12/05/2022

Final Report

Studio Buddy

Alejandro Zicavo
Elmiche Kinmakon
Tubatsi Moloi

Links

Contents

Introduction

Studio Buddy is a cross platform mobile application created using React Native that allows teachers and students to connect for music lesson coaching sessions. Teachers will be able to post practice material and assign the material to students. Professors will be able to track student progress and tailor students practice exercises. This is not a replacement for existing teaching technologies but allows for more seamless and directed instrumental practice.

Our team worked with Douglas Lindsey, a trumpet professor from the Bailey School of Music at KSU who is the product owner to define requirements that allowed the team to construct the app to serve his needs. We used the first month of this course to solicit, document and design a **mockup** of the application using Figma. Once Professor Lindsey approved the design, we started to build the application in React Native, JavaScript, and Firestore.

Background

Douglas Lindsey is a trumpet professor, from the Bailey School of Music at KSU. For his trumpet classes, he was using an application called the GOLD method to coordinate his coaching lessons. The application had a few limitations including being solely a desktop application and the professor having to login into student's accounts to assign them exercises. Professor Lindsey was in search of an application that gave him more control and flexibility: a combination of the GOLD method and the workout applications, CrossFit and True Coach. Because he was having difficulty finding such an application to fit his needs, Professor Lindsey decided to seek out help to develop his own application. For our Senior Project the team decided to work with Professor Lindsey to develop a mobile application that met his desired requirements.

Requirements

Requirements were elicited from Professor Douglas Lindsey. We setup weekly calls where we would document was required of the mobile app, digest it after the call and during the next meeting reiterate what we understood. After a total of six weeks of gathering requirements, we were able to get approval from the professor that the mobile app requirements were correct. Below you will find the summarized requirements:

1. Login

	1.1 Login Page can register a new account using email and a password.
	1.1.1 New Account have one of two roles: Professor or Student.
	1.2 Login Page must authenticate an existing user.
	1.3 Login Page must have a forgot password feature.
2. Coaching	
	2.0 Teacher landing page must allow creation of a new Studio Semester.
{Professor Perspective} - Studio Semester	2.1 Studio Semesters must have a professor landing page.
	2.1.1 Studio Semesters have the following attributes: Registration Code (must be unique), Creator (i.e.. Professor).
	2.1.2 Studio Semesters are immutable.
	2.1.3 Studio Semester contains all registered students and Creator of studio semester (i.e. Professor) can view all students.
	2.1.4 Studio Semester allows Teacher to review student progress.
	2.1.5 Studio Semester allows the Teacher an ability to create routines and update existing routines.
{Professor Perspective} - Routines	2.2 Routines must have a professor landing page.
	2.2.1 Routines have the following attributes: Name, Length (time), Frequency (per week), List of exercises.
	2.2.2 Routines fall under one of these categories: Etude or Fundamentals.
	2.2.3 Routines landing page must allow for creation of new routines and updating existing routines.

{Professor Perspective} - Exercises	2.3 Exercises must have a professor landing page.
	2.3.1 Exercises have the following attributes: Name, Description, Sample Video [Optional], Starting Tempo, Goal Tempo, Weekly Repetition
	2.3.2 Exercises landing page must allow for creation of new exercise and updating existing exercise.
{Student Perspective} - Studio Semester	2.4 Studio Semesters must have a student landing page.
	2.4.1 This page must allow for registration into a new Studio Semester by entering the code.
{Student Perspective} - Routines	2.5 Routines must have a student landing page.
	2.5.1 Students will have read only access to Routines.
{Student Perspective} - Exercises	2.6 Exercises must have a student landing page.
	2.6.1 Student must be able to interface with exercise.
	2.6.1.0 System must be able to keep track of exercise progress.
	2.6.1.1 Student must be able to mark exercises are complete.
3. Feature: Tempo Algorithm	
	FEATURE NOT CALCULATED.
4. Feature: Student Video Recording	
	4.1 Student must be able to create a video from an exercise page.
	4.1.1 Video can be up to 30 seconds in length.
	4.2 Professor must be able to access videos posted by students.
	4.2.1 Professor must be able to provide feedback on each video in the form of a comment.
	4.2.2 Professor must receive a push notification once a student has submitted a video.
5. Feature: Exercise Video Sample	

	5.1 Exercises must allow for an embedded YouTube link video as one of the exercise optional attributes.
	5.2 Students must be able to play the YouTube link video in the app.
6. Feature: Chat Messaging	
	FEATURE NOT CALCULATED.
7. Feature: Live session scheduling (Teacher and Student)	
	FEATURE NOT CALCULATED.
8. Feature: Calendar (Students)	
	FEATURE NOT CALCULATED.

Technology Analysis

During the requirements gathering phase, the team brainstormed technologies that could be used to complete the application. Some key factors that the team needed to consider were that students and the professor were using both Android and iOS mobile devices, we had limited time to develop the application, and a mode of storage was needed for storing data the professor created.

Since we needed to create an application that could support Android and iOS mobile devices and we had limited time, the team brainstormed cross platform technologies. A cross platform framework or sdk, would allow the team to create one set of code that could be compatible on iOS and Android. The two cross platform framework/sdk we decided to focus on were React Native and Flutter. React Native is a framework based on the JavaScript library, React whereas Flutter is a sdk from Google that uses the Dart programming language. No one from our team had experience with using React Native but one team member had experience creating an application using Flutter. We all had some experience using JavaScript with one of our team members having prior knowledge of React. Flutter would require the team to learn Dart along with the Flutter UI which would require us to allocate more time for learning and take time away from development. Whereas with learning React Native, it would require less time for the team to learn since we all had familiarity with JavaScript. In addition, with Flutter it would be beneficial if the application needed animations. From the requirements gathering, the team realized that there is no need for animations. Through analyzing the two options, the team decided to go with React Native.

