Number plate recognition application on Flutter

Yassir Hoossan Buksh

1st March 2024

Table of Contents

Number plate recognition application on Flutter....1
Yassir Hoossan Buksh....1
1st March 2024....1
Introduction....3
Description....4
Methodology....5
Code Methodology....6
“Common” Code....6
Design Methodology....8
Minimalism Fluxed With Functionality....8
Design....9
Strengths....15
Offline....15
Strong Algorithm....15
Modular....15
Clean UI....15
Weaknesses....16
Too strong....16
Missing Functionalities: Clean Database UI....16
Missing Functionalities: Home feed recents....16
Conclusion....17
Future Work....17
References....18

Introduction

An important part of developing an application is to view it as a project that is both modular in functionality and design. Flutter is a great tool to represent that general idealization in programming due to it's ability to achieve cross-platform connections with ease, and provide a strong foundation for a powerful and complex application. With these goals in mind -

My name is Yassir Hoossan Buksh. I am a third year student at the University Of Mascareignes (UDM), Mauritius. This project is a flutter application for the scanning of number plates. The application was completed on The 1st of March of 2024.

Description

To describe the app at it's core, it's a relatively unembellished OCR application. It allows the user to pick between taking an image or scanning from an existing image in gallery. To perform the scan I ended up using **google's ml kit** and the **image picker library**, as a whole, the application is capable of what is asked of it, without too much fluff in terms of features and capabilities.

In addition to being able to scan number plates, I thought to include the use of a simple **CRUD SQLITE** database system. When scanning an image and retrieving the text, the user can choose to add it to the database and then view previous scans.

This feature was added in a relatively primitive way. It isn't totally complete in features (being able to edit the records or deleting specific ones, or adding ones without scanning would likely be good candidates for updates to the app).

Methodology

In this section I detail the various methodologies involved in the development of the application. These methodologies include both design and code methodologies in 2 different sections. I believe that my design methodologies don’t necessarily align with the ones found in the code.

Code Methodology

"Common" Code

During this project I had a strangely different approach to coding, that had to do with the bulk of documentation and resources for the flutter package. In jest, the idea was to write code that was primarily *easy* to understand and especially to *work* with. The second importance being to write efficient code that was *readable*.

I think the difference holds quite a bit of weight when designing an application. Truthfully, all things considered, it isn't such a drastic difference that performance is ever a concern, but it does influence design decisions at play in the code itself.

To explain what I mean; it's not that the code is written to be unreadable, or to be difficult to work with, but it is distinctly *flutter* code. It is essentially the same as the concept of *pythonic* approaches in python. Following the norm of the language, instead of trying to re-invent the wheel.

My motivations for this are various, but most poignant was that for the most part, dart is compiled code that is executed as C code. Most of these libraries are existing portions of code that have had their gears running for quite some time now. There is no shortage of optimization in this code to begin with.

Readability wise? Logically, more readable code is better for the average newcomer to a project or language, but there's an advantage here to making the code similar to what one would find when trying to learn the language itself. The familiarity and similarity of what can be found in online resources strikes a strong bond with debugging, because you can easily find similar implementations of the code in different portions.

Naturally this isn't to say that originally just goes out of the window. Every line of code in this project was written by me, and I went through the effort of understanding those lines of code, and making something congruent of it that I can debug and fix if any of it goes awry. Yet, I am confident that any developer that has worked with flutter and dart before, could easily, and rather rapidly, understand what my program is doing. I wouldn't necessarily say that a junior developer in general would, but a flutter developer most probably would.

Design Methodology

Minimalism Fluxed With Functionality

Everybody likes a clean looking application. It is simply pleasing to use something that has a rather simple and appealing aesthetic without anything much to distract the viewer from the ultimate purpose of the application. That isn't to say that prettiness has to be forgone, but one can strike a good balance between good, classy design (even with quite a bit of color and character) whilst keeping things minimal. I think a generally good example of this is Instagram's interface. And that was the primary motivations here with the design.

I wanted something that was rather simple, without diminishing from it's utility, and especially without making it look "boring" per say.

There are various hints of this. For one, the color scheme of the application could have easily been black and white, with some basic fonts and an even simpler layout. But I decided against that since I felt it would be detrimental to the user experience. Contrast, color and a bit of dynamism to those concepts help with keeping users interested and captivated. It seems like a bit of a stretch for a number plate scanner app, but I still think it is worth a consideration when designing something.

