PicPicker

CP470 - Group 4 - Fall 2024

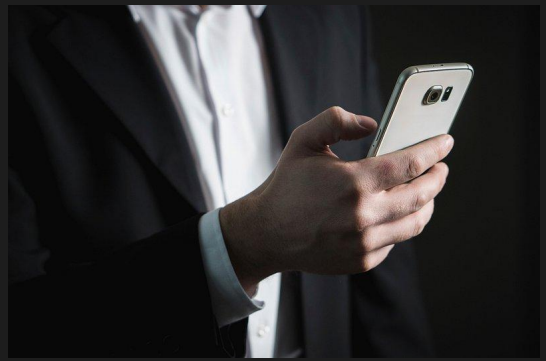
Sahil Khasnobish, Phoebe Schulman, Lewis White,
Zaid Saiyed, Muhammad Saad

Problem Statement:

Given several photos on your phone, how do you quickly determine which photos have the best quality?

Our goal:

Use machine learning to determine the “best” photos.



- Difficult to sort through 100 photos
- Manually checking quality consumes time

Project Deliverables

Primary goal

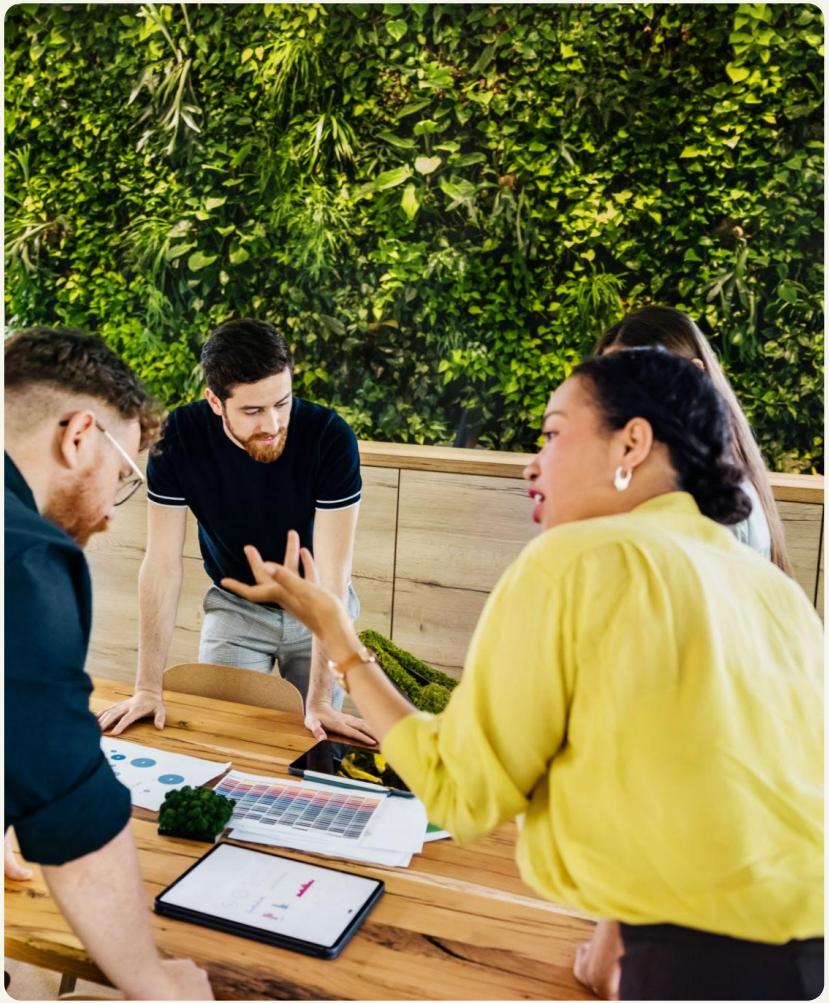
- Quickly analyze a large collection of photos
- Efficiently determine the best photos that were provided

What determines a good photo?

- Blur detection: Determine high quality images
- Eye detection: Avoid images where a subject blinks

Other deliverables:

- Log-in system
- Easy to use interface
- Best photos are saved so the user can come back to their favourite photos at any time



Target Customers

Stakeholders who must manually sort through multiple photos

- Photographers
- Travellers
- Social media influencers & content creators
- Marketing & Advertising teams
- UX/UI designers

John Doe Traveller

Has 200 pictures, across multiple countries, to share with family.

Jane Doe Influencer

Has 3000 selfies + pet pictures to share with fans.

