MID SEM REPORT

Arpit Kansal, Madhav Maheswari, Nikita Singh

October 2024

1 Abstract

Our major project builds upon the smart mirror developed during our minor project, which utilized facial recognition technology to display personalized information securely. The minor project successfully created a smart mirror with a smart display screen and a webcam, recognizing faces and providing real-time information. In this major project, we have expanded the functionality by adding a mood detection system, improving the facial recognition accuracy, and developing a custom display interface. This project aims to create a more intelligent and responsive smart mirror that not only identifies users but also adapts to their emotional state, enhancing the overall user experience.

2 Introduction

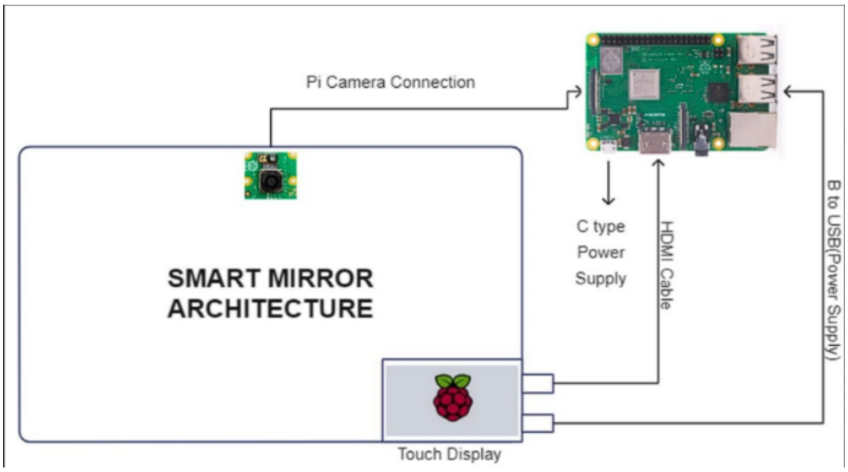


Figure 1: Project Layout

In the minor project, we developed a smart mirror as a tool for personalized security, featuring facial recognition to authenticate users and protect their information. This foundation has been expanded in our major project, where we introduce mood detection to further personalize user interactions. Additionally, the custom-built display interface replaces the Raspberry Pi screen used in the minor project, offering a more integrated and user-friendly experience. These advancements take the smart mirror from a security-focused device to a more interactive and emotionally intelligent home assistant.

3 Problem Statement

The minor project addressed the need for personalized information and automation in smart mirrors. However, this major project tackles new challenges:

1. Emotional Intelligence: Current smart mirrors, including our initial version, do not account for the user's emotional state, limiting personalized content delivery.

2. Standard Interface Limitations: The standard Raspberry Pi screen used in the minor project restricts customization and user experience. This project aims to overcome these challenges by integrating mood detection and creating a custom display interface.

4 Literature Review

The minor project involved a literature review covering IoT, facial recognition, and the use of Raspberry Pi in smart devices. Building on that:

1. Mood Detection in Smart Devices: We explore the state of mood detection technology and its applications, identifying how our project enhances current solutions.

2. Advanced Facial Recognition Techniques: We review more sophisticated facial recognition methods, comparing them to those used in the minor project.

3. Custom Interface Development: We examine studies on custom interface design in IoT devices, highlighting the benefits over the Raspberry Pi screen used in the minor project.

5 Objectives

The objectives of the minor project, which were successfully achieved, focused on facial recognition and personalized information display. For the major project, we have set new goals:

1. Mood Detection and Analysis: To develop a system that detects and responds to users' moods, enhancing the smart mirror's interactivity.

2. Enhanced Facial Recognition: To improve the recognition system's speed and accuracy, building on the foundation laid in the minor project.

3. Custom Display Interface: To design and implement a custom interface that offers a better user experience than the Raspberry Pi screen used previously.

6 Methodology

In the minor project, we gathered necessary hardware and software, trained a facial recognition model, and built the smart mirror system. For the major project:

1. Mood Detection Implementation: We expanded the system by collecting data on facial expressions, training a mood detection model, and integrating it with the existing recognition system.

2. Enhanced Facial Recognition Techniques: We improved the recognition model using a larger dataset and more advanced algorithms, enhancing the work done in the minor project.

