# Team LiveConnect

**Jacob Stone**

**Jon Hoeve**

**Jhoseph Ruiz**

**Corey Rice**

**Supervised by Franklin Wong**

# Table of Contents

## Overview

LiveConnect is a cloud-based iOS application that will allow users to organize social gatherings with ease. The real-time mobile application will allow users to chat in a group message, pick venues, share details, create polls, and vote on locations and times. The application will integrate with a web interface for establishments to manage booking information. A database and data processing will be used for customer analysis. An equivalent Android version will be developed for future market expansion.

## Languages

LiveConnect will be written in Swift programming language. Cocoapods will also be used to help automate certain features. Javascript and React will be implemented for use in web based aspects of the application. HTML and CSS will be used to create structure and layout of the website. Kotlin will be used to create the Android version of the application if time allows.

## Libraries, SDKs, and APIs

### Core Location

We are going to use Core Location to determine where a device is so we can have establishments nearby pop up.

### Cocoapods

Cocoa pods will be used to help manage certain tasks allowing us to have the app to work on multiple devices without having too many dependencies.

### Firebase

We will be using Firebase to collect and store data for the use of providing a more catered experience.

### MessageKit

MessageKit will be used with the messaging between people whether it's in a group or person to person.

### iOS chat SDK

iOS chat is an open source SDK that will help us to automate the messaging process as well as collect data from chats.

### MapKit

We will be using MapKit to help point out places of interest nearby the user. This will be helpful when it comes to suggesting establishments or meeting places.

## Code Repository Organization

### Branching

Our github repository is separated into 6 different branches, Main, Development, Documentation, Feature, Enhancement, and Bug. While all separate branches they all work together under one github repository to optimize our work towards the final prototype.

### Main

Main is the branch that contains the most up-to-date working prototype of the app. Any push/pull requests to this branch must be reviewed by more than one group member to make sure that the code going in or coming out is stable.

### Development

This section is reserved for bug free code that is still being worked on by group members. This section as well must be reviewed by at least one member when there is a pull request.

### Documentation

This branch is specifically for any documentation that relates to the project.

### Feature

This branch is for any code that adds big functionality to the application. Some examples would be adding another screen, implementing a web based API, and large logical or layout changes.

### Enhancement

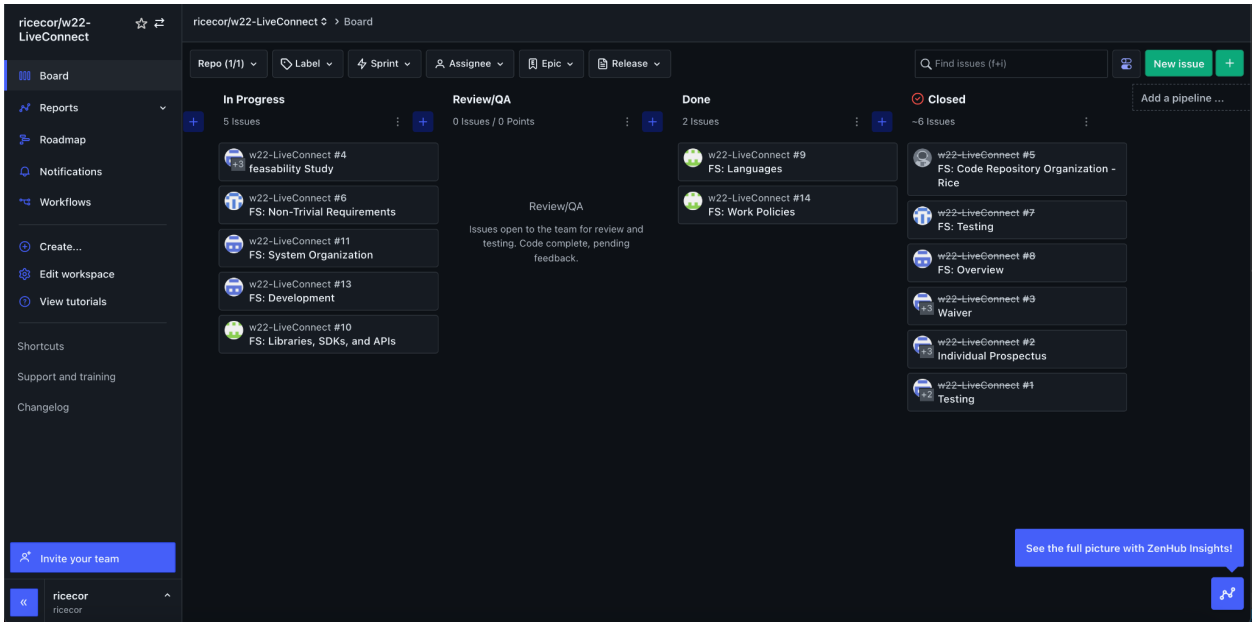
This branch is for improving on code that does not have any known bugs or issues.

