

Luis Fornes, Mary Jiang, Ha Nguyen, Zhixue (Mary) Wang, Garrett Wolfe  
CS316  
Prof. Jun Yang  
December 10, 2019

## **Loop Group - Final Report**

### **Project Overview:**

#### **What actually is Loop?**

The basis of Loop is a quick and easy way for individuals to organize events with groups they are a part of. Our web app provides support for individuals to log in, join groups, and create, RSVP to, and edit events.

More specifically, Loop is a social app that targets users who belong to a small group of people (AKA a “Loop”!) who would like to plan both long-term and short-term events within the group. People can register with a username and password and then join groups they belong to. The application visualizes a clean, aesthetic feed of all events within a group. These events are posted by a member of a group, and other members can RSVP to indicate that they will attend. By showing how many members are attending and creating a simple feed to keep track of events only, this app hopes to use the bandwagon effect to encourage members to attend more events together.

#### **Why Loop?**

Simply put, people have ideas for events that never come to fruition.

College students who belong to a group, such as a circle of close friends, club, or a group for a course project, often use messenger applications to make a group chat. These include apps like Facebook Messenger, GroupMe, and WhatsApp. In the chat, members talk to one another virtually, share videos or images, plan events with members of the chat, and more. However, since messenger applications are not specifically meant for event planning, messages from a person asking other group chat members to RSVP to an event often get lost among other messages or do not get any replies in the necessary timeframe. Friends end up canceling plans or going with only a few people, members fail to show up to group project meetings, and clubs lose out on event attendance. Even if the members of the group want to plan an event together, it often falls apart because of the nature of group chats. Sometimes, friends end up having to create multiple spin-off groups especially for coordinating events, such as “Fall Break”, “Saturday Plans”, etc. which take up extra space on messenger feeds.

Our app aims to fix this. We want to help streamline event planning, so that friends will hang out more, clubs will raise event attendance, and more. What’s novel about our app is that it is a stand-alone event planning forum, removing the clutter and distractions associated with many messaging apps. It also focuses on friends and other people users are close to, rather than

displaying events publicly, to emphasize the importance of intimacy in planning events with people the user cares about.

### **Survey of Related Work**

The two most similar applications would likely be Facebook Events and Google Calendar.

Facebook Events is an app within the larger Facebook platform. At the top of the webpage, it shows upcoming events that the user has marked as “Going”. In that same bar, you can create a new event. Below are events the user might like, probably based on location and some other factors. Then there are events popular with friends, networking events, and movies. Each event box has an image, name, date, location, and number of people attending. A user can mark themselves as Going, Interested, or Not Going. If you click on an event, there is an additional page with details, host information, and posts. This is on both Facebook’s web and mobile platforms.

Google Calendar allows users to create calendar events and invite other people. It shows a calendar format, and when a user creates a new event, you are prompted for a name, location, date and time, description, and guests. Users add guests by inputting their emails. Events can be printed, published, and they can also notify attendees. This is available as a Gmail app on the web and as a standalone app on mobile.

Other similar, less event-planning focused apps include most messaging apps like WhatsApp, GroupMe, or WeChat. These tend to have options within messenger chats to create an event with a name and time and date that people can mark to say they are going or not. They also tend to allow for notifications, either on mobile or as a chatbot.

### **System Design:**

#### **Platform and Framework**

We chose to use the Play framework built on Java and worked off of the Play template of db-beers provided to implement a simple database-driven web application. We have deployed it in our development environment and now can successfully run the application and access it from a browser with full functionality.

As a result of using the Play framework, we leveraged Play’s MVC model in our system, keeping the design of the three parts of the system separated and encapsulated. This allows us to easily make changes to the front-end, the connector, and the back-end database connection of the project. Our design is comprised of three primary parts: the LoopDB.java class serves as the model of our system, the Application.java class serves as the controller, and various Scala/HTML pages, each corresponding to a page on our site, serves as the view.

