**Virtual Hangout Scheduler Product Spec**

By: Viraj Goyal, Mikey Halim, Adam Melamed, Deepayan Sanyal

**Glossary**

* Flutter: A mobile UI Framework
* Dart: An Object-Oriented Programming Language
* Visual Studio Code: An IDE tool for creating computer programs, websites, and apps.
* GitKraken: A powerful and elegant multiplatform graphical interface for Git
* GitHub Classroom: A tool to simplify the educational use of GitHub
* Netflix: A website that lets you watch movies and television shows with a monthly subscription.
* Netflix Developer Kit: An open source software center for Netflix
* Firebase Back End Platform: A Google Platform that helps with backend services on apps
* Flask Web Framework: A microweb framework written in Python
* AWS Elastic Beanstalk: An Amazon Web Service that lets us deploy apps in the AWS Cloud
* Firebase: Google’s Platform for mobile and web app developers, especially useful for authentication.
* Firestore: A horizontally-scaling, document-based, NoSQL database framework

**Introduction**

As the COVID-19 pandemic forces people to stay in, people are turning more towards Netflix and away from traditional movie theaters. Netflix presents a new set of problems for everyone as people are having trouble choosing which movies to watch with their friends. With our app, people will be able to communicate through text and use polls to figure out what to watch and when to meet up. Our customers will be presented with a fast, easy and effortless way to choose the best movies to watch and the best dates to watch them. The trouble of scrolling through hours of movies to decide on one to watch will cease to exist as our app becomes widespread.

**General Context**

Purpose

* The purpose of our app is to provide our customers with a fast, simple, and effortless way to communicate with friends to create a virtual hangout. Our app helps provide users with quality services that allow them to choose the best movies to watch and the best dates to watch them. Together, users can decide which movies and dates best fit them and efficiently schedule a virtual hangout meeting together.

Intended Audience

* Males and females aged 15-30 that are interested in watching movies with their friends but have no virtual option to do so.

**Timeline/Test Cases**

* By October 2nd: Complete setup, everyone downloads flutter and other necessities
  + General Test Cases:
    - TC01 – Flutter – Successfully Set-up Environment
      * Windows:
        + Preconditions: Install Flutter for Visual Studio Code on your Windows 7 SP1 or later device per the following [guide](https://flutter.dev/docs/get-started/install/wguideindows).
        + Assumptions: Windows Powershell 5.0 and Git for Windows are already installed. Visual Studio Code is already installed.
        + Test Steps:

Open Visual Studio Code

Go to the following panel: *View > Command Pallete...*

Type “doctor”, then click on “Flutter: Run Flutter Doctor”

* + - * + Expected Results:

There should be no “issues” in the *OUTPUT* plane. If there are, work to resolve the problem until the test-case is passed.

* + - * MacOS:
        + Preconditions: Install Flutter for Visual Studio on your macOS (64-bit) device per the following [guide](https://flutter.dev/docs%20guide/get-started/install/gmacos).
        + Assumptions: The following command-line tools are available *bash, curl, git, mkdir, rm, unzip, which, zip*
        + Test Steps:

Open Visual Studio Code

Go to the following panel: *View > Command Pallete...*

Type “doctor”, then click on “Flutter: Run Flutter Doctor”

* + - * + Expected Results:

There should be no “issues” in the *OUTPUT* plane. If there are, work to resolve the problem until the test-case is passed.

* By October 5th: Login/Logout and settings
  + General Test Cases:
    - TC02 – Firebase – Authenticate Successfully
      * Preconditions: User already registered an account for our app
      * Assumptions: User is on a compatible device
      * Test Steps:
        1. Navigate to the sign-in page
        2. Choose a sign-in method (in this case Google Account)
        3. Enter the email and password, then click “Next”
        4. Click “Sign-In”
      * Expected Results:
        + The main page of the app should be displayed, where the user is greeted with their name.
* By October 12th: Add group and join group features, connecting with other users
  + General Test Cases:
    - TC03 – Main App – Join Group
      * Preconditions: A separate user has already created a group on the app
      * Assumptions: The user in question is already signed-into the app.
      * Test Steps:
        1. The other user shares an invite link to the group from the app
        2. The user in question clicks on the link given by the other user
        3. The user accepts the invitation to join the group
      * Expected Results:
        + The user becomes a full member of the group.
* By October 19th: Poll Channel, Bot, movie choice and admin functionality added, the main point of our project
  + General Test Cases:
    - TC04 – Main App – Join Group
      * Preconditions: The user is already in a group in the app
      * Assumptions: The user already has the app open
      * Test Steps:
        1. Navigate to the “Polling” Channel
        2. Press the “Create a Poll” button
        3. Type in as many options for the poll as needed
        4. Press the “Post” Button
      * Expected Results:
        + As other people answer the pull, the results of the poll will update and a pie chart will be displayed accordingly.
* By October 26th: Presentation is prepared. Any bugs are taken care of and any extra features are added.