Early on through speaking with Professor Lindsey, the team realized that we needed to implement login capabilities for the students and professor as well as have a database to store data that is created by the professor or anything uploaded by both types of users (student and teachers). Our first option was to use SQLlite since it is a free, web based version of SQL and we all had some exposure through the Introduction to Databases Systems course at KSU. SQLlite worked fine for the database implementation but for the login implementation, the team faced difficulty with the functionality. As a result, we made the switch to Firebase since it is known to have straightforward authentication implementation and offers storage.

Through our research with Firebase,we saw that the Firestore database is an enhanced version of the Firebase that suited the needs of the application. Through Firestore, we can manage user accounts created as well as manage and store the data created by Professor Lindsey such as practice plans and practice types. Firestore is a cloud based NoSQL database which offers the benefit of us not having to worry about storing the data locally on our machines but access the data through the Internet and allow easy transfer of ownership. The downside of using a NoSQL database is the lack of primary and foreign keys and no tables which helps with keeping the data organized and makes it easier to understand the relationships between the different tables. Even so, the documentation offered by Google was easy to understand and React Native offered support for setting up the service which proved to make the implementation relatively straightforward.

An important aspect of development was implementing a solution for version control. Using version control allows the team to work on code locally then merge our changes together. We decided to use GitHub to host our code. GitHub uses the version control, Git which we all had familiarity with and is commonly used by software teams at many companies. Through using Git, we were able to fix merge conflicts and revert to previous commits when necessary. Through GitHub, we created an organization which allowed the entire team to have full access to the repositories and create two repositories: one for the React Native application and the other for the website.

Development

We began our development efforts by learning JavaScript and subsequently learning how to use the React Native framework. We then spent some time seeing how others wrote and organized their react native projects (what folder hierarchy was used, how they named their files and how they decoupled code).

We separated the project into three major parts: (1) Authentication, (2) Professor Functionality, and (3) Student Functionality. This worked well for us since we are a team of three individuals. The three parts were worked on in parallel. The authentication component was completed first. The professor functionality second and the student functionality last.

The authentication part has two major challenges. The first challenge was to build out the pages (ie. views) to match exactly the mockup in Figma. This included the graphical views as well as the backend logic for mobile device button tapping actions and redirection. The second challenge was to connect the backend of the authentication part to Firebase Authentication APIs. This included creating an account with Firebase and enabling both Android and IOS support. Once Firebase was setup, we had to research the APIs to interface our backend code with the Firebase Authentication service to allow central management of user account(s).

The professor and student functionality were tightly knitted and required close collaboration – so we will discuss this in one section. When starting development on this part, we had a mockup that told us how the mobile app pages should look like, and we had built out models to describe the data. We had decided to start our development using SQLite as a backend database which allowed us offline development to speed up the project progress. We could not find an Object-Relational Mapping (ORM) for React Native that could allow us to interface with SQLite database using objects. So, we took a functional approach and wrote temporary functions that allowed interfacing with the SQL database. In these functions, we included the raw SQL to create database, create tables, insert values, delete values, and update values. Fast forward, once the frontend was completed, we needed to move the backend to the cloud. We decided to use Firebase Firestore service to host our application backend database. This service is uses NoSQL which presented a challenge for us. This is because when we used SQLite, we deeply nested JavaScript asynchronous functionality to allow seamless integration when waiting on database calls that were required prior to displaying data on the mobile app pages. But when we moved to Firestore, we were seeing issues were functionality would sporadically fail. Later on, we found out that Firestore has its own asynchronous APIs that we needed to use and so we had to unravel all our JavaScript asynchronous behavior and migrate that to use the Firestore '.then()' API. The also coupled this with a loading indicator. This is because when we were using SQLite, database was local, and latency was non-existent. But when we moved to the cloud, we had inconsistent latency where one call could take a second like it could take five seconds. So, we accounted for this by adding a loading indicator to any pages that required database API call.

Please see the Design section for a deep dive into our mobile apps design and architecture.

