CS 4624: Multimedia, Hypertext, and Information Access
Blacksburg, VA 24061
Virginia Tech

Practice 10k Music App
Clients: Dr. Annie Stevens & Dr. John Irrerea
Professor: Mohamed Farag
November 30th, 2023

Alexander Georgiev, Bailey McKelway, Claire Holmes, Mahlet Zemui,
Rahul Aneja
# Table of Contents

1. List of Figures............................................................................................................................... 4

2. Abstract....................................................................................................................................... 6

3. Introduction.................................................................................................................................. 7

4. Project Overview........................................................................................................................... 8
   4.1. Clients & Motivation.................................................................................................................. 8
   4.2. Brief Description........................................................................................................................ 8

5. System Requirements.................................................................................................................... 10
   5.1. Current Semester Deliverables............................................................................................... 10
   5.2. Deferred Deliverables............................................................................................................... 11

6. System Architecture...................................................................................................................... 12
   6.1. Framework............................................................................................................................... 12
   6.2. Data Flow................................................................................................................................. 13
   6.3. Database Design....................................................................................................................... 15

7. Implementation............................................................................................................................... 17
   7.1. Development Environment Setup............................................................................................ 17
   7.2. File Structure........................................................................................................................... 17
   7.3. API Setup................................................................................................................................. 18
   7.4. Authentication Setup............................................................................................................... 19
   7.5. Navigation Setup...................................................................................................................... 21
   7.6. Profile Setup............................................................................................................................ 21
   7.7. Journal Setup........................................................................................................................... 23
   7.8. Progress Setup.......................................................................................................................... 25
   7.9. Planning Portal Setup................................................................................................................. 27
   7.10. Practice Portal Setup............................................................................................................... 32
   7.11. Redux Setup............................................................................................................................ 34
# Table of Contents

8. **Developer Manual**
   - 8.2. Deployment to Linux Server ......................................................... 40

9. **User Manual** ......................................................................................... 42
   - 9.1. Login/Register Features ................................................................. 42
   - 9.2. Planning Portal Features ............................................................... 44
   - 9.3. Practice Portal Features ............................................................... 47
   - 9.4. Progress Screen Features .............................................................. 49
   - 9.5. Journal Screen Features ............................................................... 50
   - 9.6. Profile Screen Features ............................................................... 51

10. **Reflections** ......................................................................................... 54
    - 10.1. Timeline ....................................................................................... 54
    - 10.2. Challenges ................................................................................... 54
    - 10.3. Lessons Learned .......................................................................... 55
    - 10.4. Future Development ................................................................... 56

11. **Acknowledgments** ............................................................................ 58

12. **References** ......................................................................................... 59
1. List of Figures

Figure 1: Snapshot of the Wireframe’s Dashboard................................................................. 9
Figure 2: Illustration of Architectural Framework ................................................................. 12
Figure 3: Illustration of Data Flow of a Valid Authentication Process................................. 13
Figure 4: Illustration of Data Flow of an Authenticated User’s Engagement with the Application’s Interface........................................................................................................... 14
Figure 5: Snapshot of ‘users’ Collection in Firestore............................................................ 15
Figure 6: Snapshot of ‘musicPieces’ Collection in Firestore................................................ 16
Figure 7: Snapshot of ‘practiceData’ Collection in Firestore................................................ 16
Figure 8: Snapshot of the Project’s File Structure.................................................................. 18
Figure 9: Snapshots of the Application’s Start, Login and Reset Password Screens............ 19
Figure 10: Snapshots of the Application’s Register Screen..................................................... 20
Figure 11: Snapshot of the Application’s Navigation Tabs......................................................... 21
Figure 12: Snapshots of the Application’s Profile Tab............................................................. 22
Figure 13: Snapshots of the Application’s Edit Profile Page................................................... 23
Figure 14: Snapshot of the Application’s Journal Tab............................................................... 24
Figure 15: Snapshots of the Application’s Practice Details Page........................................... 25
Figure 16: Snapshots of the Application’s Progress Tab’s Carousel..................................... 26
Figure 17: Snapshot of the Application’s Progress Tab’s Music Distribution....................... 27
Figure 18: Snapshot of the Application’s Home Tab............................................................... 28
Figure 19: Snapshots of the Application’s View Modal on Non-Editable Plans...................... 29
Figure 20: Snapshots of the Application’s View Modal on Editable Plans............................. 30
Figure 21: Snapshots of the Application’s Add Modal for New Plans.................................... 31
Figure 22: Snapshots of the Application’s Add Modal for Previous Plans............................ 32
Figure 23: Snapshots of the Application’s Practice Tab......................................................... 33
Figure 24: Snapshots of the Application’s Practice Timer Page............................................. 34
2. Abstract

In the world of music education, inspiring students to maintain consistent and effective practice routines has long been a challenge. Recognizing this dilemma, our clients embarked on a journey to leverage technology and reimagine the practice experience for budding musicians. The result of their efforts was a music application that aims to revolutionize the way musicians approach their practice routines by addressing both convenience and motivation.

The innovative concept offers users a number of features that enhance their practice sessions, monitor their progress, and make the entire experience more engaging. Among its many functionalities, the application allows users to plan practice sessions and initiate them with the aid of a built-in timer to track their practice duration. Moreover, the application presents users with visualization representations of their progress on a daily, weekly, monthly, and overall basis through diverse graphs, each highlighting distinct aspects of their practice habits. Additionally, users can delve into a journal-like feature in the application, allowing them to explore and reflect on their musical journey, drawing insights from past practice sessions.

