ENGINEERING AND APPLIED SCIENCE

CS3040 MOBILE DESIGN AND DEVELOPMENT

COURSEWORK REPORT

Memori: A Mobile Travel Journal Application

A mobile application replacement for a travel journal

March 2020

Bhaven Patel
160108204
pateb107@aston.ac.uk

Word Count: 2683

Table of Contents

Introduction

This report details the development of Memori, a travel journal mobile application, from ideation through to implementation. The report begins by detailing where the inspiration for the user interface originated and then explains how this inspiration was applied. This expands further to showcase the functionality behind the UI and how several aspects have been implemented. The final section of the report will discuss how a potential evaluation would be carried out, including a self-evaluation about Memori.

Design Motivation

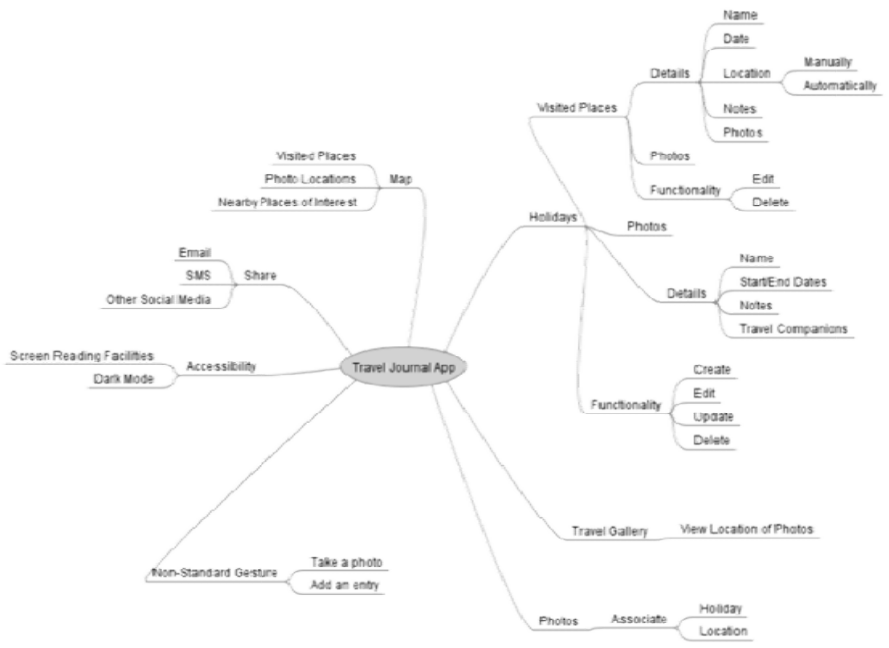
What motivated the design for Memori?

The overall design for Memori was inspired by a keen desire to ensure that users are not flummoxed by the range of functionality available. From my own experience of using a wide range of mobile applications, users are often faced with the challenge of navigating through applications that can be overwhelming, due to the difficulty of finding particular features. This coupled with the stress often related with travelling and the requirement to locate key information promptly led me to design Memori in this way. This ambition fuelled the beginning of application design and allowed me to conduct some background research for a potential structure.

What background research was conducted for the design?

The background research for Memori was first conducted around current travel journal applications. These apps provided enough insight on how a potential user would expect to see this type of application. From this, I was able to determine the key aspects of a mobile travel journal the user would expect to see.

Using this information, I was able to proceed to the conceptual design and draft a semantic network of what is required from travel journal apps and another network for what is required from the coursework specification, as shown below. The combination of these 2 networks allowed me to match up significant components which could be used within the next phase of conceptual design.



In order to further enhance my knowledge around a potential design, a card sort was used with family and friends to understand more about how users would expect to categorise the system. I used areas of functionality which would be found within a potential app and compared this to the previously created semantic networks.

Application UI Design

How does the user navigate through the app?

This sub-section will describe a user-journey on how the user will interact with Memori with the first page being the 'Holidays' page [Figure 1]. Within this page, there is functionality for creating, editing, viewing and deleting a holiday. To create a new holiday, the user can select the 'Create a Holiday' button which will take the user to another page [Figure 2]. This newly created Holiday can be viewed by selecting it within the holiday list, which will take them to the 'View Holiday' page [Figure 3]. To edit or delete a particular holiday, the user must select the toolbar where the 2 options will be provided and then subsequently select a holiday. Editing a holiday will navigate the user to [Figure 4], whereas deleting a holiday will delete the holiday.

