Mikel Bresko
Zephren de la Cerda
Andrew Strong

Multipoll

Features: The main features of this app include an account creation so you can add users to your friend list and then to your group. Once a group is made, you can send messages or create polls to send to your friends. After all the votes are in, results will be displayed to all group members within the group chat. When creating a poll, you are prompted to choose a category, items to be voted on, and which group to send the poll to.

Checkpoint 1. (Blueprints)

- Researching which tools to use (Firebase)
- Deciding which pages need to be dynamic
- Setting up fragments
- Setting up git repository
- Project skeleton

Checkpoint 2.

- Start Database setup
- Link pages
- Start functionality for groups
- Start functionality for poll creation
- Start results page

Checkpoint 3.

- Friends layout + some functionality
- Most UI done
- Formatting layers, views, and buttons (on each page)
- Information inputted into database

Checkpoint 4.

- Finish functionality for friends list/groups
- Finish functionality for poll creation
- Real time results page
- Get single user chat implementation

Checkpoint 5.

- Finish group chat

- Group/Poll Notifications

Checkpoint 1

Zephren

Page 1:
- 1 table layout
    - 2 text edits (username, password)
    - 2 rectangles for text input
    - 1 button (signup/signin)
    - 1 image view (app logo)

Page 3:
- 1 table layouts
    - 3 static buttons (account info, settings, create poll)
    - 2 text views (page title, notifications)
    - 1 image view (app logo)
- 1 linear layout
    - 2 dynamic buttons/views (recent groups/polls)
    - 1 slide button for screen transition

Page 4:
- 1 linear layout
    - 2 static buttons (friends list, create new group)
    - 1 dynamic button (group list)
    - 1 slide button for screen transition

Investigate FireBase

Pros:
- Single API with real time syncing (promotes users to collab with another)
- Ships with mobile SDKs (no need for servers)
- Local cache when users go offline (automatically synced)
- Integrates with Firebase Authentication to provide simple authentication process (email & pass, phone numbers, google, fb, twitter, etc)

Cons:
- Server-less means more code in mobile client (more code)
- Not free to use (for long term use)

- Uses JSON over SQL

What needs to be dynamic?

1. Notifications, groups, and friends list

Mikel:

Page 6 (Dynamic):

- LinearLayout of buttons
- ‘Sdd category’ button.

Page 9 (Dynamic):

- LinearLayout (no buttons) side by side to a
- LinearLayout of checkboxes
- Checkmark button in the top right leading to pg. 10.

Page 10 (Dynamic):

- Virtually a copy of page 9
- Search bar at the top to search for friends.

MongDB super simple, but not ideal for such small scale project.

Andrew:

Pages 2, 5, 11

Investigate chat plug-in (firebase DB)

What needs to be dynamic?

(Groups, chat, poll options, main page(maybe), poll categories, poll options, add friend to poll, friends list)

Page 2 (dynamic):

- Weighted Linearlayout
- Database integration
- Edit text search bar
- Dynamic buttons

Page 5 (dynamic)

- 1 Relative Layout
- Dynamic options(Checkbox)
- 1 button
- Possible outside link

Page 11 (dynamic)

- 2 Linear Layouts
- 1 Relative Layout
- Chat/database integration
- Four buttons
- 1 EditText
- Dynamic buttons
- Dynamic textview

MVC Architecture

Model

- Groups.java - One of two main user pages -> Shows groups to send polls to
- Friends.java - Dynamic friends list with search bar
- GroupSelected.java - Chosen group (dynamic)
- MainAcivity.java - Welcome screen, contains controller object
- Poll.java - Class to make poll

View

- activity_main.xml - view of main page
- activity_poll.xml - view of poll creation page
- activity_groups.xml - view of group creation page
- fragment_friends.xml - view of friends list page
-

Controller

- Controller.Java - Contains main java functions

Our current implementation should be considered as 10% of our final implementation because we not only have our basic layouts and pages planned out, but we have started building & connecting pages, even if there is little content so far.

