TMA 03

Matthew Mason: C6122243

BSC (HONOURS) COMPUTING AND IT: TM-352

Table of Contents

Question 1: The user interface for a taxi-sharing app

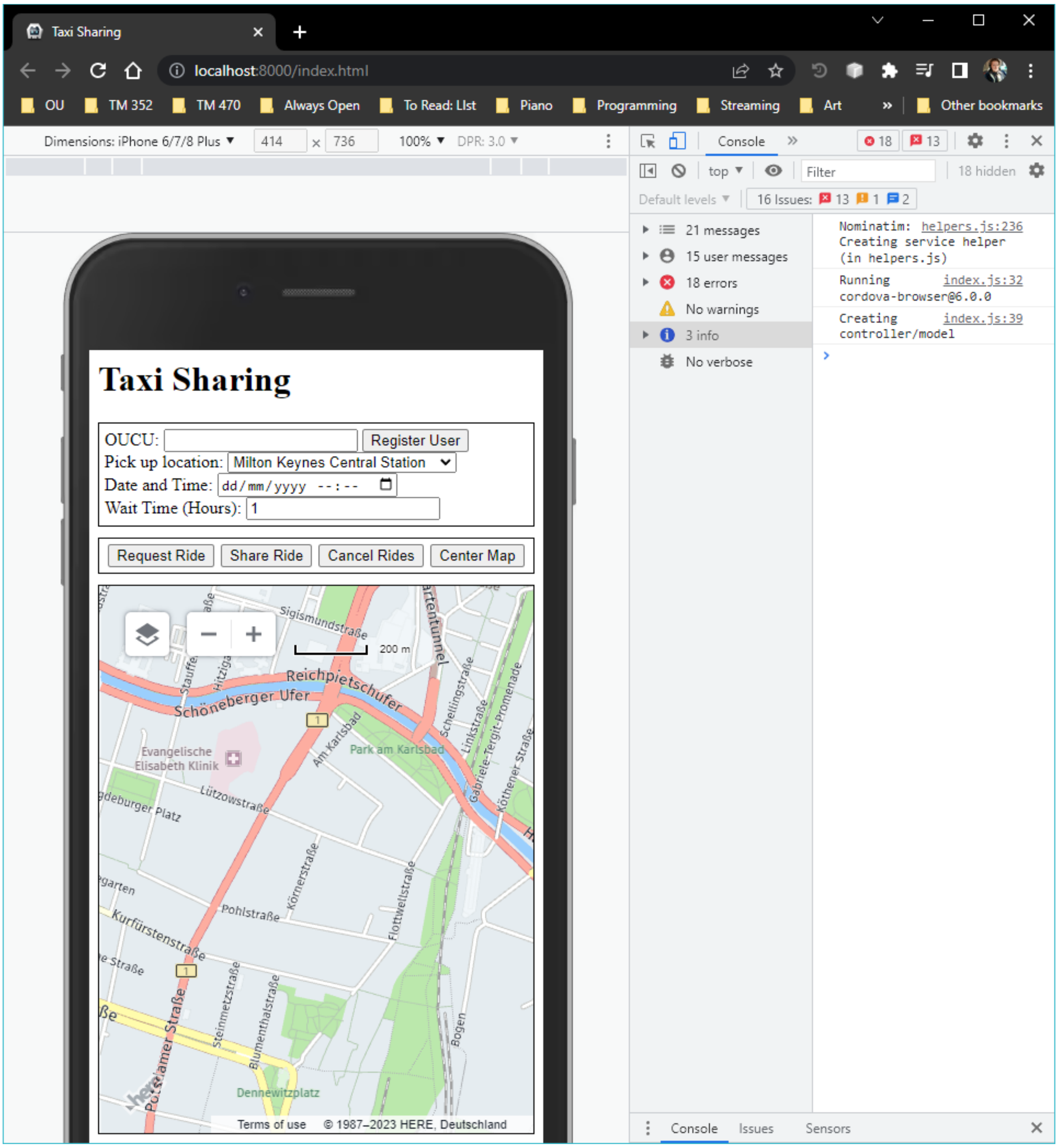
a.

FR	Name	Description	Obstacle
FR1	Feedback	Passengers will be able to leave driver and journey feedback via comments and a rating system.	The passenger may not understand the implemented rating system.
FR2	Notes	Passenger can add notes or special requirements to bookings.	Driver may not be capable of accommodating requirements.
FR3	History	Passengers can access information pertaining to past bookings; pick-up time, driver, etc.	Assuming this data is stored in the cloud, access to past journey data might not be accessible if GPS signal/internet is unavailable.
FR4	Photo	Users can take advantage of their camera to take a picture allowing them to be identified when offering rides to other passengers.	The app may not have permission to access the camera and take pictures.
FR5	Comms	Allows passengers and driver to communicate via call or text prior to pick up.	The app may not have permissions to use the microphone.

Table 1: Functional Requirements for a Taxi-sharing App.

b.

```
<body>
  <h1>Taxi Sharing</h1>
  <div style="border: 1px solid black; margin: 0px 0px 10px 0px; padding: 5px;">
    <!--- TODO: add text and form entry methods to capture required data -->
    <span>OUCU: <input id="oucu" type="text"></span>
    <button type="button" onclick="controller.registerUser()">Register User</button><br>
    <label for="location">Pick up location:</label>
    <select id="addr" name="location">
      <option value="Milton Keynes Central Station">Milton Keynes Central Station</option>
      <option value="Open University, Milton Keynes">Open University, Milton Keynes</option>
    </select><br>
    <span>Date and Time: <input id="time" type="datetime-local"></span><br>
    <span>Wait Time (Hours): <input id="wait" type="text" value="1"></span>
  </div>
  <!--- TODO: add buttons to allow the various functional requirements to be enacted -->
  <div style="border: 1px solid black; margin: 0px 0px 10px 0px; padding: 5px;">
    <span style="display: flex; justify-content: space-around;">
      <button type="button" onclick="controller.requestTaxi()">Request Ride</button>
      <button type="button" onclick="controller.offerTaxi()">Share Ride</button>
      <button type="button" onclick="controller.cancel()">Cancel Rides</button>
      <button type="button" onclick="controller.centreMap()">Center Map</button>
    </span>
  </div>
  <!--- TODO: add a placeholder for the HERE Map -->
  <div id="mapContainer" style="height: 69vh; width: auto; border: 1px solid black;"></div>
  <script type="text/javascript" src="cordova.js"></script>
  <script type="text/javascript" src="js/helpers.js"></script>
  <script type="text/javascript" src="js/index.js"></script>
</body>
```



Taxi Sharing

OUCU: Register User
Pick up location: Milton Keynes Central Station
Date and Time: dd/mm/yyyy --:--
Wait Time (Hours): 1

```
Nominatim: Creating service helper (in helpers.js)   helpers.js:236
Running cordova-browser@6.0.0                        index.js:32
Creating controller/model                            index.js:39
```

Figure 1: Taxi-Sharing app running via "cordova run browser" and chrome dev tools open.

