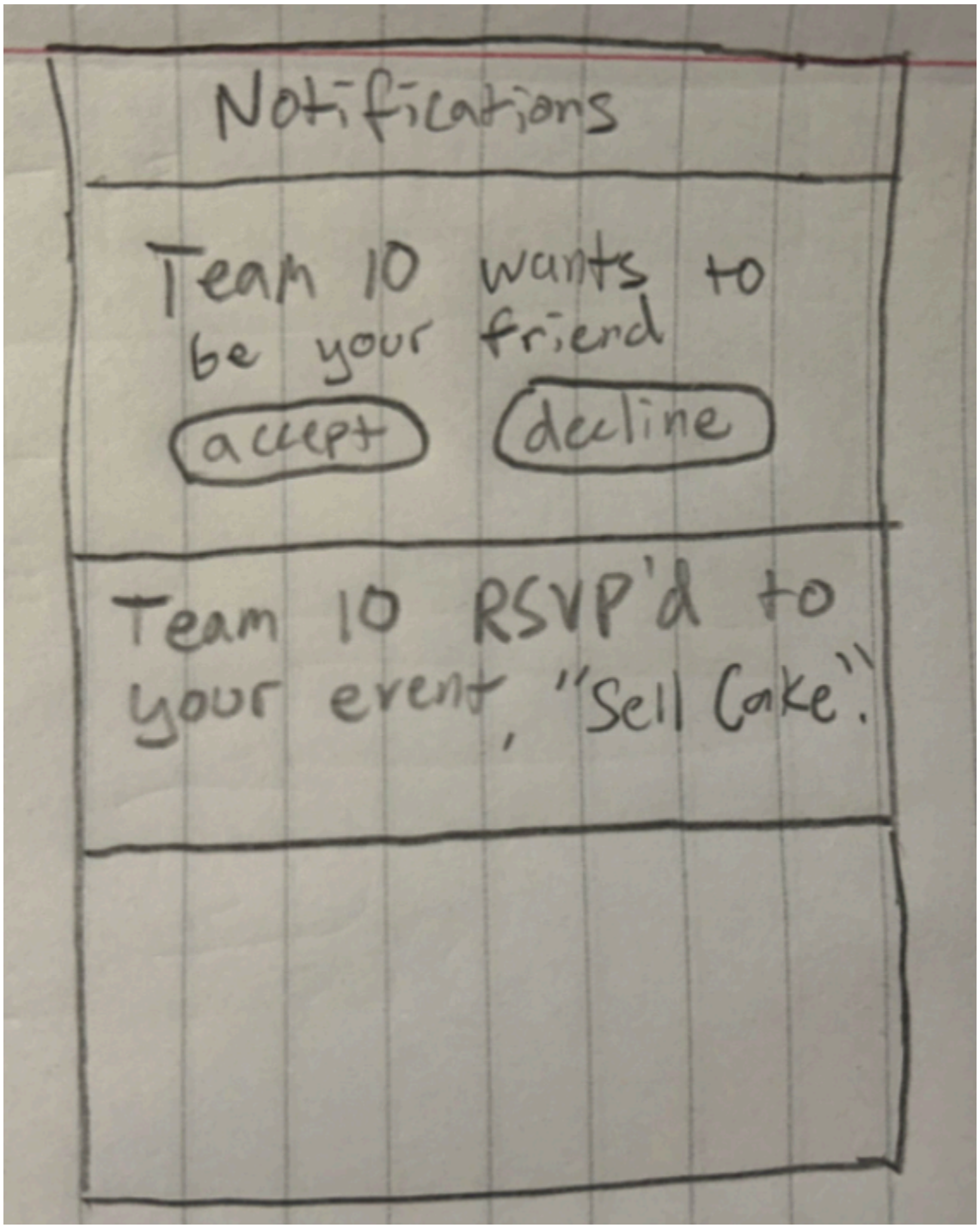
-if the user is already signed up with that account it will put a notification in their notifications box to accept or decline a friend request