Ideally, we will be spending less time planning the pages and more time debugging our implementation. Most of our pages our similar format so we're able to reuse fragments. We've also done research on what database to use, and Firebase real-time sync will be perfect for our app.

Implementation: 9

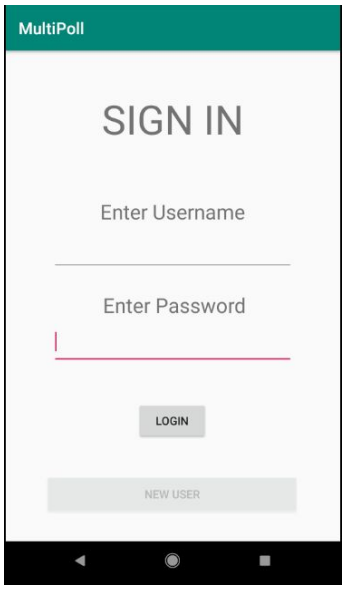
Coding standards & writeup: 9

Total: 18/24

==========

Checkpoint 2

We have a functional sign up and sign in page connected to the FireBase authentication. Once logged in, it takes you to the main activity. We have also fully implemented the model to store and handle our User, Group, and Poll objects. The main GroupSelected Page contains a list fragment of the group polls, which open poll options (also a fragment).



MVC Architecture

Model:

- User (name, email, password, groups)
    - New file that adds class to store user info to later add into database.
- Group (members, polls)
    - New file that adds class to store Group data.
- Poll (name, options)
    - New file that adds class to store polls taken from database
- FireBase DB

View:

- Acitivty_main.xml
- Activity_choosecategory.xml
    - Added displayed categories
- Actiivty_groups.xml
    - Added displayed, clickable groups
- Activity_groupselected.xml
- Activity_polloptions.xml
    - Creation
- Activity_selectfriends.xml
- Activity_signup.xml
    - Added signup page to deal with users without account.

- Activity_signin.xml
    - Added signIn page to allow users to log in and access app.
- Fragment poll_display.xml
    - Creation
- Fragment poll_results.xml
    - Creation
- ChooseCategory
    - Choose category when creating poll
- Groups (Other main page)
    - Show user's groups
- GroupSelected (Clicked on a group - shows group polls and theoretically chat)
- PollDisplay
- PollResults

Controller:

- FireBase Authentication
    - Firebase holds user login info with email, password, and username. Sets a current user for app.
- Main Activity (Home Screen)
    - Main Activity changed to default to Sign In screen when app starts
- SignUp / SignIn
    - Created two new Controllers with new views that interact with Firebase Authentication. They add user info into the database and switch to main page on successful login.
- Displays poll category data
    - Poll category data is pulled from our main controller into the respective pages