(Word-Count: 161)

Question 2: Business logic for the mobile web app

FR1.1

```
function offer(oucu, address, startTime, endTime) {
	// TODO 2(a) FR1.1
	// You need to implement this function
	// See the TMA for an explanation of the functional requirements
	function onSuccess(obj) {
	    // Inform the user what happened
	    if (obj.status == "success") {
	        console.log("User " + oucu + " has offered a taxi from " + address + " at " + startTime + ". They will
wait until " + endTime);
	    } else if (obj.message) {
	        alert(obj.message);
	    } else {
	        alert("There was a problem");
	    }
	}

	var url = BASE_GET_URL + "orders"
	$.ajax(url, { type: "POST", data: { oucu: oucu, type: 0, address: address, start: startTime, end:
endTime }, success: onSuccess });
   }
```

FR 1.2

```
function request(oucu, address, startTime) {
	// TODO 2(a) FR1.2
	// You need to implement this function
	// See the TMA for an explanation of the functional requirements
	function onSuccess(obj) {
	    // Inform the user what happened
	    if (obj.status == "success") {
	        console.log("User " + oucu + " has requested a taxi from " + address + " at " + startTime);
	    } else if (obj.message) {
	        alert(obj.message);
	    } else {
	        alert("There was a problem");
	    }
updateMapAddresses();
	}

	var url = BASE_GET_URL + "orders"
	$.ajax(url, { type: "POST", data: { oucu: oucu, type: 1, address: address, start: startTime}, success:
onSuccess });
```

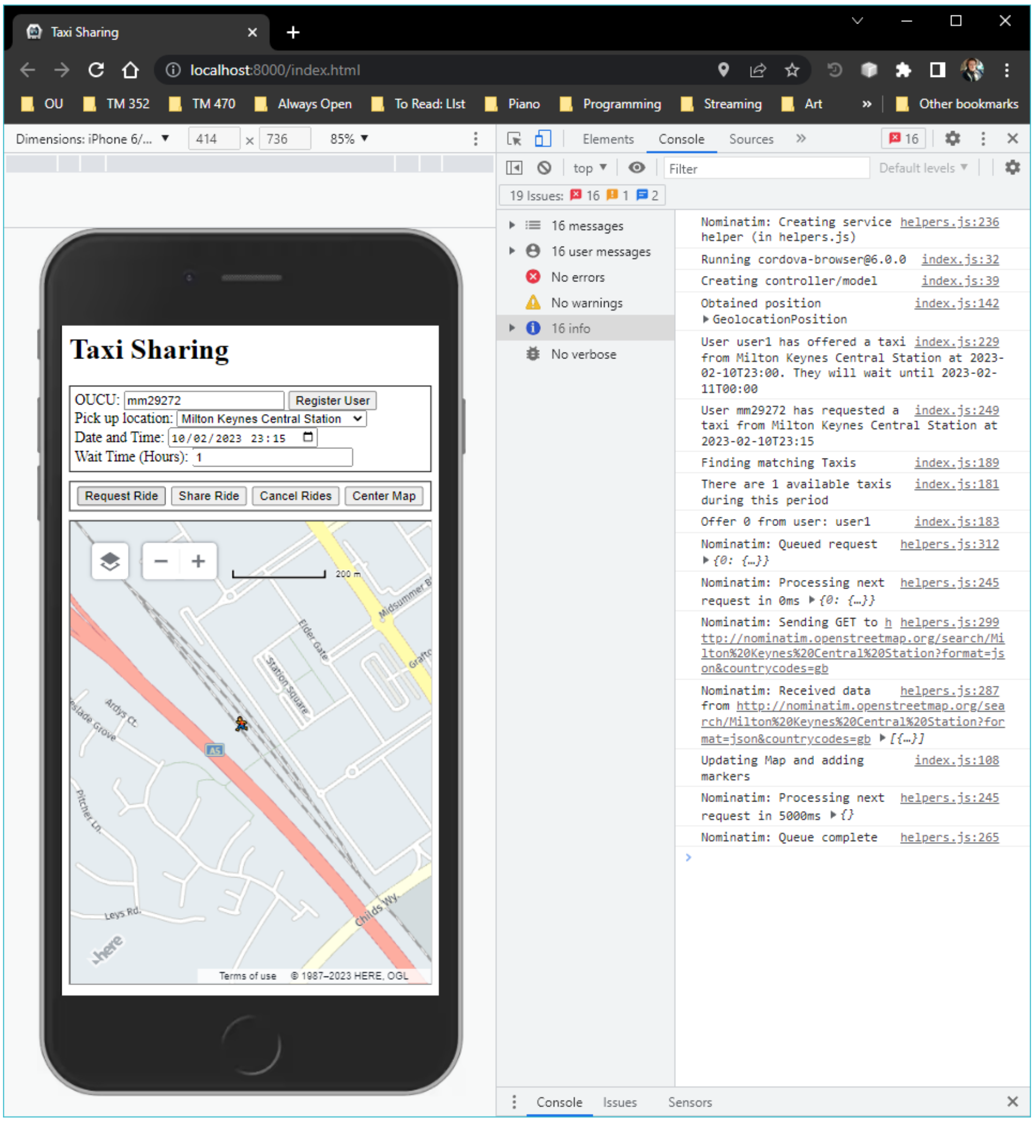
FR 2.1

```
function updateMapAddresses() {
	var oucu = getInputValue("oucu", "user1");
	// TODO 2(a) FR2.1
	clearMarkersFromMap();
	function onSuccess(obj) {
	    // Inform the user what happened
	    if (obj.status == "success") {
	        console.log("There are " + obj.data.length + " available taxis during this period");
	        obj.data.forEach(function (orders) {
	            console.log("Offer " + obj.data.indexOf(orders) + " from user: " + orders.offer_oucu)
	            addMarkerToMap(orders.offer_address);
	var matchUrl = BASE_GET_URL + "matches?OUCU=" + oucu;
```

```
    console.log("Finding matching Taxis");
    $.ajax(matchUrl, { type: "GET", data: { oucu: oucu }, success: onSuccess });
```

