CSCI 337 - Note-taking app - Final Report

Logan Humbert

May 7, 2024

Contents

List of Figures

1 Description of the application

This little mobile application aims to let students taking notes for their different classes, and organizing them by class.

2 Requirements

- **Who are the users?** The users are students (College, High School, Middle School, ...)
- **Who are the customers?** This app is only made in a school context. There is no real customer for it.
- **User stories:**
  - As a new user, I want to add my different classes, and start adding notes for them
  - As a regular user, I keep adding a new note during a lecture in my different classes, then I review them when I study. I can easily modify the content of the note if I need to.

3 Features

The user should be able to complete these different tasks easily:

- See his different classes
- Add/edit or delete a class
- Create/edit/delete a note for any class he has created
- View a note
- Search for a class or a note by typing a part of its name

4 Iterative design

4.1 Sketches on paper

Here is a first sketch of the different pages of the app, and the transitions between them:

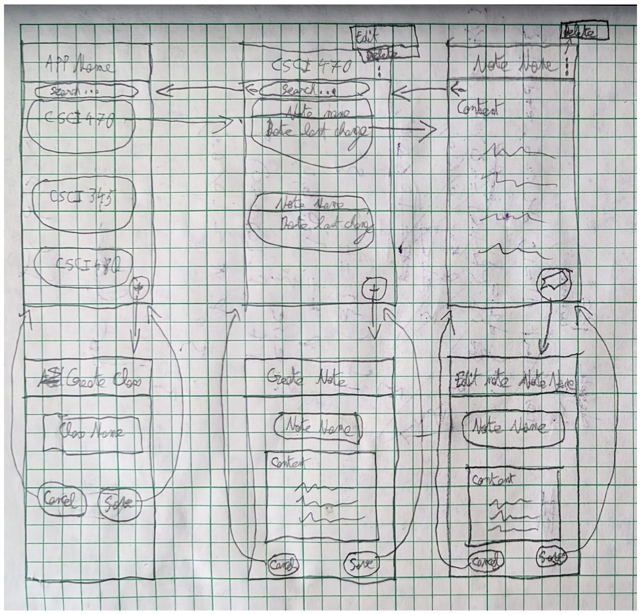


Figure 1: First sketch on paper

I imagine having 5 different screens:

- The home page, containing a list of the existing classes.
- The add/edit class form
- The list of notes in a class
- The add/edit note form
- The content of a note

All pages can be accessed by clicking on a specific element, as the arrows on the sketches show.

4.2 Designing and prototyping on Figma

After drawing my first sketches on paper, I started making a more accurate and clean design of the UI on Figma, and I used its prototyping features to create the transitions between the pages of the app.

For developing my app I decided to use Flutter, as I will explain more in details in the next section. As this framework uses the Google Material's design, I used their official figma template, which contains reusable Material component, so my Figma design could be as close as possible as the UI the final app will have.

You can access my figma design here.

I started by making the User Story "Create a class"

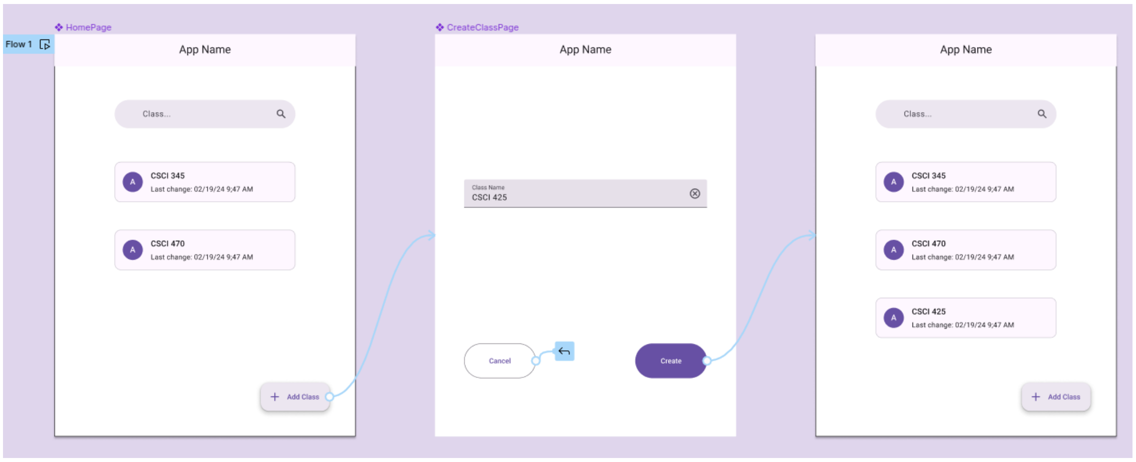


Figure 2: User story "Create Class" on Figma

From the home page that contains the list of our classes, a button lets the user create a class by giving it a name. It is then immediately available from the home page.