We have decided to keep all model-related methods inside the LoopDB.java class for simplification at this time, since the functionalities that we currently offer are limited as we are

generally working in the alpha stage. In the future, as our functionality grows, we can refactor the methods in the model to separate classes that more closely align with the way in which the View is set up, and some sort of Manager class that manages requests from the Controller and delegates the actual tasks to these smaller, separate classes. For the same reason, we have also kept all controller-related methods inside the Application.java class at this stage. In the future, a redesign of the controller will be similar to that of the Model. With the MVC model, any of this refactoring will keep other parts of the project completely unaffected as we have minimized the dependencies between the three parts. As for the View, we separated each page into scala HTML templates. This allows us to dynamically render the appropriate pages accurately depending on user interaction as part of the web application.

**Assumptions:**

- Each group has a unique group ID (gid).
- The group ID (gid) of a group is a unique code shared among group members. When new members want to join the group, they first register to the website and enter on the gid of the group they are trying to join.
- Each user has a unique user ID (uid).
- A user can join multiple groups or not join any groups at all.
- A group can have multiple user members.
- If a user leaves a group, their membership record for that group is deleted.
- If all users leave a group, the group is deleted.
- A user can only create events for groups they are in.
- A user can only RSVP to events for the groups they are in.
- Only the host/creator of the event can edit it.
- Events are single-day activities.
- In the interface, only the user who posted an event can see a button to edit the event's name, description, time, and cost and a button to delete the event.
- We assume all Loop users in the same group are in the same time-zone

**Design changes:**

- Instead of viewing the events for only one group at a time, the new default view the user will see displays all events from all groups.
- Event IDs were changed from randomized strings to incremental int values for quickness of generation and historical event creation tracking.

