Final Report
Theatr: The Movie Finder

Austin Hwa, Katie Kemp, Sahir Mody
CMSC436
May 2, 2022

Overview

The motivation behind our app is that it can be difficult to choose a movie due to the distractions and complicated structure of current websites like Fandango. Our app takes away the effort of finding a website to search for movies, putting in search criteria, and searching through a cluttered UI. All you have to do is open the app, pull to refresh, and, voila, you have a list of the movies showing in your area. It's also much more pleasing to the eye to have the display of the movies as we do on the app with all of the important information upfront and irrelevant information hidden.

At a high level, our app lists movies based on the user's location and provides showings at nearby theaters that are upcoming. The user starts a new voting session, then votes. Next, they have the option to pass the phone to another user so they can vote, or end the session. This continues until they decide to end the session and then they have the option to save the results. Lastly, they are brought to the results page which shows past results with the most recent at the top and has an option to start a new session. It also allows the user to quickly purchase tickets by providing a link to the corresponding purchase page once the results have been tabulated.

Our app makes use of the LocationManager as a mobile-specific feature and AMC API calls and share sheets as features that were not discussed in class.

Goals

For Milestone 1 we accomplished three goals: build a basic homepage, add a text box for sessions, and create a structure for a list of movies. The goal for the basic homepage was to welcome people to our app, although it would be later updated to

include information about movies. The goal of the text box was to allow the user to input their session id number, although at this point it was not functional because implementing session ids was a goal for later milestones. The goal for creating a basic structure for displaying movies was to hard code sample movies but create a VStack so that we could see a list of movies. Adding real data would be a goal for later milestones.

For Milestone 2, we originally had the goal of implementing session ids but soon realized that this would be out of the scope of this class. Instead, our main goal was to get location data from GPS. For this goal, we simply printed out the latitude and longitude coordinates after getting them from the LocationManager, and in later milestones, we made use of this data.

For Milestone 3 we added API calls to get actual movie data from AMC. This proved to be more difficult than we expected because implementing GET requests in Swift is much more complex than in other languages we have used. We also added a pull to refresh feature so that the API call uses the most recent information and the view then updates.

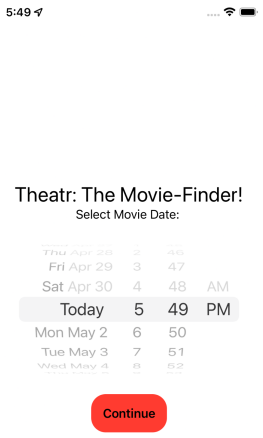
For the final submission, the remaining goals included decoding API calls, voting, date and time selection, and improved styling for different orientations, etc, looking into whether any of our stretch goals could realistically be implemented, and finally, submitting the project. Decoding the JSON return by the GET requests is not a simple task in Swift so we needed more time to complete this goal after milestone 3. Up to this point, the styling of our app was very basic, so improvements needed to be made. We added voting to the app and navigation through the process, and we added a DatePicker so that movies could be selected at any point in the future.

Stretch goals for our app included links to purchasing tickets, movie previews in-app, chat and commenting features, recommendations based on history, and collecting profile data. After milestone 2, session IDs also became a stretch goal. The only stretch goal we were able to implement was linking to ticket purchases.

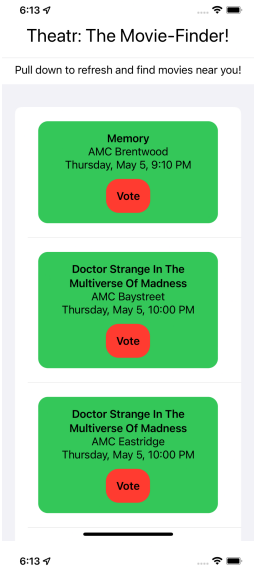
User Interactions

1. The home page of the app is visible when you open the app. It has a single button to allow the user to begin their first session on the app.