3. Custom Interface Development: We developed a new, custom display interface, ensuring a more seamless and visually appealing interaction than the standard Raspberry Pi screen used in the minor project.

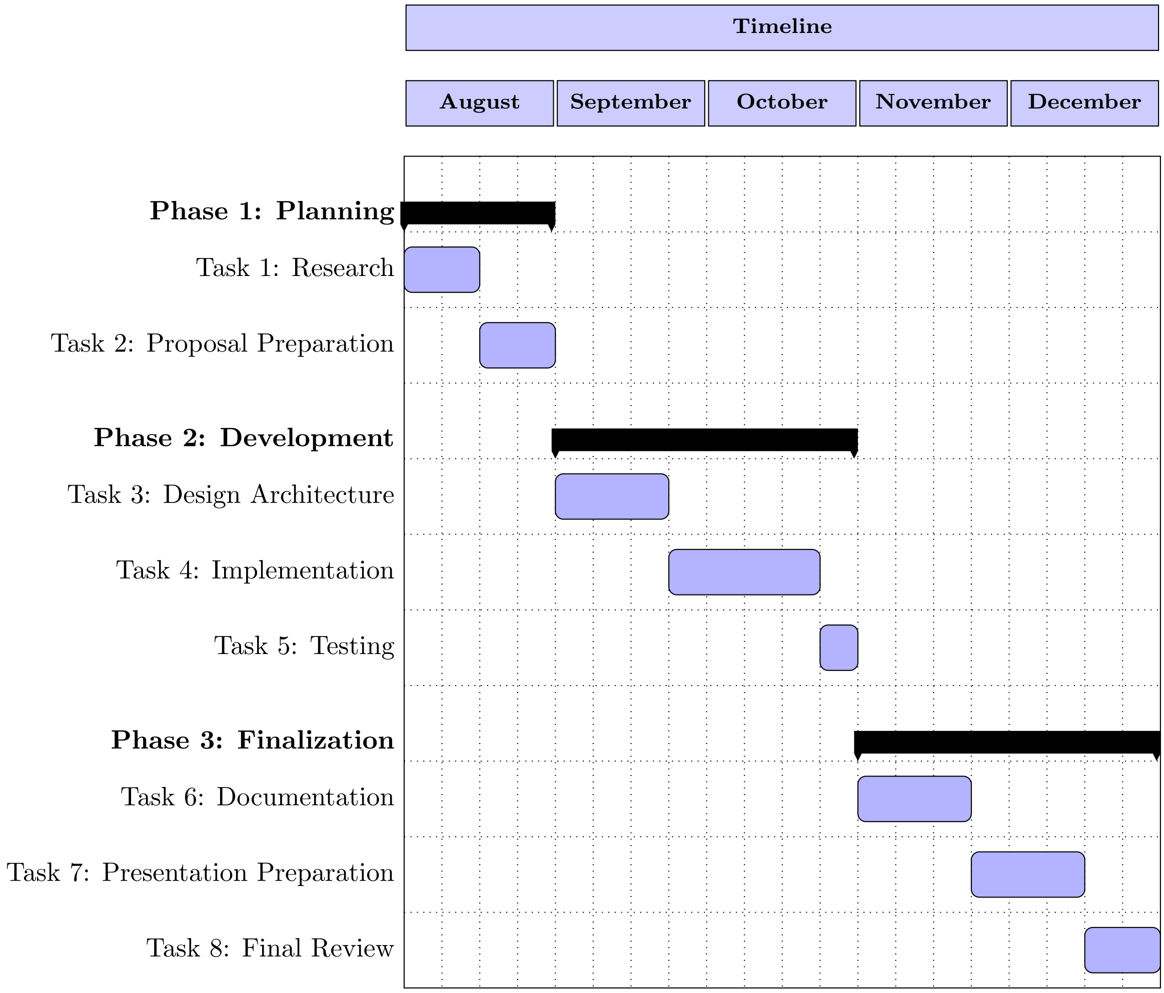
7 System Requirements

The minor project's system requirements included basic hardware like Raspberry Pi and software libraries like OpenCV. For the major project:

1. Mood Detection System: Additional processing power and AI libraries may be required for implementing mood detection.

2. Custom Interface: Specific hardware and software are needed to develop and run the new display interface, ensuring it meets the demands of the enhanced system.

