EventMe

Report for the Praxisprojekt Real-Time Mobile NLP Analysis SS21 at the University of Duisburg and Essen

Carina Erlebach
Frontend
carina.erlebach@stud.uni-due.de

Caitlin Daria Rösen
Design
caitlin.roesen@stud.uni-due.de

David Mischitz
Backend
david.mischitz@stud.uni-due.de

***Abstract*—EventMe is a mobile application that can analyse a picture via NLP. The user gets encouraged to take a picture or open one from the gallery. After accepting the picture the user sees the results and can try it again. [?] [?]**

***Index Terms*—App Design, NLP, AndroidStudio, Java, Optical Character Recognition, Python, Flask, Heroku, Cloud Vision, Github**

I. Introduction

The first idea for our application was given through partyholic. Our Main idea was to create an application to help one organising and not forgetting an event seen on e.g. an advertisement. Basically the application shall use a new picture taken or from the gallery and analyse any text on it. We are specifically looking for dates and a related location for the event. In our research we found several many calendar or event planning applications. None of them were for event management in that way we designed our to help manage and organize them e.g. connecting the app with one's Google calendar and instantly adding the event. The way our application sets events and finds the information is seemingly the only one at the moment. Thus it appears we found a gap in the market. After talking to one another and figuring out how we wanted the application to work we arranged us with what was to do.

A. Motivation

Sometimes one spots an interesting sounding event on a poster while walking through the city and thinks that this could be a nice event to attend. However one usually does not remember the name of the event because 'of course I can remember this easily' and on the way one thinks about many other things. In the end one has forgotten and/or just doesn't quite care anymore. A picture taken of the poster most of the time is forgotten in a phones gallery. For what we know this event comes back to mind when the event is the same day or even just a day late. To solve this problem we worked out how this could be fixed by coding EventMe. This application shall be used for these occasions where one comes across such a poster. Our application will scan the poster looking for a date and add it into the Google calendar. With this technique one can walk past a poster, scan it and has the event in their calendar. Thus making it easier to remember and go to the event.

B. Existing Apps / Market Analysis

In order to make sure that we would not code an already existing application or use an existing concept we made some research. This research consisted of us searching for applications with the same intention as ours in both the Google Play Store and the Apple App Store. After some time we mainly found applications which are basically calendar apps. Through this we realised that with our idea we found a gap in the market which we know can fill. I.

C. Natural Language Processing

One of the major goals of this Project was to explore the possibilities of natural language processing (NLP) in mobile app development. NLP is a scientific field that tries to make natural human language understandable for a computer. The goal is to extract interpret and organize relevant information from documents, pictures and spoken words. In our Application NLP is used to analyze the text on an image to find phrases containing date or time information. Date and time can be expressed in various different ways e.g. 12:00 or 12 o'clock, 10.03 or 10TH March. For a Computer to find and interpret these phrases NLP is needed. To tackle this Problem we used a python Module that utilizes regular expressions. Regular expressions are a part of NLP and used to match phrases in the text with specific patterns. Matching phrases are then converted into datetime objects(e.g. 2019-04-20 19:00:00). Datetime objects are standardized and easy to work with.

D. Overview

In Section II we will first discuss about the Design of the app, followed by the Front-End and Back-End development in the Sections III and IV. We conclude the report by our future vision in Section VI.

II. Design

A good app design can improve the overall user experience. Nice interfaces, noteworthy simplicity and easy navigation are fairly prominent traits of a good app. The design can be part of why people come back to the app. Of course only secondarily.

A. Name / Icon

After discussing some name ideas which were fairly simple and not as memorable, we collectively sorted out what we wanted to express with the name. We wanted the name to

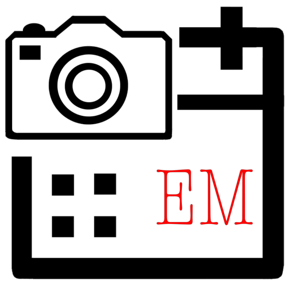

Fig. 1. EventMe Logo

express to some extent what the application does. Reminding someone of an event or rather being able to set a reminder in some sort of way to remind one of said event. The name EventMe is a composition of the phrase “remind me” of an event for example and the purpose of reminding someone which is the actual event. Thus, immediately catching the focus because the app before even opening it implies a direct use for the user. While the name catches the attraction coming directly from the user and giving a purpose for the immediate own use the logo shall rather portray the function of the application. The logo resembles the known calendar icon used in other applications or even notebooks, but with a twist. We replaced the ring chain with a camera to show the interaction between the camera and calendar function. With this design we imply a specific usage of both the calendar and the camera. The idea behind the logo was to show the two main aspects of EventMe which are using a picture and basically integrating it into ones calendar.