Project Workflow & Requirements

Login Page

Database

Username/password formatting

User authentication

DialogBox for user's consent to access photos

Button, EditText, Toast

Main Activity

Main Hub

Access to all other activities

Select Photos

Access user's photos

Add & remove photos

Threads & progress bar

Display Photos

Machine Learning

Image processing

RecyclerView of photos

Fragment in GUI

Saving photos to app (with a database)

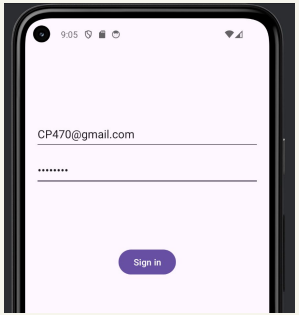
Saving photos to device

Running the App

Login page ↔ Main page

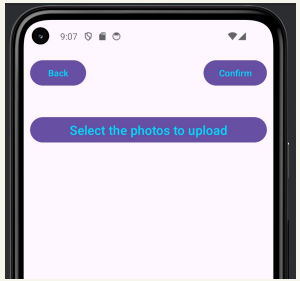
Main page → Select Photos → Display Results

Display Results ↔ Main page ↔ Login page



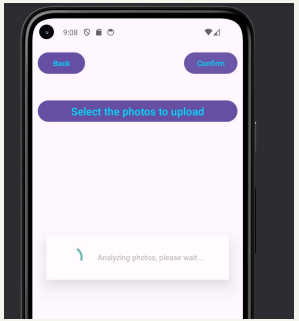
Login page

Main page



Select Photos page

Select Photos feature



Display Photos loading

Display Photos result

High-Quality Design

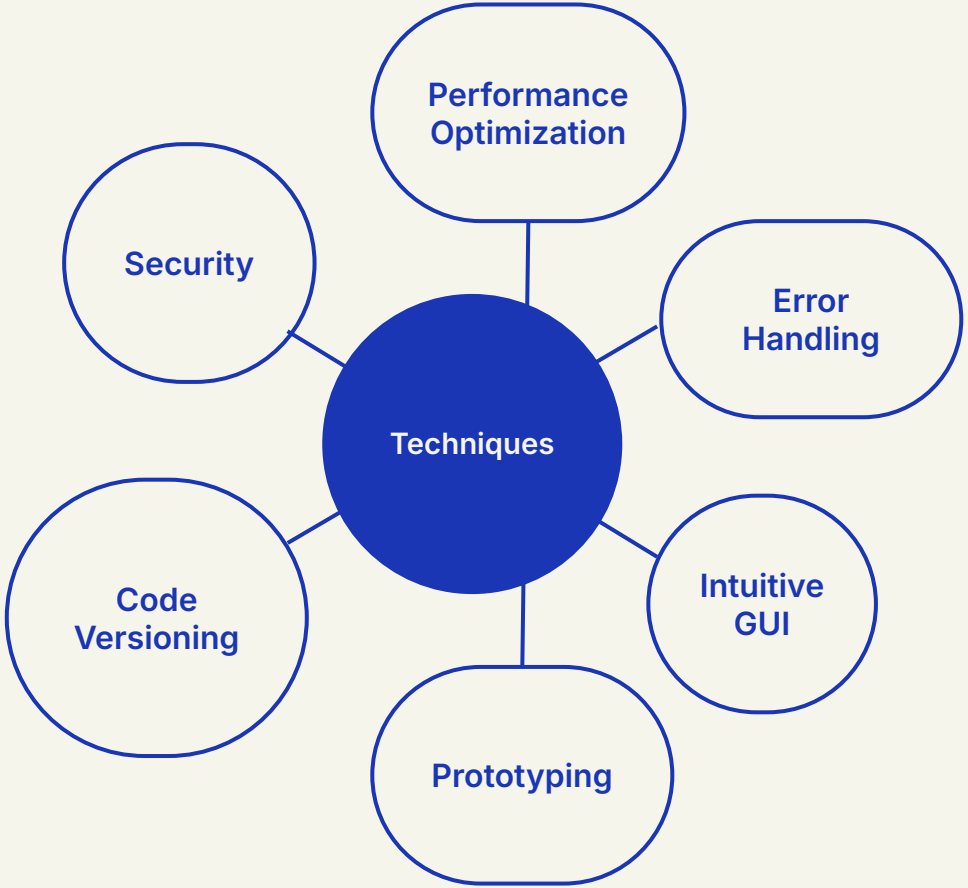
Security: user's authentication, permissions, and consent

Ensure optimal performance: threads, limiting max photos

Thorough testing and validation: error handling, Try-Catch blocks, Logging, Toasts to notify users

Designing clean code: easy to use GUI, Model View Controller architecture

Prototyping and code versioning with GitHub



Technical Challenges and Solutions

Challenge

Storing user provided images: internally and externally

Solution

Save photos to a database.

Challenge

Incorporating machine learning algorithms: for image recognition

Solution

Google ML Kit for on device machine learning.

