



Memory Frame

Cloud base Smart Photo Frame
with Voice Activation

Introduction

The internet. It provides us with a way to connect with our friends and families from all over the world. Even though the internet is now such an important component of our lives, some members of previous generations still haven't quite adapted to using it yet.

Our goal for Memory Frame is to **give people the ability to share their everyday memories with their loved ones who they may not see often. And we want to do this in an organic and natural way.**

Memory Frame

is a voice controlled digital photo frame.

The photos are first uploaded via our android app. This let's the user control what is shown in the frame anywhere they go.

User's can then tell Memory Frame to display any photos they want to see in any given moment.

By saying phases such as, "Show me a photo of Johnny"

Allowing for a personalized and familiar experience.



Market Evaluation

There are similar products offering digital photo frame on the market with the price range of \$200~\$600. However, our hand-free voice control is unique to the market. Our goal is to create an intuitive product, with our easy to use user interfaces, while keeping it affordable.

Our **target segments** are:

1. Young professionals who want to share the photos they take with their distant loved ones
2. Photo lovers who want a new way to enjoy all the photos they've taken



10.1 inch HD Smart Android
Network Version WiFi Cloud
Album Digital Photo Frame
Electronic Album Business...

CDN\$651⁴¹



NIXPLAY Original Digital Photo
Frame WiFi 15 inch W15A. Show
Pictures on your frame via
Mobile App, Email or USB. Sma...

★★★★★ ~ 4

CDN\$485⁸⁷

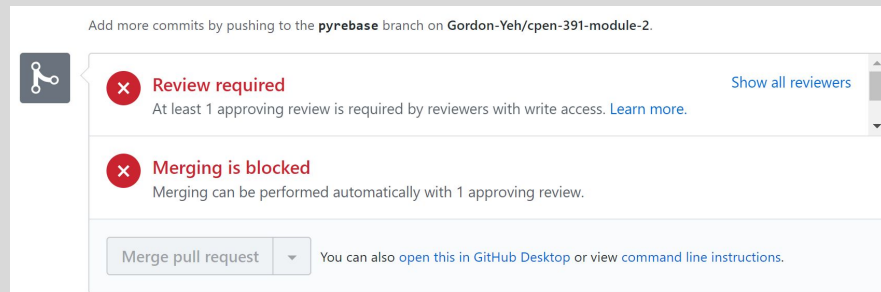
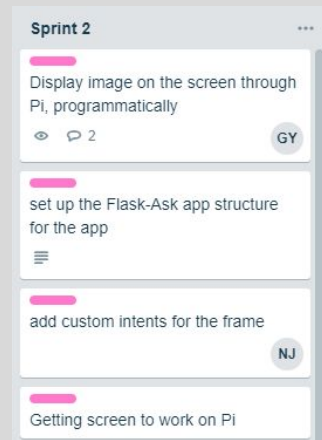
Project Management

Our team was loosely organized into 3 groups. However, we would help each other when necessary

1. Android - Richard, Claire
2. Raspberry Pi - Nathalie, Gordon
3. Database - Skylar

We used Trello assign and track tasks. We used it because we were familiar with it from module 1, and it is also free and easy to use. We found the card and board layout to be very clean and intuitive.

To ensure all code was working, we used Github's pull requests. This meant that another team member must check a branch's code before allowing the branch to merge with our master branch.