B. *Color Scheme*

The initial thought was a bright red which draws the attention to the app overall. After some research we decided to use a less vibrant colour which now became a colour gradient which is seemingly calm. Nonetheless the gradient design we decided on still has red in it as well as orange and a very light peach colour. In symbolism red and orange are the colours of not only health and technology but even more of leisure time and entertainment. [1] Moreover these are enthusiasic and emotion-handling colours which are used to appeal to an user to interact with the application. [2]

C. *Navigation*

For the navigation we wanted EventMe to be as intuitive as possible. To create more screen space we wanted upon the hamburger menu. Most users already are familiar with this type of menu and are used to using it. We weren’t able to implement this initial idea. Thus we needed to reconstruct our application. Later on we realized that a bar based navigation was what we wanted. With two options to choose from the user can start taking pictures and implementing the dates for events into their calendar.

D. *Software*

We decided to use uizard [3] as it provides the possibility of sketching your wireframes and integrating them into uizard. Furthermore one can easily switch between themes hence being able to quickly exchange different things such as buttons in the wireframe model or the actual design. or the logo from which our application shall be recognized we used Adobe Fresco Version 2.7.5. We also tried InVision Studio which could be used for animation but overall the understanding of the software was harder and more complicated than uizard. Afterwards the first thing to do was to work out how the app should look. The first drafts were easily sketched out with Adobe Fresco and were wireframe models. Uizard allowed us to switch from the wireframe to an actual aspired design draft. This draft was the most simple version of what we wanted the application to look like.

E. *Workflow + Issues*

As the design was the first part of the application to be finished we immediately started working on it. The process of the first few steps was fairly easy and rather fast accomplished. The designing of the logo took some attempts in order to correctly adjust every placement of each part of it. Furthermore in order to add the logo into the front-end we needed to adjust the picture format several times because for a still unknown reason the ’normal’ format did not work. Furthermore since we had some issues implementing the first design we needed to redo all of it which we then did directly in the front-end making it a work-in-progress whilst working on the front-end. Overall most issues were solved with some trial and error work and research.

III. Front-end Development

The goal of the Front-end development is to build an app interface. It has to combine the visual requirements of the Design and the server functions from the Back-End. The whole app needs to be looking nice and have a friction less user experience. In this chapter there will be an explanation of our software decisions followed by an elucidation of our work and some of the major problems we faced.

A. *Software Decisions*

We build our front end with AndroidStudio which uses Java as a language. We chose AndroidStudio because it has a clean interface and is very easy to use for beginners. AndroidStudio supports Java and Kotlin but we chose Java because we worked with it before and we were not comfortable with learning a fully new language. Looking back we should have chosen Kotlin because it is a shorter way than Java. This could have helped us with time and bugs because it is less code.

B. Structure

1) Permissions: In order to get access to the camera, gallery, calendar and storage usage the user gets a permission request right after the application is launched. This permission service checks which permissions need to be given. These permissions are required for the functioning of the application.

```
if (ActivityCompat.checkSelfPermission(this,
    Manifest.permission.WRITE_EXTERNAL_STORAGE)
    == PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission
            (this, Manifest.permission.
            READ_EXTERNAL_STORAGE) ==
                PackageManager.PERMISSION_GRANTED
                &&
            ActivityCompat.checkSelfPermission
            (this,
                Manifest.permission.CAMERA) ==
            PackageManager.PERMISSION_GRANTED
                &&
            ActivityCompat.checkSelfPermission
            (this,Manifest.permission.
            WRITE_CALENDAR) ==
                PackageManager.PERMISSION_GRANTED
                &&
            ActivityCompat.checkSelfPermission
            (this,
                Manifest.permission.READ_CALENDAR)
                ==
                PackageManager.PERMISSION_GRANTED
                &&
            ActivityCompat.checkSelfPermission
            (this,
                Manifest.permission.INTERNET)
                ==
                PackageManager.PERMISSION_GRANTED)
                {
        this.initializePlease();
    }
```