Then I added the other user stories and added transitions between the different pages of the app:

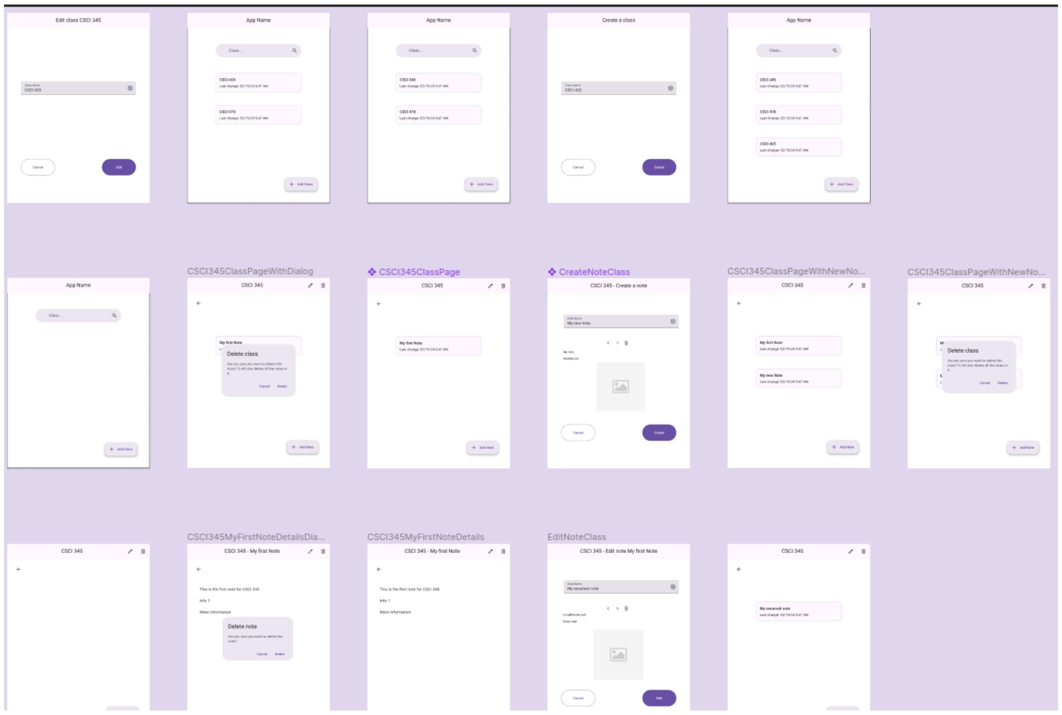


Figure 3: All user stories on Figma

Some pages of the app have been duplicated, as they simulate a change in database, or they display an additional graphical element, such as a dialog box, which is not possible to achieve directly on Figma. These are used to obtain a better experience when we execute the prototype on figma.

4.3 Development with Flutter

After obtaining a first satisfying design and prototyping my app on figma, I started to develop my app.

I decided to use Flutter, which is an open-source framework created and maintained by Google. Its main advantage is that it allows building natively compiled multi-platform applications from a single codebase.

I focused mainly on making a web version as it is one of the easiest versions to debug, and it would allow my app to be available on any platform that has a web browser, such as windows and mac computers, Android and IOS phones and tablets.

Anyway, it would be very easy to build a native Android or IOS app for example, with none or just a few modifications of the code needed.

The source code and the instructions to build and run the Flutter app are available here.

4.4 Publishing the app

After judging that my app was complete and stable enough I decided to publish it.
For this I used a service named Github pages, that allows any Github repository to be accessed as a website.
The only cons of it is that it we cannot serve a nodeJS, apache or nginx server for example with this service.
As Flutter apps are don't require any of these, Github pages is a good choice.

The latest public version of my app is available here.

4.5 UI/UX tests

After developing and publishing my app I decided to do UI/UX tests. For this, I asked other students, who were potential future users of my app to try it and give me feedback on the things they liked/disliked about the app, and reports of bugs.

Thanks to this useful information, I have been for example able to improve both the view that lists classes and the view that lists notes in a class.