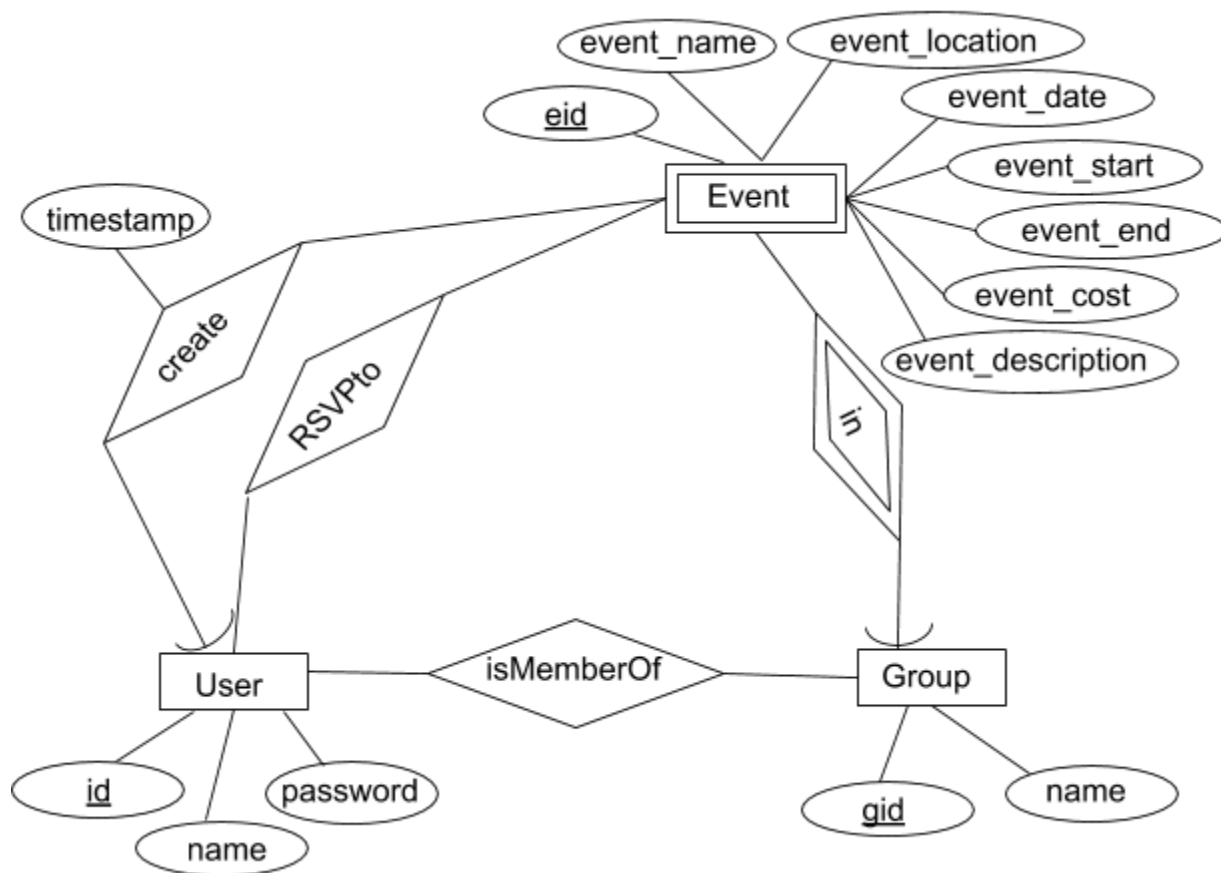
**Manufactured data**

[https://docs.google.com/spreadsheets/d/1NqGB\\_zGoRGGusiXhpNco6bMnPe0A1qw7KptbQGeNpuE/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1NqGB_zGoRGGusiXhpNco6bMnPe0A1qw7KptbQGeNpuE/edit?usp=sharing)

As we explored ways to find and utilize test data for our database, we determined that the best plan of action was to create data ourselves. We hand-crafted data for registered users, groups, events, and interactions between all of these tables and groups in order to provide a sufficient set from which to test Loop. As a result of creating this data by hand, we do not have an

extremely large data set. However, since we took the time and effort to create each data point by hand, the data is of higher quality, specific, and targeted to ensure we test important edge cases, triggers/functions, and create a dataset that will effectively and efficiently push our implementation of Loop to its limits. Although we debated scraping data or pulling from another source, we determined that the time we put into creating data would pay dividends in the future as we try to test our product as it increases in complexity.

## E/R Diagram



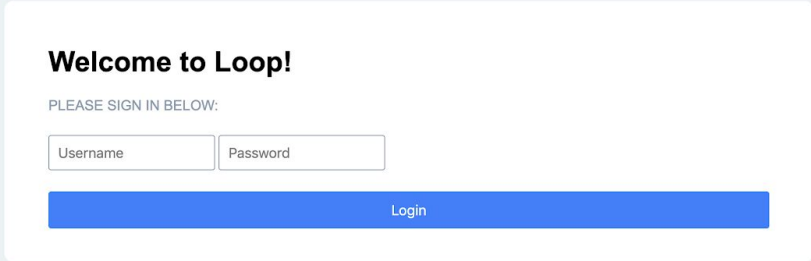
## Database Schema

A list of database tables with keys declared is below. Primary keys are underlined and foreign keys are italicized.

- User(id, name, password)
  - This table holds the user information for login authentication. It is used to check that given a username, the password provided matches in the database.
- Group(gid, group\_name)

- This table holds the group information for group joining. It is used to let users join new groups by providing the unique group code, or gid.
- isMemberOf(id, gid)
  - This table holds the membership information of users in groups. It is used to show users the groups they are a part of and the events associated with those groups. In this way, users are restricted to only viewing events of groups they are a part of and are unaware of the other existing groups. They can also filter down by group.
- AllEvents(eid, gid, id, event\_name, event\_location, event\_date, event\_start\_time, event\_end\_time, event\_description, event\_cost, timestamp)
  - This table holds all event information for all groups, including event name, location, date, start and end time, description, cost, and timestamp of creation. This information is the main information shown in the dashboard.
- RSVPto(id, eid, gid)
  - This table holds all the reservation information. Whether a user is RSPed to an event will be displayed as a green for yes or gray for no button on an event panel. The creator of an event is automatically RSVPed to it.