Work Accomplished

Android App

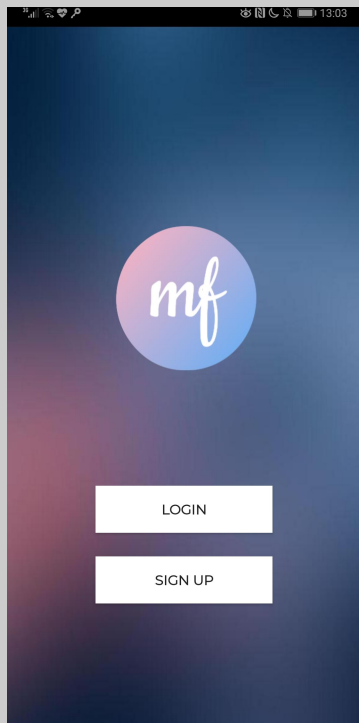
- Authenticate users with email and password
- Connect to Raspberry Pi with bluetooth
- Select picture from gallery
- Take a picture within the app
- Upload pictures to cloud
- Read from cloud database and place the pictures in the database

Raspberry Pi

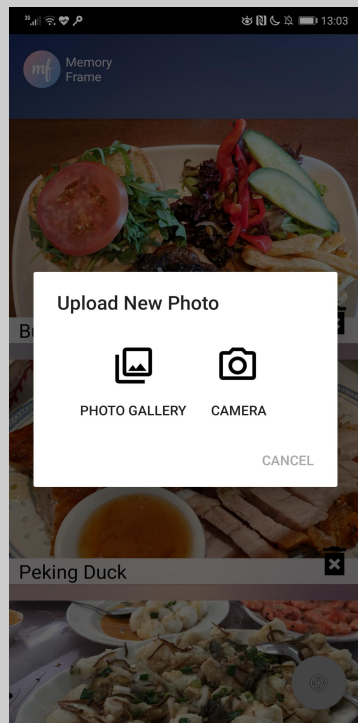
- Control picture display with Alexa commands
- Display images
- Query database with user inputs

Detailed Design: Android

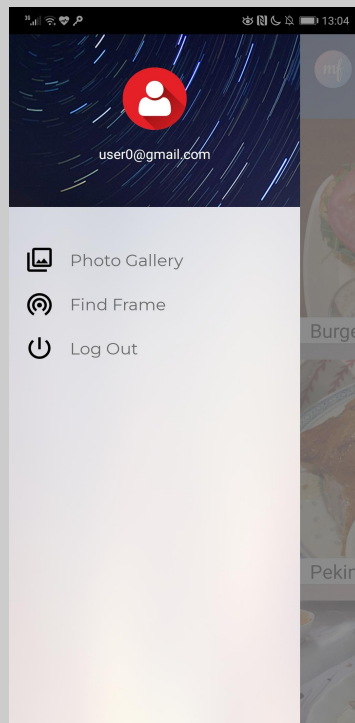
The user uploads photos to our database through our Android app.