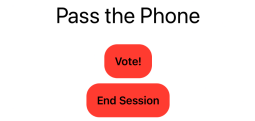
2. After a session is started, a DatePicker appears so that the user can select which day they want to see the movie. There is a button to continue to the voting screen that is labeled “continue” that allows the user to proceed to voting once they have selected the desired date.



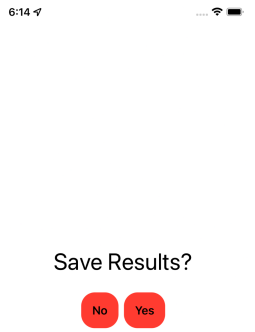
3. The voting screen appears with a list of movies. The movies are automatically loaded based on the user's location, and the user can pull the screen down to refresh in case new movies are added. Each movie has a corresponding "vote" button, and once it is clicked the user's vote is recorded.



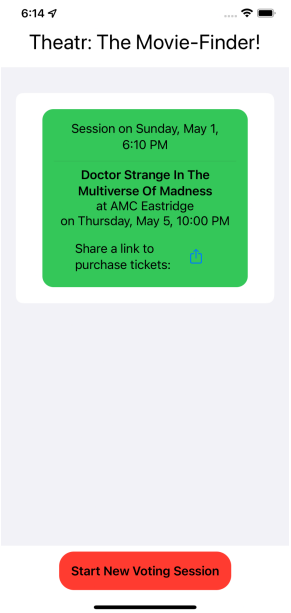
4. After a user's vote is recorded, a screen pops up asking the user to pass the phone to the next voter. When the next voter has the app in hand, they can click "vote," or if this is the last person to vote, then can click "end session." If they click "vote", the screen returns to the voting page as described in 3. If they click "end session" they proceed to the page described in 5.



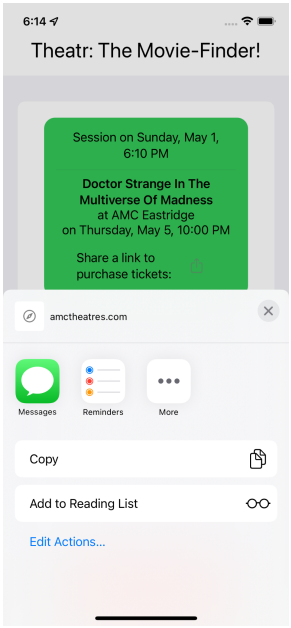
5. A screen pops up prompting the user to save results (or not). Most of the time, a user probably wants to save the results, but if they accidentally made the session or decided not to go to the movies, it might not be worth saving the result. Once they select the option, they continue to the results page described in 6.



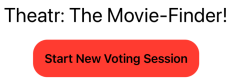
6. This screen shows a list of results with the time the session was conducted, the movie that won, and a button to share the ticket purchase link for the winning movie over text, or other means. There is also a button to start the next session which will bring the user back to the DatePicker described in 2.

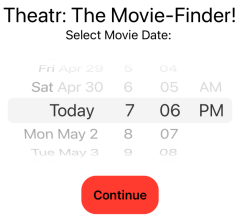


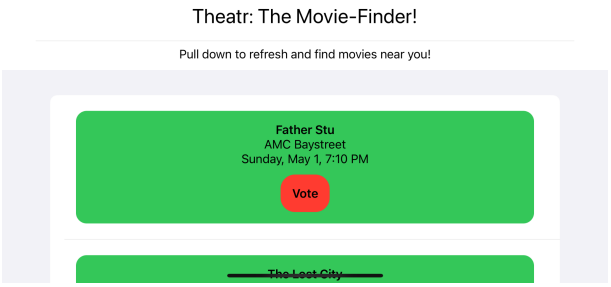
7. Clicking the share sheet button brings you to a standard menu that allows you to share the information with your contacts. As you can see, the link is to amctheatres.com, but when it is actually opened the page will be the corresponding one to the selected movie.