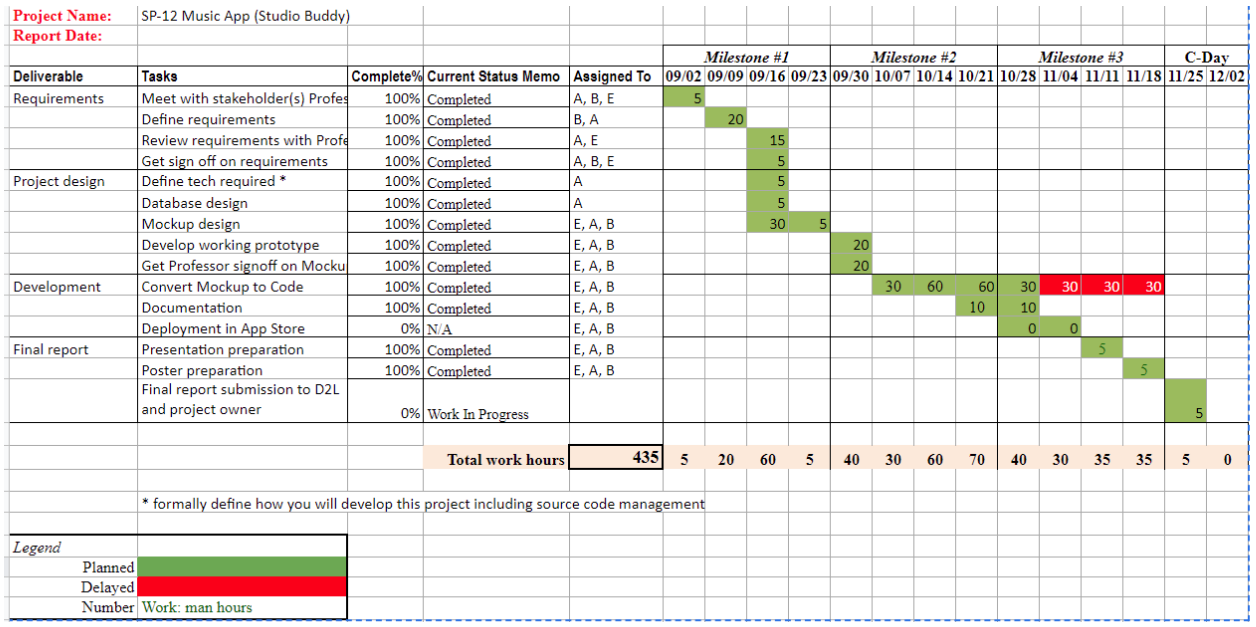
Project Planning & Management

In project planning and management, the team used multiple tools. As part of one of the deliverables for this course included created a Project Plan in which the team needed to provide a Gantt chart. The Gantt chart allowed the team, to properly allocate time each week to work on the project and to keep in mind which milestones were coming up on the schedule. To manage file sharing, the team decided to create a folder specific for this class that everyone has access to.

For meetings with Professor Lindsey and team meetings, we used Microsoft Teams since it is an application offered by KSU. We made sure to have more meetings with Professor Lindsey in August and September since that was the period that where we were gathering the requirements. In October and November, the team had some meetings to give Professor Lindsey an update on where we were with application development. Using email to communicate with Professor Lindsey also proof to be beneficial in asking clarifying questions, giving updates, and scheduling meetings.

In October, the month in which the team started constructing the application using React Native, we decided to have weekly check ins on Thursdays to discuss our progress for the week. In addition, the team would meet after the Senior Project class to discuss any concerns.

For managing our code for the, the team used GitHub as mentioned previously in the Technology Analysis. We worked primarily with one branch and push code directly. If any merge conflict arose when pushing changes, the team would deal with them manually.



Project Name: SP-12 Music App (Studio Buddy)
Report Date:

					Milestone #1				*Milestone #2*				*Milestone #3*				C-Day	
Deliverable	**Tasks**	**Complete%**	**Current Status Memo**	**Assigned To**	09/02	09/09	09/16	09/23	09/30	10/07	10/14	10/21	10/28	11/04	11/11	11/18	11/25	12/02
Requirements	Meet with stakeholder(s) Profes	100%	Completed	A, B, E	5													
	Define requirements	100%	Completed	B, A		20												
	Review requirements with Profe	100%	Completed	A, E			15											
	Get sign off on requirements	100%	Completed	A, B, E			5											
Project design	Define tech required *	100%	Completed	A			5											
	Database design	100%	Completed	A			5											
	Mockup design	100%	Completed	E, A, B			30	5										
	Develop working prototype	100%	Completed	E, A, B					20									
	Get Professor signoff on Mocku	100%	Completed	E, A, B					20									
Development	Convert Mockup to Code	100%	Completed	E, A, B						30	60	60	30	🟥 30	🟥 30	🟥 30		
	Documentation	100%	Completed	E, A, B								10	10					
	Deployment in App Store	0%	N/A	E, A, B									0	0				
Final report	Presentation preparation	100%	Completed	E, A, B											5			
	Poster preparation	100%	Completed	E, A, B												5		
	Final report submission to D2L and project owner	0%	Work In Progress														5	
			Total work hours	435	5	20	60	5	40	30	60	70	40	30	35	35	5	0

* formally define how you will develop this project including source code management

Legend	
Planned	🟩
Delayed	🟥
Number	Work: man hours

Figure 1: Team Gantt Chart

Design

Mockup

A mockup was built based off the elicited requirements. The design was done using a tool called Figma, which is a free, web-based prototyping tool, popular with web and application designers. The mockup design was iterative as we worked with Professor Lindsey in ensuring that the design matched both his requirements and was visually what he had envisioned. The team broke down the design into three parts: Login view, Teacher view, and Student view. The two types of users are a teacher (in this case specifically Professor Lindsey) and his students. Because the teacher and his students have different goals for using the app, for example the teacher would be creating and modifying content as well as managing his students while the students will be viewing content the teacher created, the screens for each type of user would differ. Both the professor and students would need to login into their created accounts to see their specific content.