Home Menu



Upload Dialog



Navigation Drawer

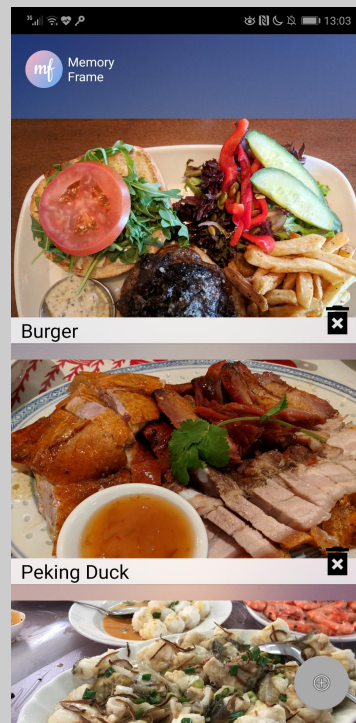
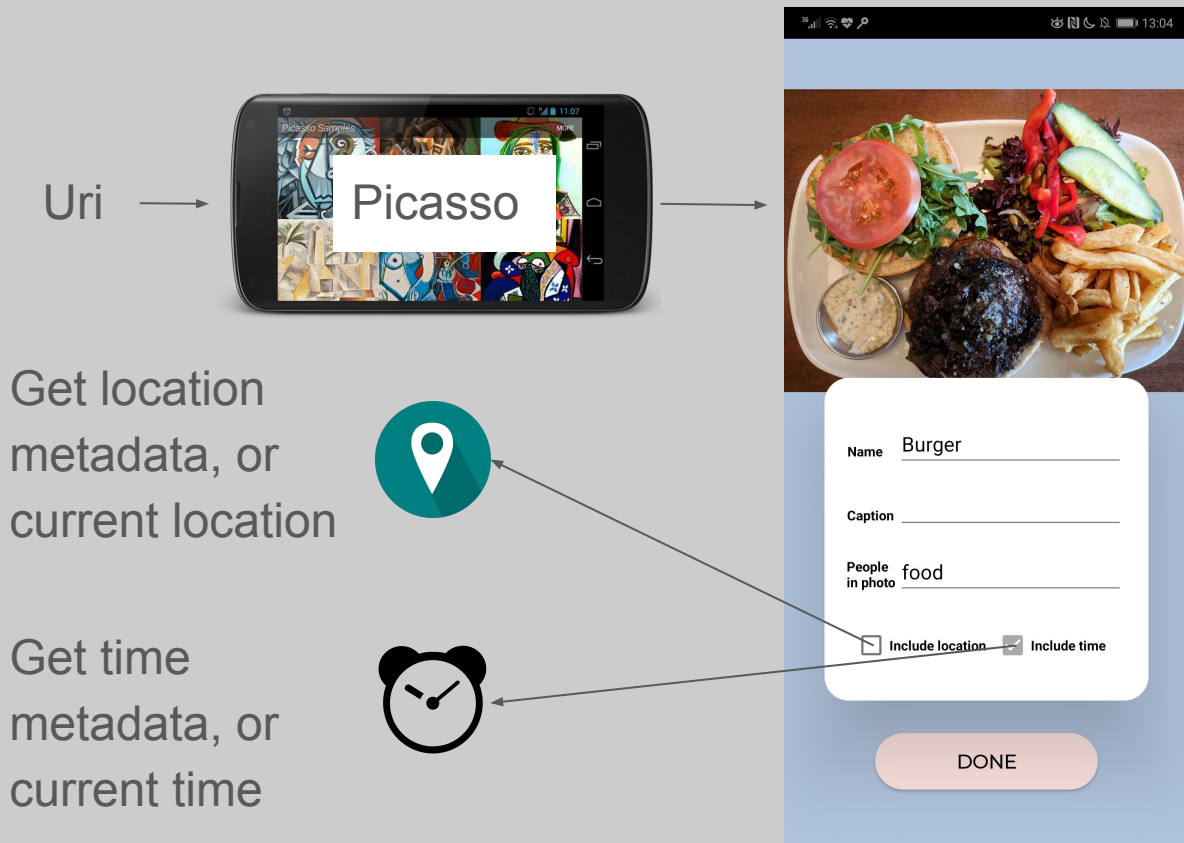


Photo Gallery

Detailed Design: Photo structure

The photo structure contains fields such as the photo itself, name, caption, people in photo, include location, and include time. Abstracted from the user is the implementation of these fields.



Detailed Design: Raspberry Pi

The photo frame part is controlled by Raspberry Pi connected to a LED screen. Consisting of 3 main running processes



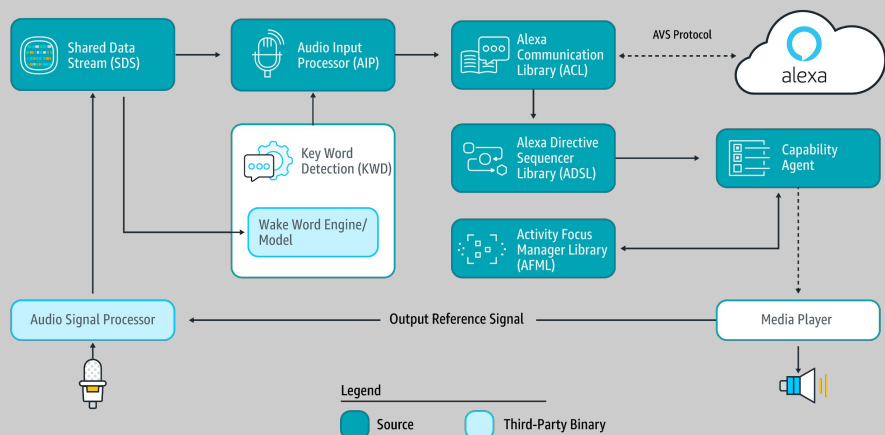
The Web front-end is where the photo is displayed. A simple web page opened via Chrome.