FR 2.2

```
function addMarkerToMap(address) {
    if (address)
        var onSuccess = function (data) {
            // TODO 2(a) FR2.2
            console.log("Updating Map and adding markers");
            var point = {
                lng: data[0].lon,
                lat: data[0].lat,
            }
            marker = new H.map.DomMarker(point, { icon: icon});
            map.addObject(marker)
            map.setCenter(point);
        nominatim.get(address, onSuccess);
```



Taxi Sharing

OUCU: mm29272

Pick up location: Milton Keynes Central Station

Date and Time: 10/02/2023 23:15

Wait Time (Hours): 1

```
Nominatim: Creating service helper (in helpers.js)        helpers.js:236
Running cordova-browser@6.0.0                              index.js:32
Creating controller/model                                  index.js:39
Obtained position                                          index.js:142
▸GeolocationPosition
User user1 has offered a taxi from Milton Keynes Central Station at 2023-02-10T23:00. They will wait until 2023-02-11T00:00    index.js:229
User mm29272 has requested a taxi from Milton Keynes Central Station at 2023-02-10T23:15    index.js:249
Finding matching Taxis                                     index.js:189
There are 1 available taxis during this period             index.js:181
Offer 0 from user: user1                                   index.js:183
Nominatim: Queued request                                  helpers.js:312
▸{0: {…}}
Nominatim: Processing next request in 0ms ▸{0: {…}}       helpers.js:245
Nominatim: Sending GET to http://nominatim.openstreetmap.org/search/Milton%20Keynes%20Central%20Station?format=json&countrycodes=gb    helpers.js:299
Nominatim: Received data from http://nominatim.openstreetmap.org/search/Milton%20Keynes%20Central%20Station?format=json&countrycodes=gb ▸[{…}]    helpers.js:287
Updating Map and adding markers                            index.js:108
Nominatim: Processing next request in 5000ms ▸{}          helpers.js:245
Nominatim: Queue complete                                  helpers.js:265
>
```

Figure 2: Running the Taxi Sharing App through Chrome Browser.

Question 3: Using a Plugin

a.

App	Photo Profile Picture
Uber	Yes
Bolt	No (must link to Facebook account)
Lyft	Yes
Taxiapp UK Passenger	Yes
Addison Lee	No

Table 2: Feature table to explore different taxi-sharing apps allowing users to take a profile picture with their camera

Having downloaded and tested a number of apps, it was surprising to see not all of them offered the ability to add a profile picture using the phones camera (Google, 2023). Having the ability to do this allows passengers and drivers to easily identify each other when requesting or offering a ride. It adds additional value by also working as a security measure for both passengers and drivers by stopping unauthorised and non-registered users from jumping in a pre-booked taxi or claiming to be the user who booked the taxi. To implement this function in our taxi-sharing app, I will simply add a button to the main index page that allows the user to take a picture with the camera via a plug-in which will then be formatted and displayed on the main page.

b.

Using the NPM inc. (NPM, 2023) website, I was able to search the database of available plugins by refining the parameters to include only those pertaining to the Cordova ecosystem and the use of the phones camera rather than including other functionality such as camera roll, etc. I refined the search further by the popularity filter to explore a number of plugins in more detail and based on the available documentation, quality filter, file size, collaborators and maintenance, I chose to use a plugin simply named 'cordova-plugin-camera' (NPM, 2023). The reason for my choice is that it provides the exact required functionality for the taxi-sharing app, it's simple to implement and it allows users to take a picture using their phones camera which can then be rendered on screen.

c.

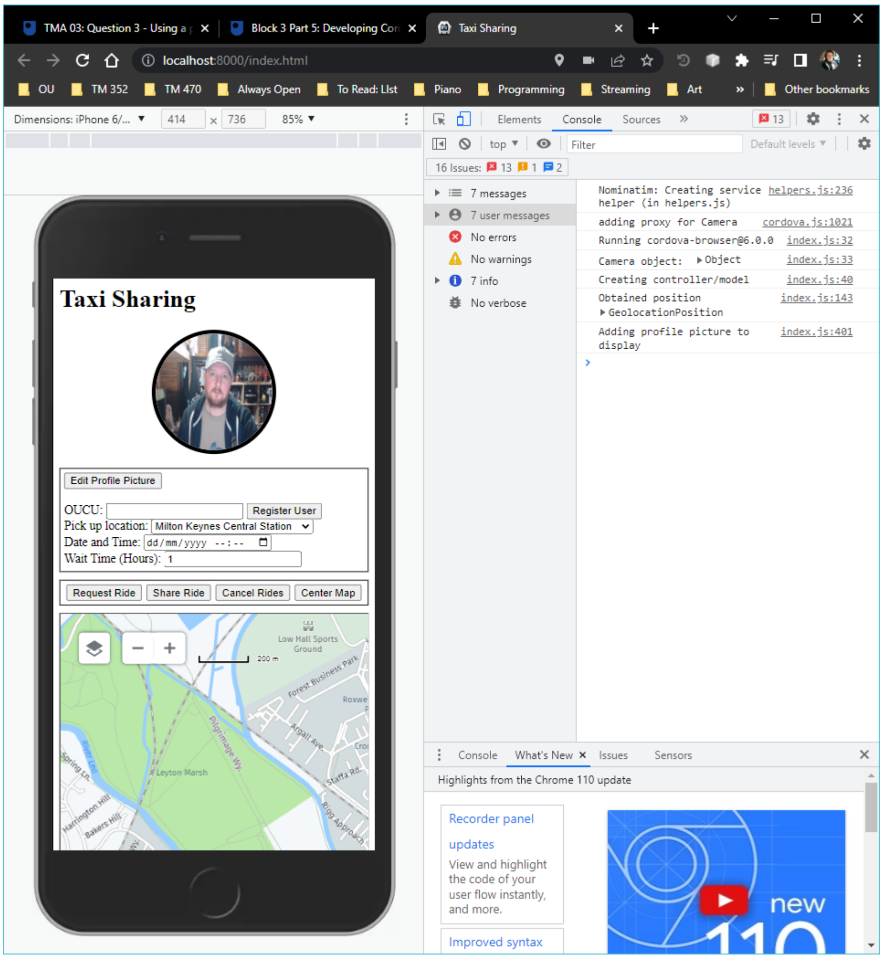
Below is the HTML that adds a hidden div element and a button to the main index page which triggers the following JavaScript function and plugin.

```
<h1>Taxi Sharing</h1>
  <!--- TODO: Implement HTML Buttons and content for Plugin -->
  <div id="photoDiv" style="display: none;">
    <img id="profPic"; style="display: block; margin-left: auto; margin-right: auto; border: 5px solid black;
width: 150px; height: 150px; border-radius: 50%;" src="">
  </div><br>

<button type="button" onclick="controller.takeProfPic();">Edit Profile Picture</button><br><br>

this.takeProfPic = function () {
        function onSuccess(data) {
            if (cordova.platformId === "browser") {
                // Set source of image on the page (plugin returns base64 data on browser platform)
                document.getElementById("profPic").src = "data:image/jpeg;base64," + data;
            } else {
                // Set source of image on the page
                document.getElementById("profPic").src = data;
            }
            document.getElementById("photoDiv").style.display = "block";
        }
        function onError(error) {
            console.log("Error calling getPicture", error);
        }
        navigator.camera.getPicture(onSuccess, onError);
    }
```