2) Pictures: As one of the main functionalities of the Front-End a well functioning picture taking is of great importance. We decided to use the build in camera or to choose a picture from the gallery. The methods we used are Intents. Intent is a Java class by Google. It allows developers to access a wide range of the Android device functionalities.

```
 Uri photoURI =
     FileProvider.getUriForFile(this,
     "com.carina.eventme.file",
     photoFileForCam);
          takePictureIntent.putExtra(
          MediaStore.EXTRA_OUTPUT, photoURI);
              startActivityForResult
          (takePictureIntent,
              REQUEST_CODE_CAMERA);
```

3) Back-End Connection: The connection to the Back-End was done with Https. It is send with a binary decoding and writing it into an output stream.

```
 URL url = new
     URL("https://eventmet.herokuapp.com/imgt");
          HttpURLConnection
              httpURLConnection =
              (HttpURLConnection)
              url.openConnection();
       httpURLConnection.setRequestMethod("POST");
       httpURLConnection.setRequestProperty
        ("Content-Type", "image/jpeg");
     httpURLConnection.setRequestProperty
     ("Content-Disposition",
         "form-data;name=image");
        httpURLConnection.setDoOutput(true);
      httpURLConnection.setDoOutput(true);//#
 //open an outputstream with the connection
          OutputStream out =
              httpURLConnection.
          getOutputStream();
          DataOutputStream os = new
              DataOutputStream(out);
 //writing the image into the data output
     stream:
          os.write(fileContents, 0,
              fileContents.length);
```

As a responds we get back a .json file which is then written into a string to access the date, time and title.

```
 JSONObject jsonObject = new JSONObject
 (responseWhole.toString());
          MainActivity.date = (String)
              jsonObject.get("text");
          MainActivity.time = (String)
              jsonObject.get("text");
          MainActivity.title = (String)
              jsonObject.get("text");
```

4) Calendar: We used an intent again to access the Google calendar. With a click on a button the calendar opens and it inserts the Strings from the Back-end. To have the option of correcting a wrong information given back we used an edit text. [4]

```
 Intent calenderintent = new
      Intent(Intent.ACTION_INSERT);
     calenderintent.setData
     (CalendarContract.Events.CONTENT_URI);
       calenderintent.putExtra
       (CalendarContract.Events.TITLE ,
            titleevent.getText().toString());
       calenderintent.putExtra
       (CalendarContract.EXTRA_EVENT_BEGIN_TIME,
            dateevent.getText().toString());
        if (calenderintent.resolveActivity
        (getPackageManager())!= null){
              startActivity(calenderintent);
        } else {
   Toast.makeText(MainActivity.this, "There
        is no app that can support this
        action",
       Toast.LENGTH_SHORT).show(); }
```

C. Problems

1) Fragments: We had a big problem with Fragments. Fragments are designed to be reusable UI layouts with logic embedded inside of them. The first problem we encountered was a button onClick. The button onClick does not work the same in Fragments as it does in Activities. It was not possible to make public methods that every button could use. So we switched every Fragment we designed into an Activity and it worked perfectly fine. [5]

2) Camera: Opening the camera function was a problem as well. Due to not having an android phone we needed to use the AndroidStudio emulators. We always got an Error message that it is doing too much on its main thread and skipped too many frames. After trying to find the error it just worked. We put the Log.d method between each line and it did not skip so many frames anymore.

```
 Log.d("camera", "eins");
```

3) Connection: Trying to get the Back-end connection to work took a lot of the time. First we never got an connection. Then we tried Retrofit2 to make it work but it did not work either. The main problem was sending the right format to the Back-end. So we just used the same method as the Partyholic App from the Praxisproject earlier but it still did not work. [6]

4) Solutions: Mainly we used Youtube and stack overflow to help build our app. We used a lot of different tutorials. The AndroidStudio developers page helped us as well. The code from the Partyholic App was a great advantage for us. Everything we have done was done before so we could get an idea how it should work and took our inspiration from there. [4] [5] [6] [7] [8]

D. Summary

To sum up the Front-End development for EventMe included some essential functionalities like the camera, permission service and the Google calendar. The most important thing is a functioning Back-End connection. Without the connection nothing would have made sense. Sadly we could not get it to work. Implementing a splashscreen and edit text was important to us as well because that is what the user sees and can interact with the App.

IV. Back-end Development

While the front end is what the user sees the backend is rather the, backround player‘ and manages how the show is run. Mainly the backend contains the server which provides the data upon requests sent, the database organizing the information and the application channeling it.

