EASYFLOW: Expense Tracker and Group Based Debt Management

Project by:

Vedansh Makharia - 220953050

Anirudh Bajaj - 220953104

Vansh Jogani - 220953126

Adit Basak - 220953134

1. Abstract

EasyFlow is an Android application designed to help users track their monthly expenses and manage shared group transactions with ease. It enables users to make friends, form groups, and log group transactions—simplifying them by reducing the number of payments using a smart debt optimization algorithm. The app leverages OCR technology through EasyOCR to scan receipts and automatically detect the highest amount for quick entry. Users can categorize expenses and visualize spending via bar charts. Furthermore, EasyFlow supports seamless UPI payments by allowing users to input a UPI ID and get redirected to Google Pay for faster settlements.

2. Introduction

Managing personal and shared expenses is a common problem faced by students, working professionals, and families. Traditional methods of recording expenses and calculating who owes what can become inefficient, especially in group settings. This inspired the creation of **EasyFlow**, an Android app that automates and simplifies expense sharing.
By integrating OCR for bill reading, automatic debt simplification, expense categorization, and UPI payment redirection, EasyFlow serves as a comprehensive financial management tool, especially tailored for the Indian audience where UPI usage is widespread.

3. Project Objective

- Allow users to register, add friends, and create friend groups
- Add group transactions involving specific group members
- Automatically simplify payments among group members to minimize transactions
- Use EasyOCR to extract the highest amount from scanned receipts for quick expense entry
- Enable users to categorize their personal and shared expenses
- Display expense breakdown by category using bar chart visualization
- Facilitate easy settlement of dues by accepting UPI IDs and redirecting to Google Pay for payments

4. Tools and Technologies Used

- **IDE:** Android Studio
- **Language:** Java
- **Database:** SQLite

- **Libraries & APIs:**
    - **EasyOCR** – for optical character recognition from bill images
    - **MPAndroidChart** – for bar chart visualization of expenses
    - **Intent-based UPI Integration** – For redirecting users to Google Pay for settlements

5. System Requirements

- **Minimum Android Version:** Android 9.0 (Pie, API level 28)
- **IDE Version:** Android Studio Hedgehog or higher
- **SDK:** Android SDK 28+
- **Hardware Requirements:**
    - Minimum 2GB RAM device for smooth performance
    - Camera access for bill scanning
- **Software Requirements:**
    - Java JDK 8 or higher
    - Android Emulator or physical device for testing

6. Project Features

1. Login

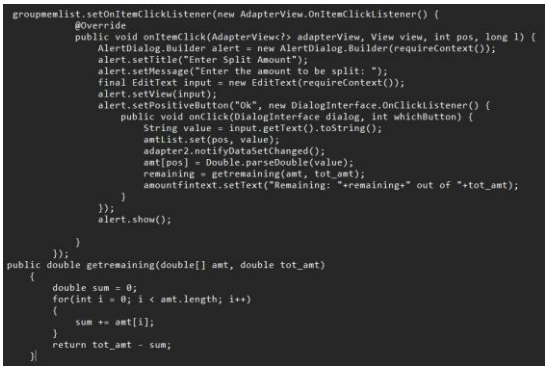
Allows registered users to securely access their account using email and password. Ensures that user data and expense records are protected and personalized.

2. Adding a Friend and Creating a Group

Users can search and add friends who are already on the app. After that, they can create groups consisting of these friends to manage shared expenses like trips, rent, or events.

3. Equal/Unequal Splitting of a Payment in a Group

When adding a group expense, users can choose to split the amount equally among members or manually assign different amounts to each member based on who owes what.



```
groupmemlist.setOnItemClickListener(new AdapterView.OnItemClickListener() {
          @Override
          public void onItemClick(AdapterView<?> adapterView, View view, int pos, long l) {
              AlertDialog.Builder alert = new AlertDialog.Builder(requireContext());
              alert.setTitle("Enter Split Amount");
              alert.setMessage("Enter the amount to be split: ");
              final EditText input = new EditText(requireContext());
              alert.setView(input);
              alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog, int whichButton) {
                      String value = input.getText().toString();
                      amtList.set(pos, value);
                      adapter2.notifyDataSetChanged();
                      amt[pos] = Double.parseDouble(value);
                      remaining = getremaining(amt, tot_amt);
                      amountfintext.setText("Remaining: "+remaining+" out of "+tot_amt);
                  }
              });
              alert.show();

          }
      });
public double getremaining(double[] amt, double tot_amt)
    {
        double sum = 0;
        for(int i = 0; i < amt.length; i++)
        {
            sum += amt[i];
        }
        return tot_amt - sum;
    }
```