To address the core functionalities mentioned above, our project relies on the integration of Firebase for user authentication and backend data storage, coupled with React Native to ensure cross-platform compatibility in the frontend. This framework facilitates effective communication between the backend and frontend, enabling the exchange of user-related data in order to meet the clients’ requirements within the application. This is notably exemplified by our organization of user information in the backend, utilizing specific collections for swift reading and writing of data as users engage with the application. That being said, as we reflect on the culmination of this semester-long project, it is evident that overcoming challenges and seizing opportunities has been instrumental in our gaining invaluable experience in both client collaboration and implementing diverse solutions. However, acknowledging the iterative nature of application development, we understand the ongoing need for refining the existing features and incorporating new ones in future development.
3. Introduction

We often become what we repeatedly do, and for successful musicians, that means embracing hard work, unwavering passion, and, above all, dedicating countless hours of practice. This universal truth, beautifully captured in the Greek saying “Excellence is a habit, not an act” resonates deeply with musicians from all walks of life as they all share a common aspiration: to improve their skills and create music that touches the hearts of others. However, the process of practicing music effectively has always presented its challenges, with musicians often struggling to maintain consistent routines, monitor their progress, and find the motivation to persevere through tough moments. In response to these challenges, our project aims to provide support to musicians worldwide, as they navigate through their musical journeys, by developing a system that assists them in planning their practice sessions, keeping them motivated to stick to their plans, and allowing them to track their progress effortlessly.

In the following report, we will delve into the specific details of the project. We will first introduce our clients and their motivation behind this project, before offering a brief overview of the envisioned system. Moving forward, we will outline the essential features of the system, encompassing both the deliverables for this semester and those slated for future developments. Next, we will explain the system’s architectural framework, including the design of both the frontend and backend, and a detailed explanation of how data flows seamlessly between these components with the configuration of the database. Subsequently, we will describe our implementation process, offering a step-by-step account of the measures we took to reach the final product. Additionally, we will include a developer manual that provides in-depth instructions for the proper execution of the current program. Furthermore, similar to a user manual, we will provide an all-encompassing guide on how the system functions, including a user-friendly walkthrough of how individuals can interact with its diverse features. Concluding our report, we will delve into reflections and heartfelt acknowledgments to all who have played a vital role in the triumph of our project.
4. Project Overview

4.1. Clients & Motivation

The motivation behind the project stems from a deep appreciation of the incredible impact of practice in the world of music. It all started with a simple, yet profound, idea popularized by the well-known author Malcolm Gladwell in his book “Outliers: The Story of Success”. He suggested that dedicating 10,000 hours of practice could transform anyone into an expert in their chosen field. This concept struck a chord with our clients, Dr. Annie Stevens and Dr. John Irrerea – both music professors at Virginia Tech – who recognized a pressing need within their students' musical journey. They recognized that students often struggle to maintain consistent practice routines and monitor their progress. Hence, fueled by their commitment to nurture their students' musical talents, Dr. Stevens and Dr. Irrerea decided to develop a solution that would make the practice process more accessible, convenient, and motivating. As a result, they came up with the idea of an innovative mobile application that not only assists musicians in planning and tracking their practice sessions, but also serves as a wellspring of inspiration. In fact, their concept combined with their meticulously designed wireframes, earned them the prestigious Alumni Enterprise Award from the Music Academy of the West – highlighting the transformative potential of the project in the world of music education.

4.2. Brief Description

At its core, the application consists of an all-encompassing, cross-platform solution that takes a holistic approach to make music practice efficient and motivating. As depicted in Figure 1, it features five key components, each tailored to enrich the user’s musical experience:

- The planning portal (effectively known as the “plan tab”) provides a snapshot of the user’s weekly plans, with a special emphasis on their daily agenda to keep them in sync with their practice goals for the day.
• The **practice portal** (or the “practice tab”) is where users can seamlessly log their practice sessions using a built-in timer to record the latter and interactive index cards to gently guide them through their practice routine. Additional built-in tools like a tuner and metronome are also readily accessible to assist them during their sessions.

• The **progress tab** offers a comprehensive view of the user's musical journey over time, meticulously tracking the hours devoted to practice as well as the pieces that have been practiced. It even fosters friendly competition among peers with a leadership board.

• The **journal tab** serves as a personal log of the user's practice sessions, allowing them to plan and reflect on their musical growth and identify areas for improvement.

• The **settings tab** allows users to tailor the application to their preferences – from profile tweaking to fine-tuning the overall app experience.

In the following section, we will delve deeper into the application’s complex features and specific requirements to provide a more detailed explanation of its capabilities.
5. System Requirements

In our initial meetings with the clients, we effectively identified and gathered a comprehensive list of system requirements essential to meet their needs for this project. After thoughtful deliberation, we made the decision to categorize these requirements into two groups: those achievable within this semester and those earmarked for future project iterations.

5.1. Current Semester Deliverables

As part of our semester's deliverables, we intend to focus on five key requirements that are fundamental to the project's initial phase:

- **User Accounts:** The system is designed to support the creation and management of user accounts, providing users the ability to input custom data (i.e., personal information and practice sessions) that can be monitored and updated to reflect their progress.

- **Practice/Planning Portal:** The system is designed to enable users to meticulously plan their practice schedules, outlining the piece(s) they intend to practice, the instrument(s) they intend to use, the duration and any necessary notes on the session (e.g., the use of a metronome to play rhythms more accurately). Additionally, users can initiate their practice sessions and have them automatically recorded in the database.

- **Practice Session Logs:** The system is designed to provide a journal-like interface that displays all completed practice sessions. Users can look through this journal and search for a specific session by date.

- **In-App Memory with Pieces/Instruments:** The system is designed to allow users to easily log their practice plans by automatically tracking the pieces practiced and the instruments used, ensuring an effortless process for users.
• **Practice Timer:** The system is designed to include a built-in timer to assist users in recording their practice sessions by tracking the duration and updating the sessions’ status in the database.

As we move forward, these requirements will lay the groundwork for future deliverables of this project.

### 5.2. Deferred Deliverables

In addition to the core system requirements mentioned above, the clients expressed interest in incorporating two additional features:

• **Built-In Tuner/Metronome:** Initially, there was a desire to include a built-in tuner and metronome to assist musicians in their practice. However, through careful discussions with both the clients and their technical expert – Ben Winters – we collectively decided to defer the integration of these functionalities. This decision was based on the complexity and resource demands associated with audio processing, which may extend beyond the current project scope.

• **Gamification of Application:** Another idea involved gamifying the practice process, allowing users and their friends to engage in friendly competition with their practice progress. Although this concept held appeal, it was not considered a core essential feature and its implementation was thus postponed to a later project phase.

Future iterations of this project should mainly focus on the implementation of the above features.
6. System Architecture

In realizing our project, we made deliberate choices regarding the tools to employ for both the backend and frontend – and outlined the data flow and interaction between these two components. The following architectural framework serves as the foundation of the project.

