OULUN AMMATTIKORKEAKOULU

Joel Ala-aho, Ossi Karjalainen

DEVELOPMENT OF A FLUTTER MOBILE APPLICATION USING SPOTIFY'S APPLICATION PROGRAMMING INTERFACE

DEVELOPMENT OF A FLUTTER MOBILE APPLICATION USING SPOTIFY'S APPLICATION PROGRAMMING INTERFACE

Joel Ala-aho, Ossi Karjalainen
Bachelor's Thesis
Spring 2022
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Option of Web Development

Authors: Joel Ala-aho and Ossi Karjalainen
Title of thesis: Development of a Flutter Mobile Application Using Spotify's Application Programming Interface
Supervisor: Teemu Korpela
Term and year when the thesis was submitted: Spring 2022
Pages: 14

This thesis followed the development process of a cross-platform mobile application in the form of personal weekly video diary entries. Each entry describes the weekly goals, finished tasks, current problems and following week's plans.

The main aim of the thesis was to develop a music application for mobile devices that utilizes Spotify's application programming interface and enhances existing features in Spotify's own mobile application. The secondary aim for the thesis was to improve skills in mobile application development.

The user interface of the application was designed using Adobe XD. The Flutter framework was used to develop the front-end for the application. Authentication and data were handled by Spotify's Web API. Google Firebase's Cloud Firestore was used for a temporary storage.

The results for the project were successful. The application was able to meet the required specifications. Development is still needed to apply for Spotify's release license. The authors continue developing the application because they want the application to end up in both Google's Play Store and Apple's App store.

Keywords: Flutter, Application Programming Interface, API, Framework, Google Firebase, Front-end development, Back-end development, Mobile development

CONTENTS

1 INTRODUCTION

The objective of this thesis was to develop a cross-platform mobile application to enhance Spotify's (1) functions. The application is used to create play sessions that share Spotify tracks between users and to provide user-friendly ways to create Spotify playlists, such as 'Song Fly', where the user must guess the track from a 30-second preview (figure 1). Play sessions are hosted by a single user that controls the playback of tracks. Other users send tracks to the session, which are sent to the host's Spotify playback track queue.