### What does this look like?



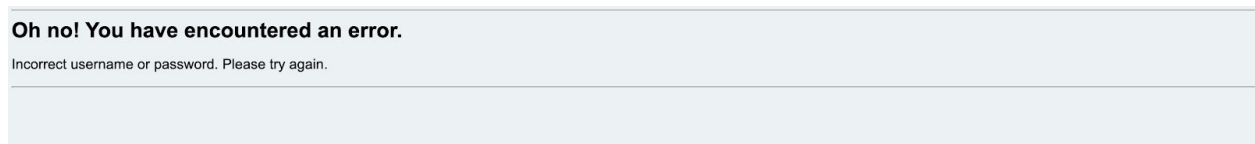
**Welcome to Loop!**

PLEASE SIGN IN BELOW:

Username Password

Login

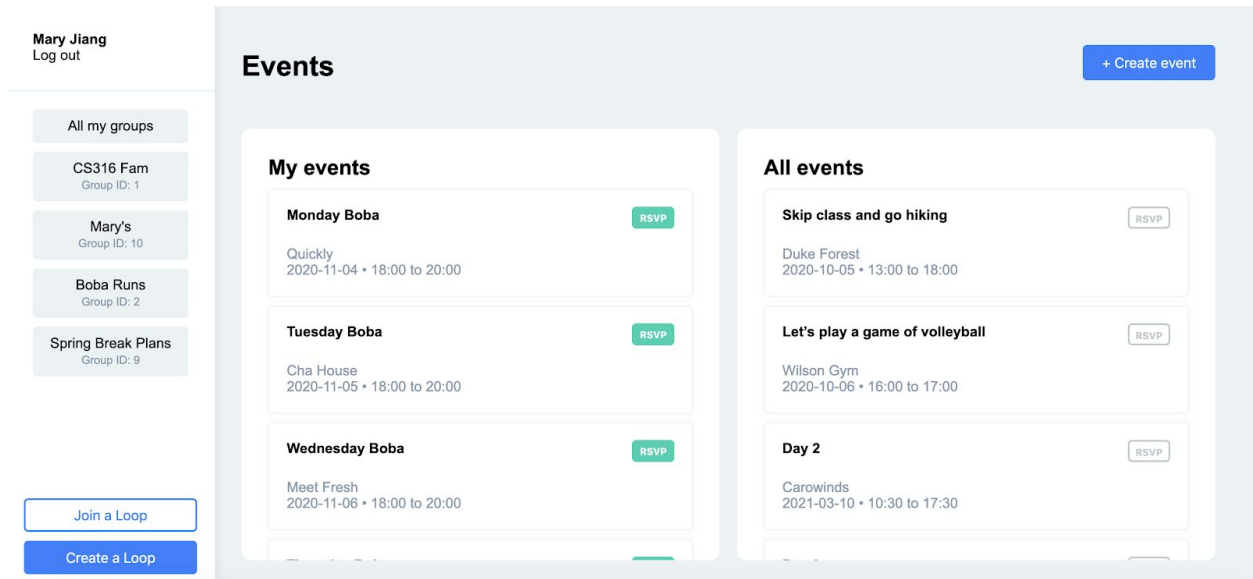
Users are welcomed to Loop with a clean interface asking them to login with their username id and password.



**Oh no! You have encountered an error.**

Incorrect username or password. Please try again.