Our current implementation should count as 30% of our project because

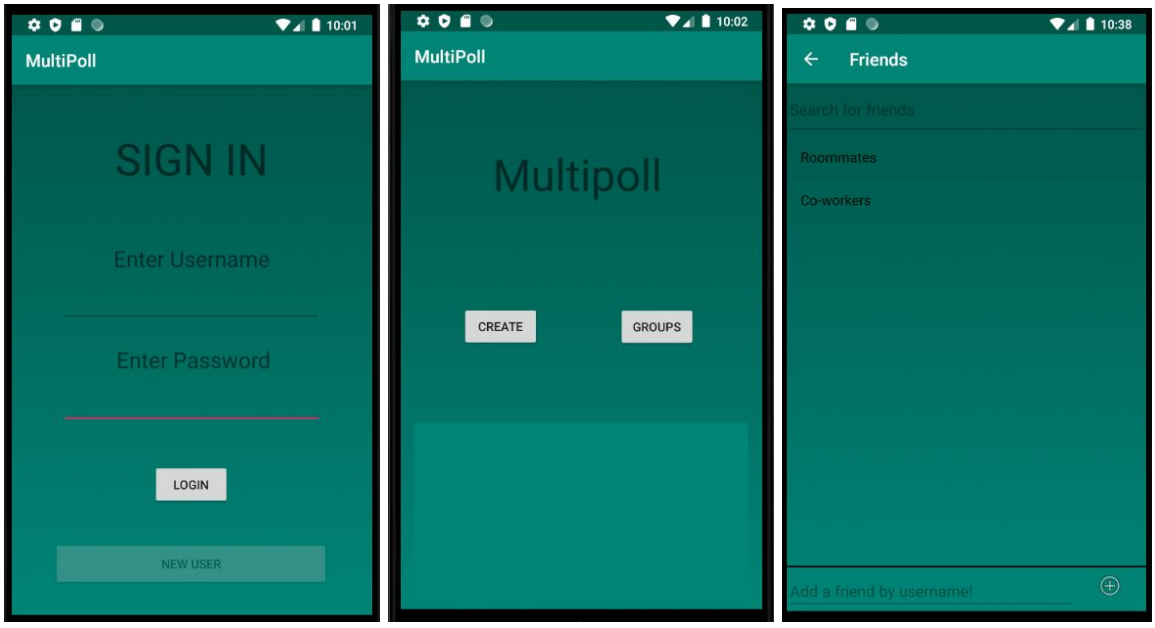
- We store all our data inside the database and have user verification
- Database is ready to be implemented to store user info
- We have (most/all) of our pages UI done
- Successfully implemented fragments
- Created structures to hold data taken from database

Expected Score:

- Implementation: 17/20 - Lacking some poll settings functionality
- Coding Standard: 5/6
- Write-Up: 3.5/3.5

==========

Checkpoint 3



MVC Architecture

Model:

- User (name, email, password, groups)
    - File that adds class to store user info to later add into the database.
- Group (members, polls)
- Poll (name, options)
- Category
    - Class that tracks user categories
- Element
    - Class that tracks name, description, and category
- Controller
    - Background controller for dynamic layouts and objects
- FireBase DB
    - Database can now be populated
- Firebase Controller
    - Object to write and read from database

View:

- Acitivty_main.xml
    - Looks much nicer
- Activity_choosecategory.xml
    - Added displayed categories
- Actiivty_groups.xml
    - Added displayed, clickable groups
- Activity_groupselected.xml
- Activity_new_group.xml
    - Created page to add new groups
- Activity_polloptions.xml
    - Creation
- Activity_selectfriends.xml
- Friends.xml
    - New page that contains list of friends
- Activity_new_friends.xml
    - Created page to add friends
- Activity_signup.xml
    - Now adds user to the database
- Activity_signin.xml
- Poll_results.xml
- Fragment poll_display.xml
- Choose_category.xml
    - Now has search function and toolbar
- Choose_groups.xml
    - Listview of groups user has joined
- Choose_elements.xml
    - Listview of elements associated with user account and category
- Groups (Other main page)
    - Now has search function and toolbar
- GroupSelected (Clicked on a group - shows group polls and theoretically chat)
- PollDisplay
- PollResults

Controller:

- FireBase Authentication
- Main Activity (Home Screen)
- SignUp / SignIn
- Displays poll category data
- Friends.java
    - Controls dynamic friends list

- Groups.java

    - Controls dynamic group list with search function
- Newfriend.java
    - New file to add friend
- NewGroup.java
    - New file to add group and add to db

Our current implementation should count as 50% of our project because

- We have a fully functional UI
- The inputted data is stored in our database
- We have implemented icons and toolbars
- We have a more beautiful design in place
- Concrete updated structure to manage database to app connection

Expected score

- Implementation: 20/20
- Coding Standard: 6/6
- Write-Up: 6/6

Bugs and Issues

- Retrieving from database giving only null values