-if the email address owner is not signed up the email message will prompt them to sign up so they can accept the friend request. Once signed up the friend request will be in notifications to accept



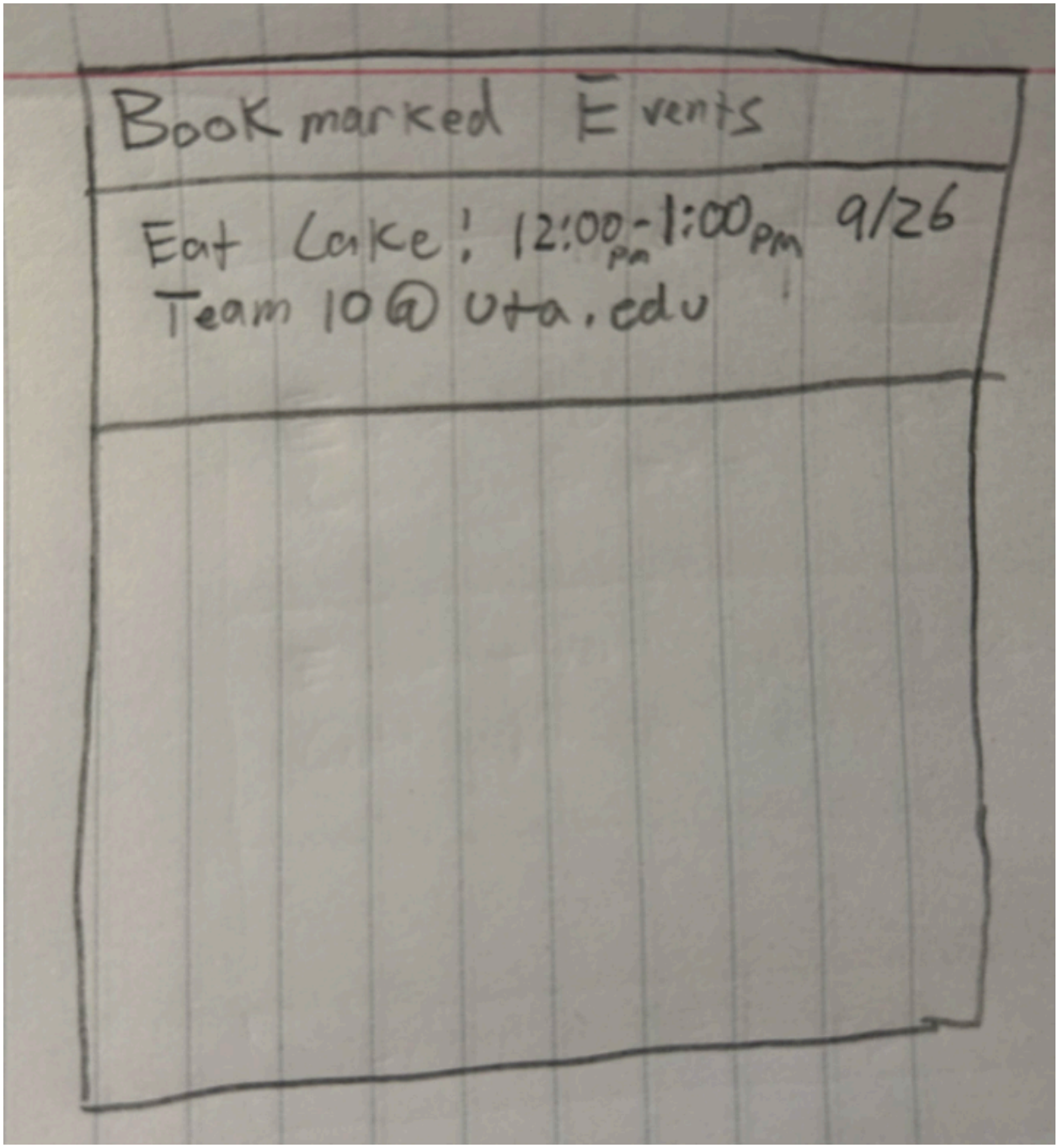
Notifications

- displays notifications for
    - Friend invite - has button to accept or decline
    - If a user RSVPs to your event
    - Invite to private event