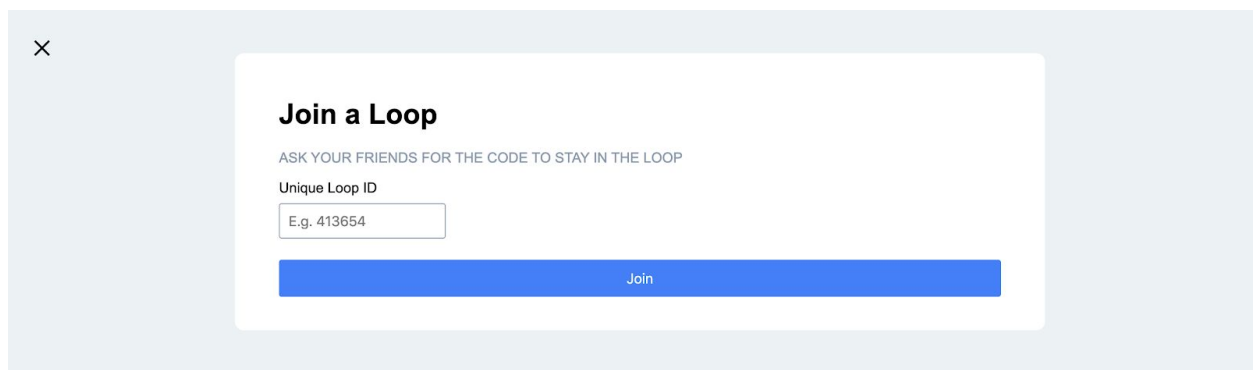
If they enter the wrong username and password combination, they are redirected to a friendly error page with the option to return to the previous page. This type of lenient error handling is common across the app.

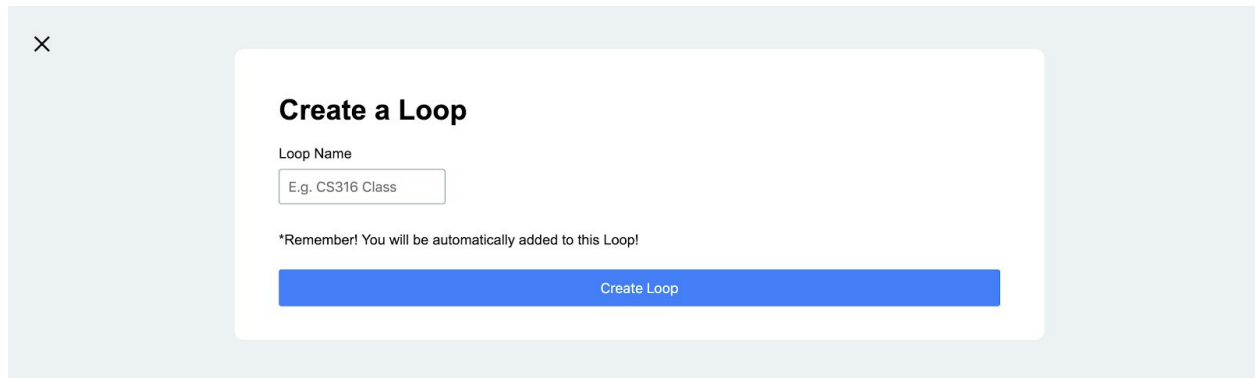


To the user, when logged in to the Loop web app, they are presented with all the events available to them in a clean, user-friendly interface. These events are made available on a user's feed through their membership in groups. The user also has the option to filter the events on their feed by group by selecting a group from the sidebar on the left.

The main events dashboard is split into two main areas, events the user has RSVP'd to already (left), and events the user has access to via group membership but hasn't RSVP'd to (right). From there, a user can also create, edit, and delete events, or create a new group. Creating/editing events involve a redirect to a form where the user fills out event details such as title/name, location, date, start and end times, description, and cost.

At the bottom of the main user page, the user sees two options to create and join a loop. Once they click these options, they will be directed to the pages below.





A screenshot of a 'Create a Loop' form. The form is white and centered on a light blue background. It has a close button (X) in the top left corner. The title 'Create a Loop' is in bold. Below it is a label 'Loop Name' and a text input field containing 'E.g. CS316 Class'. A note below the input field says '\*Remember! You will be automatically added to this Loop!'. At the bottom is a blue button labeled 'Create Loop'.