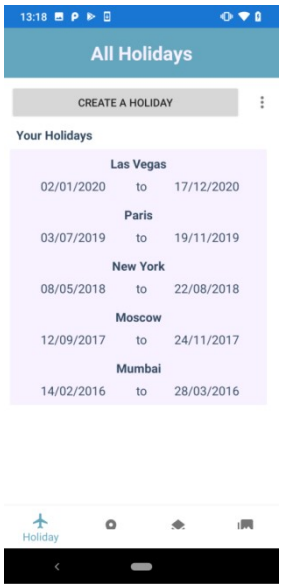


Figure 1: Holidays Page

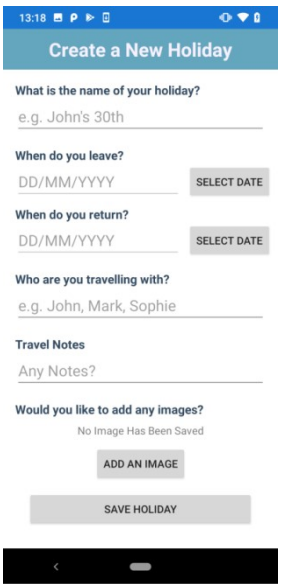


Figure 2: Create Holiday

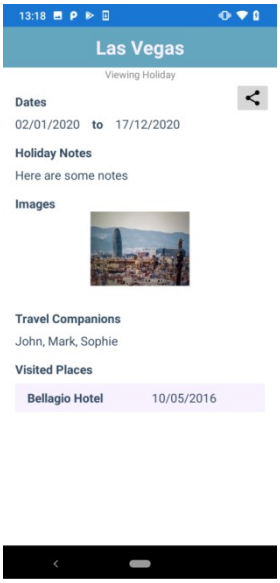


Figure 3: View Holiday

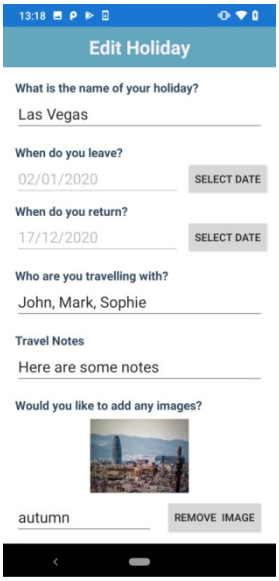


Figure 4: Edit Holiday

The next significant page within the system is the 'Visited Places' page [Figure 5]. The structure is similar to the Holidays page; hence functionality will also operate in the same manner. A new visited place can be created within [Figure 6]. Viewing a visited place created will show [Figure 7]. Functionality to edit and delete a particular visited place is in the same location (within the toolbar) as the Holiday page. Edit Visited Place will also navigate the user to [Figure 8].

