**Test Documentation - Frontend**

**Login Page:**

*Username/Password Fields*

* Make Username and Password fields appear
* Test typing (see if text appears on screen)
* Test to see if box colors change when clicked
* Test to see if they store values (print in terminal)

*Login Button*

* Make button appear
* Test to see if button clicks (print in terminal)
* Add navigation to home page
* Test to see if button routes to home page
* Test to see if button sends info to other classes

*Sign Up Button*

* Make button appear
* Test to see if button clicks (print in terminal)
* Add navigation to signup page
* Test to see if button routes to signup page
* Test to see if button sends info to other classes

**Sign Up Page:**

*Email/Username/Password Fields*

* Make the fields appear
* Test typing (see if text appears on screen)
* Test to see if box colors change when clicked
* Test to see if they store values (print in terminal)

*Sign Up Button*

* Make button appear
* Test to see if button clicks (print in terminal)
* Test to see if button routes to home page
* Test to see if button sends info to other classes
* Test to see if button click stores account information

**Home Page:**

*Create Group button*

* Make button appear
* Test to see if button clicks (print in terminal)
* Add navigation to create group page
* Test to see if button routes to create group page

*Join Group button*

* Make button appear
* Test to see if button clicks (print in terminal)
* Add popup trigger to button click
* Test to see if button click triggers popup
* Test popup typing
* Add cancel and submit button to popup
* Test to see if button work (print in terminal)
* Test to see if cancel button closes popup
* Test to see if submit button routes to group home page

*Account Drawer button*

* Make button appear
* Add icon to button
* Test to see if button clicks (print in terminal)
* Test to see if button click pulls out account drawer

*Group Drawer button*

* Make button appear
* Add icon to button
* Test to see if button clicks (print in terminal)
* Test to see if button click pulls out group drawer

*Lost Text*

* Make text appear
* Align text at bottom of page

**Account Settings Page:**

*Email/Username Fields*

* Make Email and Username fields appear
* Test typing (see if text appears on screen)
* Test to see if box colors change when clicked
* Test to see if they store values (print in terminal)

*Update Button*

* Make button appear
* Test to see if button clicks (print in terminal)
* Test to see if button changes account info (print info in terminal)

**FAQ Page:**

* Make app bar appear
* Add text to app bar, see if updates
* Add body text to screen, see if it appears
* Left align text, check if aligned

**Account Drawer:**

* Make drawer appear
* Add Account tile
* Test to see if tile clicks (print in terminal)
* Test to see if tile routes to account settings page
* Add FAQ tile
* Test to see if tile clicks (print in terminal)
* Test to see if tile routes to FAQ page
* Add Logout tile
* Test to see if tile clicks (print in terminal)
* Test to see if logout pushes you back to the login page

**Create Group Page:**

*Group Name/Group Description/Group Code Fields*

* Make the fields appear
* Test typing (see if text appears on screen)
* Test to see if box colors change when clicked
* For Group Code field, add length limiter so that only 6 characters can be added

*Create Group Button*

* Make button appear
* Test to see if button clicks (print in terminal)
* Test to see if button routes to group page
* Test to see if button sends info to other classes, and stores group information

**Group Home Page:**

*App Bar*

* Make app bar appear
* Add text to app bar, see if updates
* Pull group name from group object, update text with name

*Group Picture*

* Make place holder image appear
* Test to see if it appears
* Pull group profile picture from group object and update the picture
* Test to see if picture updates

*Group Description*

* Make text appear
* Test to see if it appears
* Pull group description from group object and update the text with the description
* Test to see if description updates

*Chat Channel Button*

* Make button appear
* Test to see if button clicks (print in terminal)
* Add navigation to chat page
* Test to see if button routes to chat page

*Poll Channel Button*

* Make button appear
* Test to see if button clicks (print in terminal)
* Add navigation to poll page
* Test to see if button routes to poll page