×

### Create a Loop

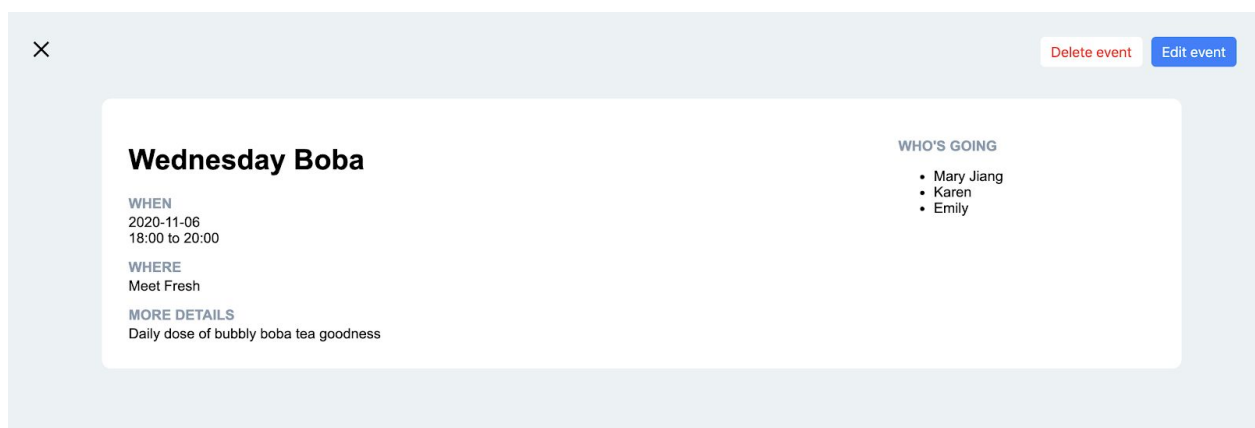
Loop Name

\*Remember! You will be automatically added to this Loop!

Create Loop

The Join a Loop page asks for a unique Loop ID. There, the user can enter the ID given to them by the Loop creator or member of which they are trying to join. The Create a Loop page asks the user to name the group.

The creator of the group will also be added as a member of the group.



A screenshot of an event details page for 'Wednesday Boba'. The page is white and centered on a light blue background. It has a close button (X) in the top left corner. In the top right corner, there are two buttons: 'Delete event' (red) and 'Edit event' (blue). The event title 'Wednesday Boba' is in bold. Below the title, there are three sections: 'WHEN' with the date '2020-11-06' and time '18:00 to 20:00', 'WHERE' with the location 'Meet Fresh', and 'MORE DETAILS' with the description 'Daily dose of bubbly boba tea goodness'. To the right of these sections, under the heading 'WHO'S GOING', there is a list of names: Mary Jiang, Karen, and Emily.

×

Delete event Edit event

### Wednesday Boba

**WHEN**  
2020-11-06  
18:00 to 20:00

**WHERE**  
Meet Fresh

**MORE DETAILS**  
Daily dose of bubbly boba tea goodness

**WHO'S GOING**

- Mary Jiang
- Karen
- Emily

When the user clicks an event from their main user page, the following screen will pop up and show all the details related to the event as well as the group members who have RSVP'd to it. Users will have the option to delete or edit an event only if they are the creator of the event.

## Create event

Event Name

What are you doing?

Location

Where is it?

Event Date

mm/dd/yyyy

Start Time

--:--

End Time

--:--

Cost

\$0.00

With which lucky group?

CS316 Fam

☐ 1

Mary's

☐ 10

Boba Runs

☐ 2

Spring Break Plans

☐ 9

Description

\*Remember! You will be automatically RSVPed to this event, so make sure you can go!

Create event

In the create event page, the user will see a form that prompts inputs necessary to create an event. These include the event name, location, date, start and end time, cost, and description, as well as a list of options from which the users can choose the group they want the event to be in. These are limited to the groups they are in.

When the user creates an event, they will be automatically also RSVPed to the event, and the event will also automatically show up in every member of the designated group's feed.