### Bug

This branch is for code that needs to be worked on because has plenty of bugs and issues.

### Zenhub

To organize our work flow as a group we are using Zenhub. This plugin allows the whole group to visualize what is being worked on and what still needs to be done. As you can see in this screenshot this document is being worked on right now.




### Icebox

The Icebox is a back burner per say where the work that is not being done in this sprint is placed.

### Backlog

Backlog is where work that needs to be done in this sprint but has yet to be worked on goes.

### In Progress

This section contains all the issues that are being worked on in Zenhub.

### Review Q/A

This is for work that has been completed, moved from “In Progress”, and is in need of review from the group.

### Done

The Done section contains the work that has gone through the review process.

### Closed

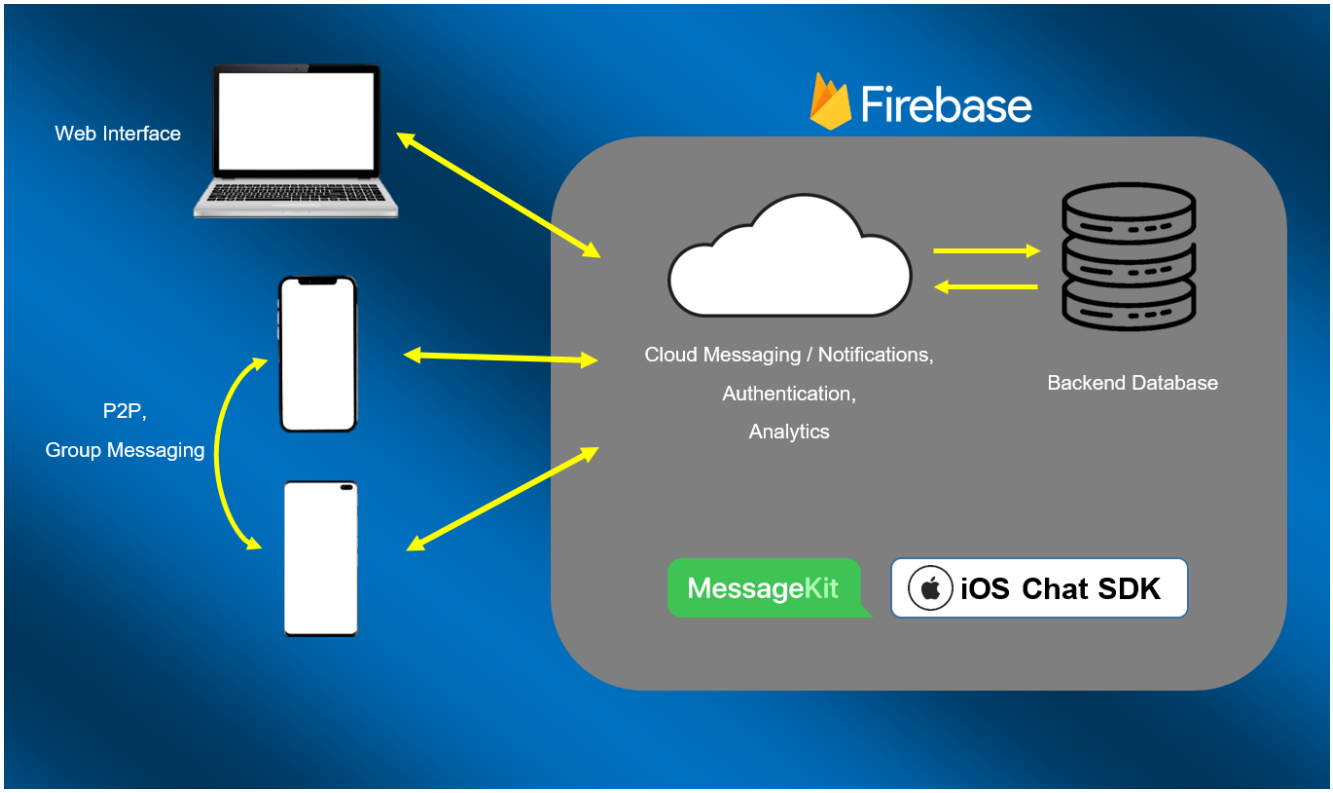
Finally, closed is the section where all issues that have been pushed to the main branch goes.