A. Frameworks and API

We used Flask as a way to create a simple web application, as it does not require particular tools or libraries. We deployed that application to Heroku which is a cloud platform using containers to run our application on a virtual environment. We tested different iterations of the application using separate containers, pushing our code directly from GitHub. The most important task of our backend is the image processing for this we used Google Cloud Vision. It is a powerful and reliable API that analyses images via deep learning. Vast quantities of training data provided from Google makes it better than most similar APIs

B. Structure

1) Receiving The Image: The frontend sends an binary decoded image to the backend server by writing it into an output stream. Once the file stream is red the image is temporarily saved to the server using functions provided by the Pillow library. [9]

```
file = request.files['image'] #image file is
    submitted to the server via POST request
img = Image.open(file.stream)
img = img.save("saved.jpg") #image is saved
    to the server overwriting a possibly
    existing image
```

2) Optical Character Recognition: The image is then sent to the Google Cloud server where it is processed. According to the function the text on the image is sent back in JSON format. [10]

```
client = vision.ImageAnnotatorClient()
   # create instance of ImageAnnotatorClient
      to get access to the Cloud Vision
      image annotater service an its OCR
      functions
with io.open("saved.jpg", 'rb') as image_file:
   content = image_file.read()
image = vision.Image(content=content) # open
    the previously saved image and send it to
    the Google Cloud servers
response = client.text_detection(image=image)
text =
    response.text_annotations[0].description
    # use the text detection function of the
    ImageAnnotatorClient to extract text from
    the image as a string
```

3) Natural Language Processing: To extract the date and time from the text a python module named datefinder is used. This module uses regular expressions to find strings which likely have date or time in them and then converts them into datetime objects. [11]

```
dates = datefinder.find_dates(text) # use the
    datefinder module to extract the date
    from the text as datetime object.
for n in dates:
   dates= str(n) # convert datetime objekt
      into string
```

4) Response To Frontend: As response to the frontend a dictonary containing date and time is sent in a .json file.

```
   result.update({ "date": date, "time":time
      }) # update the dictonary

   return jsonify(result) # return the result
      in json format
```

C. Problems

1) Optical Character Recognition: On the backend side our biggest Problems came with the API used for optical character recognition. We started out using tesseract because it was completely free and very easy to use. The first issue using tesseract was deploying it to heroku we had difficulties locating the important files once we uploaded them to the the server using a buildpack. Simply downgrading the operating system of our server solved the issue. A much bigger issue was that tesseract was not providing good enough results, it only worked with easy to read black text on a white background. Most Posters do not fulfill those requirements.

2) Cloud Vision: So we needed a more versatile and reliable API. This API was Cloud Vision. It provides vastly improved results over tesseract and has many more possibilities for image processing. But Cloud Vision also comes with a downside. The Service is not completely free. At a certain number of requests every additional one will cost a bit of money. For this reason we decided not putting the app on the Google Play Store.

3) Natural Language Processing: We would have liked to extract more information from the Poster to be able to show them in the calendar first and foremost the title of the event. When looking at a poster as a human it is very easy to know what each phrase on the Poster means. All Event Posters while containing similar Information, differ widely when it comes to their layout and word choice. Dates are limited in the way you can express them so when considering all possibilities a program can find and interpret the correct phrases. But for the event title it seems impossible to distinguish it from the rest of the information using a similar method. An AI with enough training data would probably be able to do it. Unfortunately developing an AI exceeds our skills and knowledge.

V. Conclusion

Our future ideas for EventMe include - in addition to the Google Calendar - Google Maps. This integration would give the possibility of having an easy guidance to the event location. Not only would the user be reminded by their calendar that an event is occuring but even more they would know where it is happening and how to get their. Moreover at the current point EventMe is solely accessible for Android systems thus to broaden the possible usage we'd like to make the application usable for iOS systems. For the overall usability and user experience a more reliable extraction of event title and location is inevitable. To achieve this we need to take a look into more advanced NLP methods and try to include these. The application can be downloaded from our github. [12] We used github as it is an open source repository service which allows to store code in a cloud like fashion. Not only could the code be uploaded by any collaborator but even more can be checked by a variety of people who then even can tell us what we've overlooked or forgotten or even errors can be pointed out by the github community. This way we can learn from professionals or people with more coding experience and exchange first drafts easily and with a certain feedback.

VI. Future Vision

Our Future Visions for EventMe possibly could be expanding our application in a way that it is more accessible. This meaning making the app work for iOS devices and through this getting a bigger user audience. Moreover we could change to a more advanced NLP in order to extract the important information more reliable. Furthermore it would be possible to convert location data into coordinates thus making it able to access the location coordinates in Google Maps. Our application will help people who easily forget a seen event in including and managing them in their calendar. Thereby making it a helpful application to many different people regardless of their age or gender.