Figure 1: Unequal Splitting

```
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
                         @Nullable Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_equal, container, false);

        //tottext = view.findViewById(R.id.tottextview);
        submit = view.findViewById(R.id.submit);

        SharedPreferences sharedPreferences;
        sharedPreferences = requireContext().getSharedPreferences("MyPrefs", Context.MODE_PRIVATE);
        g_id = "";
        g_id = sharedPreferences.getString("g_name",null);
        desc = sharedPreferences.getString("desc",null);
        type = sharedPreferences.getString("type",null);
        float tot_amt = sharedPreferences.getFloat("tot_amt",0);

        groupmemlist = view.findViewById(R.id.groupmemlist);

        MyDatabaseHelper dbHelper = new MyDatabaseHelper(requireContext());
        SQLiteDatabase db = dbHelper.getWritableDatabase();

        Cursor cursors = db.query(
                "Groups",
                new String[]{"group_id"},
                "group_name=?",
                new String[]{g_name},
                null, null, null
        );
        while (cursors.moveToNext())
        {
            g_id = cursors.getInt(cursors.getColumnIndexOrThrow("group_id"));
        }
        cursors.close();

        ArrayList<String> checkedTextViews = new ArrayList<>();

        Cursor cursor = db.rawQuery("SELECT users.user_id, NAME FROM GMeMbers JOIN users ON GMeMbers.user_id=users.user_id WHERE group_id=?", new String[]{""+g_id+""});


        checkedTextViews.clear();
        while (cursor.moveToNext())
        {
            String nb = cursor.getString(cursor.getColumnIndexOrThrow("name"));
            checkedTextViews.add(nb);
            //String name = cursor.getString(cursor.getColumnIndexOrThrow("name"));
            //Toast.makeText(MainActivity.this, "ID: "+id+" Name: "+name, Toast.LENGTH_SHORT).show();
        }
        cursor.close();


        //checkedTextViews.add(ctv1.getText().toString());
        //checkedTextViews.add(ctv2.getText().toString());
        ArrayAdapter<String> adapter = new ArrayAdapter<>(requireContext(), android.R.layout.select_dialog_multichoice,checkedTextViews);
        groupmemlist.setAdapter(adapter);
        groupmemlist.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);

        groupmemlist.setOnMultiChoiceModeListener(new AbsListView.MultiChoiceModeListener() {
            @Override
            public void onItemCheckedStateChanged(ActionMode actionMode, int i, long l, boolean b) {
                if(groupmemlist.isItemChecked(i))
                    counter++;
                else
                    counter--;
                if(counter == 0)
                {
                    //tottext.setText("Select atleast One Person");
                }
                else
                    //tottext.setText("Per Person total is : "+tot_amt/counter);
                    actionMode.setTitle(counter+ " Selected");

            }
```

