betcha!

Team 19 - Design Document:

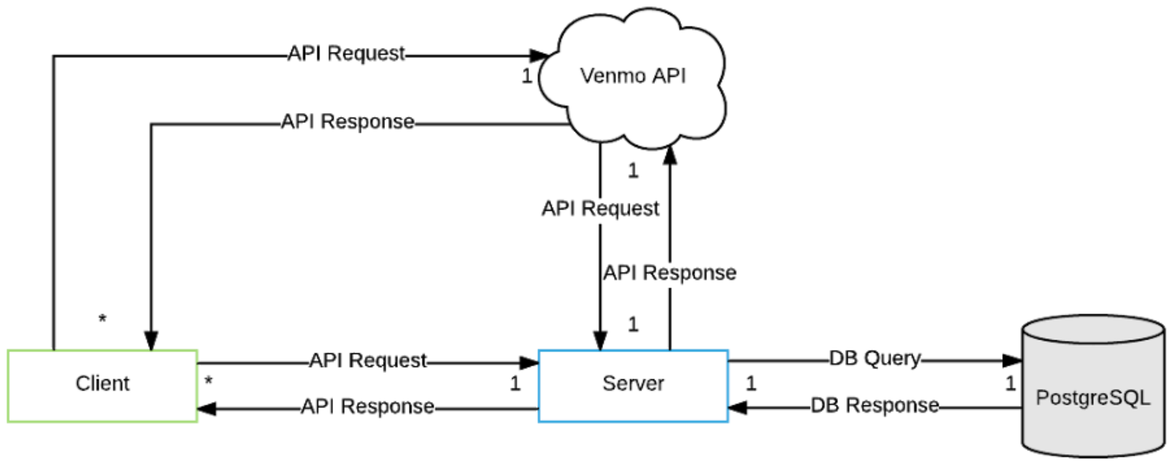
Kushagra Kushagra, Kyle Ohanian,
Noah Smith, Peter Jones, Siddharth Shah

Purpose

Casual betting between friends is a very common, very unorganized affair. Then in this space there is a perfect opportunity for an application that could keep a ledger of the bets and the details. Unfortunately, current offerings do not allow betting on custom events or interaction with a social feed. This product will allow parties to organize and keep track of their bets, as well as be able to interact with their friends' betting.

Design Outline

At a high level, the system will work in that the Android App client will communicate via API Requests to the Server. The server will then respond to those requests with data to be served via the Android Interface. Below is a high level view of the project architecture:



These technologies form the system architecture used for *Betcha!*. Below is a quick summary of the two key components of the Client-Server Architecture and their connection to other pieces of the system:

<u>Android Application (Client):</u>

- *Activity Classes* will provide Android XML from which the User Interface will be produced.
- *View Models* built with Java perform the client logic and data manipulation and call the Activity Classes.
- *SQLite* database connected to the View Model's using the Room Library will provide low latency, cached data for high use data.
- The *Venmo Android SDK* will be called to get tokens for User Authentication
- *HTTP API Requests* using the Volley Library to get data from the server.

Python Flask Hosted on AWS (Server):

- API Requests will be handled by a *RESTful Python Flask instance*.
- The server will contain a *PostgreSQL* Database for persistent data connected to the system logic using Python database connectors.
- Data will be kept additionally in a *Redis* Database to provide low latency data in memory.
- The server will also poll the *Venmo API* to confirm user identity.

Design Issues

Functional Issues:

1. **How should users create an account to use the application?**
   Option 1: Create an account using Facebook Login
   Option 2: Create an account using Venmo Login
   Option 3: Create a new login specifically for the application
   Option 4: Create an account using any of the above methods
   Decision Description
   We chose to require users to have a Venmo account to use the application. The main function of the application is to participate in bets which will require a Venmo transaction at some point. All the data relevant to our application can be obtained through Venmo which allows us to utilize Venmo Oauth. Therefore connecting a Facebook account would be redundant.

2. **When should Venmo transactions take place?**
   Option 1: Two transactions, before and after the bet, occur between the user and a designated betcha! Venmo account
   Option 2: The transaction(s) occur between users at the conclusion of the bet
   Decision Description
   We chose to have transactions only occur between users. There may be a case where one user has to distribute money among many other users. If we used a designated account the bet pool would only have to be collected from the losers, and distributed evenly to the winners. This method might elevate the number of notifications in the user's Venmo feed, but this method results in sequential transactions. These transaction amounts will be more difficult to track because it is moving through two accounts that don’t send or receive money instantly. This method is also prone to error, and possibly shifts legal liability to

our Venmo account.

3. **How will users obtain friends within the application?**
   Option 1: Import friends from Venmo
   Option 2: Import friends from Facebook
   Option 3: Add friends within the application
   Decision Description
   Our previous design decisions influenced the decision to allow users to import their friends through Venmo and add friends within the application. We chose not to allow users to import friends from Facebook because we wanted to reduce the number of APIs we need to familiarize ourselves with and because Venmo allows you to implement friends from Facebook. We chose to allow users to also add friends within the application because of the simplicity of the implementation.

4. **How many people are required to start a bet?**
   Option 1: Minimum of one user
   Option 2: Minimum of two users
   Option 3: Only two users allowed per bet
   Decision Description
   We chose to allow users to create a bet starting with only one user. Allowing only two users per bet reduces complexity because the payouts do not need to be split, but decreases usability because a user can not add others to a bet. Requiring a minimum of two users per bet introduces edge cases when people leave a bet. We chose option 1 to make our bet class more extendable and reusable.