*Use Cases*

* When friends want to meet up with each other to watch a movie
* Another use case would be a group of friends (young adults), geographically dispersed, trying to organize a movie night.

*Goals/Non-Goals*

* Goals: to allow safe and secure login, to link up with fellow users and create a channel, to chat with other users, to decide on a movie based on a fair and automated process, to decide on a date based on a fair and automated process, to allow safe and easy access in and around the app
* Non-Goals: to not be overly complicated, to not have an unfair process that gives too much power to one user or one item, to not be too slow as a program, to not be a sensory overload upon the user, to not be overtly political as to not lose any users based on world views

Budget: $20

Resources/Materials we will employ:

* Visual Studio Code latest version
* Flutter Mobile UI Framework (Dart programming language) - version 1.19.0-4.3.pre
* Provider – Flutter Library
* Python 3.6 or later
* GitKraken or GitHub Desktop
* WhatsApp for group communication
* GitHub classroom
* Microsoft PowerPoint for presentation
* Flask web framework for Python
* Firebase backend platform
* Netflix Developer Kit
* AWS Elastic Beanstalk

**Requirements**

*Functional Requirements*

* Needs to work on mobile app (iOS and Android)
* Clearly commented code
* App can be built on Google Chrome for testing on laptop
* Code is compartmentalized
* Dependencies are clearly listed in pubspec.yaml file, so others can download items necessary to run app
* Secure database stores user information
* Client Side communicates with Server Side (Amazon Web Server)
* Maximum efficiency of code
* Users need to be able to delete data, following European GDPR guidelines

*Non-functional Requirements*

* Minimalistic design
* Little to no animation
* Modern/non-distracting color scheme
* Easy user experience

**Deliverables**

Virtual Hangout Scheduler Mobile Application (can be installed on both Android and iOS)

*Features:*

1. Signup and User Login Feature
2. Create Add Group Feature
3. Generate Join Group Feature
4. Account Settings Page
5. Logout functionality
6. Group Chat Channel
7. Poll Channel with Bot asking questions
8. Add chosen movie and time to member’s calendar
9. Group Admin functionality – can decide when to start polling/which polls to release into the poll channel
10. Create user message box that contains invites from other users to join certain groups.

If time permits:

1. Forget Password Feature
2. More attractive UI
3. Admin can create more channels
4. Greater selection of entertainment choices (can select movies from other platforms like Amazon Prime)

**Pseudocode** (outline)

Flutter (dart) frontend:

home\_page.dart: User homepage. There will be a button in the middle, which will allow the user to join a group. In addition, there will be a button on the top right, allowing the user to create a new group and send invites to other users based on their name and tag (like how Discord sends friend requests).

login\_page.dart: First page that the user encounters. User will be able to login by typing in a valid username and its respective password (one username and one password field will be displayed clearly). If user does not have an account, they can sign up by clicking the signup button that will be below the login button. Furthermore, we will have our app logo displayed 50 pixels above the username field.

main.dart: Contains routes to navigate from one page to another. In addition, returns a MaterialApp object that has a title and theme.

signup\_page.dart: Once user clicks signup button on login page, they are redirected to the signup page. The signup page contains multiple text fields such as name, email, state, country, username, and password. At the bottom of the list of text fields, there is a signup MaterialButton object that can be clicked to create the user (if all fields are filled out). Once the new user is created, the person using the app is redirected to the login page and can login to the user homepage of the app using the credentials they just made.

authenticate.dart: Contains multiple methods such as createUser(), loginUser(), logout(), getUser(), updateUser() to process user requests (all of these are callback functions).