Alexa Voice Service listens for users commands and send a recorded voice sample to the Amazon's servers for parsing.

While the local web server serves as the communication link between the front-end and Amazon's servers.

Detailed Design: Voice Control

The Alexa Voice Service library is used to recognize spoken language through the attached microphone, when it recognized an activation word is spoken (in this case “Alexa”). It records the rest of the sentence that follows it and sends it to the cloud for processing.



Activation word

Invocation name: tells Alexa which device/set of skills user is referring to

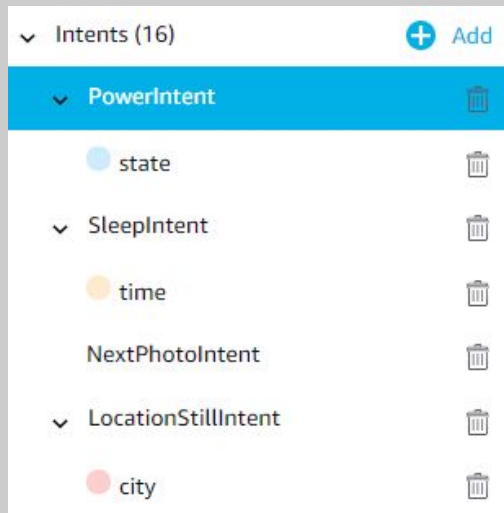
“Alexa, tell memory frame to sleep for 8 hours”

Intent: a given command. Eg. This is recognized as the sleep intent, and sends a sleep request to python server

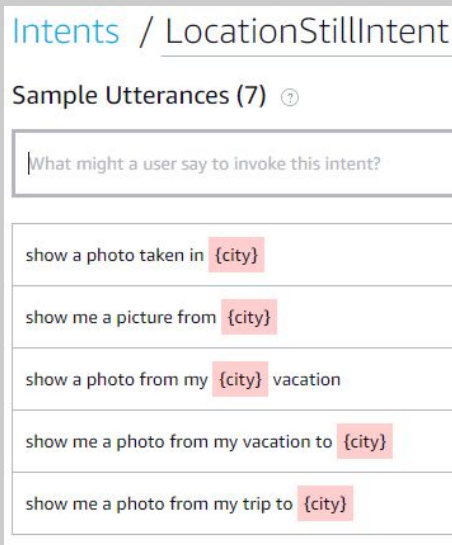
Slot: a given parameter. Eg. This is recognized as a time slot and passed as a function parameter to the python server

Detailed Design: Alexa

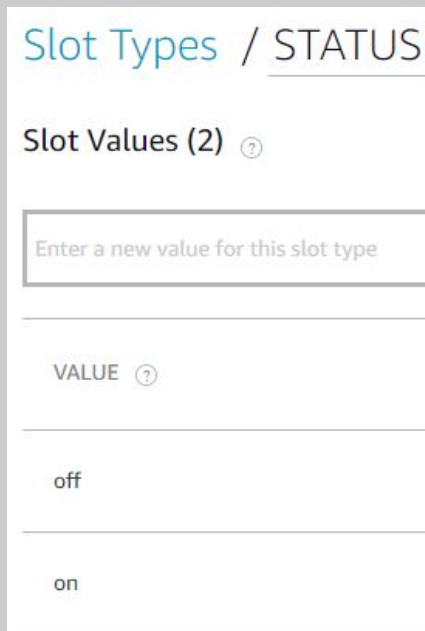
Alexa does speech to text transcription and has the structures in place to parse user commands. We used the developer console to build on these structures to create our own intents and train Alexa using machine learning to recognize them.