Non-Functional Issues:

1. **What platform should we develop our application for?**
   Option 1: Web application
   Option 2: iOS application
   Option 3: Android application
   Option 4: Cross-platform using Xamarin
   Decision Description
   A web application could introduce betcha! to a larger audience, but it doesn't really help solve the problem of causal betting. Ideally we would like to have our application on iOS, and Android, but we have limited experience with Xamarin, or iOS. We chose Android because we all have previous experience with Java

and Android Studio.

2. **What hosting service should we use for the backend of the application?**
   Option 1: Amazon Web Services
   Option 2: Heroku
   Decision Description
   We chose Amazon Web Services because they offer cheaper student pricing which Heroku does not. Also AWS is cheaper in generally and would help the applications scalability by reducing server costs.

3. **What database software should we use to store the user and bet data?**
   Option 1: PostgreSQL
   Option 2: MySQL
   Option 3: MongoDB
   Decision Description
   We chose to use PostgreSQL over the NoSQL option mongoDB because PostgreSQL offers structured storage for relevant retrieval.
   The choice of PostgreSQL over MySQL is due to familiarity and past experience as well as its robust scalability, efficiency, and ease of deployment.

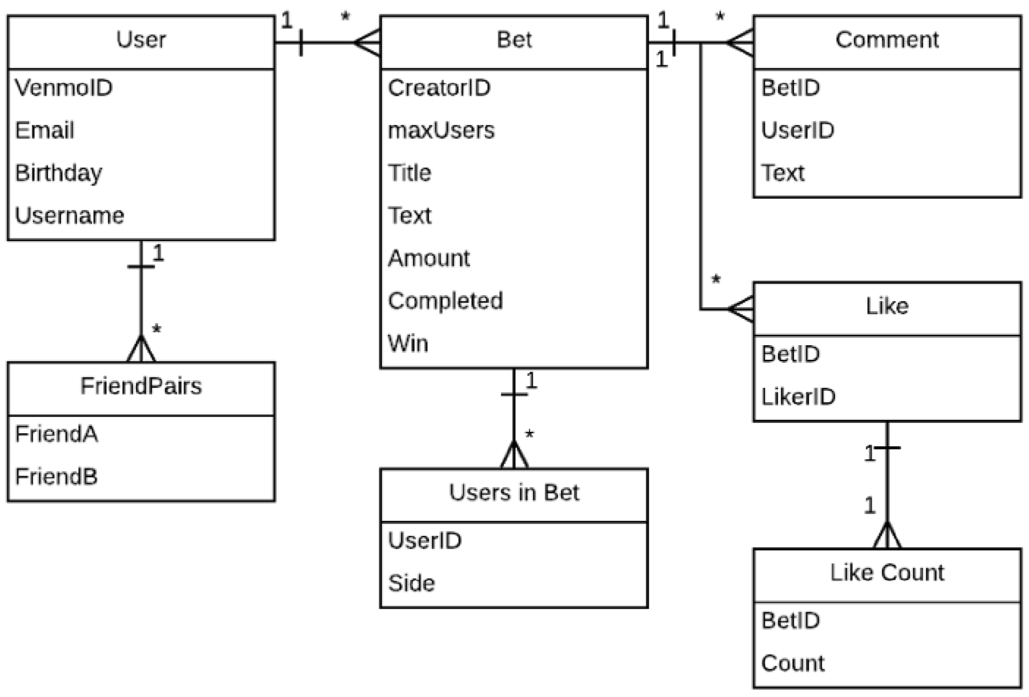
Design Details

To begin with, the client side will be an Android Application. Built using Java, the app will serve the data to the user using Android XML populated by data from View Models. These view models will be persistent classes so as to not be destroyed by the Operating System keeping memory usage low. To not waste too much battery using the network, the application will keep a subset of the data cached in a local SQLite Database. It will interact with this database using the Room Object Mapping Library capabilities built into Android. Finally, the application client will interact with the server using the Volley library. It will be able to send requests to the API using this library and also parse the JSON Responses from the Server.

The server will be a RESTful API server hosted on Amazon Web Services. The server is the central nervous system of the project. It will hold all the various social and transactional data so that it can be shared between unique users and be kept in a persistent manner. It will be built using Python Flask. It will have different endpoints for the different services the application can request. These requests will then trigger modifications to the database and other back-end tasks. The Database on the back-end will be built on PostgreSQL also hosted on AWS. The Server will interact with the Database using Python SQL Database Connectors. It will then use this data to send JSON responses back to the client. Another part of the backend will be a Redis deployment. Redis is essentially a database cache. This will provide the backend with the ability to mask the slower PostgreSQL, persistent database with this much faster in memory database. These components come together to produce a fast, persistent server.