http\_requests.dart: Methods from authenticate.dart call methods from http\_requests.dart. In http\_requests.dart, there are methods containing http post, get, and put requests sent to the backend. These requests return a value, which is then returned by the corresponding method to authenticate.dart. Authenticate.dart uses the returned value to determine what to do within the app.

user.dart: Class containing the definition of a user object. The user class has private fields such as email, name, state, country, username, but not password. There is a constructor within the user class, so new users can be made in other frontend classes.

account.dart: Class that creates the account settings page. In this class, there are multiple text fields such as email, name, state, country, and username. Whichever field you want to update, you can write in the new input into that field and press the update button on the bottom of the page to update your account information.

settings.dart: Generates the regular settings page. On this page, you can change the theme (light or dark), your password, or delete your account.

account\_drawer.dart: Returns a Drawer object with a ListView child. This is the drawer that pops up on the left side of your screen if you hover your mouse to the left on the user homepage. You see your name, state, and country on the top of the drawer. Towards the middle of the drawer, you can see an account settings button, regular settings button, and logout button. If you click the account settings button, you will navigate to the account settings page and a similar action occurs for the other buttons as well. Towards the bottom of the drawer, you can see a messages button. Clicking the messages button will navigate you to your inbox where you can see which groups you’ve been invited to.

group\_drawer.dart: Returns a Drawer object with a ListView child. This is the drawer that pops up on the right side of your screen if you hover your mouse to the right on the user homepage. This drawer contains all the groups you are in. You can see the name of the group and its profile picture. You can click any group in that drawer and navigate to that group’s homepage.

create\_group.dart: This is the class that will create the page that pops up when a user presses the create group button on the user homepage. The create group page will feature a Group Name field, a place where you can select a group profile picture, and an invite column. Within the invite column, you can add users to send a group invite to. The group admin (the person who created the group) can add users to the invite column by typing their name and #number (e.g. ViVi#5067). On the bottom of the create group page, there will be a create group button. Once the group admin clicks that button, a new group will be created, and the group admin will be directed to the homepage of that group.

bot.dart: Operating much like a discord bot, this class will contain code that allows bots to place polls into the group channel called “poll”. In this channel, users will be able to select which genre and movie they want to watch with others.

create\_poll.dart: The Group Admin will have a button on the poll channel to create a new poll. Once they click that button, the bot will generate and place the poll on the “poll” channel where all group members can interact with the poll.

message\_inbox.dart: This class will generate a page that contains any invites a user has received from group admins. A user will be able to click an invite and automatically join the group. Once they have successfully joined the group, they will be redirected to that group’s homepage.

groupHome.dart: This class will generate the group homepage. On the left side of the homepage, a regular user will be able to see a channel called “General” where all people within that group can chat with each other. In addition, a regular user will be able to click on the “Poll” channel to navigate to the polling channel (like Microsoft Teams channels). Once on the poll channel, a regular user will be able to vote for the choices they desire.

Python backend (To Be Finalized):

interface.py:

app.py:

server\_test\_suite.py:

tablecreation.py:

firebase.py:

bot.py:

send\_message.py:

AWS Backend (To Be Finalized):

*This will be done in Elastic Beanstalk*

.ebignore: This file will keep our source bundle minimally sized

Resources to create:

* EC2 instance
* Instance security group
* Load balancer
* Load balancer security group
* Auto scaling group
* Amazon S3 bucket
* Amazon CloudWatch alarms

**Mock UI’s**

A picture containing graphical user interface

Description automatically generatedA picture containing chart, treemap chart

Description automatically generatedA picture containing graphical user interface

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

**FAQ**

* What language are you coding your app in?
  + We are using python and dart (in conjunction with the Flutter mobile UI framework) to write the code for our app.
* What are you using to build the UI?
  + We are using the Flutter mobile app framework to develop the UI.
* Which movies are supported by your app?
  + A lot of movies and genres that are supported by Netflix are also supported by our app.
* Which platforms is your app compatible with?
  + iOS and Android.
* How many people can a single group allow?
  + There can be up to fifty people added to one group.
* How do you ensure user security/privacy?
  + User security/privacy will be ensured using SSL security protocols.
* When is the estimated release date of the app?
  + The estimated release date is October 26th.
* How will group invites be sent to other users?
  + Invites will be sent through the app itself and show up as a notification in a user’s message box.