The code above demonstrates how the camera plugin fulfils the functional requirement of allowing passengers or drivers to take a profile picture using their phones inbuilt camera. The area reserved to display their picture is initially hidden, it is only when the user clicks on the button marked 'Edit Profile Picture' that the 'takeProfPic' function is triggered. The plugin, once installed, creates a global object, 'camera' to provide access to functions that allow the app to take pictures. The camera object is accessed via the @navigator namespace such as navigator.camera.getPicture(). Callback functions then respond depending on the success of the task and appends the image source to the HTML element. The display property of the hidden div element is then set to show the image and styled accordingly. If an error occurs it's logged to the console to let the user know something went wrong.



Taxi Sharing

OUCU:
Pick up location: Milton Keynes Central Station
Date and Time: dd/mm/yyyy --:--
Wait Time (Hours): 1

```
Nominatim: Creating service helper (in helpers.js)    helpers.js:236
adding proxy for Camera                               cordova.js:1021
Running cordova-browser@6.0.0                         index.js:32
Camera object:  ▶Object                               index.js:33
Creating controller/model                             index.js:40
Obtained position                                     index.js:143
▶GeolocationPosition
Adding profile picture to display                     index.js:401
```

Figure 3: Taxi-sharing app displaying the use of inbuilt camera to take a profile picture.

The plugin worked almost exactly as I'd hoped. Styling the image and placement provided a small challenge but the only obstacle I encountered was that when taking a picture in portrait mode, the resulting image was displayed at a 90 degree angle to what I'd anticipated. The only solution I found was to take the picture in landscape mode, only then was the image the correct orientation within the app.

(Word-count: 562)

Question 4 – Invent a Possible App

a.

	Pocket Crochet	Crochet Genius	Crochet.land	CrochetStudio	Knitting Buddy
Conversion Chart	🟥	🟩	🟥	🟥	🟩
Pattern designer	🟥	🟩	🟥	🟩	🟥
Pattern generator	🟥	🟩	🟥	🟩	🟥
Tutorials	🟥	🟩	🟥	🟥	🟥
Photo to pixel	🟥	🟥	🟥	🟥	🟥
Add project	🟩	🟩	🟩	🟥	🟩
Pallet generator	🟥	🟥	🟩	🟩	🟥
Inventory	🟩	🟩	🟥	🟥	🟩
Project pricing tools	🟥	🟥	🟩	🟥	🟥
Glossary	🟥	🟩	🟥	🟥	🟩

Table 3: Feature table for a number of different crochet apps. Key: Green = Feature of the app, Red = Not a feature

As a crochet hobbyist, I use a variety of apps, as individually, they provide a number of functions that I find important. After researching a range of different apps, there was still one feature I required which didn't exist within any; The ability to pixilate photos. Each pixel is the equivalent to a 'block' in a C2C (Corner to Corner) pattern so this feature would allow unique photos to be turned into patterns with ease such as in figure 4 below.

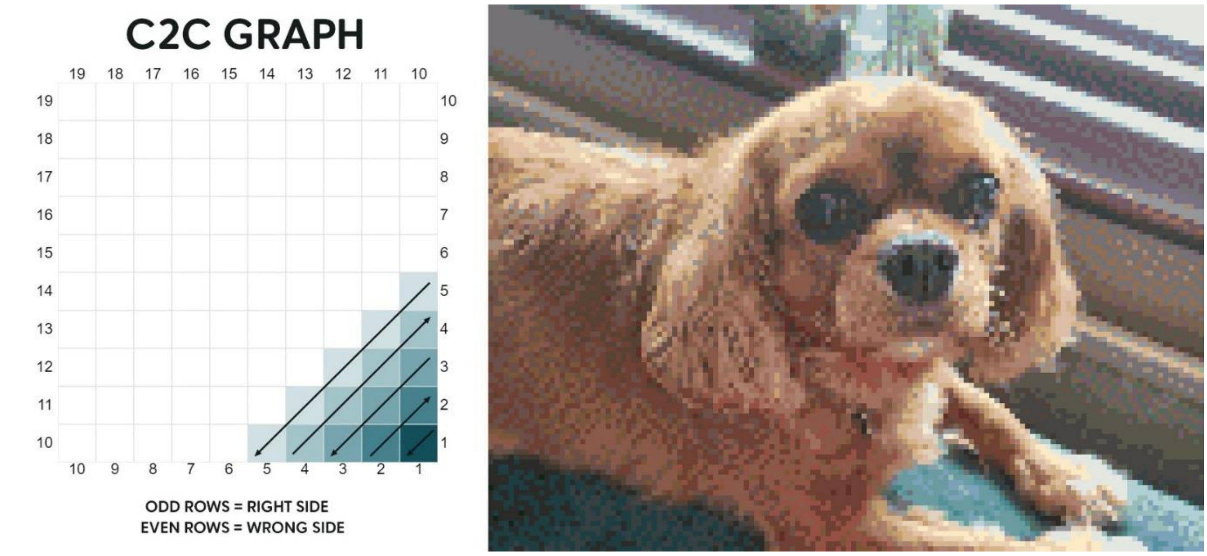


Figure 4: How Corner to Corner patterns are worked (Maker, 2023) with demonstration of how a photo can be pixelated (Pixel-Stitch, 2023).

b.

