**FACEBOOK IDS**

**Azure Classes and Controllers:**

* Startup
  + Sets up routing configuration for all controllers
  + Initializes database context using Entity Framework Code First
  + Adds several test users, households, and household members to the context
* Friend Controller
  + Primary role is to retrieve the households of Facebook friends of the current user who also use the app, but the user is not yet a member of. This is accomplished via an authenticated GET request.
  + The friend controller is called on the app’s main/welcome screen after the user logs in to allow the user to see the households of his or her friends that also use the app. The user needs to view these households in order to request to join them.
  + If the user is already a part of a friend’s household, this household will not appear in this request’s results.
* Household Controller
  + The primary role of the household controller is to get, create and update data related to households. A household is the basic communal space that users can be a part of. The relationship is a user can have many household memberships (HouseholdMember), and a household can have many memberships. Two other functionalities are also associated with households- voting and messages. Household members can participate in voting and messaging which is associated with a particular household. All household members can view and participate in the voting and messaging.
  + GET request that returns a list of households that a user is a member of.
  + GET request that returns a specific household by its id with the following info:
    - Household name
    - Household description
    - Name of currency
    - Name of the Landlord
    - List of household members
  + GET request that returns a household membership (HouseholdMember) based on a household id and user id.
  + POST action that creates a new household. This is called from the client app's home screen when the user would like to create a new household. The user that created the household will automatically become the landlord. Returns a message indicating success or failure of the creation.
  + POST action that updates the basic info about a household such as the household's name, a description, and the name of the currency. This method is called from a household's home screen to allow the landlord to edit these fields. Only the landlord may edit these fields:
    - Household name
    - Household description
    - Name of currency
  + GET request that returns list of household members for a specified household id.
* Message Controller
  + The primary role of the message controller is to store and retrieve messages for each household in Azure Document DB. Household members can create/post and view messages for a household, which are viewable to the entire household. Document DB is useful for this purpose because it allows for flexible schema which is useful if other data fields should eventually be added to the messaging functionality, and it also allows for Azure Search which will allow users to search prior messages.
  + GET request that returns all of the messages in the collection
  + GET request that returns all messages for a specified household.
  + POST action that creates a new message in Azure Document DB
* Registration Controller
  + The primary role of the registration controller is to register a user, i.e. save their info in Azure SQL, once they've authenticated through Facebook. This allows their Facebook user ID and name to be stored in Azure SQL rather than having to make a call out to Facebook's Graph every time that info is needed. The user does not have to initiate this action: once they authenticate through Facebook these methods are automatically called by the client app.
  + GET method that returns the registration status of the current user.
  + POST method that allows any authenticated user to register. This method is called automatically by the client app after the user logs in / authenticates with Facebook. If the user already exists no action is taken, but if it’s a new Facebook user their info is added to the Azure SQL database.
* User Controller
  + The primary role of the user controller is to retrieve data about users.
  + GET method that returns all users (mainly used for dev/testing)
  + GET method that returns a user by the id specified.
  + GET method that returns the current, authorized user.
* Voting controller
  + The primary role of the vote controller is to allow users to view, vote, and create proposals in the household. A "vote"/"proposal" relates to changes in membership, user karma, and the landlord. For example, if a user wanted to join a household, there's a "Join" button on the client app that will create a new vote (NewMember type) via this controller. The new vote will then appear in the list of votes for the households, available for all household members to view. Votes must be approved by a majority of the existing members in order to pass. The current version of this app allows for new member votes and karma change votes. Future releases will allow votes on electing a landlord and evicting existing members.
  + GET method that returns all the votes for a household. This is called by the client's "Voting" screen to disply all votes to the user. Returns the following for each vote:
    - Type of proposal (i.e. “Karma”, “New Landlord”, “New Member Approval”, “Evict Member”)
    - Status of the vote (ex. “2 members for, 1 against, 3 needed for majority” – 3 needed for a household of 5 for example for a karma vote)
    - Member the vote applies to (member if karma vote or landlord vote, user if a new member vote)
    - Member who proposed the vote (or anonymous)
    - Brief one-line message/description
    - List of member ids that voted for/against the vote
  + POST action that creates a new vote for a household.
    - Household ID
    - Type of proposal (i.e. “Karma”, “New Landlord”, or “Evict Member”)
    - Member to which it applies
    - If karma proposal, amount of karma to be added/deducted
    - Whether proposal should be made anonymously
    - Brief one-line message/description
  + POST action that applies a household member's vote to an existing proposal/vote. Each member is only given one vote per proposal and may not take back their vote. The controller will accept a vote containing:
    - The id of the vote
    - The vote, “For” or “Against”

