Flashlight

A PROJECT REPORT

Submitted by

Bhanu Sharma(22SCSE2030723)

in partial fulfillment for the award of the degree of

MASTER OF COMPUTER APPLICATION

BONAFIDE CERTIFICATE

Certified that this project report **flashlight** is the bonafide work of **Bhanu sharma**who carried out the project work under my/our supervision.

Dr. Farhan Sufyan
Assitant Professor

Submitted for the project viva-voce examination held on ____________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Acknowledgement

In successfully completing this project, many people have helped us. We would like to thank all those who are related to this project.

Primarily, we would thank my Supervisor **Dr. Farhan Sufyan** under whose guidance we learned a lot about this project. Their suggestions and directions have helped in the completion of this project.

TABLE OF CONTENTS

ABSTRACT

The Android Image Viewer project endeavors to create a user-friendly and versatile image viewing application for Android devices. By harnessing the power of Java programming language and the Android SDK, the application aims to enrich the user experience by providing essential features for efficient image management. Through intuitive navigation, zooming capabilities, and compatibility with various image formats, the Android Image Viewer seeks to offer a seamless platform for users to explore and interact with their diverse image collections.

As a testament to the application's commitment to user satisfaction, the Android Image Viewer employs a comprehensive testing strategy, encompassing both unit testing for individual components and functional testing for the entire application. This rigorous testing, coupled with an agile development methodology, ensures a reliable and compatible application across diverse Android devices and versions.

The Android Image Viewer project not only addresses the current needs of users for a feature-rich and intuitive image viewer but also anticipates future enhancements. The inclusion of a dynamic slideshow mode, efficient in-memory data management, and a user-friendly interface positions this application as a versatile tool for anyone seeking a captivating and interactive means of exploring their visual memories.

CHAPTER 1: INTRODUCTION

In the contemporary era of pervasive digital imagery, the ubiquity of smartphones has transformed these devices into repositories for a myriad of personal memories and visual experiences. Recognizing the need for a more sophisticated and user-centric solution to navigate this visual tapestry, the Android Image Viewer project emerges as a response to elevate the image viewing experience on Android devices.

This project introduces a dynamic and versatile Android application designed to transcend the limitations of conventional gallery functionalities. By leveraging the capabilities of Java programming language and the Android SDK, the Image Viewer strives to offer users an immersive and intuitive platform for managing their diverse image collections. In a landscape inundated with images captured on the go, the application seeks to enhance the exploration, organization, and enjoyment of these visual narratives.

The Android Image Viewer aspires to be more than a utilitarian tool, aiming to provide a visually engaging and responsive interface. Through thoughtful design considerations, intuitive gesture controls, and an emphasis on compatibility with various image formats, the application is poised to redefine the way users engage with their photos. Whether exploring a portfolio of professional photography or simply revisiting cherished moments, the Android Image Viewer is crafted to seamlessly integrate into the daily lives of users, fostering a more enriched and enjoyable image viewing experience.

1. TASK MANAGEMENT

From an articulation standpoint, task management entails more than just organizing virtual and physical collections and scheduling activities.

Recent research has begun to address the problem of generic task management in the context of email. This development is hardly surprising, given that many digital device users are overloaded by the number of chores done through email. According to this research, any successful productivity tool must be tightly connected with email functionalities. Recent researches looked at task management strategies more generally because email and related technologies are unlikely to be the whole picture.

2. Image Viewer as a solution to task management

The concept of to-do list has existed for a very long time and it is one of the primary methods for management of tasks, use of a to-dos as a reminder system, to-dos as a system for note management, etc. In the simplest and most primitive form, a to-do list can be implemented on a pen and paper as a checklist of items which can be crossed of or ticked against when completed.

This can be further extended to calendars, by writing tasks against dates where the dates can also act as deadlines for particular tasks. Other possible extensions of to-do lists can be on whiteboards, journals, text editors,etc.

The functionalities of to-do lists naturally evolve to perfectly fit android applications and applications on mobile devices. Equipped with modern tools and technologies, engineers can build an application to create a minimal and powerful application that can help boost productivity without loss of focus and attention.