## System Organization

LiveConnect will initially prompt the guest to login as a user, or as an establishment. If the guest registers as a user, then the user will be asked to authenticate their identity by sending a SMS message to its phone number. Once the user has been authenticated, the user will have access to all the user features. If a guest registers as an establishment, then the guest will be asked to authenticate its ownership of the establishment by sending a verification code to the email address registered under the claimed business. Once the ownership of the establishment has been verified, the establishment will have access to all the business features.

When the user wants to create an event, it will be presented with a template for event creation, with upcoming events, important dates, and recommended venues. After setting up an event, notifications will be sent to invite other users using their phone numbers, and direct bookings with local establishments will be provided.

When a user books a venue, the establishment will be notified. At the same time, the establishment will have waitlist management capabilities, information about guests, and reservation reminders, along with other features.



## Development

The app will be developed using XCode on our own environments. A Github repository will be used to track the most updated files, codes, and configurations for this project. Also, Zenhub, a purpose-built agile project management, will be used to assign roles, issues, and keep track of the development progress. We will also be using Clockify to time track our progress. Web interface will be built at the end of the iOS app development.

## Non-Trivial Requirements

Our project was divided into two main parts, the user base, and the establishment side. Since the user side group messaging system takes priority over most of the qualities on the establishment side, The whole team will start working on the user side and slowly have Jhoseph and Jon move over to the establishment side of the application once the messaging system is running.

### User Side

The user side will have sub projects inside of it that will be divided between group members. The user side has the messaging system, the polling feature, and the booking feature. While everybody will start to work on the messaging system since it is the most important part of our application as told by our sponsor, the polling and booking features will primarily be done by Jacob and Corey.

The messaging system will need to allow for multiple people in a group to communicate to each other as well as peer-to-peer communication. MesageKit and iOS chat SDK will be used for the chat feature to build a fully functioning chat service. The chat service should be able to add more features into the chat such as the polling feature or be able to add media like photos and GIFs which will be completed at a much later date once the prioritized features have been completed.

After our initial meeting with our sponsor, Franklin, he told us that he wants the features of the messaging service to be similar to WhatsApp. Users will be able to sign up or invite new people by using their phone numbers to connect. Using MessageKit for our viewController, we will be able to import features that make our app similar to Message services already on the iOS platform. Using the free services of iOS chat SDK, we can send the messages through them as a server proxy to our clients devices and update the group chat or individual chat to display new messages. We believe that this is the right way to make the app function properly.

The polling feature and booking feature will be done by Jacob and Corey. Using the messaging services, the polling feature will be displayed in chat for users to click single choice or multiple choice options which answers will be sent to the admin of the group to help decide on event details. This feature should be well supported through the iOS chat SDK permissions.

The booking feature will need to use CoreLocation and MapKit. Since a lot of the booking features will rely on the location around the user. Their location and the location of the establishments around them will be crucial in making sure the booking services will be done properly. The booking feature will check from the list of establishments that have made an account with our app what times are open for reservations. It will then send a message using the MessageKit and chat SDK to the establishments with important details like size, and time of the party that will be at the establishment. The establishments will be able to change any availability or inform the client of cancellations.