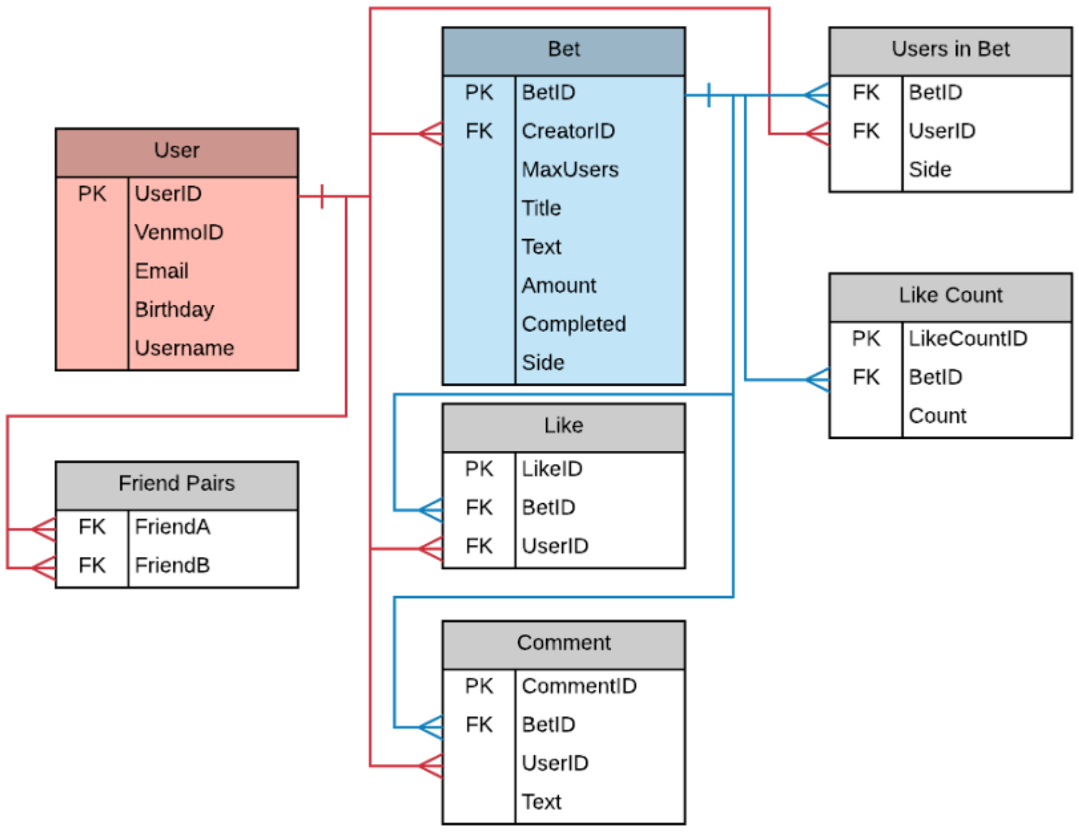
Finally, another key component of the project architecture is the Venmo API implementation. There are two places within the system that Venmo will be implemented. The first is within the client side Android application. Using the Venmo SDK, we will be able to do OAuth with the Venmo API to verify identity using a unique token. This token can then be used with each API request to the server. This is important to verify user identity on the app. But for the server to be able to check that the token ID is correct, it will have to poll the Venmo API as well. This is the second piece of the Venmo connection. In this, the server will check the token with the token it receives from the Venmo API before providing any protected services.

Class Diagram



Database Design

The Persistent database on the server will be a relational database using PostgreSQL. The basic database schema can be seen below with Primary and Foreign Keys indicated.



Database Details

User

- UserID is the Primary Key for the Table
- VenmoID is the VenmoID linked to the user account
- Email is the user's registered email
- Birthday is the user's entered birthday for age verification
- Username is the public facing username that will go with each bet

Friend Pairs

- FriendA is the first UserID in the friend pair
- FriendB is the second UserID in the friend pair

Bet

- BetID is the Primary Key for the Table
- CreatorID is the UserID of the Bet Creator

- MaxUsers is the max users allowed to be connected to the bet
- Title is the title of the bet
- Text is a description of the bet
- Amount is the amount bet
- Completed if the bet is completed
- Side is the side of the bet the creator chooses

Users in Bet

- BetID links to the Bet the user is a part of
- UserID is the ID of the user attached to the bet
- Side is the side the user selected

Comments

- CommentID is the primary key
- BetID is the ID of the bet being commented on
- UserID is the ID of the user who commented on the bet
- Text is the text of the comment