**Edit your event**

Event Name  
Wednesday Boba

Location  
Meet Fresh

Event Date  
11/06/2020

Start Time  
06:00 PM

End Time  
08:00 PM

Cost  
15.00

Cost  
15.00

Description  
Daily dose of bubbly bc

Submit edit

On the edit event page, the user is presented with multiple fields with the existing event details. The details are greyed so the user can easily see the existing values and make necessary changes to their liking. Once they are finished making changes as they see fit, the user presses the large blue submit edit button found at the bottom of the page and the event submission is revised.

This page is only accessible to the original creator of the event.

### **Algorithms & Approaches:**

Following our assumptions above, on the SQL side, we have built in a number of triggers as part of our algorithms and approaches. These triggers include the following:

- Checks to make sure that only registered users are able to join a group. If a user attempts to join a group without having been registered, the request to join will be unsuccessful. This approach allows us to customize our user experience to be user-oriented by showing registered users the loops that they are apart of as well as the events in these groups of which they can RSVP for.
- Checks to make sure that a user must be a member of a group in order to create events in the group. This is a merely logical as the point of Loop is to allow close friend groups to easily plan events together.

- Checks to make sure that a person who creates an event is automatically RSVP-ed to it. This makes sense because the logical assumption is that, if a person is trying to create an event in a close friend group, this person will most likely attend this event (they are free to unRSVP to the event if they would like).
- Checks to make sure that when an event is deleted, everybody who have previously RSVP'd to the event is unRSVP'ed to it.
- Checks to make sure that when the last membership record for a group is deleted, then the group is removed. This is to keep our database from being overloaded with obsolete groups while both keeping the user experience smooth and easy (each person only has to click leave group) while simultaneously preventing accidentally deletes of groups.

One other SQL triggers were removed (the one where a user must be a member of a group to RSVP to an event in a group), because over the course of the project, we realized that some restrictions are better implemented in either the SQL-side, back-end, or front-end. Specifically, if the front-end only shows a user the groups that they are in, they will never see the events for groups they are not in, and thus cannot RSVP to a group they are not in as that button will not show up on the page.

## **Evaluation**

### **Open Issues**

Currently, for the purpose of getting a prototype up and running as quickly as possible, we have simplified the gid generation process to automatically incrementing numbers. This, however, creates a security risk of group gid's being easily mixed up, or people joining groups that they are not apart of by randomly guessing gid's. In comparison with other systems, Google documents, for example, uses links as ways to share a document, as a link is essentially a long string that is hard to imitate. In the future, we will move towards more complex gid that are harder to generate and correctly guess. These gid should also be randomly generated rather than following some type of sequence. As a result, we have continued to keep the type for the GID in our schema as strings with 256 possible characters — this gives us the flexibility to move to generating randomized group ID's in a real production setting. Nothing other than the method for GID generation will have to change.

### **Comparison**

In creating Loop we sought to examine the benefits and drawbacks of today's most commonly used event planning spaces. These mediums include Facebook Events, GroupMe, Google Calendar, and more.

Facebook Events, albeit being a function within one of the world's most widely used social media applications, has several drawbacks. For starts, Facebook is being used less and less around the world as Facebook's Monthly Active Users statistic is [dropping](#). As a result, less people are using Facebook, less people are seeing, creating, and RSVPing events, and there is thus a decreased yield in creation->attendance. Additionally, Facebook Events normally serves as an official forum for larger or official events. Events such as birthday parties, food festivals,

concerts and such are common to find on Facebook Events. In addition, large sponsored events that charge an entrance fee or promote an event are often common, leading to events people do not want to receive invites to. However, smaller, more intimate events with smaller numbers of attendees are very uncommon. This is because Facebook's UI does not cater to smaller events between a single group of friends as it is meant for large event planning.

Another platform we looked at was Google Calendar. In a similar manner as Facebook, Google has an even more formal platform for creating and diffusing events. It is connected to your Google Account and invites are sent via email. The UI is clean and simple, but as events are sent/added to your email calendar, Google Calendar feels very formal, professional and corporate, like you are sending a calendar invite to your boss to go over a pitch deck or meet for a coffee chat. This is not the ideal level of intimacy people want to have with friends and family. This results in people not responding to events, being unable to discuss them, and thus an unsuccessful event planning space.