With the computing power and persistence of modern devices and databases, losing track of tasks will not be a problem people will have to face anymore and they can rest assured, only focused on the tasks they will have to accomplish as with modern technology and the power of digital devices, integration will be seamless and tasks can be synced across multiple devices all at once, without any hassle.

Essential Functionalities:

- A variety of methods for examining and managing to-dos that go beyond lists and mirror the advantages of current resources.
- The inconvenient property, such as when it becomes the default spot for everyday tasks where reminders can be satisfied.
- Immediately turns on, allowing for quick input and clear view. Conventional to-do lists are commonly abandoned due to slow, time-consuming input and weak output.
- No formal job description, classification, or decomposition is required from users, and any degree of abstraction for atomic task entries must be permitted.
- A mechanism for dealing with stale, low-priority to-dos that are becoming less likely to be performed but have not been explicitly deleted.

CHAPTER 2 :Methodology

The development methodology for the Android Image Viewer app embraces a user-centric and iterative approach. Regular feedback sessions and usability testing have been integral to refining the user interface and ensuring that the application aligns with user expectations. Java, renowned for its cross-platform compatibility and object-oriented principles, was the natural choice for development. The Android Studio IDE facilitates seamless integration with the Android ecosystem, streamlining the development process.

The methodology employed in the development of the Android Image Viewer project is rooted in a user-centric and iterative approach, ensuring a seamless and responsive application. The following key aspects characterize the methodology:

The methodology employed in the development of the Android Image Viewer project is rooted in a user-centric and iterative approach, ensuring a seamless and responsive application. The following key aspects characterize the methodology:

2.1. **User-Centric Design:**

- User Feedback: Regular feedback sessions were conducted to gather insights from potential users, allowing for the integration of features aligned with user expectations.

- Usability Testing:Iterative usability testing played a pivotal role in refining the user interface and interaction patterns, ensuring an intuitive experience for a diverse user base.

2.2. **Iterative Development:**

- Agile Methodology: Embracing an agile development methodology facilitated continuous refinement of features based on evolving requirements and feedback.

- Incremental Releases: The project progressed through incremental releases, allowing for the timely incorporation of user feedback and feature enhancements.

2.3. **Java Programming Language:**

- Compatibility: Java was chosen for its cross-platform compatibility, ensuring the Android Image Viewer's accessibility on a broad range of Android devices.

- Object-Oriented Approach:Leveraging the object-oriented features of Java facilitated the development of a scalable and maintainable codebase.

2.4.**Android Studio Development Environment:**

- Integrated Development Environment (IDE): Android Studio served as the primary development environment, offering a suite of tools that streamlined the development and debugging process.

- SDK Utilization: Utilizing the Android Software Development Kit (SDK) provided a standardized framework for building robust Android applications.

2.5. **Model-View-Controller (MVC) Architecture:**

- Separation of Concerns:Adopting the MVC architecture facilitated a clear separation of responsibilities, enhancing modularity and maintainability.

- Scalability: The modular structure allowed for the seamless integration of new features and functionalities without compromising the existing codebase.

2.6. **Comprehensive Testing Strategy:**

- Unit Testing: Individual components underwent rigorous unit testing to ensure their functionality in isolation.

- Functional Testing: The entire application was subjected to comprehensive functional testing, utilizing both emulators and physical Android devices to ensure compatibility across diverse environments.

2.7. Responsive User Interface Design:

- Gestures and Controls: Gesture controls, including pinch-to-zoom and swipe-to-pan, were seamlessly integrated to enhance the interactive nature of the user interface.

- Visual Appeal: A focus on aesthetically pleasing design elements contributed to an engaging visual experience for users exploring their image collections.

By combining user-centric design principles, agile development methodologies, and the utilization of robust technologies, the Android Image Viewer project methodology prioritized the creation of a responsive, intuitive, and versatile image viewing application for Android devices.ChatGPT

CHAPTER 3: System Design

The system design of the Android Image Viewer project encompasses the architectural decisions, user interface design, and data management strategies to ensure the efficient and effective functioning of the application.