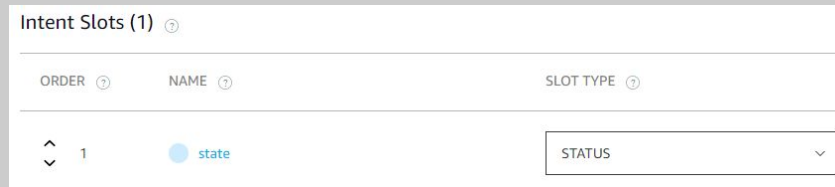
Some the of the commands we built and the slots they take



Adding sample user sayings so similar sentences also get mapped to the right intent



Creating a custom slot type



Mapping slots to their slot type

Detailed Design: Alexa

Intents:

- Power (on/off)
- Sleep for a set amount of time
- Switch to next photo
- Filter by location
- Filter by person
- Say hello
- Ask for help

Alexa Responses:

Controlled by developer console if a follow up question is required before a request is made to the python server like if a required slot is not filled:

Alexa speech prompts ?

|What will Alexa say to prompt the user to fill this slot?



I'm happy to help you learn more about your memory frame. Would you like to get a general introduction, a list of possible commands, an explanation of filtering or of sleep?



Otherwise responses are returned using the Flask-Ask framework, allowing the user to receive information like how many photos were found of a city when the location intent was called:

```
return statement('I\'ve found {} photos from {}'.format(len(images), city))
else:
    return statement('Sorry, no photos were found from {}'.format(city))
```

Detailed Design: Local Server

Whenever the Alexa server recognizes a valid command from the user, it generates an HTTPS webhook request to a specified domain name. This is to notify and specify to the user app which type of command was spoken, so the app can handle it appropriately.

This is why we run a Python server locally with the framework **Flask** to handle these requests. We chose this framework because of its extension, **Flask-Ask**, for working with Alexa.

The server waits for Alexa server's request and handles it.

For our case, when we get a command such as "Show me a photo with dogs," the server filters the database for photos of dogs and sends that information to the front-end so it can populate the frame with new found dog photos.



The communication between the the front-end and the server is really interesting because a standard webpage does not keep a TCP connection with the server after the HTML file is retrieved. We solved this by keeping a TCP connection open via Socket.io. Which was also supported by Flask with the extension **Flask-Socketio**

Detailed Design: Client

The photo is displayed through a web page on Chromium running on the Pi. We chose a web page over a custom GUI app because browsers are built in to most systems, and they are also cross-platform. Javascript is also very familiar and powerful, while HTML/CSS are simple, so it saved a lot of development time.

As mentioned in the previous page, the client communicates with the server through sockets. The client handles predefined events with Javascript. Here is a sample lists of events:

1. **Switch Photo** - Take in a new set of photos and add them to the front of the queue
2. **Next Photo** - Display the photo that's next in the queue
3. **Power** - Turns on/off the display
4. **Sleep** - Turns the display off a specified amount of time



The communication goes both ways. An example when client needs to communicate with the server is a **Update Photos** event, where the client requests the server for another set of photos, when the queue empties.

We also want to note that the client organizes the photos in a queue, where newly requested photos would be put right in the front. The queue structure is chosen because it's also used by music player, something familiar to most users.

Detailed Design: Firebase

Raspberry Pi

The Pi talks to the realtime database through Firebase REST API. We used a wrapper library, pyrebase, to query the database based on user inputs and fetch the URLs of the requested images.