In general I also decided to implement a lot of google's newer looks for it's UI and the newer elements available to flutter were a big part of this. I used Inkwells and grids to form pretty and smooth buttons with convenient gestures.

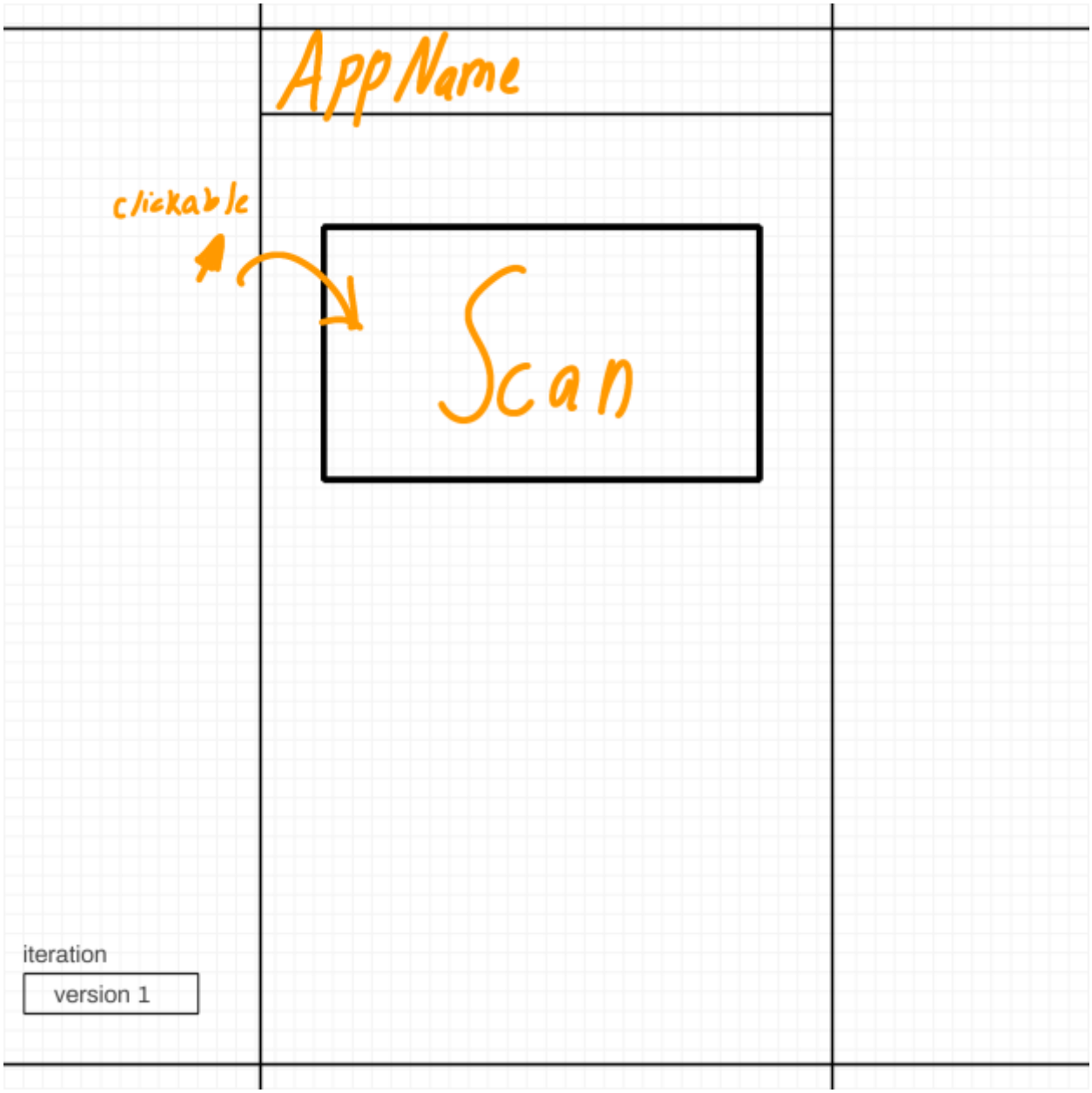
Design

For the design of the application there have been a lot of revisions made to it overtime to implement something more complete and isolated.