Figure 2: Equal Splitting

4. Simplification Algorithm

The app automatically simplifies debts within a group.
For example:
If A owes B and B owes C, it reduces unnecessary transactions by directly making A owe C.
This reduces the total number of payments and simplifies settlement.

```
public List<String> simplifyDebts() {
        MyDatabaseHelper dbHelper = new MyDatabaseHelper(this);
        SQLiteDatabase db = dbHelper.getReadableDatabase();
        Map<Integer, Double> netBalance = new HashMap<>();

        // Query to retrieve debts from the database
        String query = "SELECT paid_by, payee_user_id, pay_amt " +
                "FROM Money";

        query = "SELECT payer_id, payee_id, owed_amt " +
                "FROM Owed";



        Cursor cursor = db.rawQuery(query, null);
        if (cursor.moveToFirst()) {
            do {
                int payer = cursor.getInt(0);  // Who paid
                int payee = cursor.getInt(1);  // Who owes
                double amount = cursor.getDouble(2);

                // Update net balance
                netBalance.put(payer, netBalance.getOrDefault(payer, 0.0) + amount);
                netBalance.put(payee, netBalance.getOrDefault(payee, 0.0) - amount);
            } while (cursor.moveToNext());
        }
        cursor.close();
        //db.close();

        return minimizeTransactions(netBalance, db);
    }

    // Algorithm to minimize transactions
    private List<String> minimizeTransactions(Map<Integer, Double> netBalance, SQLiteDatabase db) {
        List<String> simplifiedTransactions = new ArrayList<>();

        // Separate creditors and debtors
        List<int[]> creditors = new ArrayList<>();
        List<int[]> debtors = new ArrayList<>();

        for (Map.Entry<Integer, Double> entry : netBalance.entrySet()) {
            int user = entry.getKey();
            double balance = entry.getValue();
            if (balance > 0) {
                creditors.add(new int[]{user, (int) (balance * 100)}); // Convert to cents to avoid floating errors
            } else if (balance < 0) {
                debtors.add(new int[]{user, (int) (-balance * 100)});
            }
        }

        // Match debtors with creditors
        int i = 0, j = 0;
```

```
while (i < debtors.size() && j < creditors.size()) {
    int debtor = debtors.get(i)[0];
    int debtAmount = debtors.get(i)[1];

    int creditor = creditors.get(j)[0];
    int creditAmount = creditors.get(j)[1];

    int settledAmount = Math.min(debtAmount, creditAmount);

    String dname = "";
    String cname = "";
    Cursor cursor = db.query(
            "Users",
            new String[]{"name"},
            "user_id = ?",
            new String[]{""+debtor+""},
            null, null, null
    );
    while (cursor.moveToNext())
    {
        dname = cursor.getString(cursor.getColumnIndexOrThrow("name"));
    }
    cursor.close();
    cursor = db.query(
            "Users",
            new String[]{"name"},
            "user_id = ?",
            new String[]{""+creditor+""},
            null, null, null
    );
    while (cursor.moveToNext()) {
        cname = cursor.getString(cursor.getColumnIndexOrThrow("name"));
    }


    simplifiedTransactions.add("User " + dname + " pays " + (settledAmount / 100.0) + " to User " + cname);

    // Update remaining amounts
    debtors.get(i)[1] -= settledAmount;
    creditors.get(j)[1] -= settledAmount;

    if (debtors.get(i)[1] == 0) i++; // Move to the next debtor
    if (creditors.get(j)[1] == 0) j++; // Move to the next creditor
}

return simplifiedTransactions;
```

Figure 3: Simplified Transaction Algorithm

5. OCR (Optical Character Recognition)

Integrates EasyOCR to scan physical bills/receipts. The app intelligently detects the largest numeric value on the receipt and auto-fills it for quick expense addition in group transactions.

```
        imagePickerLauncher = registerForActivityResult(
                new ActivityResultContracts.StartActivityForResult(),
                result -> {
                    if (result.getResultCode() == Activity.RESULT_OK) {
                        Intent data = result.getData();
                        if (data != null && data.getData() != null) {
                            Uri selectedImageUri = data.getData();
                            Bitmap selectedBitmap = handleSelectedImage(selectedImageUri);
                            if (selectedBitmap != null) {
                                processImage(selectedBitmap); // Call your processing method
                            }
                        }
                    }
                }
        );
private void processImage(Bitmap bitmap) {
        // Load image from drawable
        // Load image from drawable
        //Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.img2);
        InputImage image = InputImage.fromBitmap(bitmap, 0);

// Initialize Text Recognizer
        com.google.mlkit.vision.text.TextRecognizer recognizer =
                TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS);

// Process the image for text extraction
        recognizer.process(image)
                .addOnSuccessListener(visionText -> {
                    String extractedText = visionText.getText();
                    ocrOutput.setText(extractedText);
                    String totalAmount = findTotalAmount(extractedText);
                    String amt[] = totalAmount.split(" ");
                    Toast.makeText(this, ""+amt[1], Toast.LENGTH_SHORT).show();
                    textOutput.setText(totalAmount.isEmpty() ? "Total not found." : totalAmount);
                    //Toast.makeText(this, totalAmount, Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(getApplicationContext(), GroupExpenseActivity.class);
                    double amt1 = Double.parseDouble(amt[1]);
                    intent.putExtra("amt", amt1);
                    String gname = getIntent().getStringExtra("GROUP_NAME");
                    intent.putExtra("GROUP_NAME", gname);
                    startActivity(intent);
                });
                //.addOnFailureListener(e -> textOutput.setText("Error: " + e.getMessage()));
        // Error handling
    }
```