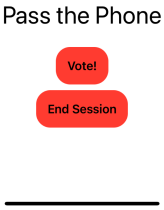


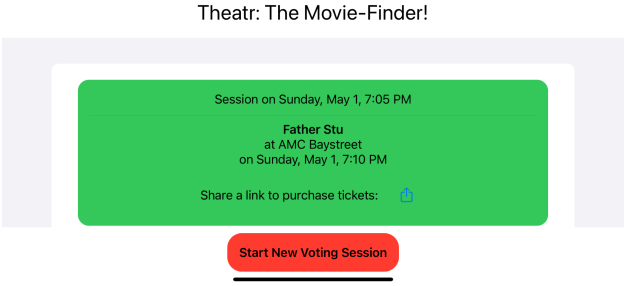
8. Each screen of our app is aesthetically pleasing in both the portrait and landscape orientations.











Development Process

For our design process, we met weekly, assessed the previous week's work, and continued to build on it. If we hadn't met our goals, we re-assessed them. For example, this happened when we had difficulty setting up the API. We had to wait on API permissions and then decode the API response, which proved to be more difficult than expected. We learned a lot about how the development process can take time and that there are often unexpected issues.

One feature not used in class that we learned a lot about was using external APIs in our application. Although URLSession was briefly discussed in class,

implementing it with an API call was actually more complicated than expected. We learned how to effectively use external APIs in Swift and how it can be difficult to parse JSON responses in Swift because we have to create a struct for each individual field or category in each section of the returned JSON, essentially having to build out the structure laid out in the JSON. As a result, that process proved too lengthy due to AMC's intricate JSON layout, with numerous nested categories that needed to be individually coded, whereas other languages have a different and easier approach to decoding JSON files. We also had to learn about Swift's libraries for handling HTTP requests, as well as setting request headers.

Another new feature we used was UIKit's share sheet. This allows the user to share things from the app to a text, email, or another app. This was perfect for our app because sharing the movie ticket link makes it even more useful. This feature was not covered in class and was very different from other features we had used that were part of SwiftUI, so we had to use external reference sources like Stack Overflow and Official Apple iOS Documentation to learn how to implement it correctly and effectively. While understanding the actual code was not too difficult, we found that when running our app on the XCode simulator, the share sheet would not actually work. We had to do more research to realize that this was not the result of a permission issue or a mistake in our code, but rather a feature of the simulator and that it only works when used on a physical device running iOS. This led us to testing our app on our own phones, which was a very cool experience and helped us confirm that the functionality we needed was in fact there.

Another aspect of development we learned about was refactoring code. As we added more features to our app, such as voting and then date selection, navigation became more complex. Originally we were using boolean state variables in the view itself to keep track of what part of the session we were in, but as we needed more information this led to many nested if…else statements, and it became more straight-forward to implement a state variable in our model that could be checked in the view. This way, our app can be represented with a simple state diagram with 6 different states.

Future Directions

A big idea for our app that we did not have a chance to implement was the creation of group sessions. Essentially, one user would be able to start a session that other users could join, and they would all be given the same list of movies that they could then vote on. This would help a group of friends easily pick out a movie that a majority of them agree upon in a quick and easy way. This would further streamline the process of selecting a movie because the group wouldn't have to be together in person in order to vote.

Another potential expansion of the app could be to incorporate more movie location APIs. In our project, we limited ourselves to the AMC API to have a working proof of concept. By adding more APIs, such as the Regal API or the Fandango API, we could potentially get more movie listings and more data about locations and prices. This would help users buy tickets for the cheapest showtime more efficiently because they could do it all in the app with several theater options and locations to choose from. In

fact, building on this, we could even incorporate a payment API like Stripe to create a one-stop-shop for our users where they can see what movies are playing nearby, pick a movie with their friends, and buy tickets as a group, all in one convenient app.

Other stretch goals we were unable to implement were movie previews in-app, chat and commenting features, recommendations based on history, and collecting profile data. Handling the files to preview movies in the app would be complex and they are not provided by AMC so we would have to turn to another API for this. Chat features, commenting features, profile data, and recommendations based on that profile data could all be implemented with a backend database like Firebase, but it would be extremely complex to keep track of all of this data so there was not enough time to implement it for this project.