*View Members Button*

* Make button appear
* Test to see if button clicks (print in terminal)
* Add navigation to member view page
* Test to see if button routes to member view page

**Chat Page:**

*App Bar*

* Make app bar appear
* Add text to app bar, see if updates

*Messages*

* Make message appear (add text and background)
* Test to see if multiple messages can appear (using itemBuilder and a loop)
* Test to see if you can change the value of the message
* Change value of message based on message object
* Test to see if messages can appear on either side, depending on if you or someone else sent it
* Verify that the messages are appearing on the appropriate side – compare userID to message owner’s ID and see if they match

*Send a Message Field*

* Add message field at bottom of page
* Test to see if text can be typed
* Test to see if text that is typed is stored in a variable

*Send Message Button*

* Make button appear
* Test to see if button clicks (print in terminal)
* Test to see if button click successfully sends a message

**Poll Page:**

* Make question text appear
* Make option text and button appear
* Test to see if button clicks (print in terminal)
* Make options appear dynamically, test with various values of options (for example, 5 options, 10 options)
* Test to see if question text can be pulled from the poll object
* Test to see if options text can be pulled from the poll object
* Test to see if when option is clicked, no other options can be clicked
* Make text appear when option is clicked to indicate percentage
* Calculate the percentage of each option selected, display it when button clicked (test to see if value is correct)

**View Members Page:**

* Make member text and box appear
* Test to see if it appears
* Test to see if multiple members can appear
* Test to see if member usernames can be pulled from the group object and updates successfully
* Add text next to member to indicate member type
* Test to see if member type (member, admin) can be pulled from the user object and updates successfully

**Group Object Class:**

* *Declare Fields* 
  + int groupID
  + String name
  + int adminID
  + String description
* *Create Group Constructor with all fields and run class*
* *Testing*
  + Create a new group object in group\_home.dart by importing the group class
  + Print out the values placed in the fields for groupID, name, adminID, and description by using the format groupname.fieldname in the Visual Studio Code terminal
  + Pass group object to other test class from group\_home.dart
  + Print out group fields on app UI to ensure that groups can be passed between classes and accessed

**Message Object Class:**

* *Declare Fields*
  + User user
  + String time
  + String date
  + String content
* *Create Message Constructor with all fields and run class*
* *Testing*
  + Create a new message object in chat\_page.dart by importing the message class
  + Print out the values placed in the fields for user, time, date, and content by using the format messagename.fieldname in the Visual Studio Code terminal
  + Pass message object to other test class from chat\_page.dart
  + Print out message fields on app UI to ensure that messages can be passed between classes and accessed

**Poll Object Class:**

* *Declare Fields*
  + String question
  + bool active
  + var options
* *Create Poll Constructor with all fields and run class*
* *Testing*
  + Create a new poll object in poll\_page.dart by importing the poll class
  + Print out the values placed in the fields for question, active, and options by using the format pollname.fieldname in the Visual Studio Code terminal
  + Pass poll object to other test class from poll\_page.dart
  + Print out poll fields on app UI to ensure that polls can be passed between classes and accessed

**AppInfo Object Class:**

* *Declare Fields*
  + String username
  + String currentGroupCode
  + var groups
* *Create AppInfo Constructor with all fields and run class*
* *Testing*
  + Create an appInfo object in login\_page.dart by importing the AppInfo class
  + Set the username value in appInfo based on what
  + Pass appInfo object to home\_page.dart as a parameter when switching to the HomePage screen
  + Print out appInfo fields in HomePage to ensure that this object passes key information between classes
  + Repeat the above steps with signup\_page.dart instead of login\_page.dart

**Test Documentation – Backend**

**Authenticate Class:**

*Sign In*

* Make sure signing in with the correct credentials returns a user object
* Make sure signing in with incorrect credentials returns null
* Test to see that “Signed In Date” updates in the backend if the login was successful