8 Project Timeline



9 SWOT Analysis

1. Strengths: Innovative technology, convenience, enhanced security, and significant market potential. The new custom interface and improved recognition accuracy enhance the system's competitiveness.

2. Weaknesses: Dependence on technology, cost, privacy concerns, and limited functionality. Increasing the system's complexity may also add to development and operational costs.

3. Opportunities: Expansion into new markets, partnerships, technological advancements, and customization options. Enhanced features may open new opportunities for commercial applications.

4. Threats: Regulatory challenges, competition, cybersecurity risks, and economic factors. Competition in the smart security market remains a threat, as does evolving privacy legislation.

10 Code

In [21]:
```
model_json = model.to_json()
with open("emotiondetector.json",'w') as json_file:
    json_file.write(model_json)
model.save("emotiondetector.h5")
```

In [22]:
```
from keras.models import model_from_json
```

In [23]:
```
json_file = open("facialemotionmodel.json", "r")
model_json = json_file.read()
json_file.close()
model = model_from_json(model_json)
model.load_weights("facialemotionmodel.h5")
```

In [24]:
```
label = ['angry','disgust','fear','happy','neutral','sad','surprise']
```

In [33]:
```
def ef(image):
    img = load_img(image,grayscale =  True )
    feature = np.array(img)
    feature = feature.reshape(1,48,48,1)
    return feature/255.0
```

In [37]:
```
image = 'images/train/sad/42.jpg'
print("original image is of sad")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
```

```
original image is of sad
```

In [17]:
```
y_train = to_categorical(y_train,num_classes = 7)
y_test = to_categorical(y_test,num_classes = 7)
```

In [18]:
```
model = Sequential()
# convolutional layers
model.add(Conv2D(128, kernel_size=(3,3), activation='relu', input_shape=(48,48,1)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(256, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(512, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(512, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Flatten())
# fully connected layers
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))
# output layer
model.add(Dense(7, activation='softmax'))
```

In [19]:
```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = 'accuracy' )
```

In [20]:
```
model.fit(x= x_train,y = y_train, batch_size = 128, epochs = 100, validation_data = (x_test,y_test))
```

In [5]:
```
print(train)
```

```
                                image     label
0            images/train\angry\0.jpg     angry
1            images/train\angry\1.jpg     angry
2           images/train\angry\10.jpg     angry
3        images/train\angry\10002.jpg     angry
4        images/train\angry\10016.jpg     angry
...                               ...       ...
28816  images/train\surprise\9969.jpg  surprise
28817  images/train\surprise\9985.jpg  surprise
28818  images/train\surprise\9990.jpg  surprise
28819  images/train\surprise\9992.jpg  surprise
28820  images/train\surprise\9996.jpg  surprise

[28821 rows x 2 columns]
```

In [6]:
```
test = pd.DataFrame()
test['image'], test['label'] = createdataframe(TEST_DIR)
```

```
angry completed
disgust completed
fear completed
happy completed
neutral completed
sad completed
surprise completed
```

In [7]:
```
print(test)
print(test['image'])
```

```
                                image     label
0       images/test\angry\10052.jpg     angry
1       images/test\angry\10065.jpg     angry
2       images/test\angry\10079.jpg     angry
3       images/test\angry\10095.jpg     angry
4       images/test\angry\10121.jpg     angry
...                               ...       ...
7061  images/test\surprise\9806.jpg  surprise
7062  images/test\surprise\9830.jpg  surprise
7063  images/test\surprise\9853.jpg  surprise
7064  images/test\surprise\9878.jpg  surprise
```

In [1]:
```
from keras.utils import to_categorical
from keras_preprocessing.image import load_img
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
import os
import pandas as pd
import numpy as np
```

In [2]:
```
TRAIN_DIR = 'images/train'
TEST_DIR = 'images/test'
```

In [3]:
```
def createdataframe(dir):
    image_paths = []
    labels = []
    for label in os.listdir(dir):
        for imagename in os.listdir(os.path.join(dir,label)):
            image_paths.append(os.path.join(dir,label,imagename))
            labels.append(label)
        print(label, "completed")
    return image_paths,labels
```

In [4]:
```
train = pd.DataFrame()
train['image'], train['label'] = createdataframe(TRAIN_DIR)
```

```
angry completed
disgust completed
fear completed
happy completed
neutral completed
sad completed
surprise completed
```