Android




The Android app uses Firebase authorization, and Firebase commands to upload entries, check if the database has changed, and modify entries

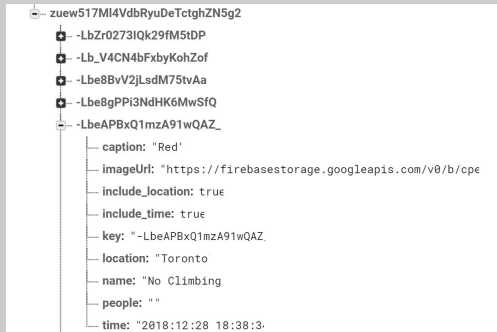




Detailed Design: Firebase


Three functions from Firebase used in this project are: user authentication, file storage and the real time database. When an authenticated user uploads an image from the app, the picture will be sent to Firebase storage and a corresponding database entry will record the URL to this picture.

Using an integrated platform like firebase instead of having separate authentication and data management system provides us a higher level of consistency and security. We can use the authentication information to organize the database and secure data access.

Identifier	Providers	Created	Signed In	User UID ↑
new@gmail.com		Apr 4, 2019	Apr 4, 2019	0GQYmehsQeeDZlci5fRo7ele0xC3
qwerty@gmail.com		Apr 4, 2019	Apr 4, 2019	0tICEOdgtXcITY3sSAz7PCN83O2
user1@gmail.com		Apr 4, 2019	Apr 4, 2019	2Vg1lmsv7TXgDPDQalM33jfqhFG2

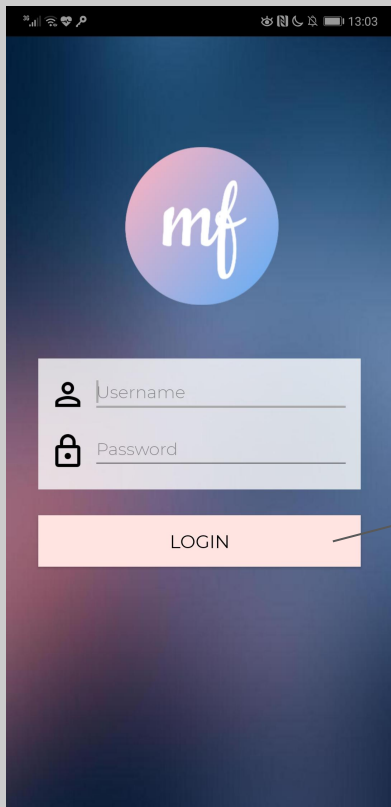


<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	 1554045911335.jpg	129 ...	image/jpeg	Mar 31, 2...
<input type="checkbox"/>	 1554322819729.gif	138 ...	image/gif	Apr 3, 2019
<input type="checkbox"/>	 1554333584048.jpg	2.46 ...	image/jpeg	Apr 3, 2019
<input type="checkbox"/>	 1554411472458.jpg	1.99 ...	image/jpeg	Apr 4, 2019
<input type="checkbox"/>	 1554411600418.jpg	5.25 ...	image/jpeg	Apr 4, 2019

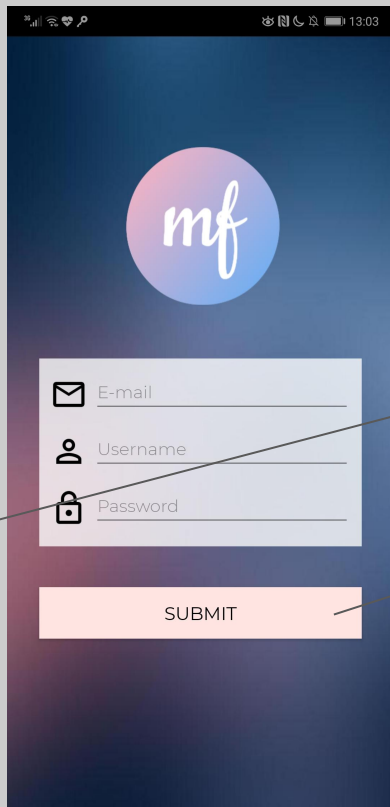
 1554412051654.jpg ×

Name
1554412051654.jpg [↗](#)
Size

```
"rules": {
  "$user_id": {
    ".write": "auth.uid == $user_id",
    ".read": "auth.uid == $user_id",
    ".indexOn": ["imageUrl", "name", "people", "caption", "time", "location"]
  }
}
```