Like

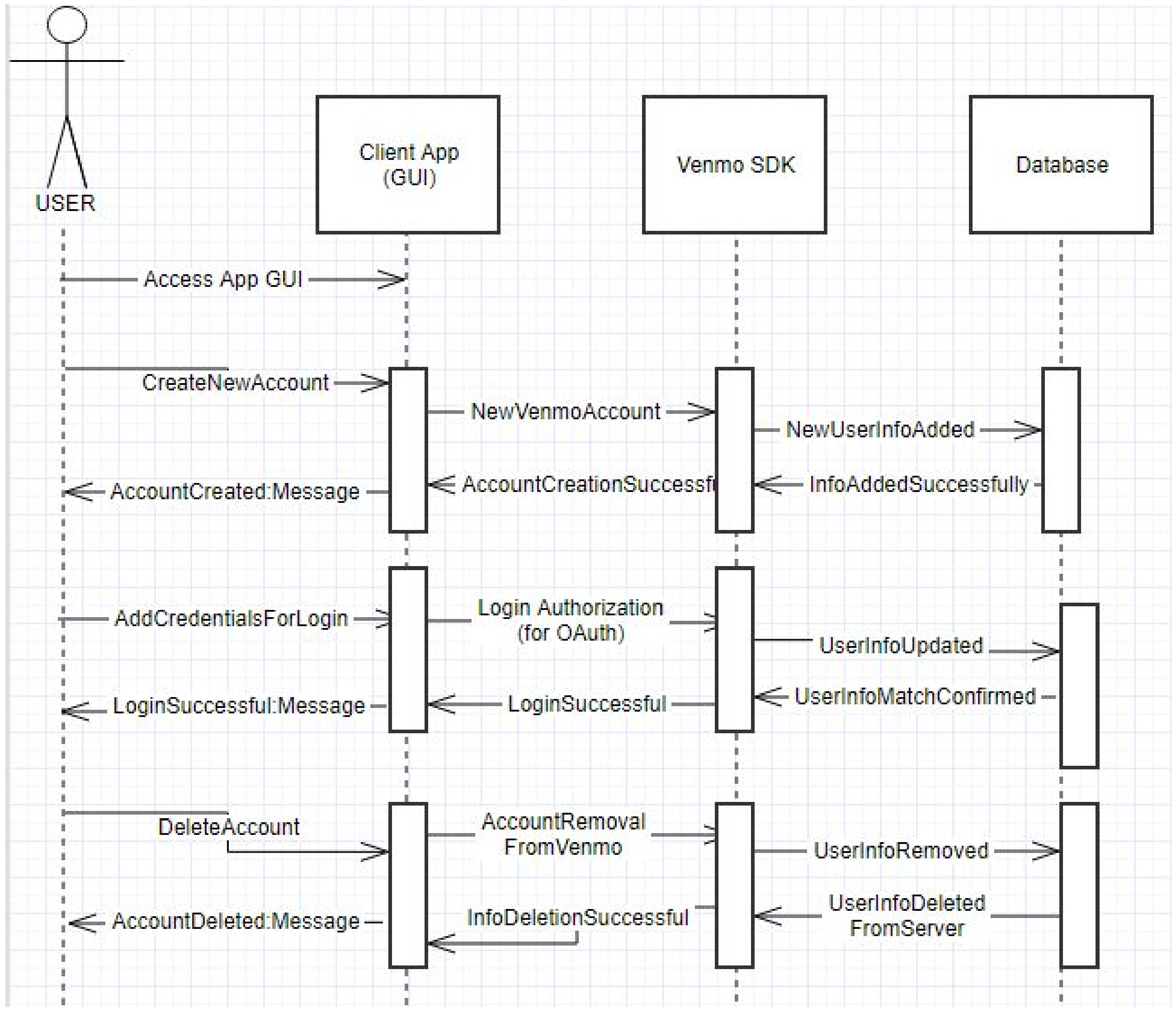
- LikeID is the primary key
- BetID is the ID of the bet being liked
- UserID is the ID of the user who liked the bet

Like Count

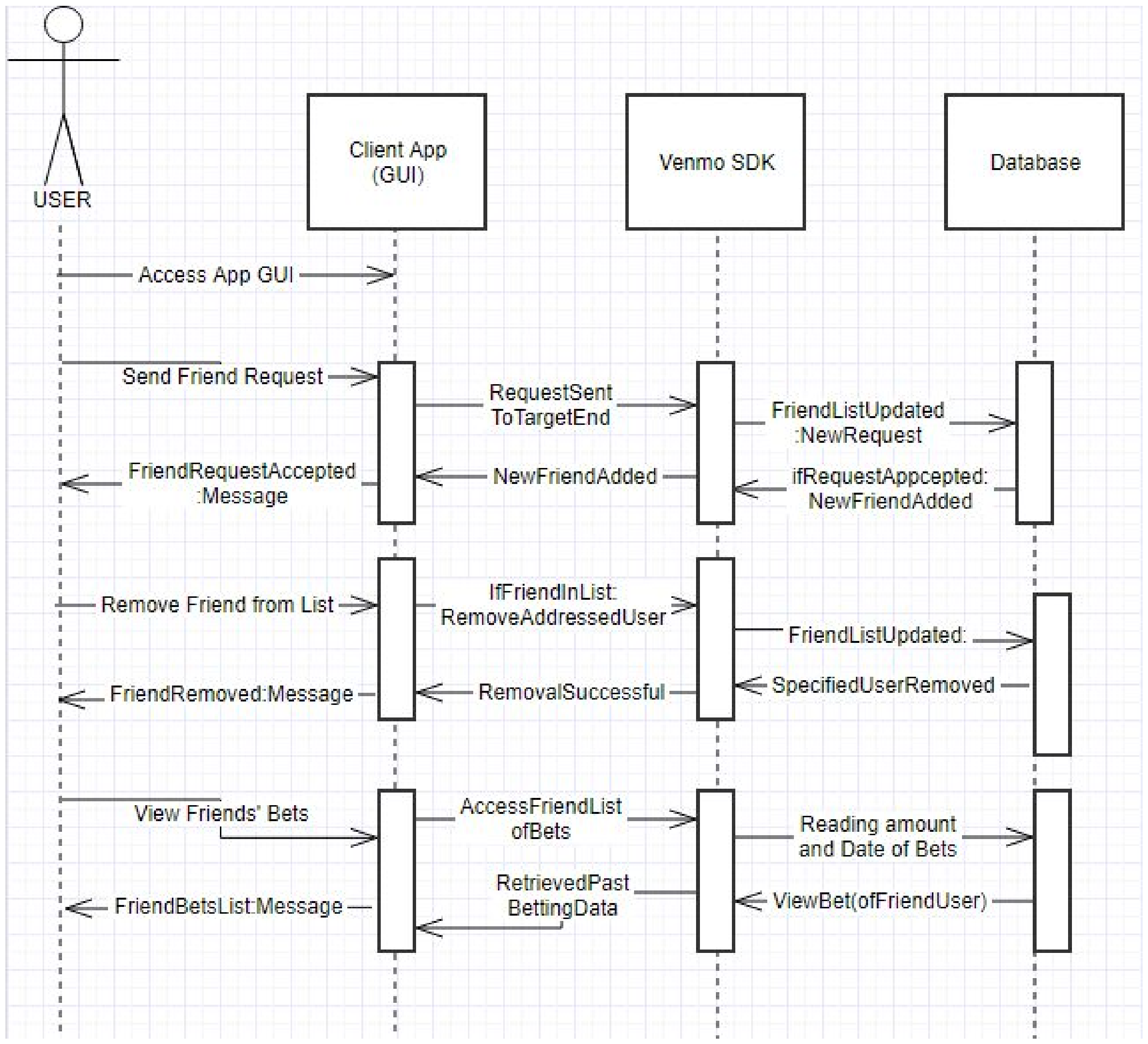
- LikeCountID is the primary key
- BetID is the bet the counts are related to
- Count is the number of likes on the bet