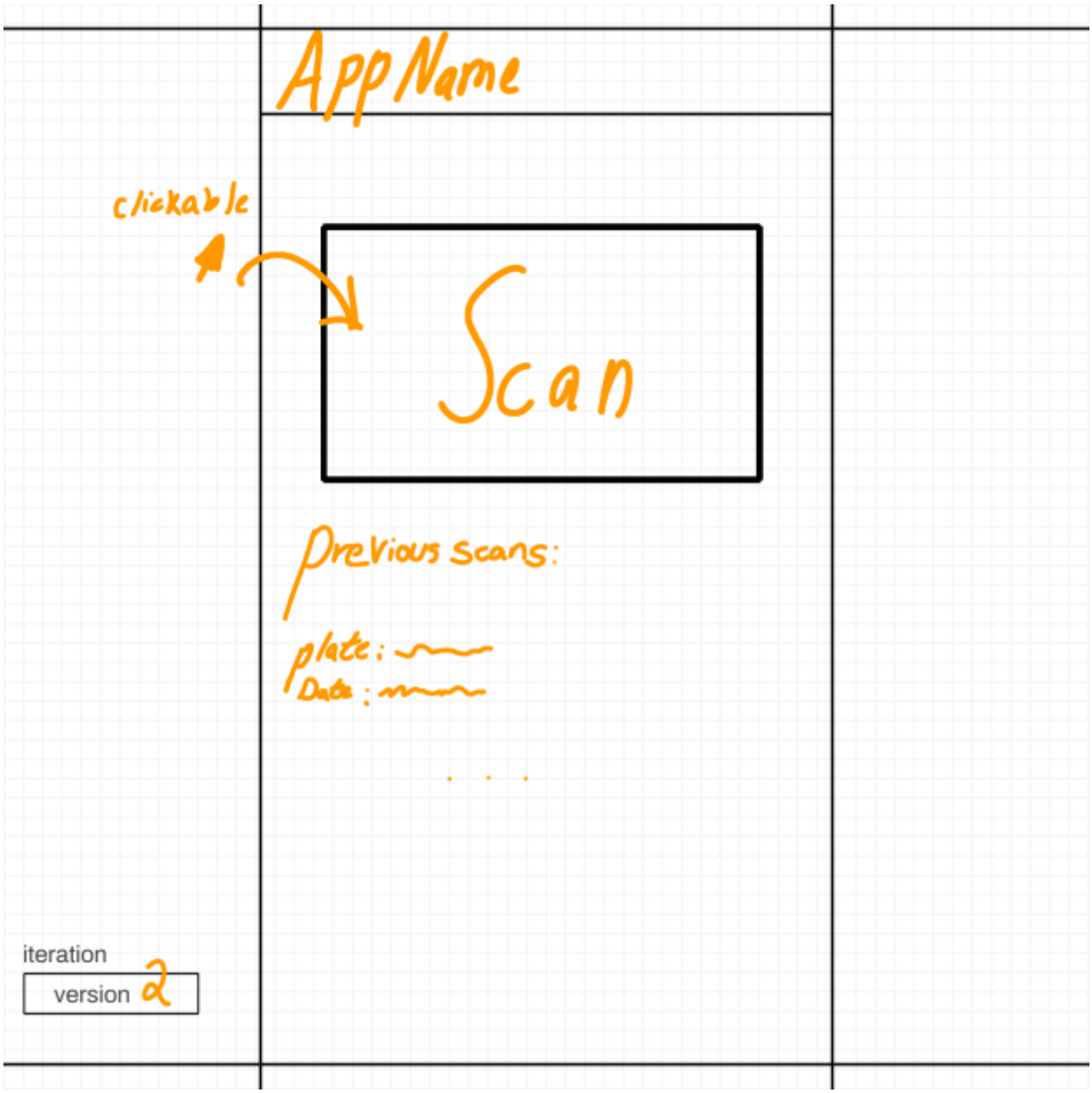
The design phase was honestly relatively short on this application as I preferred to focus on the code. I find myself to perform better when I design the app as the code comes along and api's and existing UI elements fit the standard that would make the most sense for the app.

That being said I had some rough ideas of how the app would look from the get go, some more congruent than others, some being earlier designs. I preferred to do this using a more primitive app instead of something like figma.

