Take and Teach Project Report

Team Members: Abyel Romero, David Oligney, Goma Niroula

Client: Jason White

I. About

TakeNTeach is an app created for two “user” types, enabling learning content taught by certified instructors to be purchased by the application’s user’s. Instructors have the ability to create courses, upload videos to their courses, set prices, add tags for searching, and publish their courses for users to purchase. Instructors are able to later update their course information or delete their courses should any information need to be changed. Users have the ability to search for courses to purchase and can subsequently view all of their purchased courses in one place. Users also have the ability to organize their learning content by creating folders to sort their pictures and videos. Take and Teach aims to bridge the gap between users and instructors by allowing the users to email their instructors with any questions they may have about the course material so they can benefit from their instructor’s feedback and improve upon their work. If the user’s needs go beyond the capabilities of email, instructors have the ability to create calendar events with zoom links for one-on-one or group classes to further explore their student’s needs. Our goal for this project was to provide an app which fosters a student-teacher dynamic and allows easy communication for user’s to ask questions, make mistakes and improve based on instructor feedback, all of which TakeNTeach includes features for.

II. Our Process

A. Gathered Requirements

We met with our client to understand the features he wanted in the TakeNTeach app. We received previous documentation which allowed us to understand the app and its features better and created a requirements document which detailed everything we knew about what our client wanted for the app.

B. Picked Technologies

We researched various technologies while keeping in mind our requirements and our clients budget. We settled on using Ionic which satisfied our cross-platform requirements, chose MongoDB and Google Cloud Service for our backend, and went with Heroku to host our app.

C. Divided Work

We created a timeline detailing the different features we needed to implement with tentative dates for completion and assigned each feature to a member of our group.

D. Started Coding

We created an Ionic project with Capacitor dependencies. We also created a Github account so that we could collaborate without any issues. We created a repository for our API as well as the App. We pushed the empty Ionic project we had created so that each of us could start working on each of our features. We watched tutorials for Ionic and Angular to learn the requirements of the technologies we chose and began coding our initial UI, and later implemented the backend.

E. First Round of Testing

We created an Apple account to use Testflight Beta Testing. We pushed our code to Test Flight, added test users and created a public link for the App. We sent an email to our testers which included the instructions on what to test as well as links to the feedback forms and the public application download link. We received and read the feedback from the testers about what worked and what didn't work to fix various issues..

F. Changes

Based on the feedback from our testers, we fixed each of our individual features that were not working. We also fixed all of the little bugs that the testers noticed, for example small UI errors or navigation issues.

G. Second Round of Testing

We pushed an updated build to TestFlight for our second round of testing and sent a second testing email out which included the updated instructions, updated feedback form, and the app link. We got feedback from the testers on everything that was working after our changes and any other bugs they found.

H. Final Changes

We made the final round of changes to our app based on the feedback from the testers as well as the smaller things that we noticed.

III. What feedback we got from testing and how we fixed the issues.

A. Feedback:

1. Instructor create course/upload video page was not working because the app wasn't registering the fact that the instructors were logged in so it was not letting them create the course.
2. Instructors weren't able to manage the courses because they weren't able to create it.
3. Users could purchase the same video multiple times.
4. Users were not able to stream the videos they purchased.
5. Users were not able to search for the courses.

B. Changes:

1. Before the users/instructors were automatically taken to their home page so it was not registering they were logged in so we changed the routing and instead took them to the login page. This fixed the issue. Instructors were able to create courses and upload videos to that course.
2. Instructors were able to manage courses now that they could create courses.
3. We changed the code so that once the users have purchased the video, they can't purchase it anymore.

IV. How the Project Changed (Requirements vs Final Product)

A. Our client wanted a gallery section with folder functionality which we had not included in our initial requirements, and had to incorporate with limited time. We did some research and found out that the technology we were using did not have many useful plugins that we could use for this purpose. After more research, we put together something that let the user create folders, but were unable to get additional functionality working beyond the creation of the folders

B. Our client wanted instructors to be verified by emailing the instructor information someone qualified to verify the instructor. The application was able to send basic emails including the user or instructor information, but we could not get functionality working to include anything allowing the account to be created based on something clicked from the email so the email functionality was taken out.