3.1. Architecture

The app adopts the Model-View-Controller (MVC) architecture to maintain a clear separation of concerns. This architectural choice enhances modularity and facilitates future updates and improvements. The Model is responsible for managing image data, the View handles the user interface, and the Controller governs the flow of data between the Model and the View.

3.2. User Interface Design

User interface design is a pivotal aspect of the Android Image Viewer project. The application features an aesthetically pleasing and intuitive design, ensuring that users can effortlessly navigate through their image libraries. Gestures, such as pinch-to-zoom and swipe-to-pan, are seamlessly integrated, enhancing the interactive experience. The user interface also includes convenient features like image metadata display and quick access to image details.

The Benefits of Using a To Do List

The Android Image Viewer project offers a multitude of benefits to users, enhancing their overall experience with image management on Android devices:

1. Versatility and Compatibility:

- Supports a wide range of image formats, ensuring compatibility with diverse photo collections.
- Provides a versatile platform for users with varying needs, from casual browsing to professional image review.

2. User-Friendly Interface:

- A well-designed and intuitive user interface facilitates effortless navigation through extensive image libraries.
- Incorporates gesture controls, such as pinch-to-zoom and swipe-to-pan, enhancing the user experience and making interactions more natural.

3. Efficient Image Management:

- Enables efficient organization and management of images without relying on an external database, ensuring quick and responsive interactions.
- Intuitive controls for zooming, panning, and slideshow mode enhance the overall image browsing experience.

4. Enhanced Visual Experience:

- Aesthetically pleasing design and layout contribute to a visually engaging experience for users exploring their image collections.
- Quick access to image details and metadata provides a comprehensive understanding of each photo.

5. Future Enhancement Opportunities:

- Lays the foundation for potential future enhancements, such as integration with cloud.

Source Code :