*Sign Up*

* Make sure to not validate if email is already used
* Make sure to not validate if username is already used
* Make sure to validate the user if the email and username are original, and the password is complex enough
* Make sure the authentication details are added to the backend
* Make sure a new User document is created in the document for the new user
* Test to see that the backend updated with this new user after signing up

*Reset Password*

* Make sure to not follow through if not user is registered with that email
* Make sure users are locked out after requesting a rest too many times
* If conditions are satisfied, make sure a reset password email is sent to the user
* Test to see that the user receives the reset password email
* Test to see that the email is properly formatted

**Database Class:**

*Get User by Username*

* Make sure that the correct user document is returned
* Make sure that null is returned when that user doesn’t exist
* Test to see that the user can’t access a user that doesn’t belong to them
* Test to see that only one user is returned

*Get User by Email*

* Make sure that the correct user document is returned based on the email
* Make sure that null is returned when that email isn’t registered
* Test to see that the user can’t access a user that doesn’t belong to them
* Test to see that only one user is returned

*Get Group by Code*

* Make sure the correct document is returned based on the group code
* Make sure that no documents are returned if there aren’t any matching groups
* Test to see that the user can’t access groups that they don’t have the code for
* Test to see that only one group linked to each code

*Check Group of User*

* Make sure the correct document is returned based whether the user is in that group
* Make sure that no documents are returned if the user isn’t in that group
* Test to see that the user can’t access groups that they aren’t in
* Test to see that only the requested group is searched for, not other groups

*Get Group of User by Id*

* Make sure the correct document is returned based on whether the user is in a group matching the group code
* Make sure that no documents are returned if there aren’t any matching groups
* Test to see that the user can’t access groups that they aren’t in
* Test to see that only one group is returned if the user is in one

*Get Groups of User by Id*

* Make sure the correct documents are returned based on what groups the user is in
* Make sure that no documents are returned if the user is not in any groups
* Test to see that the user can’t access groups that they aren’t in
* Test to see that every group the user is in is returned

*Post Group to User*

* Make sure that the user isn’t already in the group
* Make sure that the group already exists
* Make sure that the appropriate location in the database is updated with the new group when these conditions are met
* Test to see that the user can’t join groups they are already in
* Test to see that the user can’t join groups that don’t already exist
* Test to see that the user can join existing groups that they aren’t a part of
* Test to see that the database updates after the user joins the group

*Post User Details*

* Make sure that no account is registered with that username
* Make sure that no account is registered with that email
* Make sure that the appropriate location in the database is updated with the new user when these conditions are met
* Test to see that the user can’t take a username or email that is taken
* Test to see that the details are posted when both the username and email are available

*Get Poll*

* Make sure that group exists before trying to get a poll from that group
* Make sure to see that there is a poll in the group that matches the given number
* Make sure that the entire map associated with a valid poll is returned if these conditions are met
* Test to see that the user can’t get the poll from a group they don’t have access to
* Test to see that no poll is returned if the requested poll doesn’t exist
* Test to see that user can get the poll if all conditions are valid
* Test to see that the map contains all of the needed information: question, creationOrder, answers, votes, whether the poll is currently active

*Post Poll to Group*

* Make sure that the group already exists
* Make sure that the user has permission to post polls
* Make sure that there is not duplicate poll
* Make sure the poll has the correct fields: question, answers, whether it is active, and its creation order
* Make sure that poll collection in the group is updated in the database
* Test to see that the user can’t posts polls to groups they don’t have access to
* Test to see that non-admin user can’t post a poll
* Test to see that the poll was updated in the database
* Test to see that other users in the group have access to the newly created poll

*Update Answer in Poll*

* Make sure that the user is part of the group
* Make sure that the poll is active before being able to vote
* Make sure that the poll/answer exists
* Make sure the vote is changed in the backend when this action is executed correctly
* Test to see that the user can update their vote even if they haven’t voted in a poll yet
* Test to see that the user can’t update their vote on polls that are inactive
* Test to see that the user cannot double vote
* Test to see that the updated vote is visible to other members of the group