6.1. Framework

As shown in Figure 2, we have chosen to employ React Native to build the application’s interface, and Firebase to handle user data and operations within the app.

![Figure 2: Illustration of Architectural Framework](image)

When it comes to the frontend of the application, React Native allows us to create a cross-platform mobile application compatible across various platforms. Within this React Native environment, the application is organized into distinct sections, each managed by the UI Handler. These sections include functionalities such as ‘Login and Signup’ for user access, a ‘Planning Portal’ for scheduling practice sessions, ‘Practice Logs’ for viewing practice data, and a ‘Timer’ to track each practice session. To facilitate smooth communication between the application's
frontend and backend, API calls – managed by the API Handler – will be used to ensure the efficient transfer of data between the user interface and the backend.

On the backend, Firebase plays a crucial role in the application’s system architecture. It offers specific functionalities dedicated to distinct tasks:

- **Firebase Authentication**: When users access the ‘Login’ or ‘Register’ functions on the frontend, their provided details, such as email and password, are verified using this module. Only authenticated users can access the application's primary functions.

- **Firebase Firestore**: This cloud-based NoSQL database manages the storage and retrieval of user data. Various frontend actions such as inputting practice details, initiating practice sessions and accessing practice logs involve data transactions with Firestore. Whether it's storing new entries or fetching existing data, Firestore handles these operations, maintaining the data's integrity and security.

The clear delineation of functions and efficient communication channels ensures the application's reliable performance.

### 6.2. Data Flow

As users engage with different features within the application, data seamlessly flows between the frontend and backend. Figure 3 offers an overview of the data flow during the authentication process, while Figure 4 depicts how data moves when users engage with the application's features.

*Figure 3: Illustration of Data Flow of a Valid Authentication Process*
When a user intends to access the application, they initiate the process by inputting their login or registration details on the frontend interface (i.e., the ‘App’). Once the user submits this information, the application promptly forwards it to the API, which then collaborates with Firebase Authentication to validate the user’s credentials. If the provided information is correct, Firebase Authentication returns an authenticated user status to the API, and the latter subsequently communicates this status back to the application, granting the user access.

Upon successful authentication, the user can fully utilize the application's features. Firstly, when the user inputs practice data related to the musical pieces they are working on, the application channels this information through the API for storage and processing. The API then communicates with Firebase Firestore to safely store the data and send confirmation back to the application. Secondly, when the user initiates their practice session with the built-in timer, the application not only records the duration of the session but also communicates with the API to log this data in Firestore. This ensures that the user can subsequently view and analyze their practice details at a later time when Firestore returns this information to the application via the API. Lastly, when the user opts to view their practice data, the application sends a retrieval request through the API, which fetches the required data from Firestore and presents it back to the application, displaying it in a user-friendly calendar format.

In essence, the API serves as a bridge between the frontend and backend, ensuring secure communication and data transfer with Firebase Authentication and Firestore. This seamless integration facilitates a robust and user-friendly experience, allowing musicians to focus on their practice session while the application takes care of the data intricacies.
6.3. Database Design

Given the simplicity and expandability of our data structure, Firestore aligns perfectly with our project's requirements. While it has some limitations in terms of querying capabilities and potential read/write costs, these concerns should not raise any red flags as it more than meets our querying needs and offers a low cost solution for the current project scope by organizing any application related data into collections and documents.

To begin, we maintain the ‘users’ collection to store a comprehensive list of user information such as name, email, date of birth, musical instruments, levels of experience and profile picture, as illustrated in Figure 5. Additionally, each user collection has an associated ‘musicPieces’ and ‘practiceData’ collections.

![Database Design Diagram]

Figure 5: Snapshot of ‘users’ Collection in Firestore

Depicted in Figure 6 is the ‘musicPieces’ collection which contains information about the pieces the user has practiced, including the title of their practice plan, piece name, composer name, instrument used, and any additional notes. This information serves as the backbone for our in-app memory feature.
The ‘practiceData’ collection houses all of the user's practice data, including essential details such as the music piece, practice date, status, and duration, as shown in Figure 7.

Hence, Firestore's versatile data structure, though accompanied by some limitations and costs, allow us to effectively organize the core data elements within the application into well-structured collections and documents, ensuring a robust foundation for our application's functionality.
7. Implementation

Throughout the past semester, substantial strides have been achieved, successfully delivering on all promised deliverables. The following discussion will provide a comprehensive overview of our implementation process.

7.1. Development Environment Setup

To begin, we created a private GitHub repository to use throughout the semester for version control. We also configured Expo (i.e., an open-source platform and set of tools to help us develop and demo our application on Android and iOS) on our environment systems by following the step-by-step guide outlined in the Expo documentation. The initial codebase in our repository was created using the ‘yarn create-expo-app’ command, providing us with a fundamental starting point. As per our technical consultant’s recommendation, we modified the generated project files to use TypeScript inorder to improve the overall development experience with the incorporation of types. With these foundational components in place, we started the development server by running the ‘yarn expo start’ command, and then scanning the generated QR code to test the current application’s state on our phones with Expo Go.

7.2. File Structure

As depicted in Figure 9, our project is meticulously organized following a strict structure. While files outside the ‘src’ folder are predominantly generated by Expo as necessary configurations (i.e., standard for all Expo applications), our ‘src’ directory houses all the files we have written. Within the ‘src’ folder, we've structured our project into key sections. The ‘assets’ folder contains images, constants (e.g., values for musical proficiency levels and instruments), and common styling. Following that is our ‘components’ folder which contains all of the components we have built from scratch, encompassing elements like specific modals, dropdown windows, graphs… Moving on, we have the ‘helpers’ folder which consolidates all of the helper functions we have created for various purposes throughout the project, including validation
functions and general utilities. The ‘redux’ folder, which will be discussed in detail later, is responsible for the configurations essential for locally storing the user's information when the application initializes. Next is the ‘screens’ folder which holds the screens that users interact with – from authentication screens that serve specific authentication purposes to tab screens that showcase various application tabs once the user is authenticated. Finally, the ‘services’ folder houses essential files connecting our backend to the application. This includes configuration files, state change listeners, and crucial API calls handling the core functionalities of the application.