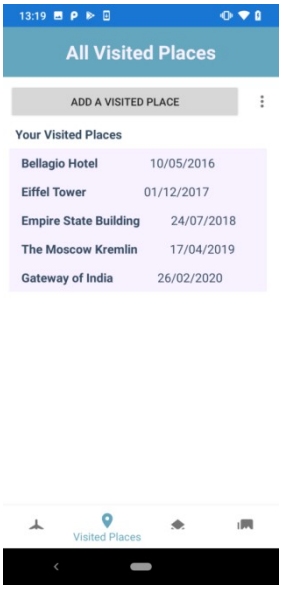


Figure 5: Visited Place Page

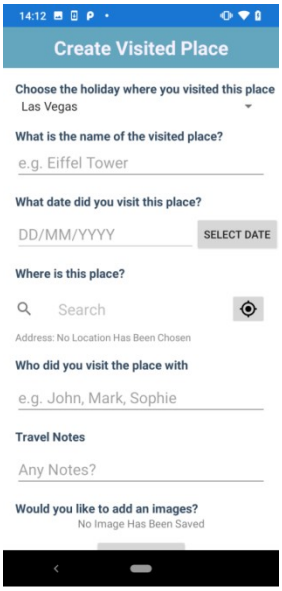


Figure 6: Create Visited Place

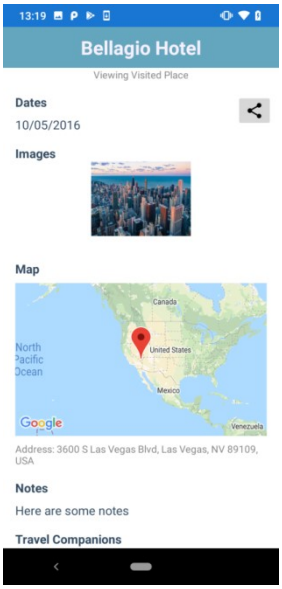


Figure 7: View Visited Place

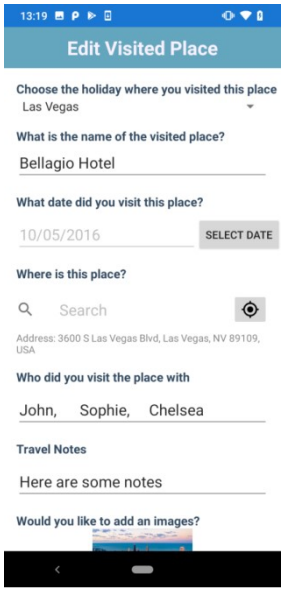


Figure 8: Edit Visited Place

In addition to the Holidays and Visited Places pages, there is also the Maps page [Figure 9]. From this page, the user will be able to view the location of any Visited Places they create and also any images they take. The map will show 2 different sets of markers; blue markers represent a visited place and pink markers represent an image. Clicking on these markers will take the user to another page where they will be able to view more information [Figure 10 & 11].

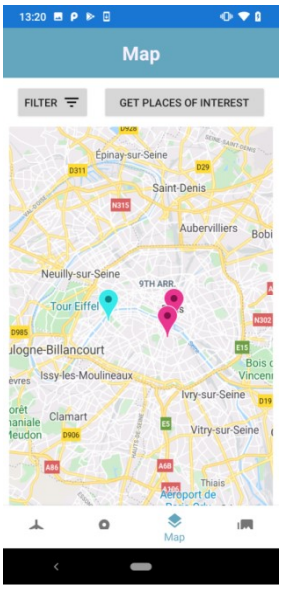


Figure 9: Map Page

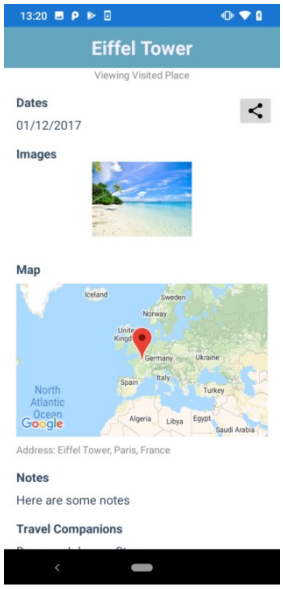


Figure 10: Viewing Visited Place (from Map)

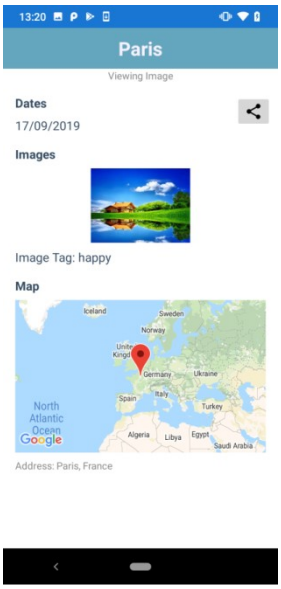


Figure 11: Viewing Image (from Map)

If the user wants to be more specific about which Visited Place markers are displayed, they can select the 'Filter' button. This will provide allow the user to filter the marker by different options [Figure 12 & 13]. The filtered markers will then show on the map, this functionality applies for both the Filter by Date option and Filter by Companions option.