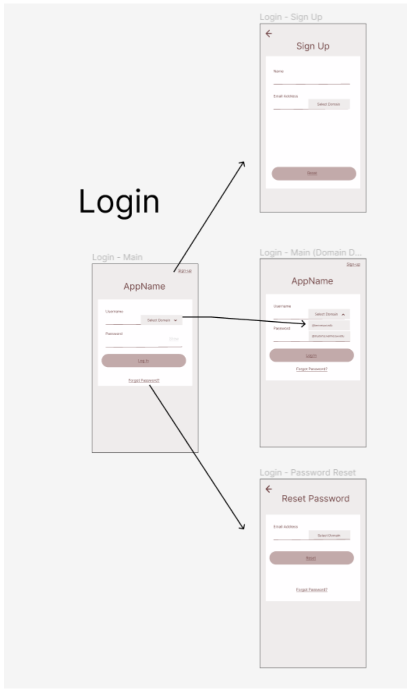


Figure 2: Figma Authentication Mockup

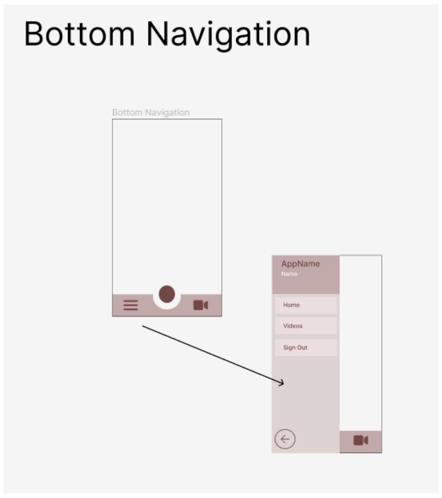


Figure 3: Figma Bottom Navigator Mockup

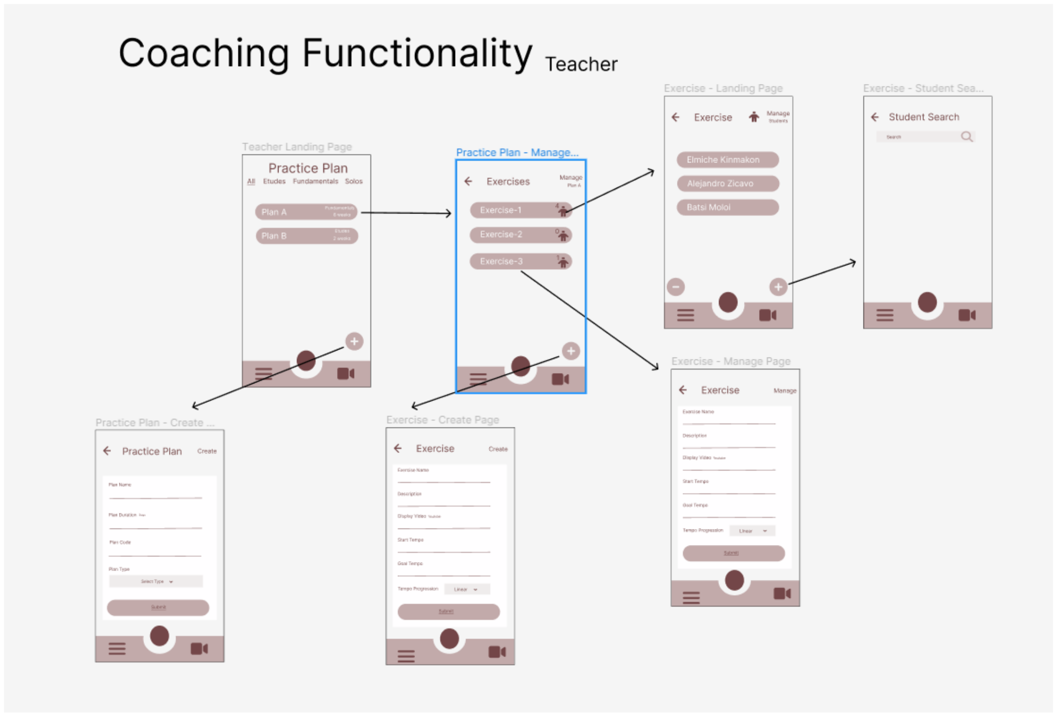


Figure 4: Figma Coaching Functionality (Professor) Mockup

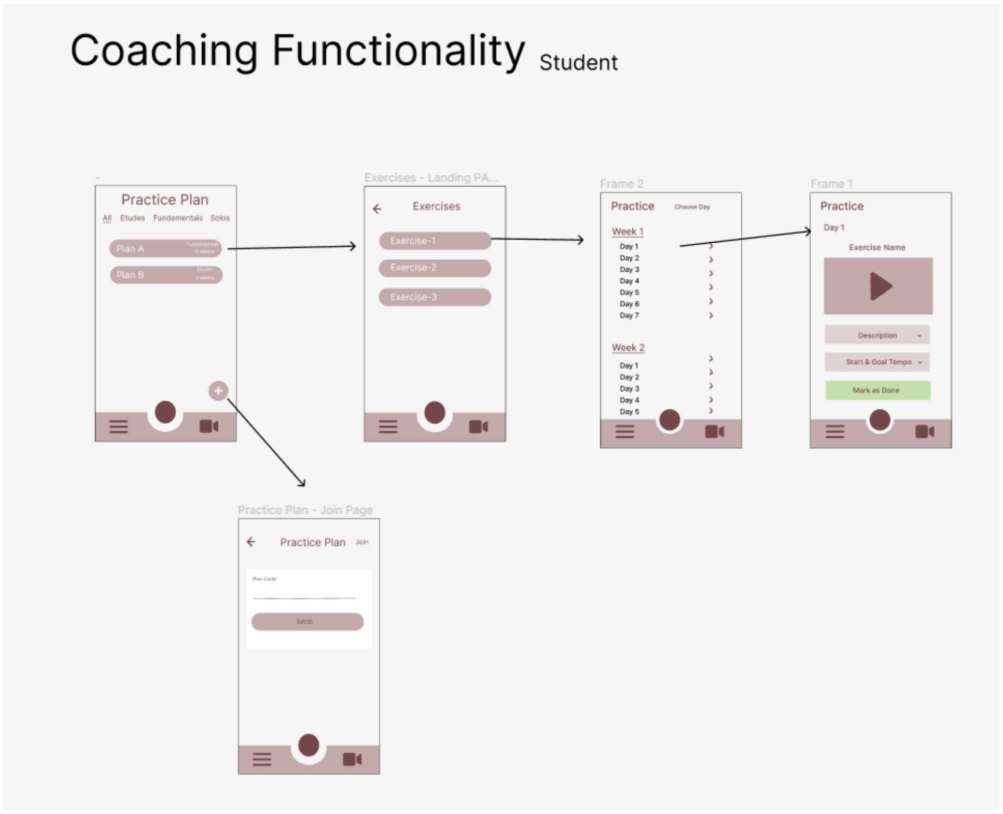


Figure 5: Figma Coaching Functionality (Student) Mockup

Architecture

The Mobile app was organized into four major sections. These sections are depicted in the high-level interactions diagram on Figure 6. At the core, a professor required the creation of Exercise(s). These Exercise(s) were organized into Practice Plan(s). The Practice Plan(s) have a defined Practice Type. Lastly, an Exercise can be Practiced. In general, a professor was given create, update, and delete functionality while a student was given only read functionality. These sections are talked about in more depth on the next section titled '*Database*'.