FR	Name	Description	Obstacle
FR1	Pixelization	Convert a photo stored on the users devices into a pixilated version based on setting such as width (number of stitches), colours and size.	The app may not have permission to access the photo storage of the phone.
FR2	Counter	The user can increase or decrease the number of rows or rounds they have completed.	Persistence of current count has potential to be lost if app is closed down or there is an unscheduled closure.
FR3	Timer	The user can start and stop a timer to show how long they have currently worked on a specific project.	There may be a delay between the user pressing the timer button and the code running causing inaccurate timing.
FR4	TCO	The user can add the cost of materials and time taken on a project to calculate cost prices of projects and commissions.	Users may want to convert between different currencies for international orders. Constantly changing exchange rates will make this difficult.
FR5	Add Project	The user can add individual projects within the app to keep track of a number of aspects such as completion percent, inventory, patterns, etc.	The user may have difficulty typing in all necessary information using the touch screen controls.

Table 4: Functional Requirements for a potential Crochet Assistance application

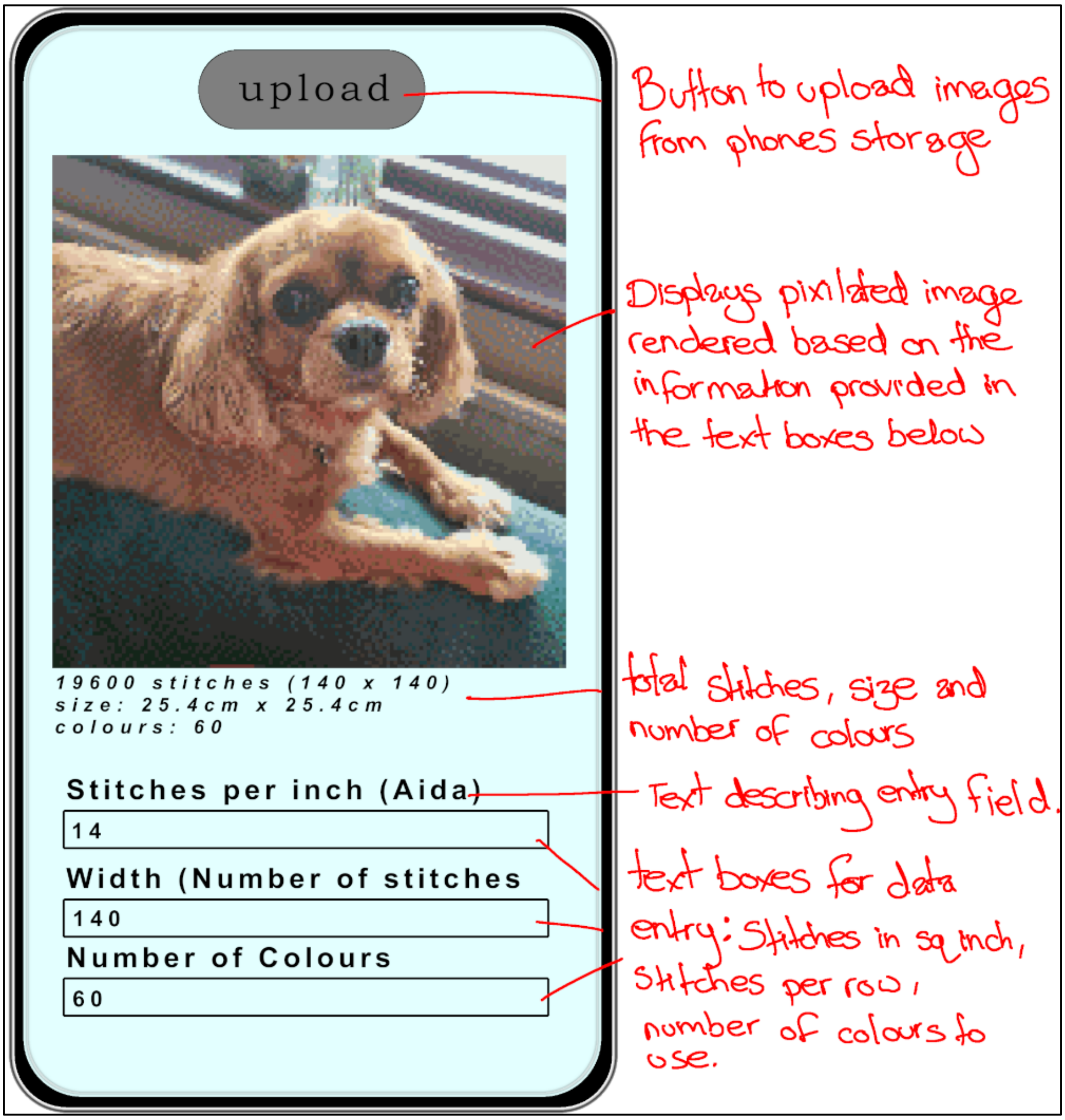
c.

i.

This app is the most comprehensive crochet companion you'll need for any project. Create your own designs and patterns in no time from your own perfectly pixelized photos and artwork. Use the pallet generator to find the ideal colours and yarns or create your own. Keep track of all your projects, equipment, and plan your commission costs using our pricing tools.

If you're new, you can use out video tutorials to learn the basics along with a great glossary of all the terms you'll ever need to know. All your crocheting needs are all in this single app! Happy crafting!

ii.



19600 stitches (140 x 140)
size: 25.4cm x 25.4cm
colours: 60

Stitches per inch (Aida)

14

Width (Number of stitches

140

Number of Colours

60

Button to upload images from phones storage

Displays pixlated image rendered based on the information provided in the text boxes below