![Figure 8: Snapshot of the Project's File Structure](image)

### 7.3. API Setup

To enhance the modularity of our project and facilitate future development and updates, we decided to organize all API-related functions within a dedicated folder, including authentication and data management api files. Given our reliance on Firebase Authentication and Firebase Firestore, each section corresponds to the unique functionalities of these services. Within the authentication API file, comprehensive handling of authentication processes takes place. This includes the creation of user accounts, logging in and out, account deletion, and password changes. Within the data management API file, operations related to backend data – such as adding, editing, deleting, and fetching user information – takes place. The backend data spans a range of user attributes, including the user's personal information (i.e., name, date of birth, musical proficiency level…) and practice data (i.e., plans, logs, progress…). This
systematic organization allows us to ensure clarity and efficiency in managing distinct functionalities within our project structure.

7.4. Authentication Setup

To implement user authentication within our application, we initially focused on email and password registration. As seen in Figure 9, when users first open the application, they are presented with a ‘start screen’ that gives them the option to either login or register. When logging in, validation checks are performed both on the frontend and backend. The frontend validation includes checking for the presence of an email and password, adequate password length, and proper email formatting, while the backend validation includes checking that the email exists in the backend and the given password is correct for a valid email. In case of an incorrect password, a ‘Forgot Password’ link can trigger an email from Firebase Firestore with reset instructions.

![Figure 9: Snapshots of the Application’s Start, Login and Reset Password Screens](image)

In addition to the above features, Figure 10 depicts the registration process with the essential personal details (e.g., name, date of birth, instruments…) needed to create an account by email and password. Similar to the login procedure, both frontend and backend validation checks are implemented to ensure the validity of the entered email and password.
Figure 10: Snapshots of the Application’s Register Screen

While the design of the above screens was crafted using various React Native libraries by closely following the project's initial wireframe design, the backend primarily relied on the establishment of a Firebase project that provides free authentication and data storage as long as the number of monthly active users stays under 50k and stored data is less than 1 GiB. To set up Firebase, we first enabled Firebase Authentication's email and password authentication method, which also provides options for email verification and password recovery. Next, we integrated the Firebase SDK into our Expo project, configuring it within a file named firebase.ts that contains our API keys and other settings. For security reasons, we moved the API keys to a separate file (i.e., ‘.env’) that was added to our .gitignore and not pushed to our GitHub repository. Finally, we established access to the API keys in app.config.js, making them accessible throughout the entire application as long as the ‘.env’ file is present in the repository.
7.5. Navigation Setup

In our development process, we decided to split the application into two distinct views: a logged in user view and a logged out user view. A user who is logged in should be able to enjoy the privilege of uninterrupted access to all of the application’s featured tabs – as shown in Figure 11 – without the need for recurrent logins; while a user who is logged out should be confined to the authentication functions. This separation corresponds to two screen stacks – one for logged in users and one for logged out users. To effectively manage the two screens stacks, we created a custom hook to check if a user is currently logged in by listening for the ‘onAuthStateChanged’ event. Upon application launch, the ‘App.tsx’ file dynamically selects and presents either the logged-in user stack or the logged-out user stack, depending on the authentication status of the current user.

![Navigation Tabs](image)

*Figure 11: Snapshot of the Application’s Navigation Tabs*

7.6. Profile Setup

Given the limited details in the initial wireframes for the application’s ‘Settings’ tab, we opted to transform it into a comprehensive ‘Profile’ tab, consolidating all user-entered information from the registration process. As depicted in Figure 12, the tab not only displays user details but also incorporates essential functionalities allowing users to log out or delete their account. While the log-out functionality is relatively straightforward, the delete account feature involves thorough validations with the user’s password. Once a password is provided, API calls are initiated to validate the password, facilitating a re-login process to confirm the deletion of the account from Firebase. This deletion ensures the complete removal of all user-related information from both Firebase Authentication and Firestore.
Recognizing that users may wish to edit certain fields or update their password, we opted to integrate an edit functionality in the ‘Profile’ tab. Initially, our plan was to allow direct editing of fields within the tab; however, after careful consideration, we acknowledged the potential for user errors and the complexity of validating changes without a structured process. Hence, we decided to link an ‘Edit Profile’ page to the current tab through a new stack navigator separate from the main application's tab navigation stack. As seen in Figure 13, this page offers editable fields, including the option to change their password, and then confirm the changes by clicking the save button, which triggers validation checks before applying the modifications. The majority of the validations are performed on the frontend, confirming that all fields receive valid input (i.e., ensuring name, instrument & musical level are not empty, and date of birth is not set to the future). Changing the password involves additional validation, ensuring that both the old and new passwords match and that the new password meets the appropriate length criteria. The initial check is conducted securely in the backend through Firebase, while the subsequent check is performed in the frontend to promptly address any issues with the new password. Additionally, we included a ‘back’ feature to discard any modifications and return to the ‘Profile’ tab.
7.7. Journal Setup

Our original implementation for the journal included using React Native’s agenda component. This component would have allowed us to easily link a calendar to a list of events by specific time periods – when an element on the calendar is selected, the list would jump to that event. However, the issue with this approach was that the created calendar would only show events one week at a time instead of the clients’ preferred monthly view outlined in their wireframes. Furthermore, by only showing one week, the user would not be able to see their practice trends at a glance and would need to pull down on the calendar’s knob. Hence, to address this issue, we switched to using a separate calendar and scrollview component. As presented in Figure 14, in this adaptation, we ensured that dates on the calendar associated with a practice session are marked with a red dot, while all practice sessions for the presently selected month are presented as a list in the scrollview. When the currently selected month changes, the list of practice sessions is dynamically updated to showcase only those within the current month. This involves applying a date filter, utilizing a where clause in the query to the 'practiceData'
collection in our backend, specifying the relevant month. The documents returned from this query are then utilized to populate information about each practice session in the scrollview.