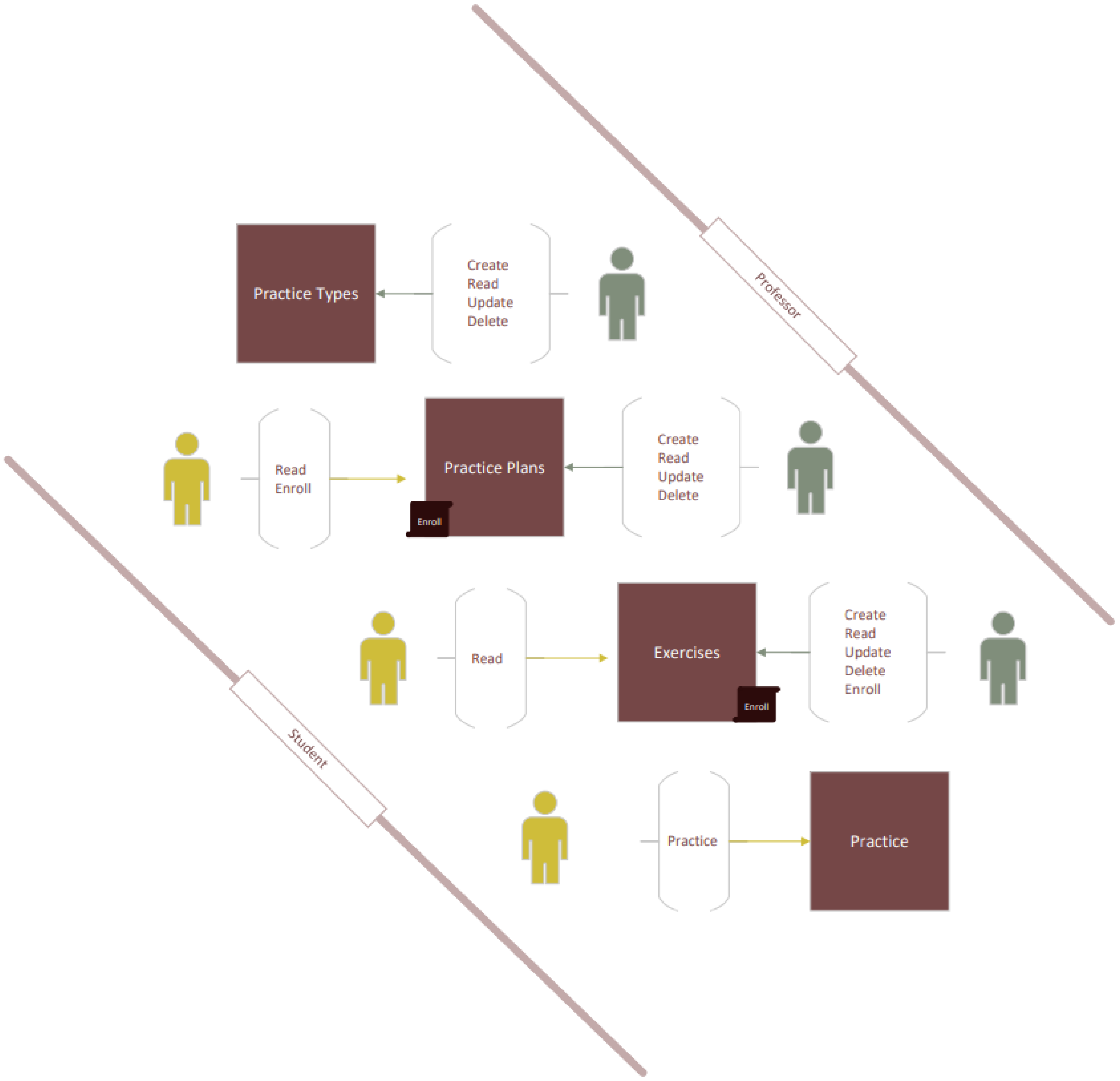


Figure 6: Diagram showing high-level interactions between a Professor and a Student.

Database

The database implementation was done in both SQL using SQLite and NoSQL using Firebase Firestore API(s). Both implementations were done independently, and even though they are very comparable, subsequent discussion will only focus on the NoSQL database since this is what ultimately made it to our final product.

We used Firebase Authentication service to manage our users and ensure their integrity of their credentials in a store safeguarded by Google Firebase. Consequently, we realized that professors did not have access to information that was needed to review their students. This was because the authentication service essentially segregated users from each other – this was a security feature. So, what we did is we created a User Settings collection. This collection was responsible for holding a User UID (which is unique and created by Firebase Auth) and storing the following attributes for a given user:

A.	Display Name	*Mutable*
B.	Email	*Immutable*
C.	User UID	*Immutable*

With these attributes, we were able to use a student and professors User UID in other collections and be able to reference their email and display name.