Bookmarked Events

- emulates event list
- Displays all booked mark events in a scrolling list
- same functionality as event list
- after event time has passed bookmark is removed



Data structures

Note: hashed Passwords not stored in database but handled by firebase auth library which utilizes a separate secure data structure for handling user_ids and passwords

1. User

- **Attributes:**
  - uid: Unique identifier for each user (primary key).
  - email: User's email address (must be unique).
  - name: User's full name.
  - is_verified: Boolean to indicate if the user has verified their email.
  - created_at: Timestamp when the user registered.
  - updated_at: Timestamp when the user's data was last updated.
- **Relations:**
  - One-to-Many with Events (hosted_events).
  - Many-to-Many with Events (RSVPed_events).
  - Many-to-Many with Friends.
  - Many-to-Many with Notifications.
  - Many-to-Many with BookmarkedEvents.

2. Event

- **Attributes:**
  - event_id: Unique identifier for each event (primary key).
  - host_id: ID of the user who is hosting the event (foreign key to User).
  - title: Name of the event.
  - description: Description of the event.
  - location: String with detailed location description or address.
  - latitude: Latitude of the event location (for map).
  - longitude: Longitude of the event location (for map).
  - start_time: Timestamp for when the event starts.
  - end_time: Timestamp for when the event ends.
  - contact_info: Contact information for the event.
  - organization_links: Social media or organization links, if any.
  - is_private: Boolean indicating if the event is private.
  - is_rsvp_enabled: Boolean indicating if RSVP is enabled.
  - created_at: Timestamp when the event was created.
  - updated_at: Timestamp when the event was last updated.
- **Relations:**
  - Many-to-One with User (host).
  - Many-to-Many with User (RSVPed_users).
  - Many-to-Many with User (bookmarked_by_users).

3. Friend

- **Attributes:**
  - friendship_id: Unique identifier for each friendship (primary key).
  - user_id: ID of the user (foreign key to User).
  - friend_user_id: ID of the friend user (foreign key to User).
  - status: Status of the friendship (e.g., pending, accepted, declined).
  - created_at: Timestamp when the friend request was sent.
  - updated_at: Timestamp when the friendship status was last updated.
- **Relations:**
  - Many-to-Many with User (friends).

4. Notification

- **Attributes:**
  - notification_id: Unique identifier for each notification (primary key).
  - user_id: ID of the user receiving the notification (foreign key to User).
  - type: Type of notification (FriendRequest, RSVP, EventInvite).
  - message: Notification message content.
  - is_read: Boolean indicating if the notification has been read.
  - created_at: Timestamp when the notification was created.
- **Relations:**
  - Many-to-One with User (recipient).

5. RSVP

- **Attributes:**
  - rsvp_id: Unique identifier for each RSVP (primary key).
  - event_id: ID of the event being RSVPed to (foreign key to Event).
  - user_id: ID of the user RSVPing (foreign key to User).
  - status: Status of RSVP (e.g., Going, Not Going, Maybe).
  - created_at: Timestamp when the RSVP was made.
- **Relations:**
  - Many-to-Many with User.
  - Many-to-Many with Event.

6. BookmarkedEvent

- **Attributes:**
  - bookmark_id: Unique identifier for each bookmarked event (primary key).
  - event_id: ID of the event being bookmarked (foreign key to Event).
  - user_id: ID of the user who bookmarked the event (foreign key to User).
  - created_at: Timestamp when the event was bookmarked.
- **Relations:**
  - Many-to-Many with User.
  - Many-to-Many with Event.

Relationships Overview

1. **User** - A user can host multiple events, bookmark events, RSVP to events, and have friends.
2. **Event** - An event can be hosted by one user, have multiple RSVPs, and be bookmarked by multiple users.
3. **Friend** - A friendship is a many-to-many relationship between users.
4. **Notification** - Notifications are tied to users and represent various actions such as invites or RSVPs.
5. **RSVP** - Links users and events with their response status.
6. **BookmarkedEvent** - Links users and events with their bookmarked status.