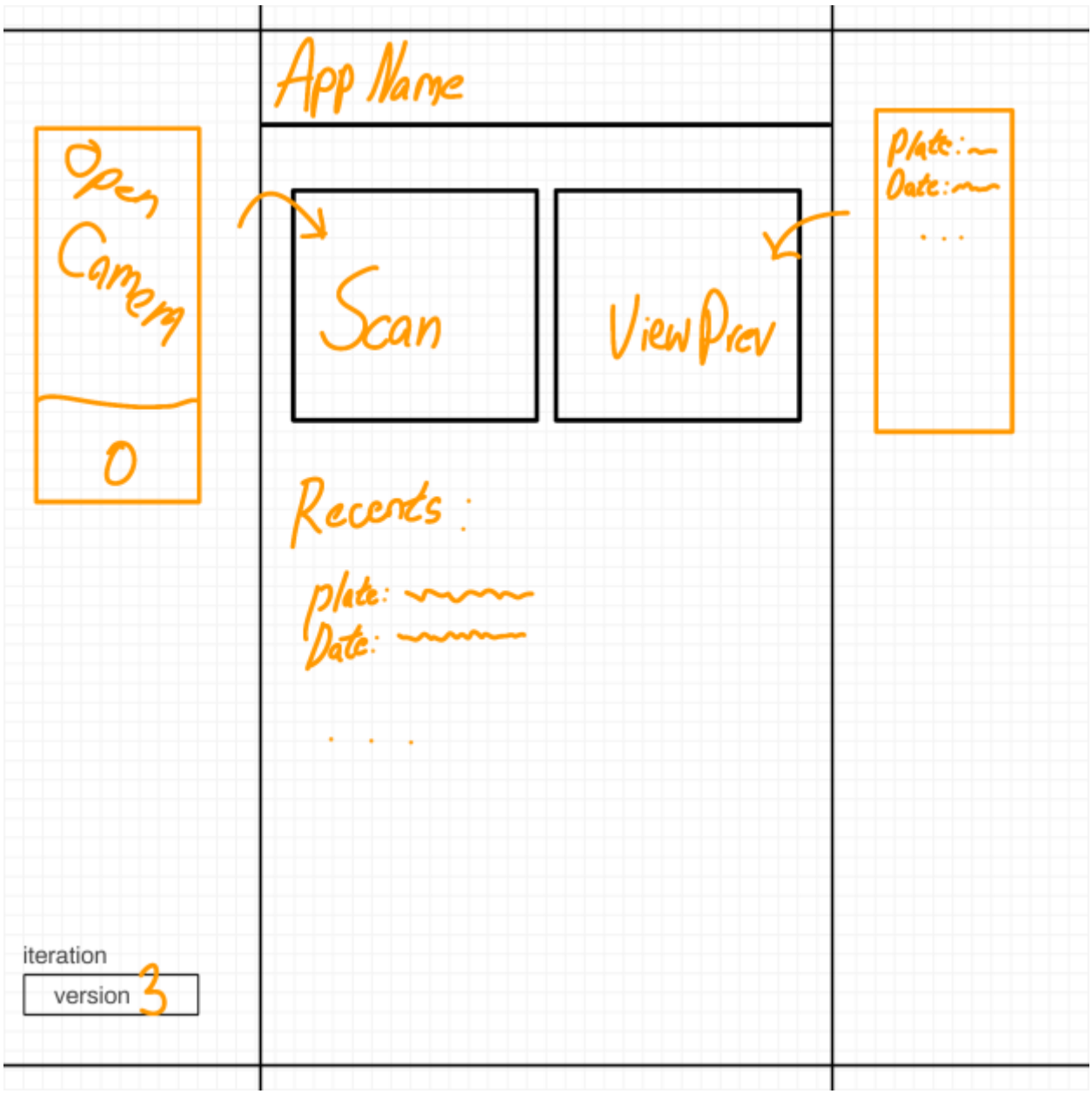
There are a few reasons for this but at it's most essential I find painting to be more familiar with my designs in general especially at the rough draft stage. I did make final revisions on the figma flow for the app however, which is referenced in this report quite often. I also find it quite useful for designing basic flows and outlining the use cases for the program