Next, we needed to be able to maintain a list of Practice Types and per requirements, a professor needed to be able to create, update and delete Practice Types. We created a collection that hosted documents with the following attributes:

A.	Name	*Mutable*
B.	Sub Type	*Mutable*
C.	User UID	*Immutable*

The name attribute is that name of the Practice Type. The sub type attribute is to allow an additional tag where needed. The User UID is the ID for the Professor who manages that Practice Type on the collection. Essentially, to find the available practice types of a given professor, we would search all practice types were User UID was equal to the professors UID.

Next, we needed to create Practice Plan collection that will later be used to organize exercises. We created a collection that hosted documents with the following attributes:

A.	Code	*Immutable*
B.	Duration (in days)	*Mutable*
C.	Name	*Mutable*
D.	Practice Type UID	*Mutable*
E.	User UID	*Immutable*

The name attribute represents the name of the Practice Plan. The duration represents the length of the practice plan in days. The Practice Type UID represents an existing practice type already tied to the professor. The User UID is the professor who owns the Practice Plan. Lastly, the code is a globally unique code that students use to register to a Practice Plan.

Next, we needed a collection to tie a student User UID to a Practice Plan UID. A collection was created for this purpose with the following attributes:

A.	Practice Plan UID	*Immutable*
B.	User UID	*Immutable*

Intuitively, the Practice Plan UID points to the Practice Plan while the User UID points to a unique student that is enrolled into the Practice Plan.

Next, we needed a collection to track exercises. We created a collection with the following attributes:

A.	Name	*Mutable*
B.	Description	*Mutable*
C.	Video Link	*Mutable*
D.	Start Tempo	*Mutable*
E.	Goal Tempo	*Mutable*
F.	Tempo Progression	*Mutable*
G.	Practice Plan UID	*Immutable*

The name attributes is the name of the Exercise while the description is used to provide more clarity to students in terms of what is expected of them during this exercise. The video link is an optional value where a professor can post a YouTube video with material, they would like a student to preview. Start tempo, goal tempo and tempo profession are all exercise parameters. The first two define a tempo they should follow. The Tempo profession is an algorithm that defines how a student should be doing from a starting tempo to eventually reaching a goal tempo. Currently, only a Linear progression was defined which increases starting tempo linearly until it reaches the goal tempo based on the duration of the Practice Plan. Lastly, an exercise is tied to a Practice Plan and thus we need the UID.

Next, we need a way to tie a user to an exercise. This is a requirement that allows the professor granular control to assign any exercise to student(s). A collection with the following attributes was created to track the student enrollment into exercises:

A.	Exercise UID	*Immutable*
B.	User UID	*Immutable*
C.	Start Date	*Immutable*

Intuitively, the Exercise UID is tied to a student UID with an additional start date. The start date is based on the requirement that a professor must be able to select a start date for a given exercise – that is a date when the student should see the exercise so that they can Practice it.

Results

Some of the requirements that we originally planned to complete in Phase 1, we had to push to Phase 2.

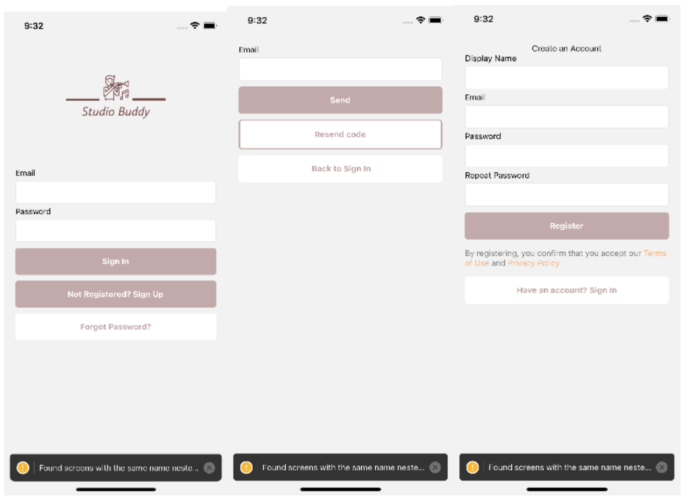
Our application was created using React Native as originally planned. For the login, we were able to create all the screens and implement the capability for KSU faculty and students to create an account and sign in. To reset passwords, KSU blocks emails from Firebase and we were not able to come with an alternative solution. The backend logic for the login is handled through Firestore.

For the Teacher view, the professor can create practice plans which students can enroll in and exercises for those practice plans. The professor can update and delete practice plans and exercises. In addition, the professor can create and update practice types. Additionally, the professor can see the students who are enrolled in his classes. We used Firestore to store the practice plans and practice types that the Professor creates.

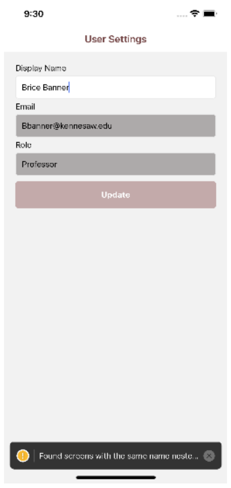
For the Student view, the student can join a practice plan, view practice plans they are enrolled in, and see the exercises assigned.

Below are the screenshots of the screens the team developed so far.

Login Views



Shared Views



9:30

User Settings

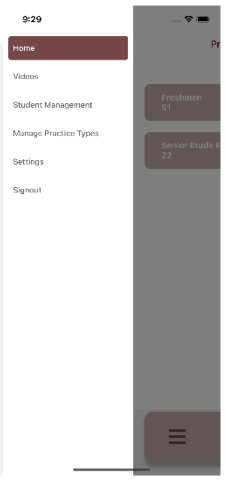
Display Name: Brice Banner

Email: Bbanner@kennesaw.edu

Role: Professor

Update

Found screens with the same name neste...