Finally, another subset of event planning mediums are those embedded in common group messengers such as GroupMe, WeChat, and WhatsApp. These platforms are extremely intimate as members join groups with family, friends, student groups, coworkers, etc. That being said, communication frequency in these group chats is extremely high, with several hundreds and sometimes thousands of messages being sent a day. As a result, proposed events and their important details can be lost and buried in the chat as people begin to post more and more messages. This leads to events that get low numbers of RSVPs and low attendance.

Loop's strengths lie in its base functionality. Loop is a stand-alone, group-based, event organizing application. It is not a side function of a larger application. As a result, its core functionality and resources are dedicated to event planning and organizing instead of other things that Facebook, Google, and group chats need to cater to. Loop's UI presents its users with an easy to read and understandable list of proposed events for all groups in addition to events sorted by group, allowing you to choose the level of specificity and priority you have in planning/RSVPing at any given time. Loop can have anywhere from high to low levels of intimacy since in Loop users can join various groups. Users can create and join groups of limitless possibility. From family Loops, to friend and coworker Loops, users have the ability to create a space to focus solely on organizing events with people they care about. No getting bogged down by photos, videos, or messages that hinder the ability to plan and RSVP. This way you can focus on events only with the people you care about, with zero distractions. This level of intimacy and attention to design and focus peg Loop as what we believe is a unique and unparalleled application for planning events with the people that you care about.

### **Directions for future Loop Work**

One of the main motivations for creating Loop was convenience. The lower the barrier to actually creating and RSVPing to events, the more likely friends and groups are to actually meet and go to events together. A more convenient approach then would be to also develop a mobile version of our app, since users are much more likely to access such social apps on their

phones. This would require setting up a publicly accessible database that is shared between both the web app and the phone app as well as creating a mobile-friendly user interface. Such an interface would likely have only one column and filters near the top rather than the sides, or the filters could be hidden.

Another shortfall in typical event-planning applications is the method by which users can typically decide whether to attend an event. As we all know, many times during the planning phase, the time and location of an event are often up in the air and rapidly changing based on the people who are interested in attending. In typical event planning implementations, this decision-making process is binary: Yes I will Attend, or No I will not attend. In future iterations of Loop we could look into creating a vote-based decision for an event location/time or another methodology of polling and collectively deciding important event details.

Additionally, in our Event table schema, we have given event creators the ability to give an event a value for event\_cost. In future implementations we could integrate Loop with payment platforms such as Zelle, Venmo, Paypal, Cash App, etc to increase simplicity in splitting the cost of an event with attendees whether the event is a dinner, going to the movies, paintball, or buying a friend a gift.'

In a world where we are all connected by social media constantly, it is often highly desired for users to be able to share their experiences with friends, family, and more. In the future, we may look to integrate Loop with social media services such as Facebook, Instagram, Snapchat, and more to increase the visibility of events currently being planned on Loop. This could include posting a link on an Instagram story that would redirect you to join a group on Loop, or even something as simple as enabling a user to publicize an event on their various social media outlets. This is similar to Instagram users who share songs from Spotify or Apple Music on their Instagram Stories, prompting their friends' sense of curiosity to go and listen.

On another note, we believe it would improve event success and attendance if we were to implement desktop notifications for event plans. Things such as reminding/notifying a user of an event they RSVPed to, reminding a user that an event is currently "hot" (being actively RSVP'd to), or reminding a user to RSVP to an event in general could further improve the yield of RSVP'ed and attended events. This can be extended by building integration with a user's phone's native calendar app.

Smaller improvements include building time-zone functionality into Loop so friends can make plans across time zones, integrating the event location with users navigation app of choice, sharing event details with those not on Loop via native messaging applications, adding a photo to link to the event, adding "All-Day" option to event start and end times, and even adding a "Event History/Stats" Page which would give each user a breakdown of how many events they have (or have not) attended, how many friends they have "looped" with, and other relevant and interesting statistics.