Sequence Diagrams

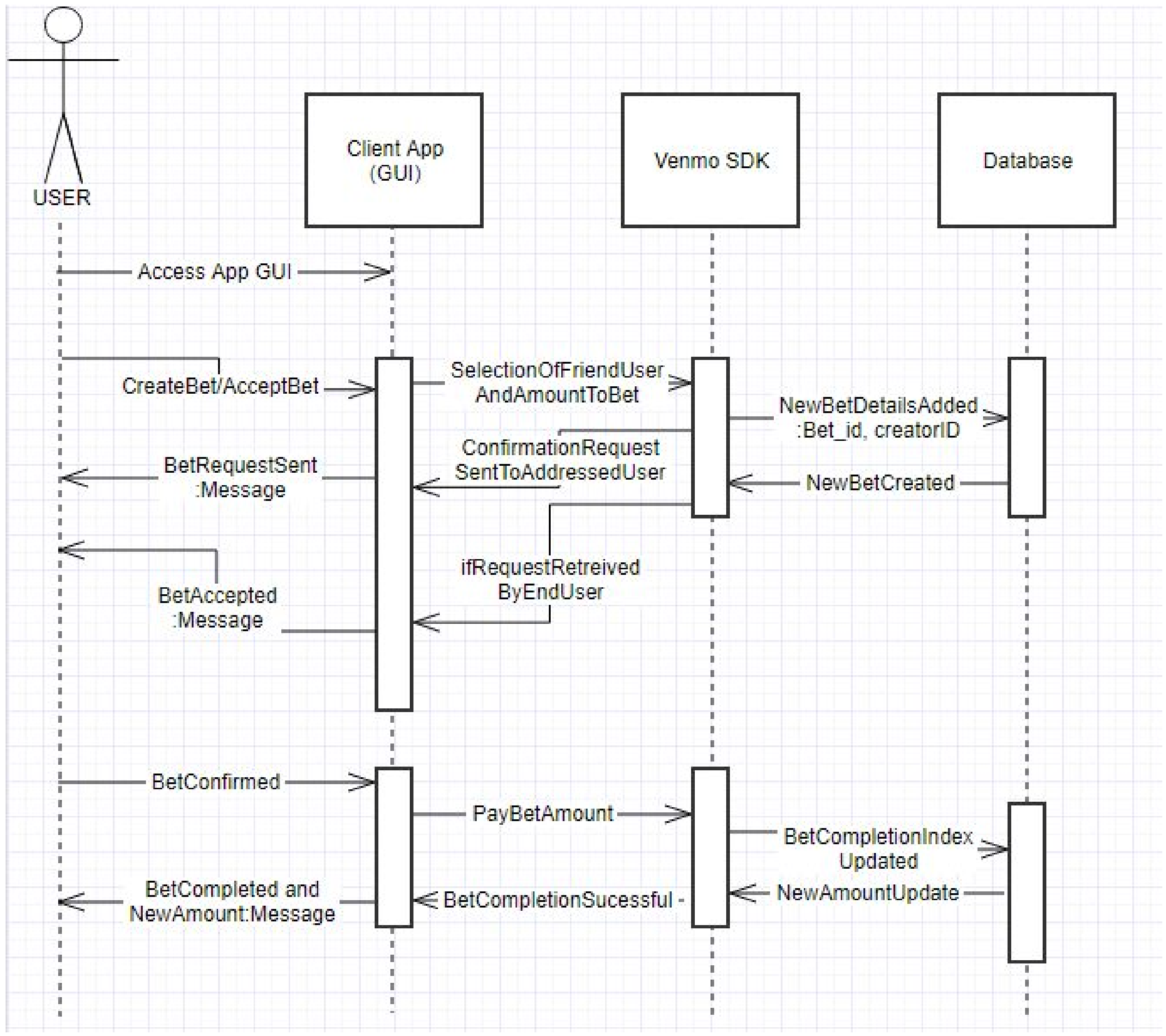
Login Sequence



Managing Friends

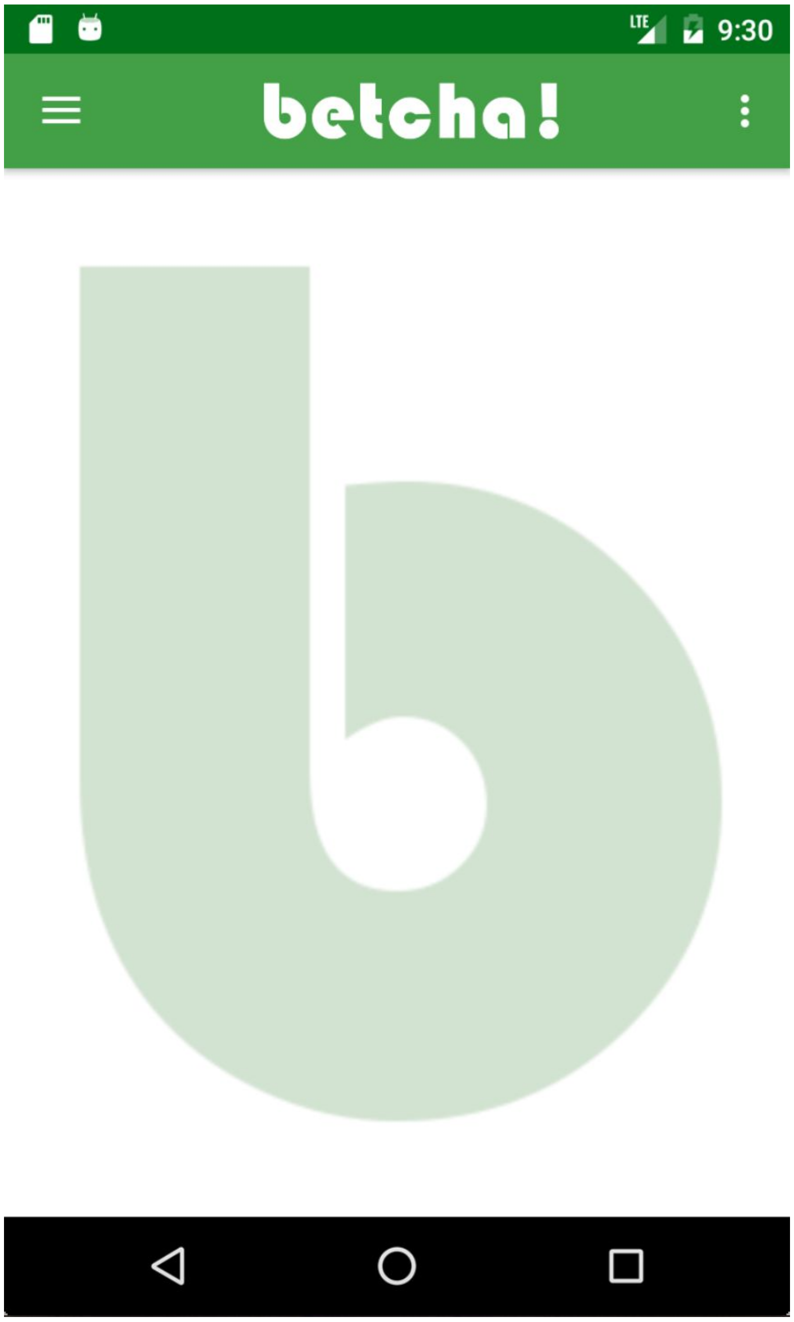


Betting and Payment

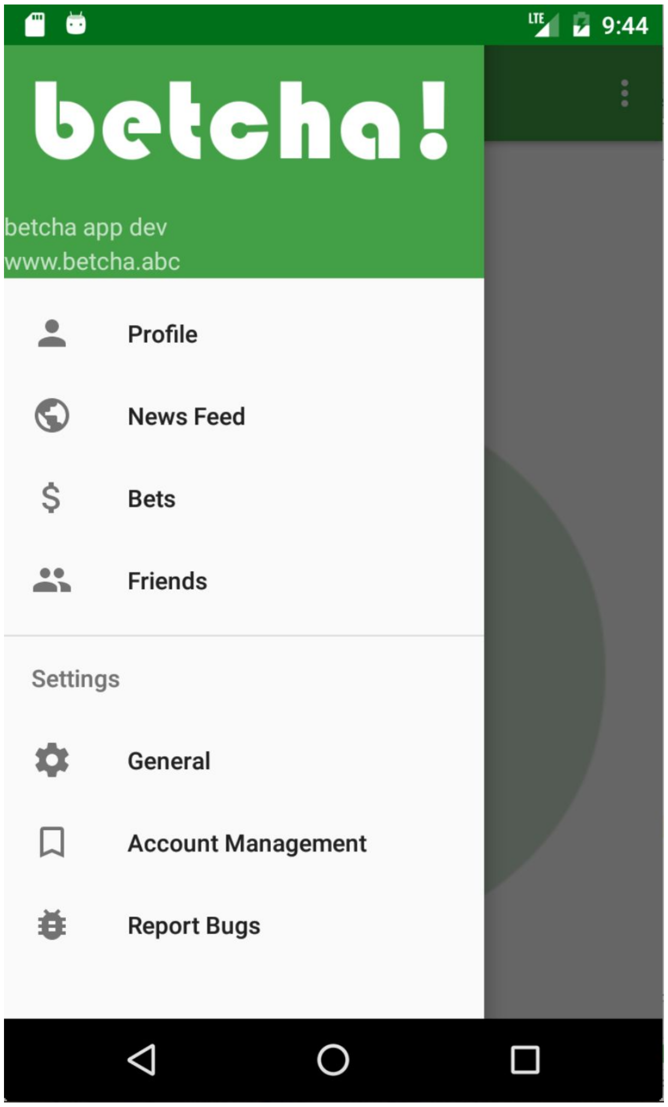

USER
Client App (GUI)
Venmo SDK
Database
Access App GUI
CreateBet/AcceptBet
SelectionOfFriendUser AndAmountToBet
NewBetDetailsAdded :Bet_id, creatorID