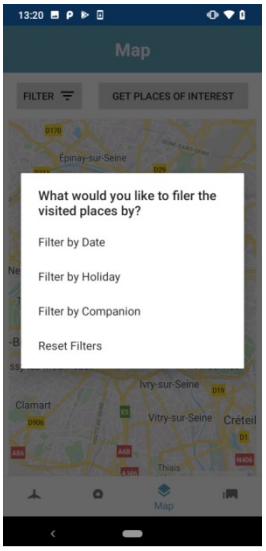


Figure 12: Map Marker Filter Options

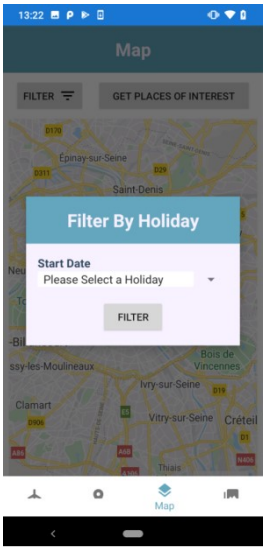


Figure 13: Filter Markers by Holiday

The user will also be able to obtain nearby places of interest via the Get Places of Interest button. By clicking on this button, a pop-up menu will be presented which shows all the categories of nearby places of interest [Figure 14]. By selecting a category, the user will be taken to another page where they will see another map with 2 sets of markers [Figure 15]. The red marker will represent the user's current location, whereas the blue markers will represent nearby places. By clicking on a blue marker, the user will be able to use Google Maps by selecting the button in the bottom right-hand corner, to navigate to the place of interest, [Figure 16].

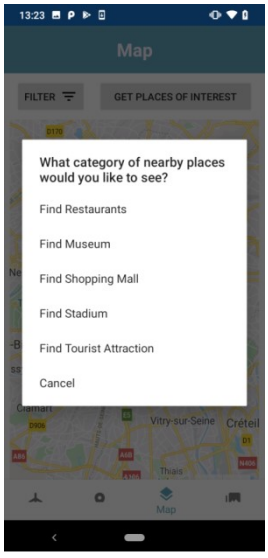


Figure 14: Find Nearby Places of Interest

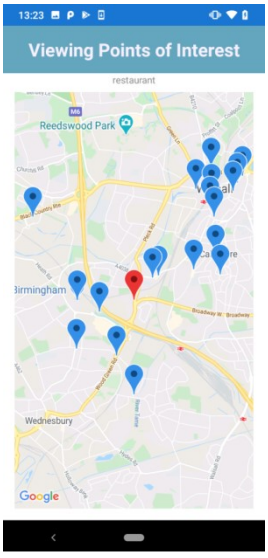


Figure 15: Viewing Nearby Restaurants

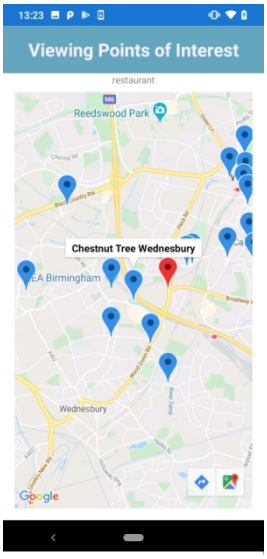


Figure 16: Selected Restaurant Marker

The final page within the system is the Gallery page which is where the user will be able to view all of the images that are saved within Memori [Figure 17]. The user will be able to select any image to learn more information [Figure 18]. There is also functionality to search and filter the images according to the user's choice, via the Search button. Upon selecting Search, a pop-up menu will be presented, showing the user the different ways that the images can be searched [Figure 19]. Each method of searching will present a new pop-up dialog which will use the user's input to search through the images and return the matched items [Figure 20]. However if there is no matched items, the user can reset the gallery by selecting Reset within the Search menu.

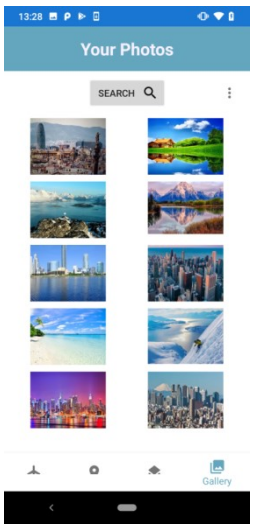


Figure 17: Gallery Page

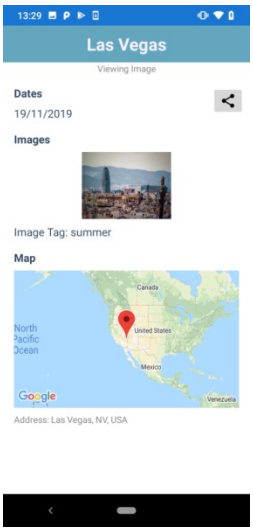


Figure 18: View Image (from Gallery)