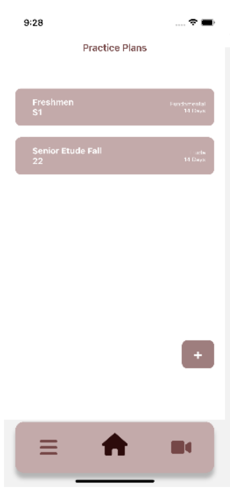
9:29

Home
Videos
Student Management
Manage Practice Types
Settings
Signout

Freshmen S1

Senior Etude F 22

Professor Views

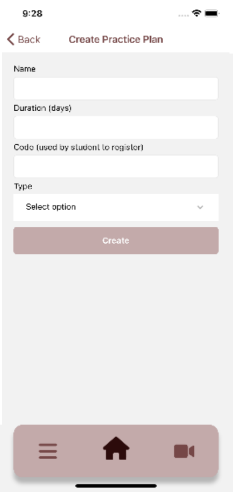


9:28

Practice Plans

Freshmen S1

Senior Etude Fall 22



9:28

Back   Create Practice Plan

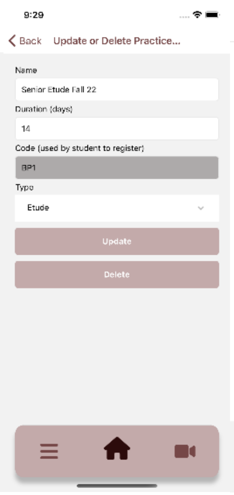
Name

Duration (days)

Code (used by student to register)

Type: Select option

Create



9:29

Back   Update or Delete Practice...

Name: Senior Etude Fall 22

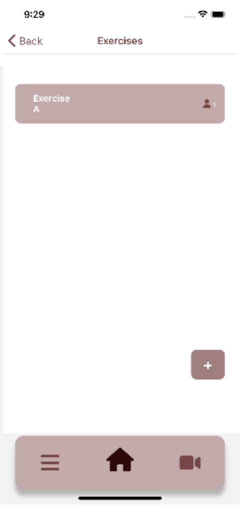
Duration (days): 14

Code (used by student to register): BP1

Type: Etude

Update

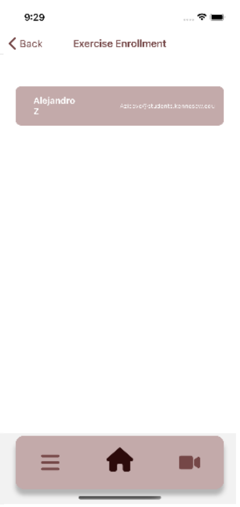
Delete



9:29

Back   Exercises

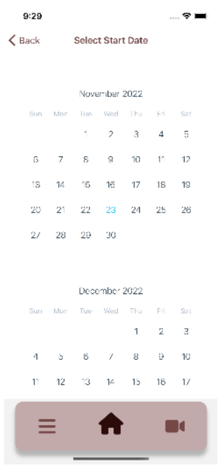
Exercise A



9:29

Back   Exercise Enrollment

Alejandro Z



9:29

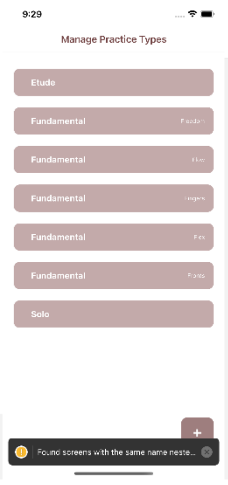
Back   Select Start Date

November 2022

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

December 2022

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17

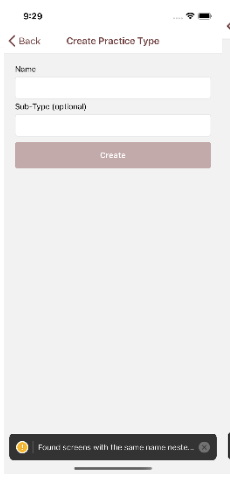


9:29

Manage Practice Types

Etude
Fundamental
Fundamental
Fundamental
Fundamental
Fundamental
Solo

Found screens with the same name neste...



9:29

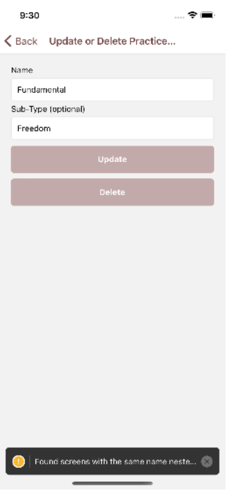
Back   Create Practice Type

Name

Sub-Type (optional)

Create

Found screens with the same name neste...



9:30

Back   Update or Delete Practice...

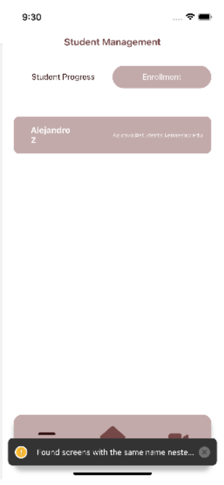
Name: Fundamental

Sub-Type (optional): Freedom

Update

Delete

Found screens with the same name neste...



9:30

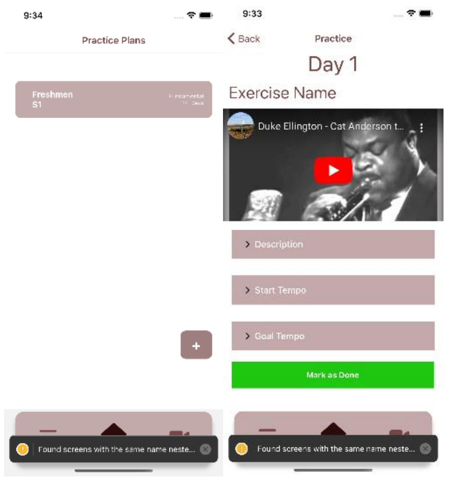
Student Management

Student Progress   Enrollment

Alejandro Z

Found screens with the same name neste...