VII. Where to find the App

Currently our application cannot be found in the app store. Still the link to our github repository is accessible to anyone through this link: https://github.com/caiidar/Event-Planer-NLP . [12]

References

[1] J. Thomas. Farben kombinieren – so verbesserst du die usability deiner webseite. [Online]. Available: https://makeusershappy.de/farben-kombinieren-usability-webseite/

[2] none mentioned. App design: Die bedeutung und wirkung der farben. [Online]. Available: https://de.yeeply.com/blog/app-design-die-bedeutung-der-farben/

[3] C. R. Carina Erlebach, David Mischitz. Design eventme. [Online]. Available: https://app.uizard.io/prototypes/1Al0eW9BwYiWEM6Q0WAr

[4] [Online]. Available: https://www.youtube.com/watch?v=NK_-phxyIAM&t=904s&ab_channel=CodingDemos

[5] [Online]. Available: https://stackoverflow.com/questions/21192386/android-fragment-onclick-button-method

[6] [Online]. Available: https://github.com/smmohash/Partyholic-0.1.0

[7] [Online]. Available: https://developer.android.com/

[8] [Online]. Available: https://www.youtube.com/

[9] A. C. Fredrik Lundh. Image module. [Online]. Available: https://pillow.readthedocs.io/en/stable/reference/Image.html#module-PIL.Image

[10] Imageannotatorclient. [Online]. Available: https://googleapis.dev/nodejs/vision/latest/v1.ImageAnnotatorClient.html

[11] G. C. Alec Koumjian. datefinder - extract dates from text. [Online]. Available: https://datefinder.readthedocs.io/en/latest/

[12] C. R. Carina Erlebach, David Mischitz. Event-planer-nlp. [Online]. Available: https://github.com/caiidar/Event-Planer-NLP

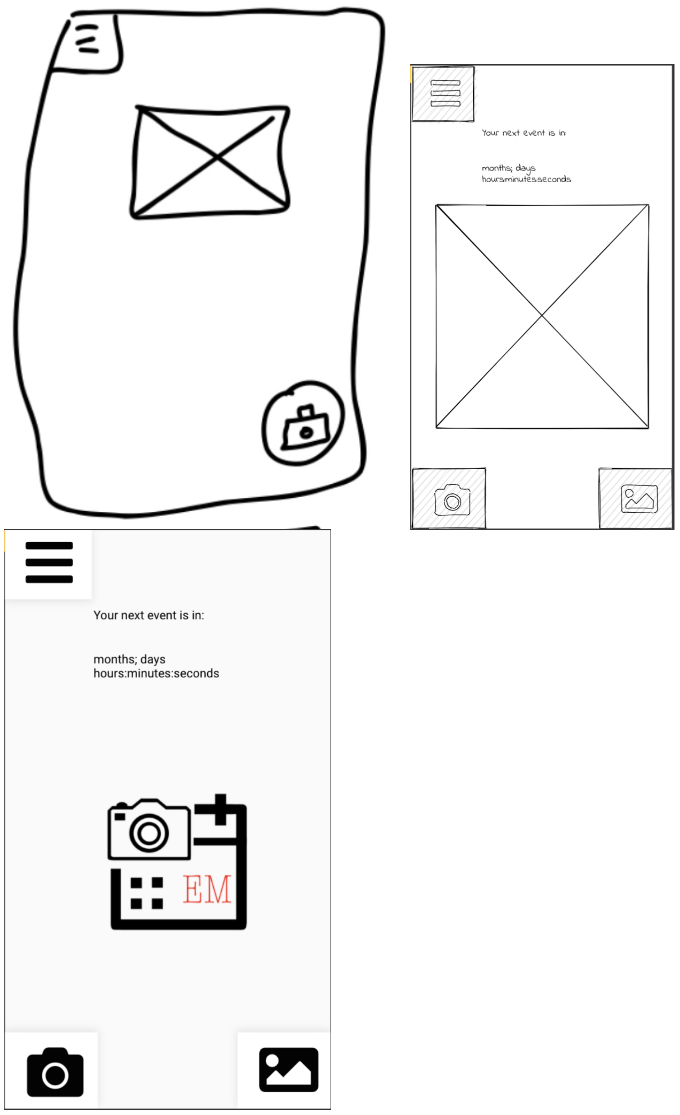

Fig. 2. initial design