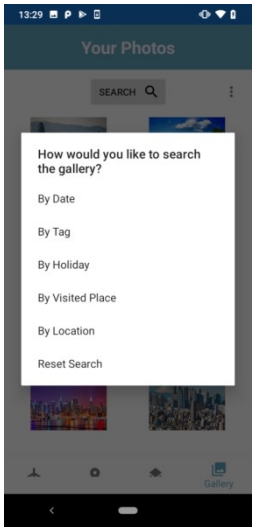


Figure 19: Search Options

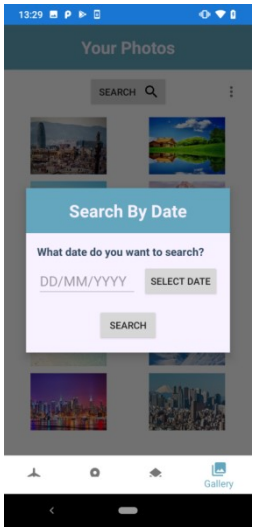


Figure 20: Search by Date Dialog

The toolbar within the Gallery page will allow the user to sort the images [Figure 21]. Each category can be further sorted depending on the arrangement, for example sorting by name can be sorted by A-Z or Z-A [Figure 22]. This is a similar case with each category.

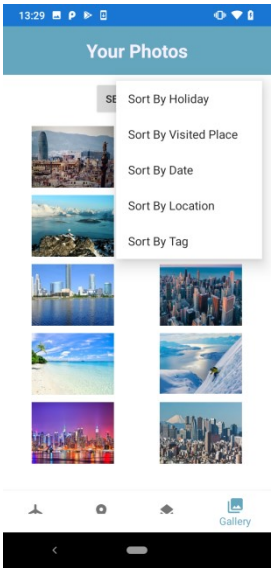


Figure 21: Sort Gallery (from Toolbar)

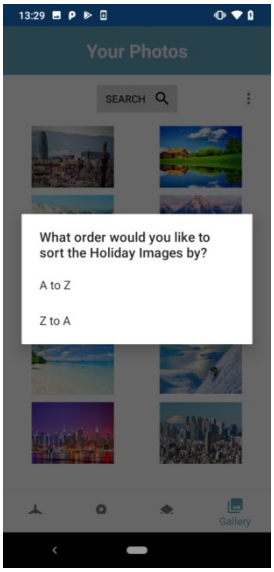


Figure 22: Sort by Name

Within the view page of each element, there will also be a share button which will allow the user to share the Holidays, Visited Places and Images to a friend through SMS or social media. An example of this share screen is shown in [Figure 23].

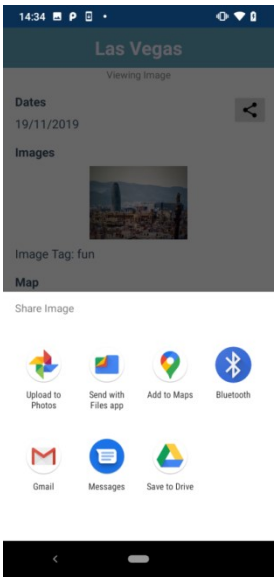


Figure 23: Sort Gallery (from Toolbar)

As a hidden easter-egg within Memori, clicking the title bar within 1 of the 4 main pages will perform an action. For Holidays and Visited Place, it will present the user with a form to create a new Holiday/Visited Place respectively. However for other pages; just clicking the title bar will present a comical message [Figure 24].

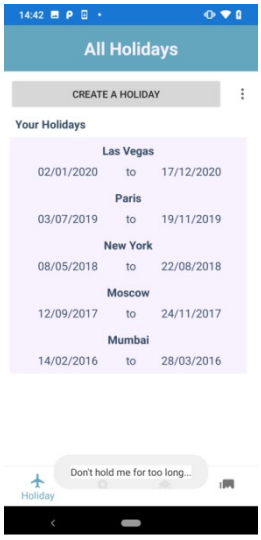


Figure 24: Hidden Easter Egg Message

Why the UI was designed this way?