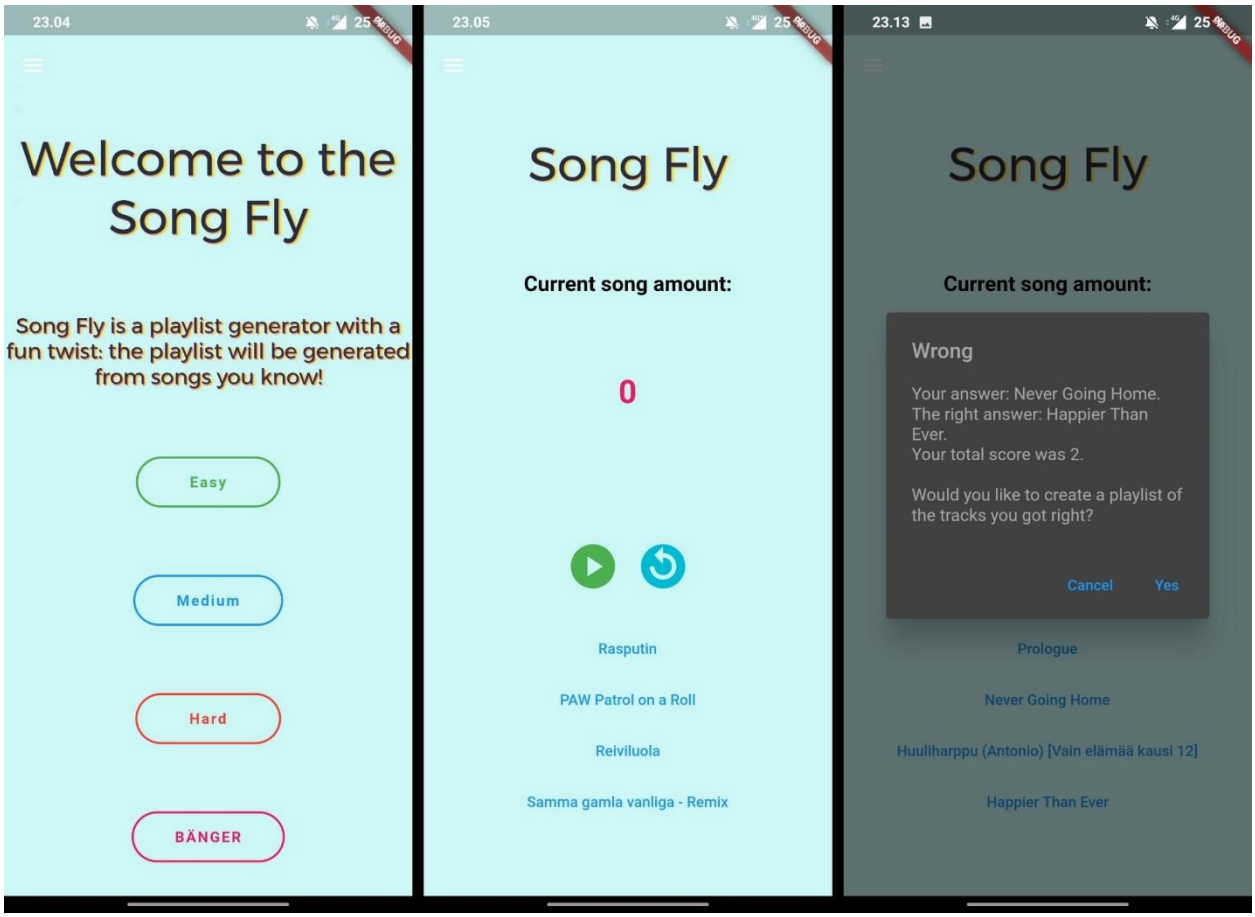

FIGURE 1. 'Song Fly'-mode on a real device.

2 WORKING METHODS

The thesis work started with an introduction video where the authors explained about tools, working methods and idea for the project that they would develop. In the introduction it was discussed that the thesis would include eight weekly updates and a final video where the authors would conclude everything together. Six weeks in, the authors also published an update video where they talked about major changes that the application would go through. All the videos can be requested from the thesis supervisor Teemu Korpela. Source code is not available for public use.

The authors chose to use Trello (2) as their backlog. Each author was assigned several tasks for each week, depending on the expected difficulty of the tasks. Weekly meetings were decided to be held on each week's Monday, where the authors would go through objectives for the week. Version control rules were set at the beginning of the project through a template for pull requests and branching.

The development work was set to be done with Visual Studio Code (3), Android Studio (4) and Xcode (5). Adobe XD free (6) was set to be used as a user interface designing software. For designing data modeling the authors used UMLet (7). The version control was handled with a private GitHub (8) repository. For back-end data the authors used Google's Firebase Cloud Firestore (9).

During development, issues were faced in both front- and back-end. Since modern mobile operating systems have introduced strict rules on how applications are allowed to run in the background, the application had to be developed in a way that can evade these rules. This was achieved with Dart's (10) isolate-class, which is where the play sessions' database streams are run. Spotify's Developer Terms of Service (11) and Privacy Policy (12) had to be followed precisely to enable the application for an API quota extension. These terms and policies affected the application mostly on design and data storage.

The benefits of a cross-platform development framework were seen early in the development process. With no previous experience in iOS development, the application was quickly in a state where it could be run on such devices. The framework's packages and widgets were used to quicken the development of the application.

3 WEEKLY DIARY ENTRIES

In this section of the thesis, both authors explain briefly about their weekly video diary updates. Each entry is described in greater detail in the video diary entries.

3.1 Week 1

3.1.1 Joel's entry for Week 1

During the first week of the project, I created a base structure for the project. I also managed to authorize both the application and users through Spotify application programming interface. (13, 14, 15, 16, 17)

3.1.2 Ossi's entry for Week 1

In the first weekly video I went through the design process and discussed why we chose our colors and font for the application. I explained about spending the first week designing how the user interface would look like for the application. (18, 19)

3.2 Week 2

3.2.1 Joel's entry for Week 2

My second week was filled with troubleshooting since we faced issues in background processing and on running the application on Google Chrome. I tried to fix the Chrome issue with no success and thus we decided to drop the support for Chrome. I found solutions for the background processing issue. I managed to connect the application to Firebase and create a basic function to begin sending tracks into a Firebase collection. (20,21, 22, 23)