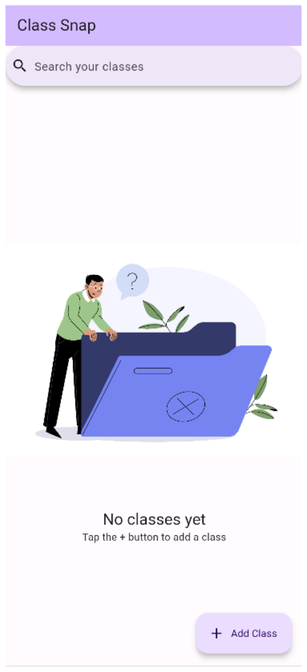


Figure 4: Home page with empty list of classes

The graphics and text on the screenshot above displayed whenever no class has been created yet, were not present at first. Therefor, some users did not know what to do and did not necessarily see the "Add Class" button in the lower right corner of the page. Running UI/UX tests helped me being aware of the issue and addressing it.

If these tests helped me improving some aspects of the app, they also helped me detecting bugs I did not noticed while I was developing the app and fixing them.

For example, if we named a class or a note with a '/' in it and we tried to access it, the application threw an error. This is because this special character is used in the routes leading to a class or a note, so the route was invalid in this specific case.

5 App improvements

After obtaining a first working version of my app, completed by UI/UX tests, I decided to work on new improvements for it.

5.1 Export as pdf

One major feature absent from the app is the possibility to export the notes. I decided to implement a pdf export feature to begin with.

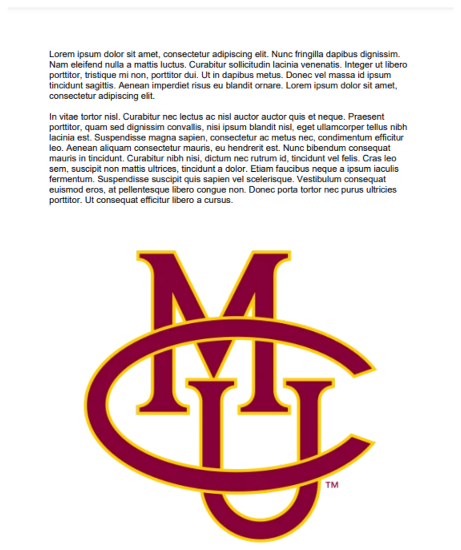
I use the flutter_quill library to make the notes editor and its toolbar. Officially, this library only supports exporting the text as html or raw text. To be able to export my notes as pdf, I explored two possibilities:

- Find a library that is directly able to convert the content of the text editor into a pdf.
- Find a library that can convert html into pdf.

I started with the first option I had in mind. I found some libraries made for flutter_quill able to directly export the text from the editor to a pdf file, but they were either deprecated and incompatible with newer version of flutter_quill, or were only compatible with Android, not with the web version of the app.

After multiple fails, I decided to try the second option. It is easy to export a note as html using the tools made available by flutter_quill. Thanks to the library htmltopdfwidgets I have been able to take this html text as input to create a pdf files.

Then, on the web I create a base64 url to download the file, and on Android and IOS, I write the file as binary in the external storage.


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc fringilla dapibus dignissim. Nam eleifend nulla a mattis luctus. Curabitur sollicitudin lacinia venenatis. Integer ut libero porttitor, tristique mi non, porttitor dui. Ut in dapibus metus. Donec vel massa id ipsum tincidunt sagittis. Aenean imperdiet risus eu blandit ornare. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

In vitae tortor nisl. Curabitur nec lectus ac nisl auctor auctor quis et neque. Praesent porttitor, quam sed dignissim convallis, nisi ipsum blandit nisl, eget ullamcorper tellus nibh lacinia est. Suspendisse magna sapien, consectetur ac metus nec, condimentum efficitur leo. Aenean aliquam consectetur mauris, eu hendrerit est. Nunc bibendum consequat mauris in tincidunt. Curabitur nibh nisi, dictum nec rutrum id, tincidunt vel felis. Cras leo sem, suscipit non mattis ultrices, tincidunt a dolor. Etiam faucibus neque a ipsum iaculis fermentum. Suspendisse suscipit quis sapien vel scelerisque. Vestibulum consequat euismod eros, at pellentesque libero congue non. Donec porta tortor nec purus ultricies porttitor. Ut consequat efficitur libero a cursus.

Figure 5: Example of note exported as pdf

5.2 Cloud Storage

After adding support for exporting notes as pdf, I started thinking about adding support for cloud storage.

On Figma, I created a new user story representing the pages shown to the user the first time he opens the app. He can login/register to an account (email/passord, Google, Twitter, Apple, ...) or skip the process to use the app without any account.