MainActivity.java

```
package com.anni.imageviewer;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.ParcelFileDescriptor;

import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import com.anni.imageviewer.R;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;

import java.io.FileDescriptor;
import java.io.IOException;
```

```
public class MainActivity extends AppCompatActivity {


    private static final int SELECT_PICTURE = 1;

    private ImageView imageView;


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        Toolbar toolbar =  findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);


        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);

        fab.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View view) {

                Intent i = new Intent(

                        Intent.ACTION_PICK,

android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);


                startActivityForResult(i, SELECT_PICTURE);

            }

        });
        imageView = (ImageView) findViewById(R.id.imageView);
    }


    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == SELECT_PICTURE && resultCode == RESULT_OK &&
null != data) {
            Uri selectedImage = data.getData();
            try {
                Bitmap bmp = getBitmapFromUri(selectedImage);
                imageView.setImageBitmap(bmp);
            } catch (IOException e) {
                e.printStackTrace();
                Snackbar.make(imageView, "Error getting image",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        }
    }


    private Bitmap getBitmapFromUri(Uri uri) throws IOException {
        ParcelFileDescriptor parcelFileDescriptor =
                getContentResolver().openFileDescriptor(uri, "r");
        FileDescriptor fileDescriptor = parcelFileDescriptor.getFileDescriptor();
        Bitmap image = BitmapFactory.decodeFileDescriptor(fileDescriptor);
        parcelFileDescriptor.close();
        return image;
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
//        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

//        if (id == R.id.action_settings) {
//            return true;
//        }

        return super.onOptionsItemSelected(item);
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

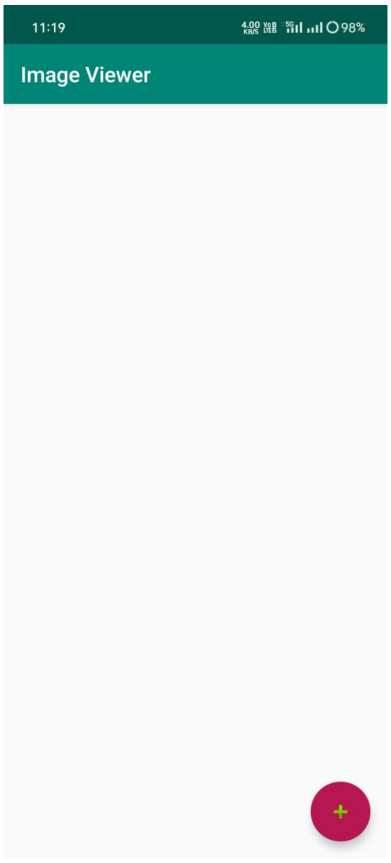
```
android:layout_width="match_parent"
    android:layout_height="match_parent" android:fitsSystemWindows="true"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.AppBarLayout
android:layout_height="wrap_content"
        android:layout_width="match_parent" android:theme="@style/
AppTheme.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar android:id="@+id/toolbar"
            android:layout_width="match_parent" android:layout_height="?attr/
actionBarSize"
            android:background="?attr/colorPrimary" app:popupTheme="@style/
AppTheme.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_main" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
android:id="@+id/fab"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_gravity="bottom|end" android:layout_margin="@dimen/
fab_margin"
        android:src="@android:drawable/ic_input_add" />
```

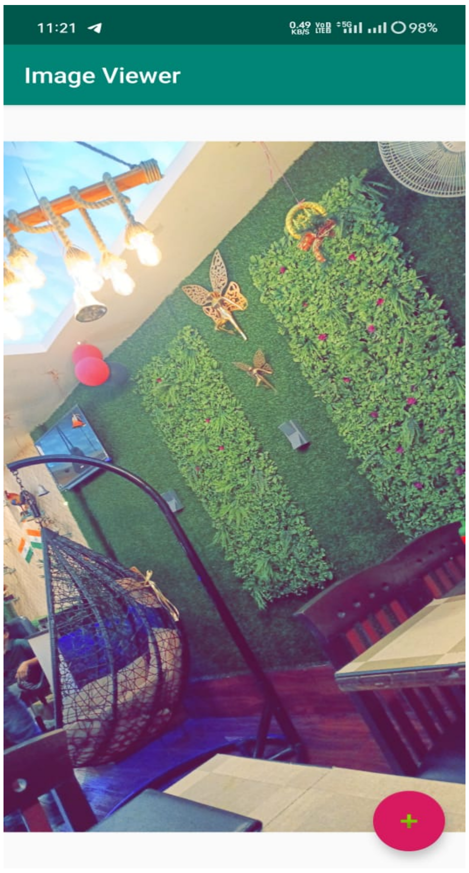
```
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

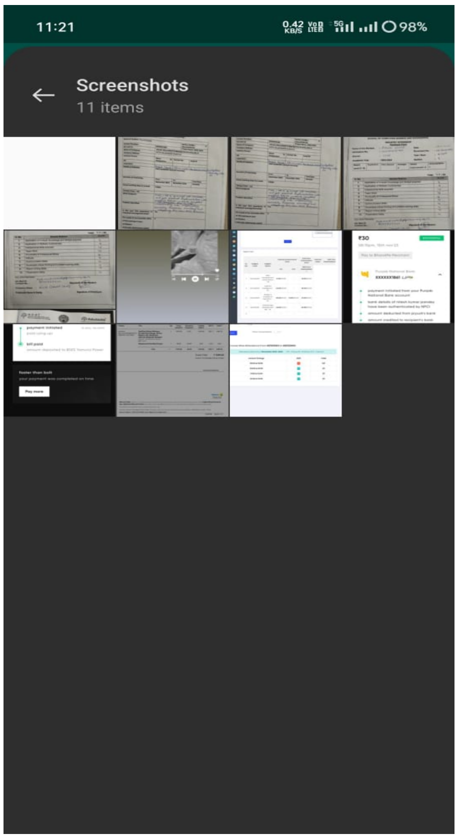
<u>content_main.xml</u>

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
    android:layout_height="match_parent"


    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main" tools:context=".MainActivity">

    <ImageView android:id="@+id/imageView" android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>
```

OUTPUT:


11:19
98%
Image Viewer
+


11:21
98%
Image Viewer
+


11:21
98%
Screenshots
11 items