total stitches, size and number of colours

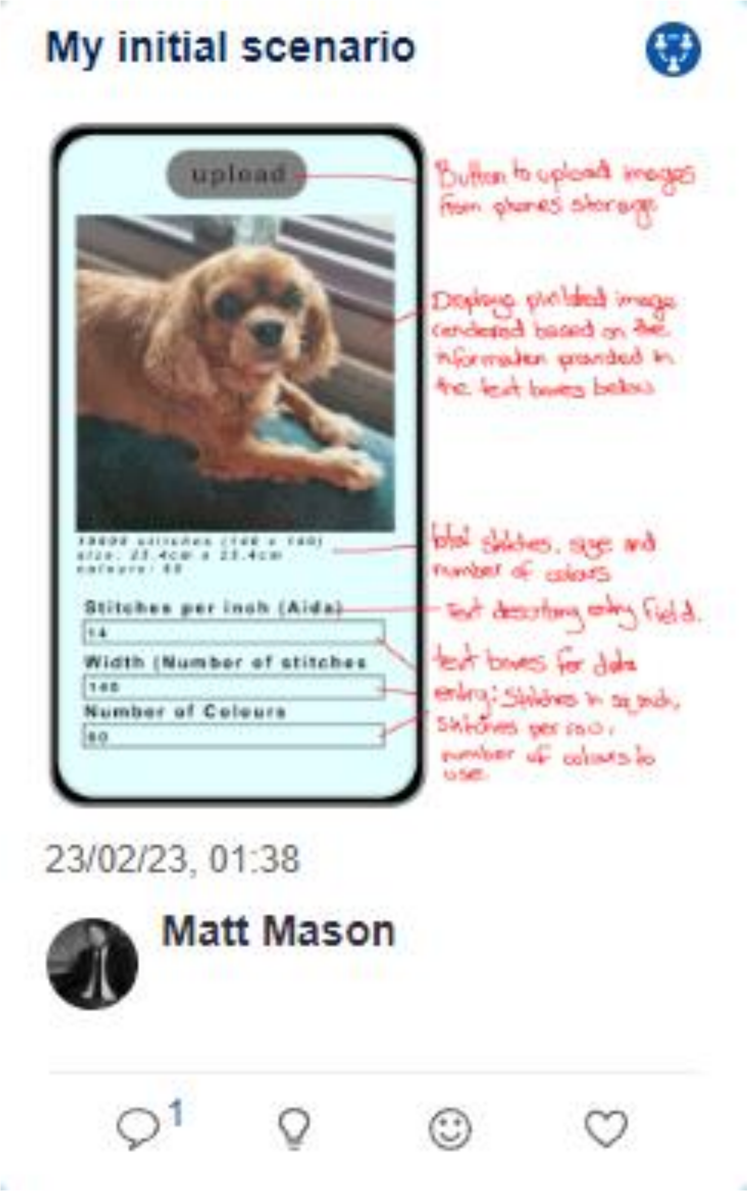
Text describing entry field.

text boxes for data entry: Stitches in sq inch, Stitches per row, number of colours to use.

Figure 5: Wireframe of FR1 displaying how a user can pixelate a photo based on a number of settings.

iii.

Uploaded on 23/02/2023.



My initial scenario

upload

19600 stitches (140 x 140)
size: 25.4cm x 25.4cm
colours: 60

Stitches per inch (Aida)

14

Width (Number of stitches

140

Number of Colours

60

Button to upload images from phones storage

Displays pixilated image rendered based on the information provided in the text boxes below

total stitches, size and number of colours

Text describing entry field.

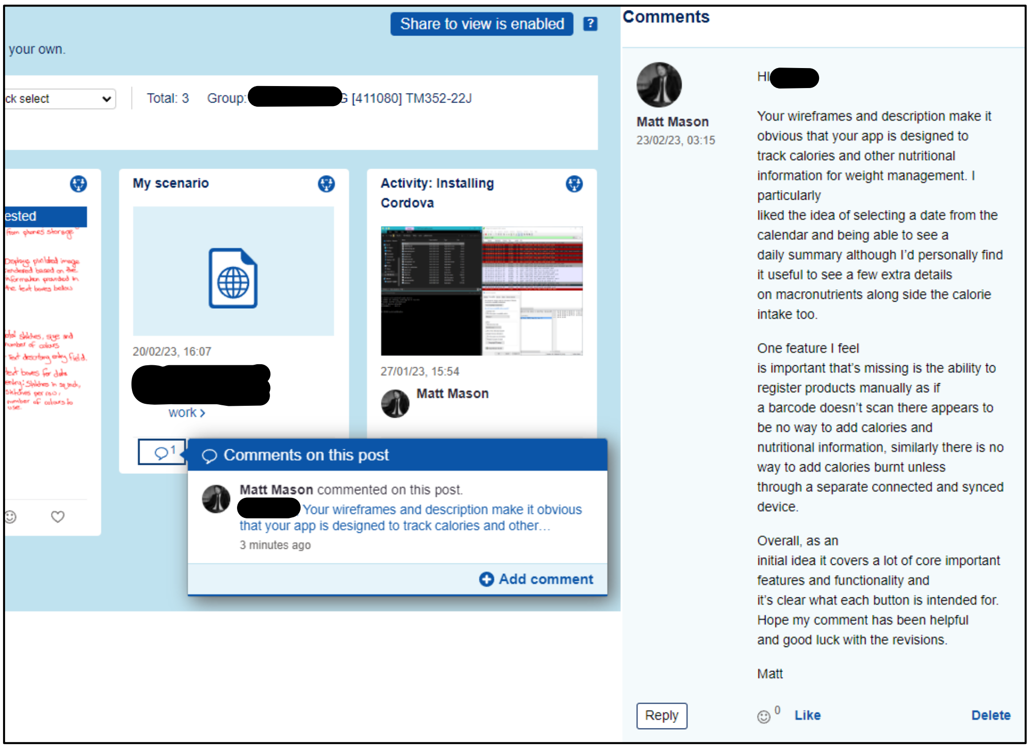
text boxes for data entry: Stitches in sq inch, Stitches per row, number of colours to use.

23/02/23, 01:38

Matt Mason

Figure 6: Evidence of upload to OpenStudio.

iv.



20/02/23, 16:07

[redacted]

27/01/23, 15:54

Matt Mason

Comments on this post

Matt Mason commented on this post.

[redacted] Your wireframes and description make it obvious that your app is designed to track calories and other…

3 minutes ago

Comments

Matt Mason
23/02/23, 03:15

HI [redacted]

Your wireframes and description make it obvious that your app is designed to track calories and other nutritional information for weight management. I particularly liked the idea of selecting a date from the calendar and being able to see a daily summary although I'd personally find it useful to see a few extra details on macronutrients along side the calorie intake too.

One feature I feel is important that's missing is the ability to register products manually as if a barcode doesn't scan there appears to be no way to add calories and nutritional information, similarly there is no way to add calories burnt unless through a separate connected and synced device.

Overall, as an initial idea it covers a lot of core important features and functionality and it's clear what each button is intended for. Hope my comment has been helpful and good luck with the revisions.

Matt

Figure 7: Initial Scenario by [redacted] of a calorie and nutrition tracking app with provided feedback.