ConfirmationRequest SentToAddressedUser
BetRequestSent :Message
NewBetCreated
ifRequestRetreived ByEndUser
BetAccepted :Message
BetConfirmed
PayBetAmount
BetCompletionIndex Updated
BetCompleted and NewAmount:Message
BetCompletionSucessful
NewAmountUpdate

UI Mockups

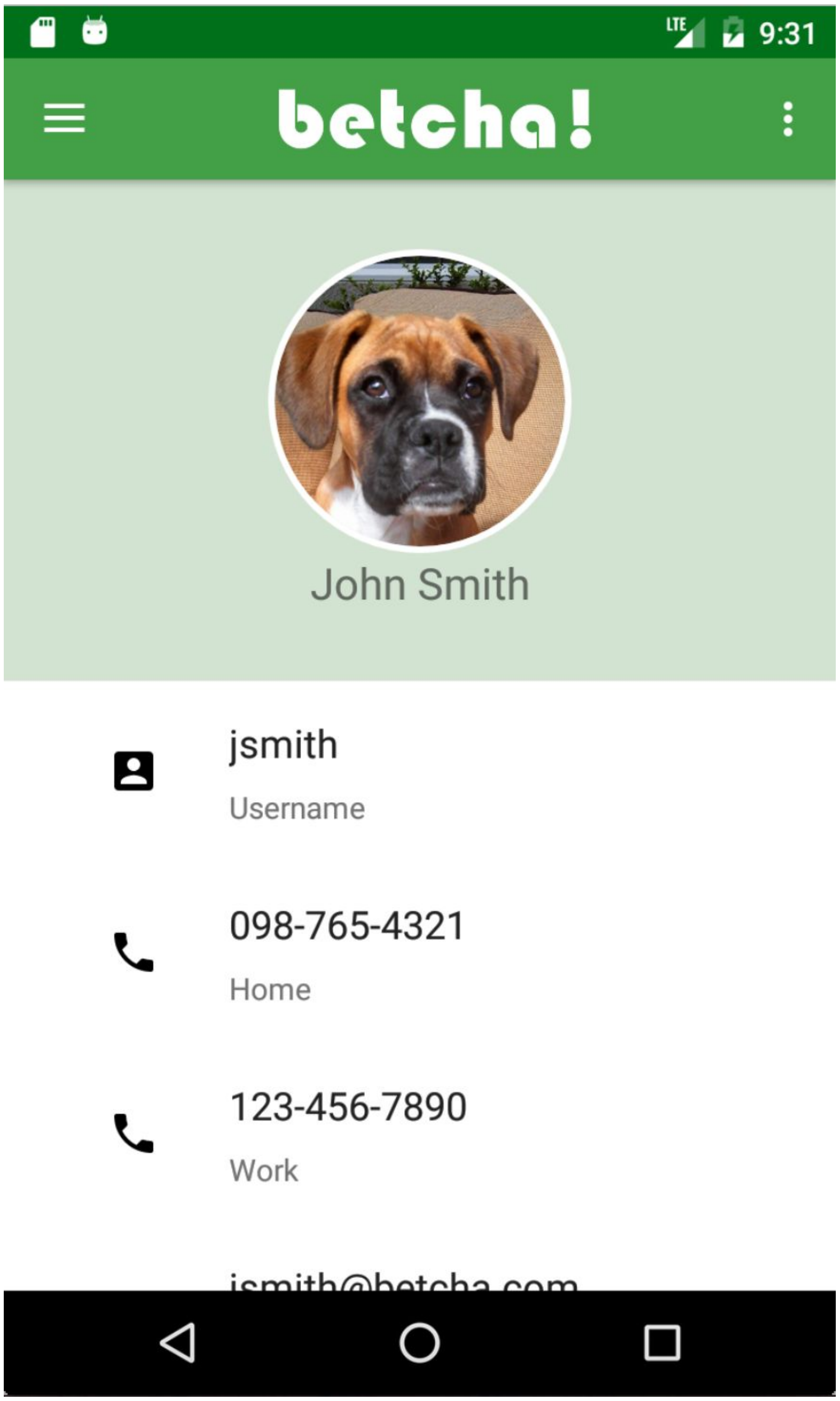
The UI will consist of a nice bar at the top with the logo. The home page is a transparent view of the ‘b’ in betcha, and will be seen every time someone logs into the app. It also has a navigation view on the side that the user can use to navigate through the app. Any feature that they want to use will be have to go through this navigation view. All other views will be display at the click of a navigation option.