The user interface of Memori adapts 2 major design patterns; the first of which being the Global Navigation Bar. This is to ensure that the user is aware of what page they are currently on and also, what other pages are available to them. The second pattern is the Visual Framework pattern; which will assist the user as it will provide a sense of familiarity as they will know where to find a particular function, regardless of what page they are currently on.

In addition to the layout of the UI, my main objective was to ensure that Memori achieved the best colour contrast possible. In order to confirm if the colours were correctly chosen, I implemented a colour blindness test to the Holiday page from the UI design (Coblis — Color Blindness Simulator – Colblindor, 2020). This was to ensure that elements were still visible, regardless of any colour-blindness that the user may have, as shown in [Figure 25 – 28].

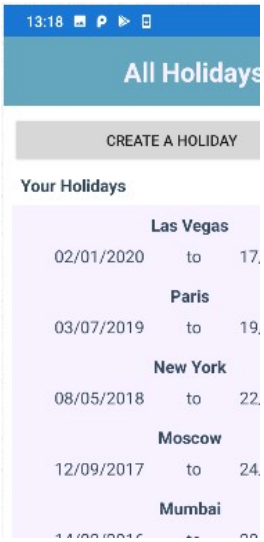


Figure 25: Normal Screenshot

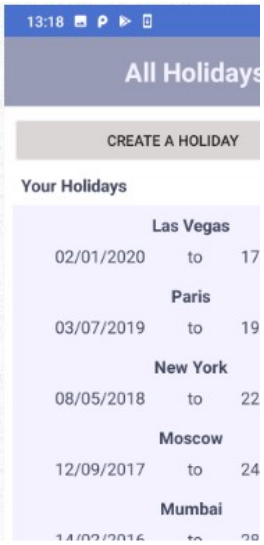


Figure 26: Protanopic Screenshot

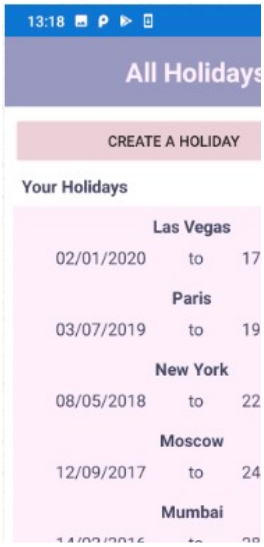


Figure 27: Deuteranopic Screenshot

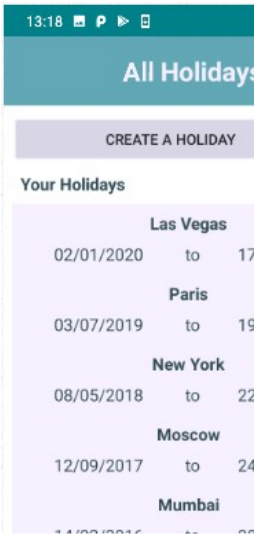


Figure 28: Tritanopic Screenshot

Implementation

What was the overall structure/architecture of Memori?

The overall architecture of the system revolves around the combination of 2 software design patterns; Singleton and Observer. The Singleton pattern was necessary as it involved using a single class (which can be referenced by one of the classes within the Entities sub-package) to help manage its own instance and still be able to manipulate freely throughout the application. This ability to modify itself allowed for an easy implementation of the Observer pattern. As there is a possibility that there will be several Holidays created, each Holiday must follow a strict structure to reduce likelihood of runtime errors. This is key when saving Holidays to a database as any inconsistencies could cause some more significant errors in the future.

Which aspects were difficult to implement and how were they achieved?