**Android Client Screens and Classes:**

* **Main Activity/Screen**
  + This class represents the main and first screen any user sees when the app is launched. It's a landing page where the user can login, quit the app, create a household, view their household, and households their friends belongs to. Clicking on one of these households brings the user to that household's screen.
  + A login button allows the user to login and logout using Facebook authentication. The user must login to use any additional functionality.
    - After the user successfully logs in the client also makes a call to the Registration resource to check and make sure the user is registered, meaning their info retrieved from Facebook is saved in the Azure SQL database. If the user does not already exist, the Registration controller will save the user’s id and name. The user id is saved by the client so that Facebook doesn’t need to be called every time info about the user is needed.
  + Once the user logs in, the screen will show either households the user belongs to, or households the user’s Facebook friends belong to (but the user is not yet a member of), depending on a toggle switch at the top of the screen. This is done by making calls to the Household or Friend controller in Azure. When the switch is toggled the display is automatically updated with the user’s or user’s friends’ household list.
  + A “Create Household” button sends a request to Azure Household resource to create a new household. The new household appears in the user's household list after the screen is updated.
  + If the user navigates to/from this screen, the user’s authentication data is preserved so that the correct data is displayed on the screen and the user does not need to authenticate again.
  + A “Quit” button allows the user to quit the application.
* **Household Activity/Screen**
  + The household screen contains basic info about the household such as its name, description, currency, landlord, and a list of members. The landlord can edit this info. To users that are members of the household, there are buttons that take the user to the Voting and Messaging screens, and for users that are not members there's a button that allows the user to request to join the household.
  + The “Request to Join” button allows the user to request to join the household if they are not a member. It makes a POST request to the Household resource in Azure, adding a NewMember vote to the household's list of votes with the user as the target member. The result is that a proposal to allow the user to join the household appears in the household members' voting screen, and if it's passed the user will become a member of the household.
  + If the user navigates to/from this screen, the user’s authentication data is preserved so that the correct data is displayed on the screen and the user does not need to authenticate again.
* **Voting Activity/Screen**
  + This class represents the voting screen that allows the user to view all the votes for the household. Information about each vote is displayed, including the type of vote, the target member, its status, and two voting buttons for the user to vote for or against the proposed vote. Users are able to propose a vote of their own and select a member (either themselves or another household member) and propose that that member have currency added or subtracted from their balance.
  + All votes from the household are displayed as a scrollable list. Each vote has info about the target member, its status, and a “Vote Yes” and “Vote No” buttons. These buttons become disabled if the voting is over or if the user already voted.
  + If the user navigates to/from this screen, the user’s authentication data is preserved so that the correct data is displayed on the screen and the user does not need to authenticate again.
* **Message Activity/Screen**
  + The messages screen displays all the messages for a household. Users can create/post messages to this forum and view messages posted by other users. All messages on this screen are viewable by all household members, and the user has the ability to search/filter previous messages. The messages are stored in Azure Document DB and the search ability utilizes Azure Search.
  + This screen allows the user to enter a text message and the “Send” it to the rest of the household. The message will then be displayed on this screen for all household members. When the “Send” button is sent, a POST call is made to the Message controller in Azure, which then in turn sends the message contents (along with timestamp and user info) to Azure Document DB for storage.
  + The "Search" button allows the user to search through the household's messages based on a search term. The household that meet the search criteria are then displayed to the user. The messages are stored in Document DB and Azure Search is used to implement this functionality. The client makes the call to the Azure Search service directly. The indexer runs every five minutes so users will have to wait a few minutes before their message becomes searchable.
  + If the user navigates to/from this screen, the user’s authentication data is preserved so that the correct data is displayed on the screen and the user does not need to authenticate again.

**Areas for improvement**

* Address security
  + Implement protections against SQL injection attacks
  + Ensure proper authorization for methods
  + Need to update workflow for Azure Search so that only relevant messages for the caller’s household are returned to client
* Refactor client class structure to reduce repetitive code
  + For example, methods like navigationClick and getIntentParameters are used by all the Activities and could benefit from an inheritance structure