![Figure 14: Snapshot of the Application’s Journal Tab](image)

In response to our client’s request, we also implemented a dedicated ‘Practice Details’ page for each practice session, accessible by selecting one in the scrollview. This page is configured through a different stack navigator separate from the main application's tab navigation stack. When a session in the scrollview is selected, information about that session is passed to the ‘Journal Details’ page, as shown in Figure 15. This page presents all essential details pertaining to the selected plan, utilizing non-editable fields.
7.8. Progress Setup

Our primary focus on the ‘Progress’ tab centered on enhancing the presentation of user progress. Hence, following the wireframe designs, we decided to divide the screen into two sections: a top section featuring goals and a bottom section showcasing musical distributions. Our implementation process commenced with the top half, with the development of a goal tracker component. This component queries completed practice hours and pieces from the backend, using the ‘practiceData’ collection, and compares them against predefined goals. Having established this foundational framework, we introduced a carousel mechanism to present this information on a daily, weekly, monthly, and overall basis to assess progress over different time periods, as shown in Figure 16.

Figure 15: Snapshots of the Application’s Practice Details Page
Figure 16: Snapshots of the Application’s Progress Tab’s Carousel

In the bottom section of the screen, we introduced a music distribution component that offers a concise overview of weekly practice habits by composers. This involves a distribution graph illustrating practice time dedicated to various composers throughout the week, as shown in Figure 17. The chart highlights the top five composers based on total practice time using specific queries from the ‘practiceData’ collection in Firebase. While the current implementation of the music distribution component only supports the top five composers for the week, the system is intentionally designed to be scalable, allowing the flexibility to increase the number of displayed composers as needed.
7.9. Planning Portal Setup

Our primary focus on the ‘Home’ tab centered around planning practice sessions for the week. While the screen also features the same goal tracker component used in the ‘Progress’ page for overall goals, the central focus of this screen lies in the components below it. The elements beneath the goal tracker consist of interactive calendar days representing the current week, with each day linked to a corresponding planner component, as illustrated in Figure 18.
Upon selecting a day, the application fetches the corresponding practice plans for that date from the ‘practiceData’ collection in our backend using specific queries. These plans, whether complete or incomplete, are then presented in the planner along with their titles. If the chosen day precedes the current date, only read operations are available for the plans, opening a modal of non-editable fields with detailed information about the plan such as title, piece name, composer name, instrument used, practice date, duration, status and any notes, as depicted in Figure 19. This precaution ensures that no inadvertent edits are made to practice plans that have already occurred.
Figure 19: Snapshots of the Application’s View Modal on Non-Editable Plans

If the selected day is on or after the current date, both read and write operations are available for existing plans and new ones. For existing plans, a modal with editable fields is opened, featuring save and delete functions as seen in Figure 20. The save button triggers a validation process for any changes made to the plan, ensuring that all fields have valid inputs. While the inputs for title, piece, and composer must be non-empty, the notes field can be left blank. Furthermore, the instrument field must contain a single instrument chosen from the instruments listed in the user's profile. This is achieved by presenting all of the user's selected instruments in the same dropdown component used for the ‘Register’ and ‘Edit Profile’ pages. After successful validation of the changes, saving reflects the edits in the backend’s ‘practiceData’ collection and then in the application. Moreover, if the changes to the plan result in it no longer matching any of the plans in the backend’s ‘musicPieces’ collection, the updated plan is added to that collection. The delete button is more straightforward, removing the plan from the backend’s ‘practiceData’ collection and then from the application. In case of any issues
accessing the backend during any of these two operations, error messages describing the problem are displayed and no changes are applied to the application.

Figure 20: Snapshots of the Application’s View Modal on Editable Plans

As for new plans, at the end of the planner, there is an add piece button on the planner linked to a modal that provides two options. As shown in Figure 21, the first option is to add a new plan, which, when selected, opens another modal with input fields for the plan's title, piece name, composer name, instrument to use, and notes – similar to the edit plan modal discussed earlier. However, this modal only features a save button, as the plan has not yet been added to the system and thus should not have delete operations. The save button functions in the same way, adhering to the validation process discussed above and reflecting the changes back to the backend’s ‘practiceData’ and ‘musicPieces’ collection, and – if successful – updating the application, with any errors promptly displayed.
Illustrated in Figure 22, the second option for adding a plan is to select one from a set of previous plans. When chosen, this opens another modal containing all of the saved plans in the backend’s ‘musicPieces’ collection. Each plan's title, piece name, composer name, instrument used, and notes are displayed. By selecting a plan from the displayed options, all input fields for the plan are automatically prefilled, and the ‘practiceData’ collection in the backend is updated, before then adding the plan to the planner in the application for the selected date.
7.10. Practice Portal Setup

The practice portal, accessible through the application's ‘Practice’ tab, serves as a dynamic list of all the incomplete practice plans users have scheduled for the current date with the ability to commence working on these plans. As seen in Figure 23, the list is implemented with the help of a planner component – similar to the one designed for the planning portal – for the current day. While this component will be thoroughly discussed in the following section, it’s important to note that its primary function is to facilitate, in a user-friendly manner, the addition, modification, and removal of plans, not only from the user's local device but also from the backend.
Figure 23: Snapshots of the Application’s Practice Tab

The core feature of the practice portal lies in the ‘Practice Timer’ page, which activates when plans are scheduled for the current date and users choose to initiate the timer by selecting the start timer button. This button is linked to the ‘Practice Timer’ page through a different stack navigator separate from the main application's tab navigation stack. As showcased in Figure 24, this page adopts an index card-like structure, sequentially presenting all plans scheduled for the date. Each plan on the screen is equipped with start, pause, next, and stop functionalities to effectively monitor progress. Upon selecting the start or pause button for a new plan, the system tracks engagement by duration, with the elapsed time displayed at the top (i.e., at the user's discretion). Following this, the system saves the user’s progress for each plan to the backend when either the stop session or next piece button is selected. The stop session function halts the entire practice session and returns the application to the ‘Practice’ tab, leaving the last piece worked on as ‘in-progress’ in the backend and ensuring it remains in the list of plans to work on for the day. The next piece function marks the current plan as ‘completed’ and advances to the
next one, if available. This action updates the backend, removing the completed piece from the list of plans to work on and logging the related data in both the ‘Journal’ and ‘Progress’ tabs.

![Figure 24: Snapshots of the Application’s Practice Timer Page](image)

7.11. Redux Setup

To optimize the efficiency of our data retrieval process from Firebase and eliminate the need to pass the same variable throughout various source files, we made a strategic decision to implement Redux. Redux is a popular JavaScript library for managing and centralizing application state. It is commonly used to manage the complexity of state changes in large applications, functioning as a global object accessible and updatable by every component in a react application.