The Home Page will be displayed every time the user logs into the application. The reason the ‘b’ is faded green is because we don’t want there to be too much “noise” when they first open the application.



This is the **Navigation View**, where the user will be able to access all features of the application. **Profile** goes to their betcha! Home page and will show their profile as everyone else in the world sees it. **Bets** will show the list of available bets that their friends have created. **Friends** will go to their list of friends. **General** goes to any general settings they would like to change on the application. **Report Bugs** is where they can submit a defect that the application has. The other tabs are described below. Inside of each tab are a whole bunch of other features that will be set, such as creating bets, joining them, liking bets, adding friends, etc. This is the “first step” to using all features in the application.



John Smith

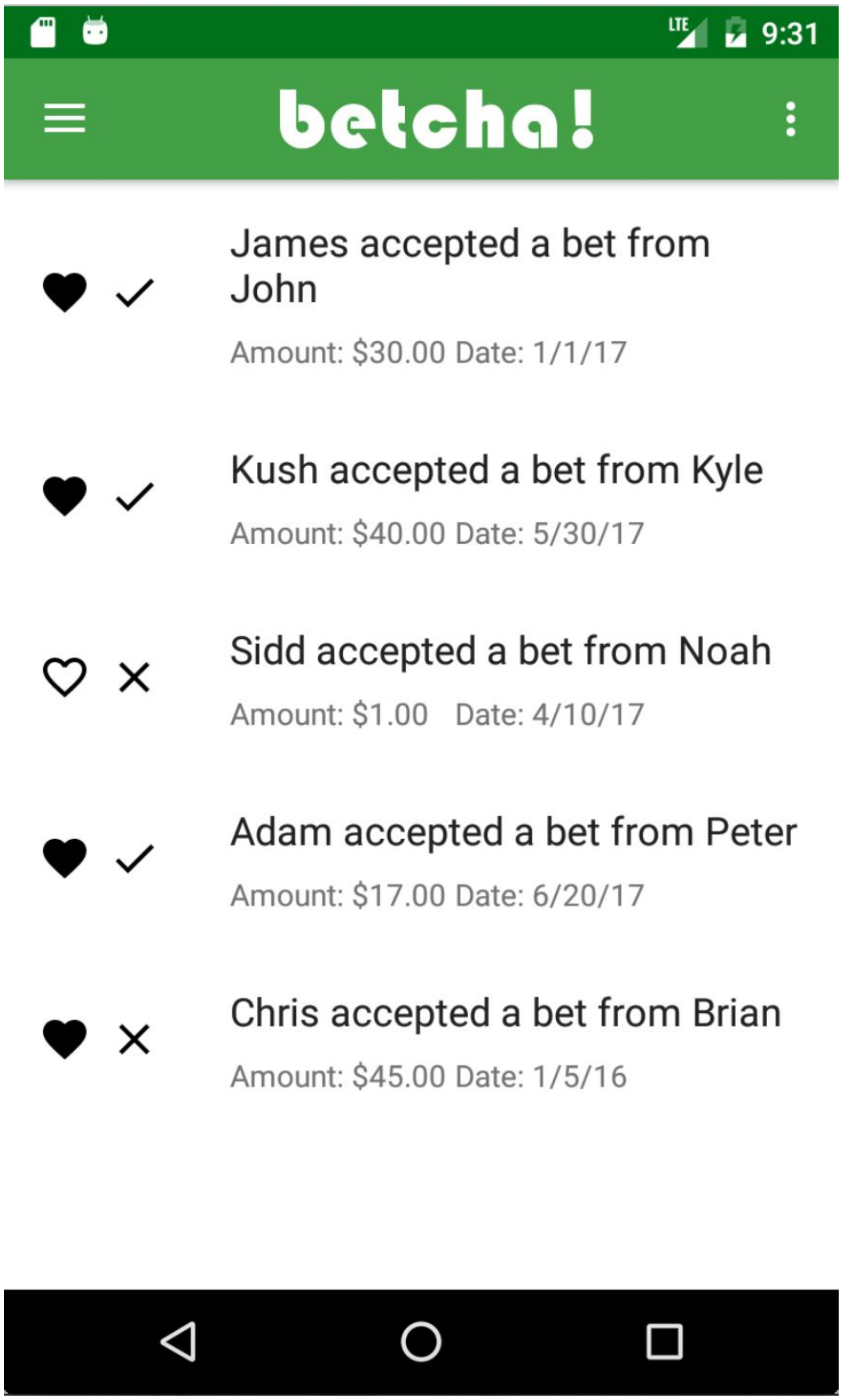
jsmith
Username

098-765-4321
Home

123-456-7890
Work

jsmith@betcha.com

This is the **Account Management** page, which is where the user can view their user information, such as Username, emails, etc. It will also feature a nice rounded picture.



James accepted a bet from John
Amount: $30.00 Date: 1/1/17

Kush accepted a bet from Kyle
Amount: $40.00 Date: 5/30/17

Sidd accepted a bet from Noah
Amount: $1.00 Date: 4/10/17

Adam accepted a bet from Peter
Amount: $17.00 Date: 6/20/17

Chris accepted a bet from Brian
Amount: $45.00 Date: 1/5/16

This is the **News Feed** Page, which will gather information from the user’s circle, and display what kinds of bets are going around their friends. They are able to click on bets to see more details about what the bet is about, the wagers, etc. Users are also able to like other bets as well.