A particular aspect which was difficult to implement was the functionality involved to display the map markers within Map page. In order to display and edit all the markers shown within the map, a method of iteration was required. Each map marker takes several seconds to load and display, however if a for/while loop was used then the system would add a marker before it was ready. This caused null reference exception error at runtime. In order to overcome this, I used recursion to help 'slow' down the system and correctly display the current marker before moving to the next. This is shown in the following images, or further viewed within the MapFragment.java class within the directory 'app\src\main\java\com\example\memori\ui\map' of the application:

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    retrieveTables();

    if(filtersActive){
        displayToast("All filters have been reset, lo
        filtersActive = false;
    } else {
        displayToast("Loading all Map Markers");
    }
```

This is where the method is first called from. The onMapReady() method is called when the Map fragment is ready within the Map page

```
private void setAllVPlaceMarkers(List<VisitedPlace> impAllVPlaces){
    if (impAllVPlaces != null) {
        int numVPlaces = impAllVPlaces.size();

        if (i < numVPlaces) {
            VisitedPlace visitedPlace = impAllVPlaces.get(i);
            String placeID = visitedPlace.getLocation();

            if (placeID != "") {
                if (!Places.isInitialized()) {
                    Places.initialize(getActivity(),  s: "AIzaSyDMPsU2SV31
                }

                // Define a Place ID.
                String placeId = placeID;

                // Specify the fields to return.s
                List<Place.Field> placeFields = (Arrays.asList(Place.Field
                        Place.Field.ADDRESS, Place.Field.LAT_LNG));

                // Construct a request object, passing the place ID and f
                FetchPlaceRequest request = FetchPlaceRequest.newInstance(
                PlacesClient placesClient = Places.createClient(getActivit
                placesClient.fetchPlace(request).addOnSuccessListener((res
                    setMarker( type: "VPlaceFragment", response.getPlace().
                            response.getPlace().getLatLng().longitude, vis
```

This acts as an iterator which will iterate through the array

This line calls another method which draws the marker onto the map. This line occurs too fast within normal loops

This is where the recursion occurs, in addition to the changes to the iterator to ensure the next item is used

Another aspect which was particularly difficult to implement was the sort function for the images within the Gallery. The main cause of this was the number of different fields that the user could sort by; this involved implementing a unique method of sorting for each field. Below is an extract of the code which was developed to sort the images in the order of old to new, which can be found within the GalleryFragment.java file within the following directory, 'app\src\main\java\com\example\memori\ui\gallery'.

```
case SORT_OPTIONS_DATE_OLD_TO_NEW:
    Log.d( tag: "OrderGallery",  msg: "GalleryFragment: Sorting Date Old
    images.clear();
    imageSortField.clear();

    ArrayList<Date> imageONDates = new ArrayList<>();

    HashMap<Date, Images> hmONImageToString = new HashMap<>();

    for (Images currentImage : mImages){
        String currentDateString = currentImage.getDate();

        Log.d( tag: "OrderGallery",  msg: "-----------------");

        int currentYear = Integer.parseInt(currentDateString.substring
        int currentMonth = Integer.parseInt(currentDateString.substrin
        int currentDay = Integer.parseInt(currentDateString.substring(

        Calendar calendar = Calendar.getInstance();
        calendar.set(currentYear,  month: currentMonth - 1, currentDay);

        imageONDates.add(calendar.getTime());
        hmONImageToString.put(calendar.getTime(), currentImage);