3.2.2 Ossi's entry for Week 2

In this weekly entry I went through the process of designing and implementing a drawer menu for the application. I also talked about designing the search page for the application. I showed my code for the drawer menu and went through how the code works. I talked about general problems that I faced when trying to implement my design. (24)

3.3 Week 3

3.3.1 Joel's entry for Week 3

The third week consisted of implementing an isolate solution to fix the background processing problem and developing a way to send tracks to other users' Spotify queues through Firestore. I also added basic group functions and renamed global files according to guidelines. (25)

3.3.2 Ossi's entry for Week 3

During week three I went through finalizing the design for the search page of the application. The week three was quite challenging as the search page was not as easy to build as I had thought. I was not able to build the search page based only on Flutter documentation, but I had to rely on a guide that used previous version of Flutter. (26)

3.4 Week 4

3.4.1 Joel's entry for Week 4

In the fourth diary entry I went through the development process of the "Party"-mode. I also described how I managed to fix an isolate token issue and what we planned together for the UI and states of the application.

3.4.2 Ossi's entry for Week 4

The weekly entry for the week number four was quite brief due to me being ill. I talked about continuing my work on the search page and went through designs that I polished in Adobe XD.

3.5 Week 5

3.5.1 Joel's entry for Week 5

During the fifth week, I started working on a new feature called "Song fly" by implementing main functions for it. Besides that, I had to make small fixes to "Party"-mode. We also faced new problems with Spotify's Developer Terms and Policies. (27)

3.5.2 Ossi’s entry for Week 5

In week 5 I was able to advance on the search page. I was able to connect our applications front-end to use Spotify’s endpoint correctly. Data retrieved from the endpoint was also correctly mapped into our application. The search page still was not fully functional as it lacked the feature to send the song to our Google Firebase. I also continued my weekly work on user interface components.

3.6 Week 6

3.6.1 Joel’s entry for Week 6

I started the sixth week by refactoring the “Song fly”-mode. During the week, I also developed a new feature called “Bänder”.

3.6.2 Ossi’s entry for Week 6

In week 6 I started my work on components that Joel had finished. We were missing reusable components and I wanted to start my work on them. I also talked about researching more about Spotify guidelines as I had misunderstood how Spotify as a brand wants to use their logo on third-party applications.

3.7 Week 7

3.7.1 Joel’s entry for Week 7

The seventh week consisted of implementing a way to track the Spotify queue, fixing plenty of bugs and refactoring the code to match our intentions. (28)

3.7.2 Ossi’s entry for Week 7

During the seventh week, I continued the work I had left from the week six. I spent my time on fixing the user interface to be in the form we wanted it to be.

3.8 Week 8

3.8.1 Joel’s entry for Week 8

During my last week of the project, I worked on features that had to be developed because of the Spotify Privacy Policy and their Developer Terms of Service. I continued the work on bug fixes and refactoring.

3.8.2 Ossi’s entry for Week 8

In the last week of the project, I spent my remaining time to finalize the application. I spent my time on researching Spotify’s design guidelines and making sure that all of our components were correct.

4 CONCLUSION

Both authors of this thesis improved in mobile application development. The project introduced the authors to modern and widely used technologies. Both authors had previous experience of working with Android applications and application programming interfaces, but iOS development and a completely new framework introduced welcoming challenges.

Working in a professional-level project where the goals and the ways of work were set by the two authors was both challenging and rewarding. Setting a goal to create an application that does not have an existing audience is a hard task. You must conclude a research work where you introduce your idea of an application to persons that have never heard of it. During the start of the thesis, the authors concluded a survey where the participants were asked about music applications. The survey had questions about features that music applications lack, about the applications that they use and where they listen to music. The research concluded to confirm the ideas what the authors had for the application.

The usage of a pre-built Web application programming interface brought along several downsides. The absence of an endpoint for users' track queues meant an additional work on the client. The documentation for the application programming interface was under construction during the development. Application programming interface requests resulted in enormous responses, which were not documented and thus were hard to parse into usable form.