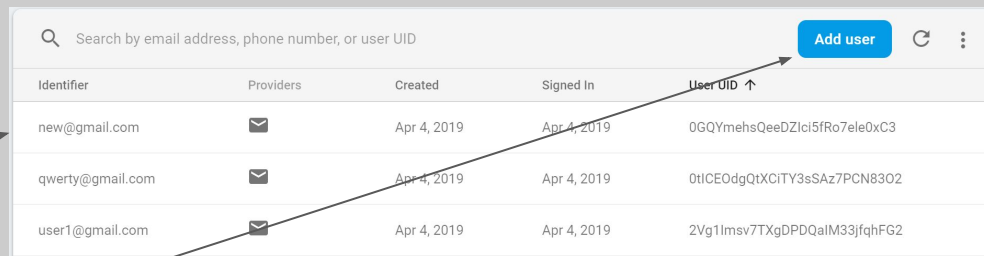

Detailed Design: Firebase Authorization



Mobile app login screen. At the top is a circular logo with the letters 'mf' in white on a blue-to-purple gradient background. Below the logo are two input fields: the first is labeled 'Username' with a person icon, and the second is labeled 'Password' with a lock icon. At the bottom is a pink button labeled 'LOGIN'.



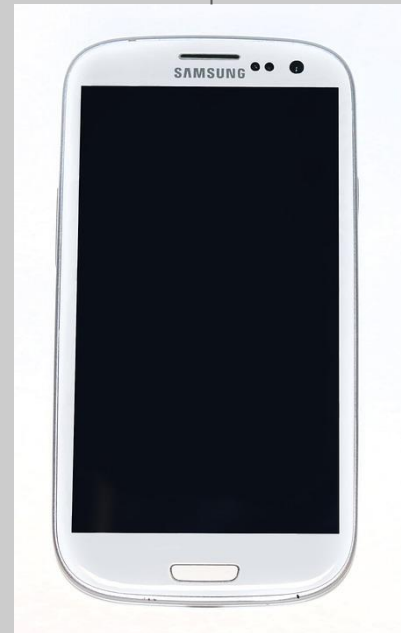
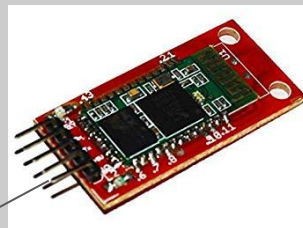
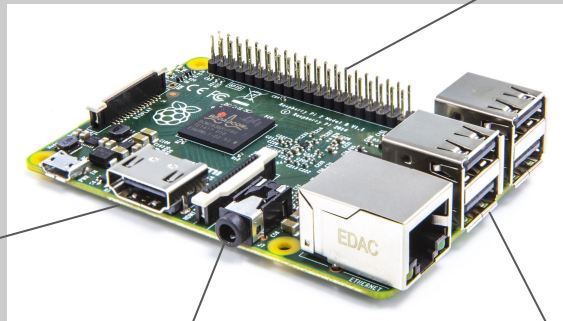
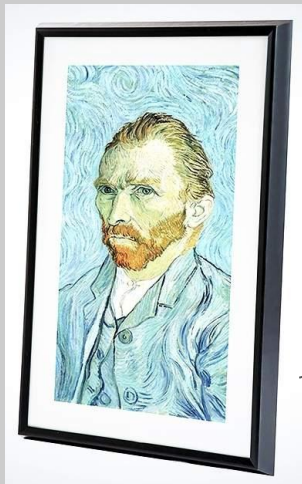
Mobile app registration screen. At the top is a circular logo with the letters 'mf' in white on a blue-to-purple gradient background. Below the logo are three input fields: the first is labeled 'E-mail' with an envelope icon, the second is labeled 'Username' with a person icon, and the third is labeled 'Password' with a lock icon. At the bottom is a pink button labeled 'SUBMIT'.