Challenge

Performance concerns: for high quality images

Solution

Gray scaling to remove colours.

Challenge

User Interface

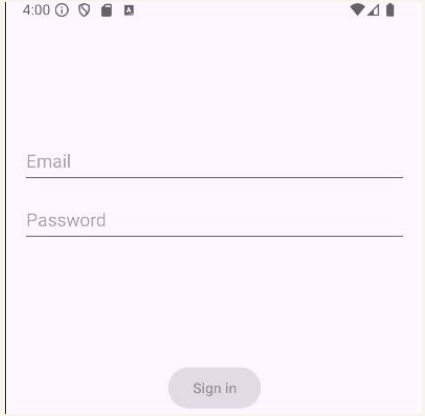
Solution

Consistent UI elements such as buttons, fonts, colours. Place buttons so they are easy to access.

Live Demo

Main Functions

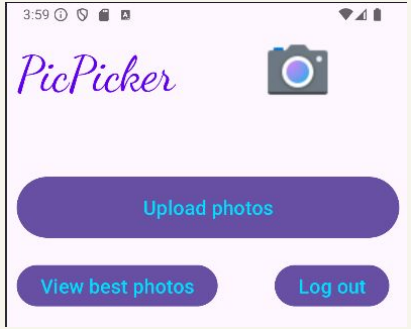
Login Page



Email & Password Fields:
-Validates syntax

Sign in:
-Validates login details
-Requests photo access
-Calls MainActivity

Main Page

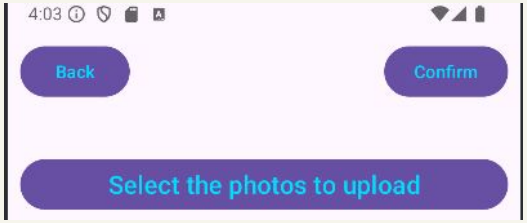


Upload Photos:
-Calls clickUploadPhotos() to start SelectPhotosActivity

View Best Photos:
-Calls clickDisplayPhotos() to start DisplayResultsActivity

Log Out:
-Calls clickLogout() to return to login page

Select Photos Activity



Select the photos to upload:
-Calls selectPhotos() to get photos from user's camera roll

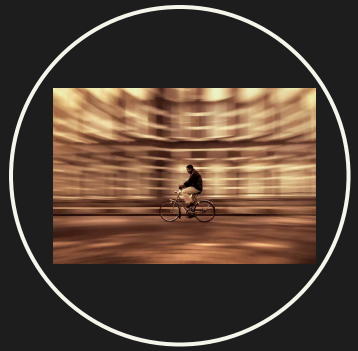
Back:
-Calls MainActivity to return to main page

Confirm:
-When photos are selected, calls DisplayResultsActivity()

Machine Learning

Gray Scaling

Remove colors: less time and resources



Blur Detection

Remove photos that are too blurry

Eye Detection

Determine if people are blinking with Machine Learning

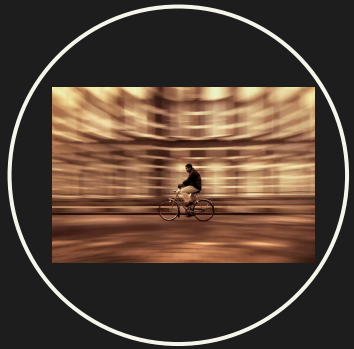
Gray Scaling in Image Processing

- Reduces computational load and speeds up processing
- Essential for accurate blur detection as it simplifies the image data
- Helps in focusing on textural and shape attributes important for further analysis

Gray Scaling

Blur Detection Using the Laplacian Method

- Pixel Intensity Variance: We use the variance in pixel intensity as a measure to detect sharpness. Low variance indicates that the image lacks clear, defined edges, which are essential for sharpness
- Edge Detection Sensitivity: Highlights edges by measuring rapid intensity changes, effective for identifying blur levels
- Automated Quality Checks: Automates sharpness assessments to exclude blurry images efficiently, enhancing photo selection processes



Blur Detection

Eye Detection using Google ML Kit

- Real-time Eye Status Recognition: Utilizes ML Kit to detect whether eyes are open or closed, ensuring optimal photo moments
- Photo Authenticity: Enhances photo authenticity by capturing natural expressions with eyes open
- Efficiency in Photo Selection: Reduces the need for retakes by filtering out photos with closed eyes, improving user satisfaction

Eye Detection

Conclusion

Privacy and Security:

- 100% offline functionality.
- No data sent to servers or stored externally.
- Complete user data privacy.

Conclusion:

- Ideal for privacy-conscious users seeking quality and convenience.
- Empowers users to capture and keep only the best moments without privacy concerns.

Thank You!

Questions?