Progress Update Document

Proposal Changes:

Several changes have been made to the original plan. The project name has been changed from RSVP Collection System (RSC) to AllEvently, following a recommendation from our professor. We have also decided that email will be the only planned method for sending notifications, as text messaging involves additional costs. Some features, such as creating signup sheets, uploading images to an event gallery, and adding a waiting list, have been deferred to the "if there's time" section. We have also decided not to include social media sharing for events.

There have been changes to the technology stack as well. We switched from Django to Express (with Node.js) for the backend, and we changed our development environment from PyCharm to Webstorm, another JetBrains tool. We also replaced Google Firebase with PostgreSQL, alongside Node.js, Express, and Vercel for hosting. Lastly, we are now sourcing icons from FontAwesomeIcon. These changes reflect our evolving approach to meeting project requirements and improving overall functionality.

Some key issues remain unresolved, which may impact the project. We still need to determine a clear method for sending email notifications, as this could significantly affect the system's usability. The approach to allowing users to customize invitations is still undecided, which may also impact usability. Another pending decision is how to obtain and store images for invitations. Whether or not users will be allowed to upload their images remains undecided, which could affect the level of customization available. The functionality for saving partially completed invitations has not yet been developed.

Progress:

User Interface:

Items that are completed:

The AllEvently project has significantly progressed, with several key features successfully implemented. The login page is fully functional, featuring error messages to guide users when incorrect information is entered. Similarly, the signup page is complete, with relevant error messages to ensure a smooth user registration process. We have successfully integrated the Google API, enabling users to log in or sign up with their Google accounts. Additionally, the login page includes a reset password link that directs users to the Password Reset page, which is also fully implemented with error messages and a helpful dialog box.

The Reset Password button now navigates users to the Update Password page, which includes error messages and a dialog box for guidance. The Account Settings page has also been developed, with dialog boxes confirming actions like saving changes or sending email links. On the event management side, the Public Events page has been created to display Public Event Cards, while the Private Events page supports public and private event cards for Hosts and Guests. Event Cards on the Private Events page correctly navigate the Host and Guests to relevant sections, such as the Event Chat for both roles and the Guest List for the Host.

Items that are partially completed:

There are, however, some partially completed features that require further development. The Account Settings page needs to include navigation to the Password Reset page and the Event Creation Page. The Public Events page still needs links to the public invitation, integration of a printable PDF version of public invitations, and navigation to the Event Creation Page.

Items needed to implement:

Several features and functionalities remain to be completed. The implementation of the Google Maps API for displaying event locations is still pending. The Event Creation Page, including differentiation for recurring events, must be developed along with the Invitation Page/RSVP form and a printable PDF page. Currently, Public Events, Private Events, Chat pages, and Guest Lists use placeholder data, and data integration with the database is required to implement these features entirely. The Events Page needs functionality to filter events by name and whether the user is hosting or attending. Additionally, updates to the Event Cards are needed to display the number of people attending the event for Hosts and Guests, the number of people they are bringing, and their attendance status. The Public Events page also requires a filtering mechanism by month, week, and day, and functionality to generate QR codes for events is yet to be implemented.

Server:

Items that are complete:

The express server framework has been configured in the project’s backend code. Upon running a command, the express server can correctly run using a port on the local machine.

Login functionality has been implemented using express, node, and vue routing. The server will send a query to the database using information pulled from the Vue application, and it will handle the response from the database to either signal that the user entered the correct information or that the information was incorrect.

Items that are partially complete:

One of the partially incomplete items is the signup function. The signup function will pull information from the Vue application and send it to the database over the server. The database will then insert and store the information and create the new user.

Another partially incomplete item is the load events page function. This function will simply load the events page from the server.

Items needed to implement:

The rest of the inter-page mapping still needs to be implemented. There needs to be completed functionality with event-creation and event chatting. Testing must also be done with the server-side code to ensure everything loads and processes correctly.

Database:

Items that are complete:

A People table has been created. It contains the email address, first name, and last name. The email address is the primary key.

An Accounts table has been created. It contains the email address and encrypted password for each account. The email address is the primary key. The email address is a foreign key from the People Table.

A Sessions table has been create. It contains the session ID and the email address. The session ID is the primary key. The email address is a foreign key from the Account table and must be unique and not null.

A Reset Credentials table has been created. It contains the email address and the temporary password. The email address is the primary key. The email address is a foreign key from the Account Table.

An Images table has been created. It contains the image ID, the path to the image, and whether the image is a template image. The image ID is the primary key.

A User Images table has been created. It contains the image ID and the email address. The image ID is the primary key. The image ID is a foreign key from the Images table. The email address is a foreign key from the People table.

An Events table has been created. It contains the event ID, event host email address, host's name, event name, event location, event start date, event end date, event time zone, invitation layout, background image, font background color, font color, font, whether the event is public, whether the event reoccurs, reoccurrence frequency, end reoccurrence date, whether to request

child number, whether to limit additional guests, max additional guests, and whether to enable host notifications. The event ID is the primary key. The event host's email address is a foreign key from the accounts Table

A Chat Messages table has been created. It contains message ID, event ID, sender's email, content, sent date, and whether it is an accepted notification. The message ID is the primary key. The event ID is a foreign key from the message from the Events table. The sender's email is a foreign key from the People table.

A Guests table has been created. It contains the guest's email, event ID, whether an invitation has been sent, whether the guest accepted, child count, and additional guest count. The guest's email and the event ID are a compound primary key. The guest's email is a foreign key from the People table. The event ID is a foreign key from the Events table.

An Authenticate User function has been created. This function will verify account credentials and return a session ID on success.

A Create Account function has been created. This function will create accounts and return a session ID if it is given valid credentials.

A Create Reset Credentials function has been created. This function will generate reset credentials and return a temporary password on success.

An Update Password function has been created. If given the corresponding email and temporary password, this function will update an account's password.

A Get Attending Events function has been created. This function returns all events attended by a user.

A Get Hosted Events function has been created. This function returns all events hosted by a user.

A Get Public Events function has been created. This function returns all public events.

Items needed to implement:

A Change Name function to change an account's first and last names.

A Create Event function to create an event.

A Get Event function to return an event based on the event ID.

A Store Image function to store user images.

A Get User Images function to return all images a User can access.

A Get Image function to return an image path based on the image ID.

A Get Event Chat function to return all messages in a chat based on the event ID.

A Send Message to add a chat message to the database.

A Get Guest List function to return all guests for a given event.

An Invite function to update a guest to be invited.

An Is Guest function to determine if a user is a guest of an event.

An RSVP function to store responses to invitations.