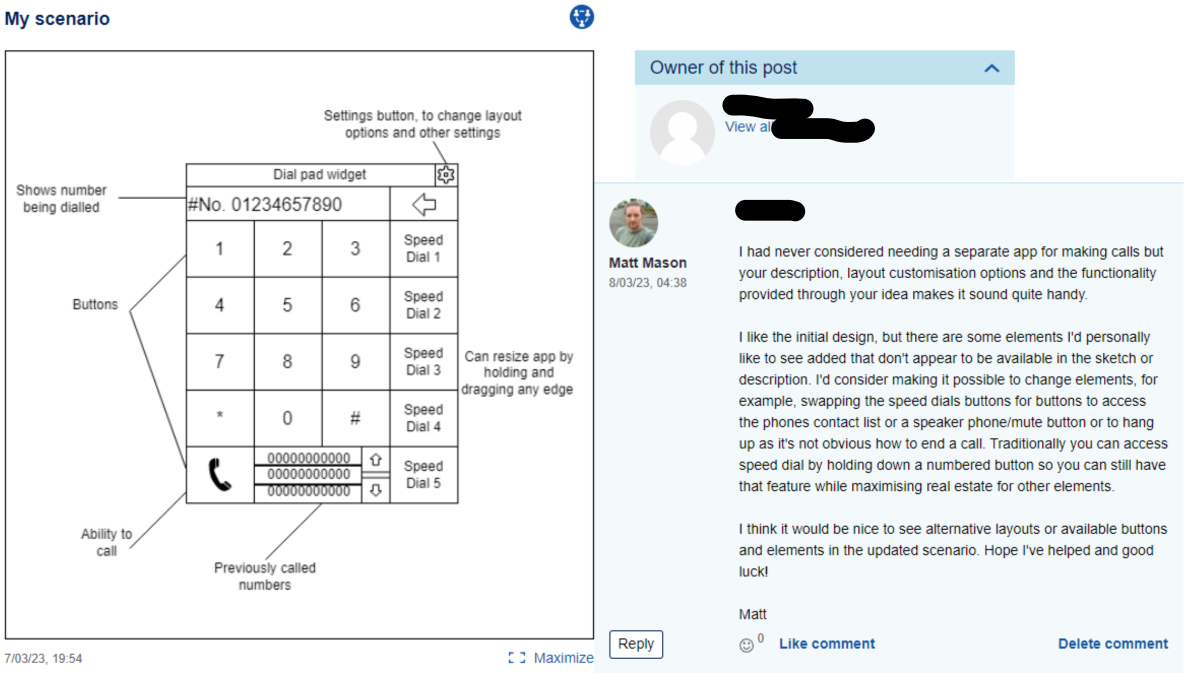
HI [redacted]

Your wireframes and description make it obvious that your app is designed to track calories and other nutritional information for weight management. I particularly liked the idea of selecting a date from the calendar and being able to see a daily summary although I'd personally find it useful to see a few extra details on macronutrients alongside the calorie intake too.

One feature I feel
is important that's missing is the ability to register products manually as if a barcode doesn't scan there appears to be no way to add calories and nutritional information, similarly there is no way to add calories burnt unless through a separate connected and synced device.

Overall, as an
initial idea it covers a lot of core important features and functionality and it's clear what each button is intended for. Hope my comment has been helpful and good luck with the revisions.

Matt



[redacted]

Matt Mason
8/03/23, 04:38

[redacted]

I had never considered needing a separate app for making calls but your description, layout customisation options and the functionality provided through your idea makes it sound quite handy.

I like the initial design, but there are some elements I'd personally like to see added that don't appear to be available in the sketch or description. I'd consider making it possible to change elements, for example, swapping the speed dials buttons for buttons to access the phones contact list or a speaker phone/mute button or to hang up as it's not obvious how to end a call. Traditionally you can access speed dial by holding down a numbered button so you can still have that feature while maximising real estate for other elements.

I think it would be nice to see alternative layouts or available buttons and elements in the updated scenario. Hope I've helped and good luck!

Matt

Figure 8: Initial Scenario by [redacted] *of a dial pad widget with provided feedback.*

Hi [redacted]

I had never considered needing a separate app for making calls but your description, layout customisation options and the functionality provided through your idea makes it sound quite handy.

I like the initial design, but there are some elements I'd personally like to see added that don't appear to be available in the sketch or description. I'd consider making it possible to change elements, for example, swapping the speed dials buttons for buttons to access the phones contact list or a speaker phone/mute button or to hang up as it's not obvious how to end a call. Traditionally you can access speed dial by holding down a numbered button so you can still have that feature while maximising real estate for other elements.

I think it would be nice to see alternative layouts or available buttons and elements in the updated scenario. Hope I've helped and good luck!

Matt

v.

comments made by other students:

[redacted]

25/02/23, 12:59

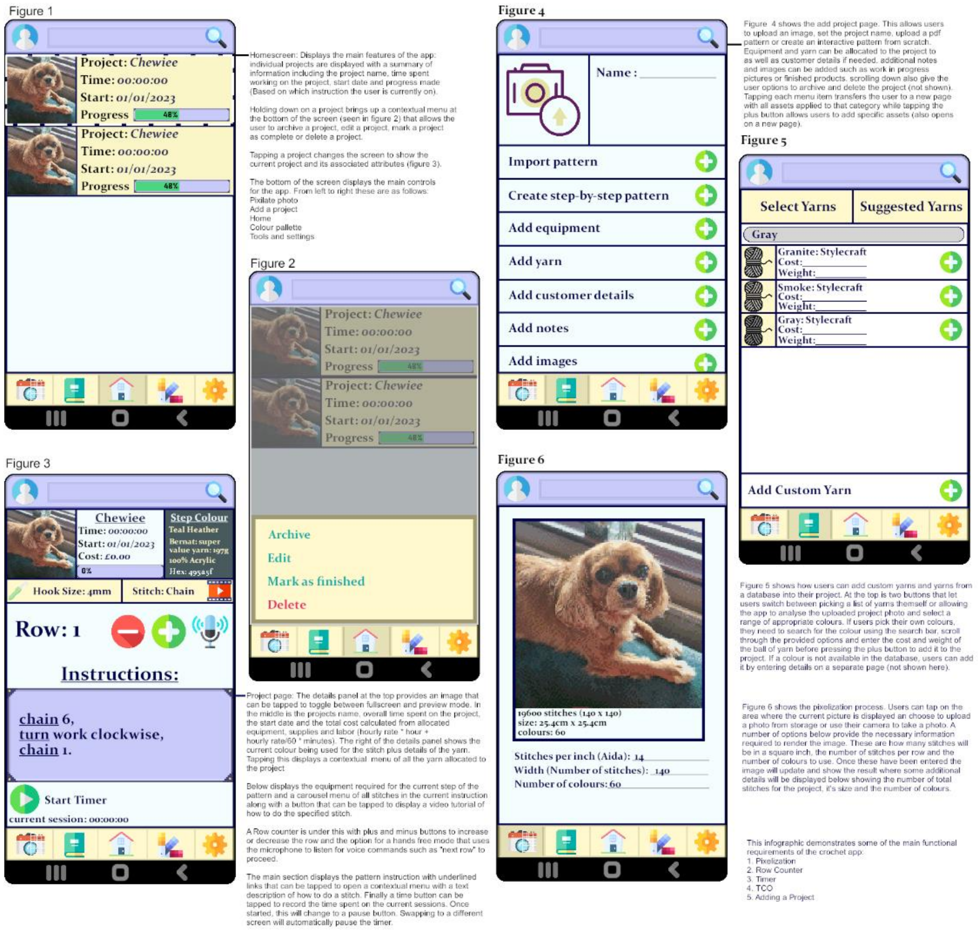
Even before reading the app description an it was quite clear what the purpose of the app was that is to take a photo and convert into a pixel format.