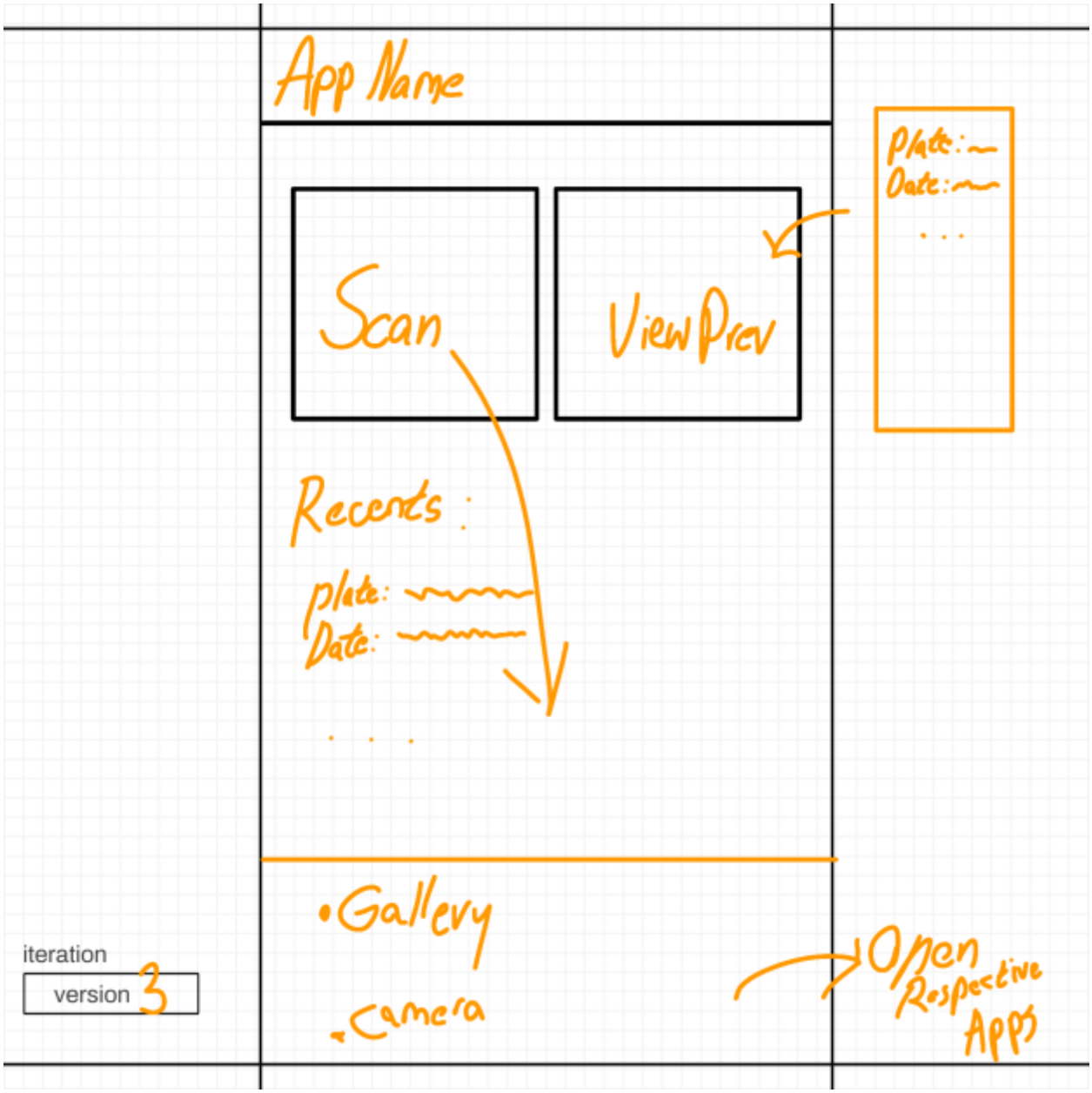
The first draft of the application was a basic app with an appbar, the appname at the top and a clickable inkwell (or button, I wasn't sure at the time) to make the camera feature. I hadn't thought through the next flow states for the application yet. You can tell that this is rather early on