We rely on Redux to manage a significant portion of user data in our application. Upon the initial mount following a user's login or registration, a fetch call is initiated to Firebase to retrieve crucial user information. This includes the user's profile via the ‘users’ collection, practice data for the week through queries from the ‘practiceData’ collection, and all of the user's music pieces from the ‘musicPieces’ collection. To optimize loading times and streamline the
application's performance, we limit Redux's information to these essential data points, which also happen to be the most important ones. The retrieved data is then seamlessly stored in Redux by dispatching the setProfile, setPracticeData, and setMusicPieces actions. Once stored, the data is accessible by any React component using the useSelector hook. This ensures that every modification made to our backend pertaining to the user is also logged in Redux, updating the current data in the application correctly with Firebase.
8. Developer Manual

In anticipation of future contributions from other teams, the following provides a step-by-step guide to set up the project for development on a computer. Additionally, following the provided guide, a discussion of the Linux server where the current project was attempted to be deployed will also be included.

8.1. Guide for Development

To set up the project on a computer for development, follow these steps:

1. Installing Expo and Expo Go
   - Install Expo on the computer by following the guide available at: [https://docs.expo.dev/get-started/installation/](https://docs.expo.dev/get-started/installation/)
   - Then, download Expo Go on a mobile device from its application store.

2. Clone Repository
   - Open the computer's terminal and navigate to the directory where you want to clone the repository.
   - Next, as shown in Figure 25, run the command:
     
     ```
     git clone https://github.com/alexsg2/Practice_10k_Music_App.git
     ```

     ![Figure 25: Snapshot of Running the ‘git clone’ Command](image)

Figure 25: Snapshot of Running the ‘git clone’ Command
The file structure should match the one illustrated in Figure 26.

![File Structure after Running 'git clone' Command](image)

**Figure 26: Snapshot of File Structure after Running the ‘git clone’ Command**

3. **Install dependencies**
   - Navigate to the cloned repository's directory and install ‘yarn’ on the computer.
   - Then, as seen in Figure 27, run the command:
     
     ```
yarn install
     ```

![Running 'yarn install' Command](image)

**Figure 27: Snapshot of Running the ‘yarn install’ Command**

The file structure should now look like the one shown in Figure 28.
4. **Insert `.ENV` file**

- Create a new file named `.env` in the project's root directory.
- Next, include all of the necessary API keys and configurations required for Firebase within the `.env` file, following the guide provided at:

  https://firebase.google.com/docs/functions/config-env?gen=2nd

The file structure should now include the `.env` file, as seen in Figure 29.
5. **Start the Program**

- Connect the computer to a trusted internet network and, as illustrated in Figure 30, run the command:

  \[\text{yarn expo start}\]

  ![Figure 30: Snapshot of Running the ‘yarn expo start’ Command]

- Now, connect the mobile device to the same network as the computer and scan the provided QR code using the mobile device's camera in order to access the program through the Expo Go application.

  **Note:** In case Expo Go encounters an internet connection issue, use the command \[\text{yarn expo start --tunnel}\] instead and then scan the generated QR code, as shown in Figure 31.
8.2. Deployment to Linux Server

Following the guide above, we were able to successfully set up the project on a Linux server that was provided for the course – as shown in Figure 32.
However, deploying the application on the server was not possible since Expo Go could not reach it on our tested mobile devices. Figure 33 shows the result of scanning the QR code and running Expo Go.

Since our application is different from traditional websites where applications can be run on a server and accessed via a url, any distribution of the application should be done through the Apple app store or Google play store.

With a number of features incorporated into our project, the following offers a user manual designed to assist users in navigating through the various functionalities. This guide aims to provide clarity on the different features, their purposes, and possible user interactions.

9.1. Login/Register Features

As illustrated in Figure 34, upon launching the application, users are greeted by a ‘Start’ screen which provides two options: login or register. Existing users can choose the login button, leading them to the ‘Login’ screen, while those that would like to create a new account can select the register button, directing them to the ‘Register’ screen.

![Figure 34: Snapshot of the Application’s Start Screen](image)

Those that choose to login, can input their email and password in the provided fields as seen in Figure 35. If the provided credentials are valid, clicking the login button will redirect the user to the application’s home screen. However, in the event of any issues, an on-screen error message is promptly displayed, providing the user with information on why the login failed.
Additionally, users who forget their password can utilize the ‘Forgot Password? Click here.’ link above the login button. Clicking this link redirects users to a ‘Reset Password’ screen, where they can enter their email and click the ‘Reset Password’ button to receive instructions via email on how to reset their password.

![Login and Reset Password Screens](image)

*Figure 35: Snapshot of the Application’s Login and Reset Password Screens*

Those that choose to register, can input their name, email and password and select the appropriate options from a dropdown window for their profile picture, date of birth, instruments and musical level, as seen in Figure 36. The dropdown for instruments allows users to choose multiple options, while the one for musical level permits only a single selection. After successful validation of these fields, clicking the register button creates an account for the user and directs them to the application's home screen. However, in the event of any issues, an on-screen error message is promptly displayed, providing the user with information on why registration failed.
Moreover, users have the flexibility to navigate between the ‘Register’ and ‘Login’ screens using the bottom links provided on these respective screens. Additionally, the back button located at the top of each of the mentioned screens, except the ‘Start’ screen, allow smooth navigation through all of the authentication screens.

**9.2. Planning Portal Features**

The ‘Home’ screen hosts the application’s planning portal. While the top half of this screen shows the user’s overall progress in terms of completed hours and pieces, the bottom half of the screen showcases their practice plans for the current week depending on the selected day, which is highlighted in yellow. For days that have already passed, such as Tuesday in Figure 37, users can only view the details of the practice plans for that day by selecting one and reviewing its details in a popup window.
Practice plans can only be added, edited, or deleted if the practice date has not yet passed. So, as illustrated in Figure 38, with the current day being Thursday, only plans scheduled from Thursday to Saturday are eligible for addition, editing, or deletion.
The process of adding practice plans begins by clicking the add piece button at the end of the list of plans for the selected day. As depicted in Figure 39, this action opens a popup window with options to either add a new plan or select from previously saved ones. Choosing the new plan option opens another popup window with fields to fill out for the plan. Upon completing all fields – except for the notes field, which is optional – clicking save should add the plan to the list of plans for the selected day. Conversely, choosing the previous plan option triggers the opening of a new popup window with all past plans – if any – displaying each plan’s title, piece name, composer name, instrument to use and notes. Clicking any of these plans will add it in the list of plans for the selected day. In case of any errors during these two options, an error message with a description of the problem will be displayed.