I would personally not download the app as I do not stich, but I wonder if the settings were tweaked weather the user could unintentionally create pixelated artwork from there photos.

I do not feel like the design could be improve its function is quite basic and simple it has one purpose an does that very well.

Most of the features mentioned in the app description has not been implemented in the initial design. Maybe a feature to capture the image straight from the camera instead from uploading a file from the device.

The interactions are clear and are self-explanatory 'Number of Colours' might not be clear is it referring to number of bits or colours the user can choose.



Homescreen: Displays the main features of the app: individual projects are displayed with a summary of information including the project name, time spent working on the project, start date and progress made (Based on which instruction the user is currently on).

Holding down on a project brings up a contextual menu at the bottom of the screen (seen in figure 2) that allows the user to archive a project, edit a project, mark a project as complete or delete a project.

Tapping a project changes the screen to show the current project and its associated attributes (figure 3).

The bottom of the screen displays the main controls for the app. From left to right these are as follows:
Pixilate photo
Add a project
Home
Colour pallette
Tools and settings

Project page: The details panel at the top provides an image that can be tapped to toggle between fullscreen and preview mode. In the middle is the projects name, overall time spent on the project, the start date and the total cost calculated from allocated equipment, supplies and labor (hourly rate * hour + hourly rate/60 * minutes). The right of the details panel shows the current colour being used for the stitch plus details of the yarn. Tapping this displays a contextual menu of all the yarn allocated to the project

Below displays the equipment required for the current step of the pattern and a carousel menu of all stitches in the current instruction along with a button that can be tapped to display a video tutorial of how to do the specified stitch.

A Row counter is under this with plus and minus buttons to increase or decrease the row and the option for a hands free mode that uses the microphone to listen for voice commands such as "next row" to proceed.

The main section displays the pattern instruction with underlined links that can be tapped to open a contextual menu with a text description of how to do a stitch. Finally a time button can be tapped to record the time spent on the current sessions. Once started, this will change to a pause button. Swapping to a different screen will automatically pause the timer.

Figure 4 shows the add project page. This allows users to upload an image, set the project name, upload a pdf pattern or create an interactive pattern from scratch. Equipment and yarn can be allocated to the project to as well as customer details if needed. additional notes and images can be added such as work in progress pictures or finished products. scrolling down also give the user options to archive and delete the project (not shown). Tapping each menu item transfers the user to a new page with all assets applied to that category while tapping the plus button allows users to add specific assets (also opens on a new page).

Figure 5 shows how users can add custom yarns and yarns from a database into their project. At the top is two buttons that let users switch between picking a list of yarns themself or allowing the app to analyse the uploaded project photo and select a range of appropriate colours. If users pick their own colours, they need to search for the colour using the search bar, scroll through the provided options and enter the cost and weight of the ball of yarn before pressing the plus button to add it to the project. If a colour is not available in the database, users can add it by entering details on a separate page (not shown here).

Figure 6 shows the pixelization process. Users can tap on the area where the current picture is displayed an choose to upload a photo from storage or use their camera to take a photo. A number of options below provide the necessary information required to render the image. These are how many stitches will be in a square inch, the number of stitches per row and the number of colours to use. Once these have been entered the image will update and show the result where some additional details will be displayed below showing the number of total stitches for the project, it's size and the number of colours.

This infographic demonstrates some of the main functional requirements of the crochet app:
1. Pixelization
2. Row Counter
3. Timer
4. TCO
5. Adding a Project

Figure 9: Redesign of crochet app

I did not realise the initial diagram needed to cover all FRs so my redesign addresses that with a variety of comprehensive diagrams and notes to rectify this based on Simon's comment on my work. I have chosen to mostly keep the original functionality of the initial design as I feel it's very simplistic and the text descriptions are adequate enough to describe what is required. I removed the upload button in favour of tapping the screen where the image is located, which then opens the camera, and the user has the option to take a picture or upload one from storage.

(Word-count: 946)

References

Flaticon, 2023. *Vector icons and stickers -PNG, SVG, PSD, and CSS.* [Online]
Available at: https://www.flaticon.com/
[Accessed 03 03 2023].

Google, 2023. *Android Apps on Google Play.* [Online]
Available at: https://play.google.com/store/apps
[Accessed 17 02 2023].

Maker, S., 2023. *How to Corner-to-Corner Crochet (C2C) for beginners.* [Online]
Available at: https://sarahmaker.com/c2c-crochet/#:~:text=C2C%20stands%20for%20corner%2Dto,scarves%2C%20shawls%2C%20and%20dish cloths.
[Accessed 22 02 2023].

NPM, 2023. *cordova-plugin-camera - npm.* [Online]
Available at: https://www.npmjs.com/package/cordova-plugin-camera?activeTab=readme
[Accessed 17 02 2023].

NPM, 2023. *ecosystem:cordova -npm search.* [Online]
Available at: https://www.npmjs.com/search?q=ecosystem%3Acordova
[Accessed 17 02 2023].

Pixel-Stitch, 2023. *Pixel-Stitch - Convert photos and images into cross stitch.* [Online]
Available at: https://www.pixel-stitch.net/index.html
[Accessed 22 02 2023].