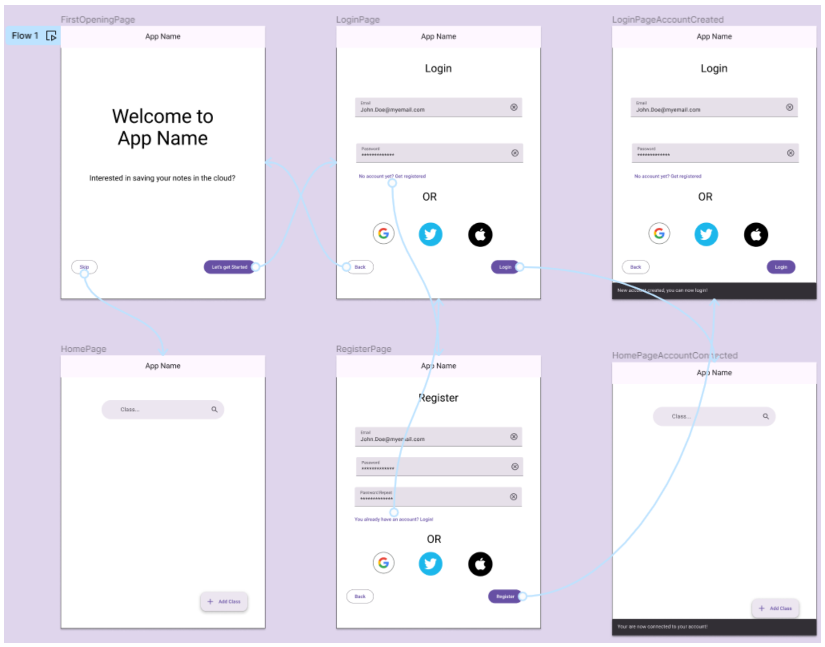


Figure 6: User story "First app opening" on Figma

Then I created a second user story representing the connection to an account from the home screen of the app, using a new button placed on the top app bar.

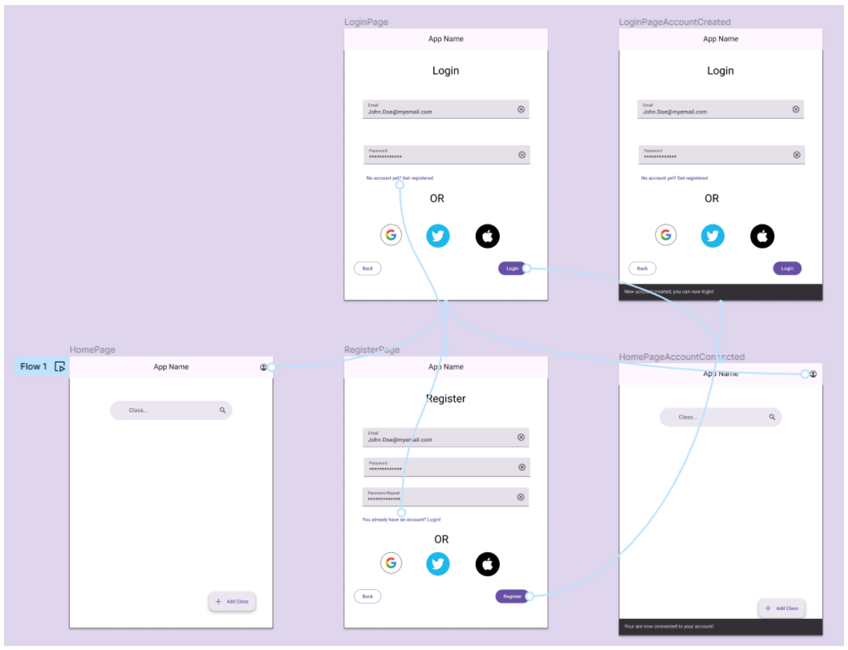


Figure 7: User story "Connect to an account" on Figma

6 Conclusion

The development of this note-taking application employed an iterative design approach to create a user-friendly tool for students.

From initial paper sketches to Figma prototyping and ultimately a Flutter-based web application and then UI/UX tests, this project has let me exploring different important aspects of the creation of applications.

I have been able to respect the different requirements I have set for the app and to implement the different features features I wanted to have at first.

Future development could explore additional features like:

- Add more useful features such as the possibility of adding hand-drawn notes
- Add cloud storage support (for example via Firebase which has a great compatibility with Flutter)
- Add support for exporting notes as word documents

7 Links

Here are the links to important resources:

- Project's repository
- Figma Design
- Web version of the app