Student Views


9:34
Practice Plans
Freshmen
S1
+
Found screens with the same name neste...
9:33
Back
Practice
Day 1
Exercise Name
Duke Ellington - Cat Anderson t...
Description
Start Tempo
Goal Tempo
Mark as Done
Found screens with the same name neste...

We created a website that gives an overview of the application as well as a place to view the deliverables from the course and C-Day such as the poster, project plan, and demo. The website was created using HTML and CSS.

Test Plan & Report

Testing was performed and results indicate Mobile app is ready for production. There was one failure which was the Password Reset option. In our testing, it looks like KSU email filtering is blocking the reset link sent by Firebase. There are other additional results marked as N/A, these indicate we could not test this feature as it was not implemented.

Section	Test	Test Result
1. Login	1.1 Login Page can register a new account using email and a password.	PASS
	1.1.1 New Account have one of two roles: Professor or Student.	PASS
	1.2 Login Page must authenticate an existing user.	PASS
	1.3 Login Page must have a forgot password feature.	FAIL
2. Coaching	2.0 Teacher landing page must allow creation of a new Studio Semester.	PASS
{Professor Perspective} - Studio Semester	2.1 Studio Semesters must have a professor landing page.	PASS
	2.1.1 Studio Semesters have the following attributes: Registration Code (must be unique), Creator (ie. Professor).	PASS
	2.1.2 Studio Semesters are immutable.	PASS
	2.1.3 Studio Semester contains all registered students and Creator of studio semester (ie Professor) can view all students.	PASS
	2.1.4 Studio Semester allows Teacher to review student progress.	PASS
	2.1.5 Studio Semester allows the Teacher an ability to create routines and update existing routines.	PASS
{Professor Perspective} - Routines	2.2 Routines must have a professor landing page.	PASS
	2.2.1 Routines have the following attributes: Name, Length (time), Frequency (per week), List of exercises.	PASS
	2.2.2 Routines fall under one of these categories: Etude or Fundamentals.	PASS
	2.2.3 Routines landing page must allow for creation of new routines and updating existing routines.	PASS
{Professor Perspective} - Exercises	2.3 Exercises must have a professor landing page.	PASS
	2.3.1 Exercises have the following attributes: Name, Description, Sample Video [Optional], Starting Tempo, Goal Tempo, Weekly Repetition	PASS
	2.3.2 Exercises landing page must allow for creation of new exercise and updating existing exercise.	PASS
{Student Perspective} - Studio Semester	2.4 Studio Semesters must have a student landing page.	PASS
	2.4.1 This page must allow for registration into a new Studio Semester by entering the code.	PASS
{Student Perspective} - Routines	2.5 Routines must have a student landing page.	PASS
	2.5.1 Students will have read only access to Routines.	PASS

{Student Perspective} - Exercises	2.6 Exercises must have a student landing page.	PASS
	2.6.1 Student must be able to interface with exercise.	PASS
	2.6.1.0 System must be able to keep track of exercise progress.	N/A
	2.6.1.1 Student must be able to mark exercises are complete.	PASS
3. Feature: Tempo Algorithm	FEATURE NOT CALCULATED.	N/A
4. Feature: Student Video Recording	4.1 Student must be able to create a video from an exercise page.	N/A
	4.1.1 Video can be up to 30 seconds in length.	N/A
	4.2 Professor must be able to access videos posted by students.	N/A
	4.2.1 Professor must be able to provide feedback on each video in the form of a comment.	N/A
	4.2.2 Professor must receive a push notification once a student has submitted a video.	N/A
5. Feature: Exercise Video Sample	5.1 Exercises must allow for an embedded YouTube link video as one of the exercise optional attributes.	PASS
	5.2 Students must be able to play the YouTube link video in the app.	PASS
6. Feature: Chat Messaging	FEATURE NOT CALCULATED.	N/A
7. Feature: Live session scheduling (Teacher and Student)	FEATURE NOT CALCULATED.	N/A
8. Feature: Calendar (Students)	FEATURE NOT CALCULATED.	N/A

Conclusion

As have closed out the last phase of this project, we have a few takeaways from this project. We have gained an increased respect for Software Engineering and project management. It took our team around six weeks to finish collecting requirements and finish the mockup (and approved by stakeholder). We believe it took us this long because eliciting requirement sounded a lot easier than it was; the reality was that we would go back and forth redefining what we understood was needed from the app multiple times. Another takeaway was that managing a Software Development project was not effortless; it was a constant back and forth between developing code, reviewing mockup and keeping stakeholder up to date on where we are, where we are going next, and ensuring we were incorporating the stakeholder's feedback into the project.

Appendix

User Guide

https://drive.google.com/file/d/1X4dtjE9qsNf6HrEHDY4WCg63VrpxcELZ/view?usp=sharing

Quick links

Project Website https://sp12-studio-buddy-website.netlify.app

GitHub Project https://github.com/sp12-music-scheduling

Marketing

UG-253

STUDIO BUDDY

Create · Practice · Collaborate

Enables music professors to customize and tailor the coaching experience for students.

Tubatsi Moloi
Alejandro Zicavo
Elmiche Kinmakon

Developed in collaboration with Dr. Douglas Lindsey from the Bailey School of Music.

9:29
Studio Buddy
Email
Password
Sign In
Not Registered? Sign Up
Forgot Password?