Overall, the objectives set at the beginning of the project were achieved. The application is in a state where it meets the requirements of the specifications given for it. The authors gained valuable experience from working with modern tools and the ways of developing cross-platform mobile applications.

REFERENCES

1. Spotify Application, 2021. Date of retrieval: 12.11.2021. Available: https://www.spotify.com/us/download/other/

2. Trello, 2021. Date of retrieval: 12.11.2021. Available: https://www.trello.com

3. Visual Studio Code, 2021. Date of retrieval: 12.11.2021. Available: https://code.visualstudio.com/Download

4. Android Studio, 2021. Date of retrieval: 12.11.2021. Available: https://developer.android.com/studio

5. Xcode, 2021. Date of retrieval: 12.11.2021. Available: https://developer.apple.com/xcode/

6. Adobe XD free version, 2021. Date of retrieval: 12.11.2021. Available: https://helpx.adobe.com/xd/get-started.html

7. UMLet, 2021. Date of retrieval: 12.11.2021. Available: https://www.umlet.com/

8. GitHub, 2021. Date of retrieval: 12.11.2021. Available: https://github.com/

9. Firebase Cloud Firestore Documentation, 2021. Google. Date of retrieval: 12.11.2021. Available: https://firebase.google.com/docs/firestore

10. Dart documentation, 2021. Dart. Date of retrieval: 12.11.2021. Available: https://dart.dev/guides

11. Spotify Developer Terms, 2021. Spotify. Date of retrieval: 12.11.2021. Available: https://developer.spotify.com/terms/

12. Spotify Developer Policy, 2021. Spotify. Date of retrieval: 12.11.2021. Available: https://developer.spotify.com/policy/

13. Spotify Web API Documentation, 2021. Spotify. Date of retrieval: 12.11.2021. Available: https://developer.spotify.com/documentation/web-api/

14. Spotify Web API Console, 2021. Spotify. Date of retrieval: 12.11.2021. Available: https://developer.spotify.com/console/

15. Spotify Dashboard, 2021. Spotify. Date of retrieval: 12.11.2021. Available: https://developer.spotify.com/dashboard/

16. Flutter documentation, 2021. Flutter. Date of retrieval: 12.11.2021. https://www.flutter.dev

17. FlutterFire documentation, 2021. Flutter, Google. Date of retrieval: 20.11.2021. https://firebase.flutter.dev/docs/overview/

18. Spotify Design Guidelines, 2021. Spotify. Date of retrieval: 14.11.2021. Available: https://developer.spotify.com/documentation/general/design-and-branding/

19. Google Fonts package, 2021. Flutter. Date of retrieval: 15.11.2021. Available: https://pub.dev/packages/google_fonts

20. Spotify Authorization guide, 2021. Spotify. Date of retrieval: 15.11.2021. Available: https://developer.spotify.com/documentation/general/guides/authorization/

21. Flutter Web Auth, 2021. Date of retrieval: 21.11.2021. Available: https://pub.dev/packages/flutter_web_auth

22. Rijckaert, Tim. ‘How to run Flutter in the background?’, 2019. Date of retrieval: 23.11.2021. Available: https://medium.com/vrt-digital-studio/flutter-workmanager-81e0cfbd6f6e

23. Concurrency in Dart, 2021. Dart. Date of retrieval: 23.11.2021. Available: https://dart.dev/guides/language/concurrency

24. Milke, Johannes. Drawer menu GitHub Repository, 2021. Date of retrieval: 21.11.2021. Available: https://github.com/JohannesMilke?tab=repositories&q=drawer&type=&language=&sort=

25. Effective Dart: Style, 2021. Dart. Date of retrieval: 2.12.2021. Available: https://dart.dev/guides/language/effective-dart/style

26. Flutter Tutorial | Restaurant Search App with Zomato API (Part 1: Querying the Public API), 2020. Date of retrieval: 9.12.2021. Available: https://www.youtube.com/watch?v=GDUfd5oDUho

27. AudioPlayer package, 2021. Flutter. Date of retrieval: 11.12.2021. Available: https://pub.dev/packages/audioplayers

28. Spotify Android SDK, 2021. Spotify. Date of retrieval: 10.12.2021. Available: https://developer.spotify.com/documentation/android/