    }
```

This enhanced for loop will traverse through each image to find its date

This section uses string manipulation to extracts the date from the Image database table and creates a new Calendar object

These 2 lines of code add the newly created date object to an ArrayList to get sorted. And also a Hashmap to ensure that the original image can be obtained via the date

This line sorts the dates from old to new accordingly

This final enhanced for loop iterates through the sorted ArrayList and returns the new list

What software testing was performed?

The majority of testing that was conducted within the development of Memori was directly related to the applications ability to meet a particular requirement. Originally, this involved a lot of black & white box testing to ensure that particular changes occurred correctly and that it did not occur by accident. This was practical in discovering a major bug in relation to the database, as it did not correctly update a field but it did update the object itself; mainly related to the SQL query involved in updating the record.

In addition to this, a combination of usability and robustness testing was used to ensure that potential functionality within the system was performed correctly. But in order to ensure that this is consistent, a large amount of robustness testing was required. This lead to the implementation of strict data validation to ensure that all anomalies entered were spotted and dealt with appropriately.

Evaluation

How would Memori be evaluated to determine its usability?

In order to evaluate Memori and its performance within the real-world, I would compare it to rival travel journal applications to assess its efficiency and performance in completing several tasks. If I were to conduct a usability evaluation, I would set up a lab-based experiment in which 2 groups of equal size would use Memori and a competitor's app, respectively. The participants would be timed to see how long it would take to perform a series of tasks. And finally, I would ask the users to complete a questionnaire regarding the user-satisfaction of the application; which would implement the NASA TLX scale. Upon collecting the results, I would be able to determine the area's that Memori performs well and be able to make potential improvements on areas where Memori did not perform so well.

Self-Evaluation & Reflection

What are the strengths/weaknesses of Memori?

I believe that Memori excels in various aspects, the first of which is its simplicity in design which allows for fast navigation throughout the application; due to the visual framework pattern. This can be very practical as a complex UI may be less efficient in completing tasks and could mean the user must allocate time from their busy schedule in order to make changes to the journal. Whereas with Memori, any potential changes could be made on a bus ride from one visited place to another.

On the other hand, there are aspects within Memori that can be considered a flaw. There are areas within the application that could have used a more efficient approach during the implementation; one of which being within the Map page. For example, displaying the markers on the Map page takes a few seconds due to the recursion algorithm previously mentioned. This waiting time is directly proportional to the number of markers the map as to display, hence there is a possibility that the user must wait a significant time before using the application, if a large number of markers have been saved.

What elements could be improved?

An element that could be improved within Memori is the aesthetics within UI. This is mainly because- in my opinion- Memori's simplistic design may not appeal to younger generations. Although this audience may not consider using a travel journal, they are the most capable with regards to being able to travel around the world. This is something I would improve on if there were to be another version.

How well does Memori meet the requirements?

Based on a requirements traceability exercise carried out, which aligns the requirements to the design and implementation, Memori satisfies all 'must have' requirements. The user is comfortably able to create, edit and delete Holidays and Visited Places. Users can also view location markers of a particular Visited Place or Image. In addition to this, all of the information saved within Memori is saved within persistent storage within the device, in the form of a database.

There are some requirements however which have achieved, but could have used further improvement. For example, the gesture could be improved to use more of the devices sensors such as shake. This could be used to trigger a new action such as editing the last holiday or adding a new image. Apart from this, I believe that Memori has achieved the requirements appropriately.

Resources List

Bibliography

Color-blindness.com. 2020. *Coblis — Color Blindness Simulator – Colblindor*. [Online] Available at: <https://www.color-blindness.com/coblis-color-blindness-simulator/> [Accessed 18 March 2020].

Extra Figures

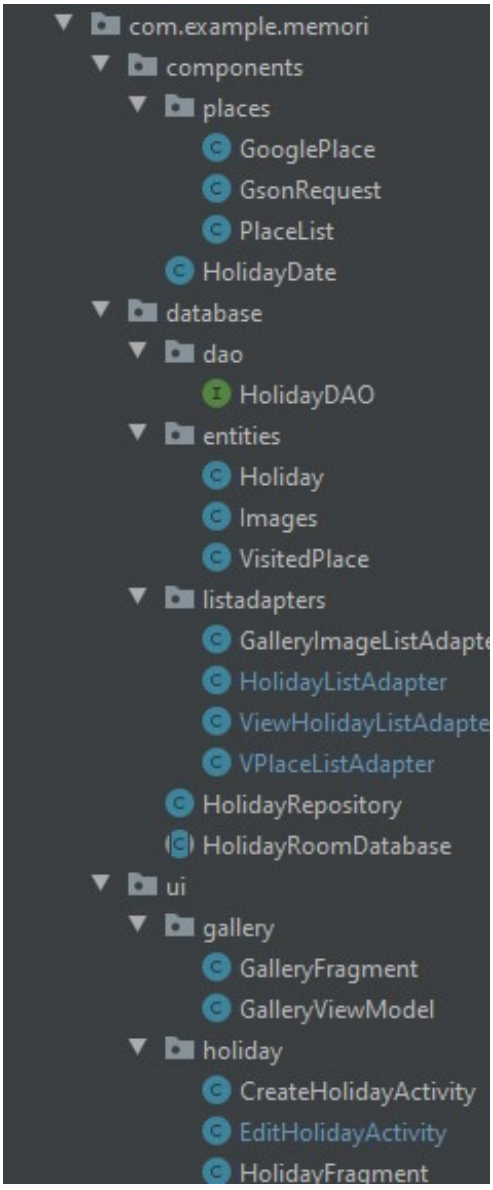


Figure 29: Screenshot of Package Organisation