Tests

In project current state testing has limited availability. This section will be populated as more features become implemented and test cases are written.

1. Sign up:
    1. Press sign up button on nav bar
    2. Enter a valid email address into email text field
    3. Enter a valid PW into text field (currently requires 6 char minimum)
    4. Press sign up

- A toast window should pop up saying you have been sent a verification email
- Check for the verification email and click the link in the email
- another toast window should pop up saying your account was verified

2. Log in:
    1. Press log in button on navigation bar
    2. Enter valid Email and password for verified account
    3. Press login button

- Toast window popup with log successful message
- login button should change to logout button

Sample code

Below are screenshots from the code for relevant or interesting functions.
So far many files contain simple code that is just serving as basic web page layout or navigation between components.

```
// Helper function to append a symbol if the email belongs to a school domain
const appendSymbolForSchoolEmail = (email, displayName) => {
  const schoolDomains = ["@mavs.uta.edu", "@uta.edu"];
  const isSchoolEmail = schoolDomains.some(domain => email.endsWith(domain));

  if (isSchoolEmail) {
    return displayName + " 🐬";  // Append special symbol to denote school email
  }

  return displayName;  // Return unchanged if not a school email
};
```

```
// Sign up new user
const signUp = async (email, password, displayName = "User") => {
  try {
    const userCredential = await createUserWithEmailAndPassword(auth, email, password);
    const user = userCredential.user;

    // Check if the user has a school domain email
    const updatedDisplayName = appendSymbolForSchoolEmail(email, displayName);

    // Store user data in Firestore
    await setDoc(doc(firestore, "users", user.uid), {
      username: updatedDisplayName,
      email: user.email,
      createdAt: new Date().toISOString(),
    });

    // Send verification email
    await sendEmailVerification(user);
    console.log("User registered and verification email sent:", user);

    alert("Verification email sent! Please check your inbox to verify your email.");
  } catch (error) {
    console.error("Error registering user:", error.message);
  }
};
```

```jsx
const Map = () => {
  // Set the initial map view (latitude, longitude, zoom level)
  const [position, setPosition] = useState<[number, number]>([32.732, -97.115]);

  return (
    <div className="map">
      <MapContainer
        center={position}
        zoom={16}
        scrollWheelZoom={true}
        style={{ height: "100vh", width: "100%" }}
      >
        {/* TileLayer loads the map tiles from OpenStreetMap */}
        <TileLayer
          url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
          attribution='&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
        />

        {/* Marker shows the position on the map */}
        <Marker position={position}>
          <Popup>
            This is the campus location!
          </Popup>
        </Marker>
      </MapContainer>
    </div>
  );
};
```

All screenshots of our own project code written with StackBlitz IDE
https://stackblitz.com/

StackBlitz Documentation

Discover how to use StackBlitz, an online development environment for frontend, Node.js and the JavaScript ecosystem.

https://developer.stackblitz.com/

Customers and Users

The goal for our web app is to make a simple free web tool for **students, faculty and other UTA visitors** to find and host events quickly and easily with a visual map tool.

This is a free service so we do not want to refer to any users as customers.

Plan for increasing users:
Invite classmates to use our website
Share our website in classmate socials such as teams groups, discords, groupmes etc.
Ask event hosts in person or via social media outreach if they would like to try hosting their events on our website and offer to demo for them

Feedback

Plans for receiving additional feedback

Document feedback here (for future iterations) or in a group feedback forum or document with other team 10 group members that is shared in the Repo.

During development:
Ask 3311 students and other students for feedback on our website prompting specific questions such as…
Would you use this web tool?
Would you use “X” feature?
What features, if any, would make you more likely to use our web tool?

After launch:
Add a “Provide Feedback” button to the website with a survey, questionnaire or simple comment type form to fill out.
Track user data and website use through database and firebase analytics tool to see which features are being used or gauge general website traffic.