### Establishment Side

For the establishment aspect of this project, the initial meeting with our sponsor, Franklin,  helped determine that a web based interface would work best from a business perspective. Since Jon and Corey have had prior experiences with Web Application Programming and JavaScript, They will be heading this part of the project. JavaScript will be used to program the specific parts of the application and after asking our sponsor about priorities with our application, The web based aspects can be slightly more rudimentary since the people using this site will be trained by other workers to use the platform. A ‘help’ tab on the website will be given for a procedural informative experience for the establishments

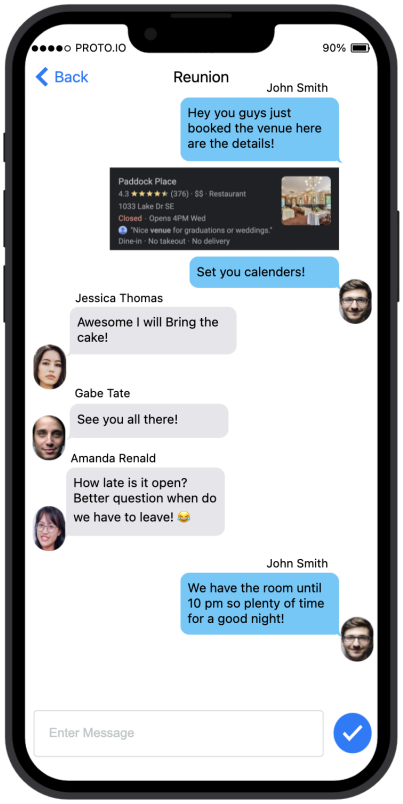
The establishment will then be able to input their availability and any current reservations to show the current open times for new reservations to be made. These times will be kept in an array and sent to the client side for the users to pick from the list a reservation that works for them. This process will send back the users information like their name, number of people in reservation, email address, and phone number with the reservation information like time and date.

Lastly, the establishment will have the ability to change their availability for reservations. They will have the ability to cancel any reservations and close their availability for reservations in case they are fully booked. Anytime a reservation is cancelled, a notification will be sent to the client to apologize.

### Documentation and Styling

No preference were given to us from the Liveconnect sponsor so we will be following GVSU's styling guidelines provided on the CIS website. Since LiveConnect wants to continue working on this app, we will not need to provide documentation for the users but must create a handoff document necessary for programmers continuing this project to understand we have created this document already and will continue to add to it throughout the semester to ensure full detailing of events and proper documentation throughout the project..

## Mock-ups




When the user is invited to a group chat for an event they will be greeted to a screen similar to this one. Users will be able to chat and share locations for the event through google. In this mock up you can see the venue with the google reviews, the address, and the closing time. Each person has their own account set up with their names and a picture that will be displayed for the group to see as shown.

## Work Policies

Roles were assigned based on experience and strengths. The areas of work were assigned to people who had background experience or interest in the task. Corey has

had a lot of experience in mobile app development making him the right fit for lead programmer. He will be in charge of most of the programming with help from the others. Corey will oversee the development of the app allowing for a smooth development. Jhoseph has experience creating a Cloud-based messaging application. He will be mainly responsible for the messaging between users. Jon and Jacob will have varying roles that will constantly be changing to help where it is needed. They will also be in charge of team presentations and documentation. The team will work together to complete the app, with members responsible for the areas they have the most experience in.

## Testing

Our group will be using a SCRUM workflow methodology meaning that not a lot of time will be spent on testing aside from basic unit testing to make sure key components and features work. The group will make sure that the app has reliable code by approving all pull requests with other members of the team present for the approval and making sure that the code works both before and after the integration of the pull. During each sprint meeting with our sponsor, the sponsor will then approve all work that has been done or tell us specific features that need to be changed in order for their application to be ideal.

## Deliverance and Milestones

Sprint 1

- Create login screen with user authentication for new users
- Create invitation feature to invite new users
- Create a P2P messaging service
- Store data on database service
- Event creation template

Sprint 2

- Create analytics from data stored
- Create rudimentary polling feature using single choice response
- Create a search feature for establishments using a map
- Create a booking feature
- Notification services
- Events list

Sprint 3

- Refine messaging service to allow for group messaging
- Expand invitation feature to invite through SMS, email, link
- Create a Web based application
- Refine search functionality with filter options

Sprint 4

- Polish polling feature to allow for multiple choice responses
- Polish web application to allow establishments to manage/view bookings
- Polish events list to allow for different sorting methods
- Bugs

Sprint 5

- Documentation / Polishing

Sprint 6

- Presentation and product submission