Firebase user management interface. At the top is a search bar with the text 'Search by email address, phone number, or user UID'. To the right of the search bar is a blue button labeled 'Add user' and a refresh icon. Below the search bar is a table with the following columns: Identifier, Providers, Created, Signed In, and User UID ↑. The table contains three rows of user data.

Identifier	Providers	Created	Signed In	User UID ↑
new@gmail.com	✉	Apr 4, 2019	Apr 4, 2019	0GQYmehsQeeDZlci5fRo7ele0xC3
qwerty@gmail.com	✉	Apr 4, 2019	Apr 4, 2019	0tlICE0dgQtXCITY3sSAz7PCN83O2
user1@gmail.com	✉	Apr 4, 2019	Apr 4, 2019	2Vg1lmsv7TXgDPDQalM33jfqhFG2

Detailed Design: Hardware



Result

Our project has met all of our goals and requirements. We ensured robustness by testing our code on a variety of different devices.

Create an account ✓


A mobile app interface for creating an account. It features a circular logo with the letters 'mf' in a pink and blue gradient. Below the logo are three input fields: 'E-mail', 'Username', and 'Password', each with a corresponding icon (envelope, person, and lock). At the bottom is a pink 'SUBMIT' button.

Link to different accounts ✓

Identifier	Providers	Created	Signed In	User UID ↑
new@gmail.com	✉	Apr 4, 2019	Apr 4, 2019	0GQYrmehtQeeDZic5fRo7ele0xC3
qwerty@gmail.com	✉	Apr 4, 2019	Apr 4, 2019	0nICE0dgQixCITY3sSAz7PCN8302
user1@gmail.com	✉	Apr 4, 2019	Apr 4, 2019	2Vg1lmsv7TXgDP0QalM33jfhF02

Upload a photo ✓

<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	1554040891335.jpg	128 ...	image/jpeg	Mar 31, 2...
<input type="checkbox"/>	1554322819729.gif	198 ...	image/gif	Apr 3, 2019
<input type="checkbox"/>	1554330384048.jpg	2.4K ...	image/jpeg	Apr 3, 2019
<input type="checkbox"/>	1554411472438.jpg	1.9K ...	image/jpeg	Apr 4, 2019
<input type="checkbox"/>	1554411605418.jpg	5.2K ...	image/jpeg	Apr 4, 2019



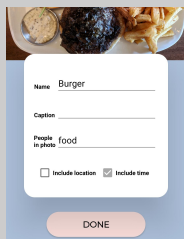
1554412051654.jpg

Name: 1554412051654.jpg

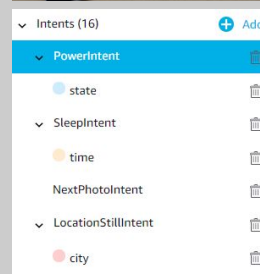
Display photos ✓



Add a description ✓

A mobile app interface for adding a description. It shows a photo of food at the top. Below the photo is a form with a 'Name' field containing 'Burger', a 'Caption' field, and a 'People in photo' field containing 'food'. There are checkboxes for 'Include location' and 'Include time'. At the bottom is a pink 'DONE' button.

Voice command ✓



Conclusion and Suggested Future Work

The overall project was a great success and we accomplished everything we set out to do. For a continuation of the project, feedback could be sent from the photo frame user to be displayed on the Android app. Facial recognition could also be used to automatically determine the people in a picture.