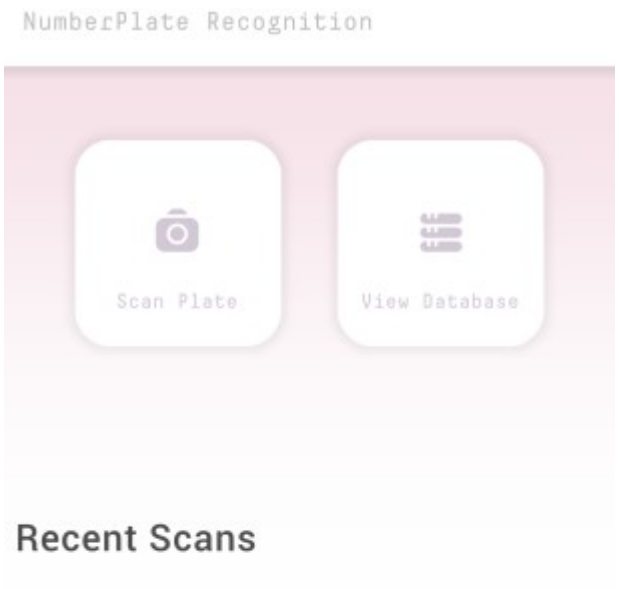
I thought about adding a “previous scans” pane under the scan button, to make it easier to see previous scans. I didn’t manage to implement this in the final application



For the next version I had some changes to the design again. Adding a second inkwell next to the original button/Inkwell. They would be aligned using a grid. “Previous scan” was renamed to “recents”. We also have a rough flow for the app



On the final initial draft I came to the decision that having an option for both the gallery and the camera would be best. I also found out about the image picker library that flutter has and thought it would be best for our implementation.



This was the final design of the screen for the interface. I did not have the time to make the recent scans section of the program.

Strengths

Here I outlined the main strengths of the application and it’s final version.

Offline

There's no need for a back-end like in previous iterations, and the latest version works entirely offline.

Strong Algorithm

google's ML kit provides an incredibly competent and hassle free OCR with a really good bet at getting the texts you're trying to retrieve. The algorithm is very strong. Providing a great platform to work in

Modular

The algorithm can easily be swapped for another without too much hassle due to the modular nature of the code. IOS support wouldn't be too hard to implement with some basic dependencies and changes to the home code.

Clean UI

A decent effort went into designing the UI. I'd like to say that that effort paid off. I used inkwells for the majority of the content, with a grid at the top, and an appbar. The UI is simplistic but also relatively functional and easy to work with.

Weaknesses

And here the weaknesses of the application;

Too strong

The OCR API hasn't been particularly specified for license plates. So the app will scan any text from any image. This can be seen as a nice double functionality but it also causes some issues, since we can get a lot of false positives. The combat this we can simply store all blocks of text and suggest them as results, if only one result is present we then omit this window since it is redundant (this was not achieved, the window will show even if only one block of text is found).

Missing Functionalities: Clean Database UI

Whilst the page is there, it isn’t terribly well designed. It lacks design choice or proper integration since its unfinished. It does have the nice point of being functional and letting users see previous scans that they have saved, but to say that it's perfected is a far cry from reality.

Missing Functionalities: Home feed recents

In the original UI designs for the app, I had planned to show recent scans at the bottom of the feed so that users wouldn't need to open the db storage. Sadly I didn't have the time to implement that in the final build of the app.

Conclusion

To conclude this journey in flutter programming – I think it's rather interesting to work in an environment like flutter. Due to how well documented the libraries are, and how rather straight-forward it is to work with when it comes to multiple various implementations of different functionalities. It was rather insightful as a whole. As for completing an application of this sort, I had quite a few ideas for new projects whilst working on this project itself. And as a whole I find the ecosystem that flutter provides to be very interesting to work with as a developer. I quite enjoyed the work on this project and hope to develop more in the flutter environment if and whenever I can.

Future Work

I believe if I had the time I would likely update the application – adding IOS support, including the recents screen, and polishing and UI and the code. With that level of effort, the app would be signed and released as an application on atleast the playstore at first. Seeing as the application is rather simple though, I doubt it would garner much attention. I do think there is potential for simple apps like this in general though, and their creation being such a breeze really makes development an enjoyable journey.

References

There were no references to other materials in this report, however here is the github to the project that contains references to the APIs I have used during them. I would like to extend my gratitude to said developers for their APIs and dependencies that made this project possible.

- yassir56069 (2024). *yassir56069/numberplate_recog*. [online] GitHub. Available at: https://github.com/yassir56069/numberplate_recog [Accessed 1 Mar. 2024].