Figure 39: Snapshot of the Application’s Add Piece Popup Windows

The process of editing or deleting practice plans begins by choosing a plan to edit or delete from the list of plans for the selected day. As shown in Figure 40, this opens a popup window with editable fields for that plan. Changes can be made directly to the fields, and upon satisfaction, clicking the save button should reflect those changes back, if valid (i.e., all but the notes field are filled). Conversely, deleting the plan is accomplished by simply clicking the delete
button. In case of any issues with either saving or deleting, an error message will explain the problem.

Moreover, users may dismiss any of the popup windows by clicking on the cancel icon in the top left corner of the windows, discarding both the window and any changes that may have been made.

9.3. Practice Portal Features

Similar to the planner on the ‘Home’ screen, the ‘Practice’ screen displays the user’s incomplete practice plans for the current date, as depicted in Figure 41. For users seeking more details about these plans or wishing to make edits, clicking on an entry opens a modal presenting editable fields with save and delete buttons. This modal, as well as the one opened when clicking the add plan button on the planner, functions in the same manner as described in the previous section. The start timer button on the ‘Practice’ screens initiates the practice session for the day, transitioning the user to a ‘Practice Timer’ screen with the plan currently in progress. This button is operational only when there are plans scheduled for practice. In the absence of any plans, users
are presented with an error message, explaining that there must be at least one plan in order to initiate a practice session.

As shown in Figure 42, the ‘Practice Timer’ screen features the current plan for practice, accompanied by intuitive buttons for starting, stopping, pausing, and progressing to the next plan. The elapsed time for the current plan is displayed at the top, along with a button to toggle the visibility of the time. Directly below, is a card-like window that displays the current plan’s details, including the title, piece name, composer name, instrument to use, and any accompanying notes. Users can toggle between the start and pause buttons adjacent to the card to track the duration they spend on a plan. To conclude the entire session, they can click on the stop session button, preserving the duration for the current plan and marking it as ‘in-progress’ before redirecting the user back to the ‘Practice’ screen. In this transition, the planner is updated by removing all completed plans. Plans are marked as ‘completed’ when users click on the next piece button. This action saves the duration for the current plan and updates its status before moving on to the next plan – if one exists. In the absence of a ‘next plan’, users are redirected back to the ‘Practice’ screen, with all plans removed from the planner – as no incomplete plans remain.

Figure 41: Snapshot of the Application’s Practice Screen with and without Plans
9.4. Progress Screen Features

The ‘Progress’ screen, showcased in Figure 43, displays an overview of the user's statistics, offering insights into their practice habits. To check their progress against the pre-defined daily, weekly, monthly, and overall goals, users can simply swipe through the graphs within the carousel at the top of the screen. These graphs provide a visual representation of the user’s progress in terms of practice hours and number of completed pieces. Moreover, users can explore the top 5 composers they have practiced during the week in the bottom section of the screen. In addition to displaying this musical distribution as a pie chart, the total time spent on these composers is also highlighted.
9.5. Journal Screen Features

As illustrated in Figure 44, the ‘Journal’ screen displays the user’s completed/in-progress practice sessions by month. The default month that is shown when the user opens the app is the current month. Users can navigate to previous and future months using the arrows above the calendar – with the marked dates changing depending on which days have practice sessions associated with them. As the month changes, the practice sessions shown below the calendar also change to reflect sessions in the currently selected month. Users can scroll through the list of practice sessions and view their date and title. If they would like to see more details, they can click on an entry in the list to open up a details page, as shown in Figure 44, which holds all information about that practice session. It includes the session’s title, piece name, composer name, instrument used, practiced date, duration, status, and notes – if provided. Users can click the back button to return to the ‘Journal’ screen.
9.6. Profile Screen Features

As illustrated in Figure 45, the application's ‘Profile’ screen, accessible through the last tab at the bottom of the navigation component, displays the user's personal information, accompanied by change profile picture, edit, logout and delete options. The user's personal details include a profile picture as well as non-editable fields with their name, email, date of birth, instruments, and musical level. The change profile picture button beneath the profile image allows users to select a new profile picture from their gallery, provided the application has the necessary permissions to access their photo library. In case permissions are not granted, an error message notifies the user urging them to allow the required permissions. At the bottom, two buttons facilitate logging out or deleting their account. Selecting the logout button redirects users to the application's ‘Start’ screen, where they can either login again or create a new account. Selecting delete account button prompts a popup field where users must enter their password to confirm the deletion of their account. Upon confirmation, users will be redirected to the application's ‘Start’ screen where they can create a new account. However, in the event of an
error during the confirmation process, a message signaling the error will appear allowing the user to make another attempt at deleting their account.

![Profile Screen](image)

*Figure 45: Snapshot of the Application’s Profile Screen*

The edit button above the fields on the ‘Profile’ screen redirects users to an ‘Edit Profile’ screen where modifications to fields such as name, date of birth, instruments, and musical level are possible, as seen in Figure 46. For password changes, users can edit the optional password fields, entering their old password, accompanied by the new one. If the old password is correct, and the new password is valid – along with other edited fields – clicking the save button redirects the user to the ‘Profile’ screen with the changes reflected back in the rest of the application. However, should an error arise in the validation of edited fields upon clicking save, an error message will be displayed, providing an explanation of the issue. Alternatively, users opting to discard changes can simply click the back button at the top of the screen to revert to the ‘Profile’ screen.
Figure 46: Snapshot of the Application’s Edit Profile Screen
10. Reflections

As we conclude this semester-long project, we have been able to reflect on the journey we have experienced thus far. The following discussion covers the adjustments made to our timeline, as well as the challenges we encountered and the invaluable lessons we learned, before outlining the work that lies ahead in future iterations.

10.1. Timeline

Through continuous adjustments to our timeline in response to emerging challenges and improved efficiency, Figure 47 encapsulates the final timeline that we crafted to fulfill the essential project requirements.

![Timeline Diagram]

*Figure 47: The project timeline from the beginning to the end of the semester*

10.2. Challenges

As in any project, we encountered a variety of challenges over the past couple of months. The first challenge we ran into simply involved learning new tools. Some group members were familiar with Firebase, Redux, TypeScript, and React Native, while others weren’t. We overcame this challenge by watching and reading tutorials as necessary, and asking other members of the team who had more experience with the selected tools. The next challenge we faced involved the need to reduce the number of backend reads. Initially, each screen in our application made a call to Firebase for data every time it was in focus to ensure users viewed the most up-to-date information. However, this resulted in an excessively high number of reads – a rate that would be
unsustainable when the application is deployed, especially with the current Firebase plan. To address this challenge, we implemented Redux to effectively reduce the number of reads necessary when navigating between screens.

In addition to the technical issues we ran into, we also faced difficulty balancing the clients’ wants with what we could deliver within the given timeframe. For instance, the clients requested including built-in tools like a metronome and tuner, but after discussing with our technical consultant we learned that we would likely not be able to implement those features due to the complexities of audio programming. Since building a mobile application from scratch was a novel task for some team members, we had trouble establishing definite deliverables for the semester. However, in the end we were pleased to deliver the 5 promised features to our clients, even though the additional features were not feasible within the current timeframe. We believe that another team will be able to pick up where we left off and incorporate the extra features our clients requested, as well as enhance the existing ones and maintain the current codebase.

10.3. Lessons Learned

Throughout the completion of this project, we learned two main lessons. The first lesson involves the importance of development order. Reflecting on the past couple of months, we recognize that the development of features and essential components in a different sequence could have not only saved us time but also streamlined the entire development process. Our mistake was choosing to postpone the development of the planning and practice portal features until the end, despite their pivotal roles in the project. Initially, we prioritized the development of other features, with a focus on frontend elements, assuming it to be more straightforward. However, this assumption proved to be incorrect, as the time and effort required to complete those features turned out to be substantial, leaving us with little time to address the crucial planning and practice portal features. Although we managed to finish the latter features within the set timeline, we regret not dedicating more time to them.

The second lesson we learned throughout the semester is the need to precisely define what can and cannot be delivered within a given timeframe. It’s imperative for any project to refrain from overcommitting and maintain a realistic outlook on achievable goals when faced with time and skill constraints. In navigating the challenges of this project, we recognized how
our limited time and skill set influenced the constant modification of our timeline. As a result, we now know how to approach future projects with an improved ability to accurately assess what we can and cannot accomplish with given constraints.

10.4. Future Development

Despite the considerable progress made this semester, our project still requires further refinement to align it with what our clients envision. This includes introducing additional features, transitioning to a different backend data storage system, and resolving certain UI issues.

As mentioned in a previous section, our clients have expressed interest in the addition of two main features: the gamification of the application and built-in practice tools. While the first feature aims to motivate users to practice more by engaging in friendly competition with friends online, the second feature aims to enhance the overall practicing experience with tools like a metronome and tuner. Hence, future development of this project should take into account these client requests and ensure the application aligns with their goals.

Moreover, recognizing the limitations of our current backend database, Firebase Firestore, future iterations of this project should explore a more robust database solution. One compelling suggestion is the adoption of MongoDB, a NoSQL database that offers flexibility and scalability – accommodating the evolving needs of the application. Nevertheless, an alternative system can be considered as long as it aligns with our current project structure, preferably as another NoSQL database. Deviating from this choice might necessitate substantial modifications to the project to accommodate different types of databases. Additionally, integrating our API using Express.js could prove to be beneficial, as the latter is a minimalist and flexible framework – that would complement MongoDB, if used.

Furthermore, given the current UI issues within the application that we have discovered, it’s important that future development of this project tackle and resolve these concerns:

● The first issue involves time discrepancies, as all times are stored in UTC, causing inconsistencies due to variations in the user’s local time on their device.
● The second issue involves a rounding error that affects the display of practice
duration in the goal tracker component. For instance, 0.50 hours may be presented
as 0.4975947569450.
● The third issue to address is to enhance database reads, ensuring we’re not
consistently fetching data every time a screen or component reloads.
● The fourth issue to address is the absence of detailed error messages with Firebase
Authentication when login or registration fail.
● The fifth issue to address involves a text overflow visual bug on the ‘Practice
Timer’ page caused by the excessive length of certain plan details.
● The sixth issue to address lies in the ‘Journal’ screen, where the calendar month
fails to dynamically update to today’s month when switching between tabs.
   Additionally, this screen should also incorporate a sorting functionality.
● Lastly, the final issue to address is on the ‘Profile’ screen and it involves the lack
of a mechanism to remove or edit a user’s profile picture.

Acknowledging that some of the mentioned issues might have been identified earlier with
the inclusion of test cases in our code, we unfortunately ran out of time and opted to defer these
contems for future development. This decision was influenced by the broader scope of including
additional features, considering the potential for numerous enhancements to significantly
improve the project and make it a more engaging experience – ultimately motivating users to
practice more diligently.
11. Acknowledgments

We would like to extend our heartfelt gratitude to the key individuals who have played pivotal roles in both the start and continued success of this project.

First and foremost, we want to thank Professor Mohamed Farag for his guidance, as it has been instrumental in keeping us on the right path throughout this semester.

Additionally, we would like to extend our deep appreciation to our esteemed clients, Dr. Annie Stevens and Dr. John Irrerea, for entrusting us with their project. Their continuous feedback has been invaluable in guiding our progress and ensuring that we meet their expectations.

Furthermore, we want to acknowledge the contributions of Ben Winters, our technical consultant, who has provided invaluable advice on project setup and guided us through the tasks assigned by our clients.

Last but not least, we want to extend our gratitude to Rachael Quan, the creative mind behind our wireframes. Her outstanding work has propelled us swiftly through the design process of the application, ensuring it aligns precisely with the client's vision.

Collectively, these individuals have been the cornerstones of our project's achievements, and we are deeply grateful for their continued support and expertise.
12. References