Figure 4: OCR

6. Personal Expense Tracking

Users can add personal (non-group) expenses and assign them to categories like Food, Travel, Bills, etc. The app shows bar charts to visualize how much was spent in each category, helping users track and manage their spending habits.

```
public void setData(List<Float> values, List<String> labels) {
        this.values = values;
        this.labels = labels;
        invalidate(); // Redraw the view
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        if (values.isEmpty()) return;

        float maxValue = getMaxValue();
        float barWidth = getWidth() / (values.size() * 2f);
        float startX = barWidth / 2;
        float bottom = getHeight() - 50;

        // Draw title
        canvas.drawText("Monthly Expenses", getWidth()/2, 40, labelPaint);

        for (int i = 0; i < values.size(); i++) {
            float height = (values.get(i) / maxValue) * (getHeight() - 150);
            float left = startX + i * barWidth * 2;
            float top = getHeight() - height - 50;
            float right = left + barWidth;

            // Set bar color based on index, cycling through the colors
            barPaint.setColor(barColors[i % barColors.length]);

            // Draw bar
            canvas.drawRect(new RectF(left, top, right, bottom), barPaint);

            // Draw value above bar
            canvas.drawText(String.format("Rs.%.2f", values.get(i)),
                    (left + right)/2, top - 10, textPaint);

            // Draw month label
            canvas.drawText(labels.get(i), (left + right)/2, bottom + 40, textPaint);
        }
    }
```

Figure 5: Bar Chart Code

7. GPay Integration for Direct Payments

The app includes a seamless Google Pay (GPay) integration for settling dues.When a user selects a friend to pay, the app automatically fetches their GPay UPI ID from the database.

The user simply enters the amount, and the app redirects them to GPay with pre-filled details—making the payment process quick and hassle-free.

```
btn.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
              if(!amount.getText().toString().isEmpty() && !upiid.getText().toString().isEmpty() && !note.getText().toString().isEmpty()) {



                  ContentValues values;
                  values = new ContentValues();
                  values.put("payee_id", payee_id);
                  values.put("payer_id", curr_id);
                  values.put("owed_amt", Double.parseDouble(amount.getText().toString()));
                  db.insert("Owed", null, values);
                  Toast.makeText(requireContext(), "Payment of Rs."+amount.getText().toString()+" sent to "+upiid.getText().toString(), Toast.LENGTH_SHORT).show();

                  values = new ContentValues();
                  values.put("payee_id", payee_id);
                  values.put("payer_id", curr_id);
                  values.put("amt", Double.parseDouble(amount.getText().toString()));
                  db.insert("Settle", null, values);

                  String GOOGLE_PAY_PACKAGE_NAME = "com.google.android.apps.nbu.paisa.user";
                  int GOOGLE_PAY_REQUEST_CODE = 123;

                  Uri uri =
                          new Uri.Builder()
                                  .scheme("upi")
                                  .authority("pay")
                                  .appendQueryParameter("pa", upiid.getText().toString())
//                                  .appendQueryParameter("pn", name.getText().toString())
//                                  .appendQueryParameter("mc", "your-merchant-code")
//                                  .appendQueryParameter("tr", "your-transaction-ref-id")
                                  .appendQueryParameter("tn", note.getText().toString())
                                  .appendQueryParameter("am", amount.getText().toString())
                                  .appendQueryParameter("cu", "INR")
//                                  .appendQueryParameter("url", "your-transaction-url")
                                  .build();
                  Intent intent = new Intent(Intent.ACTION_VIEW);
                  intent.setData(uri);
                  intent.setPackage(GOOGLE_PAY_PACKAGE_NAME);
                  startActivityForResult(intent, GOOGLE_PAY_REQUEST_CODE);
              }
          }
      });
```

Figure 6: Gpay Integration

8. Database Design

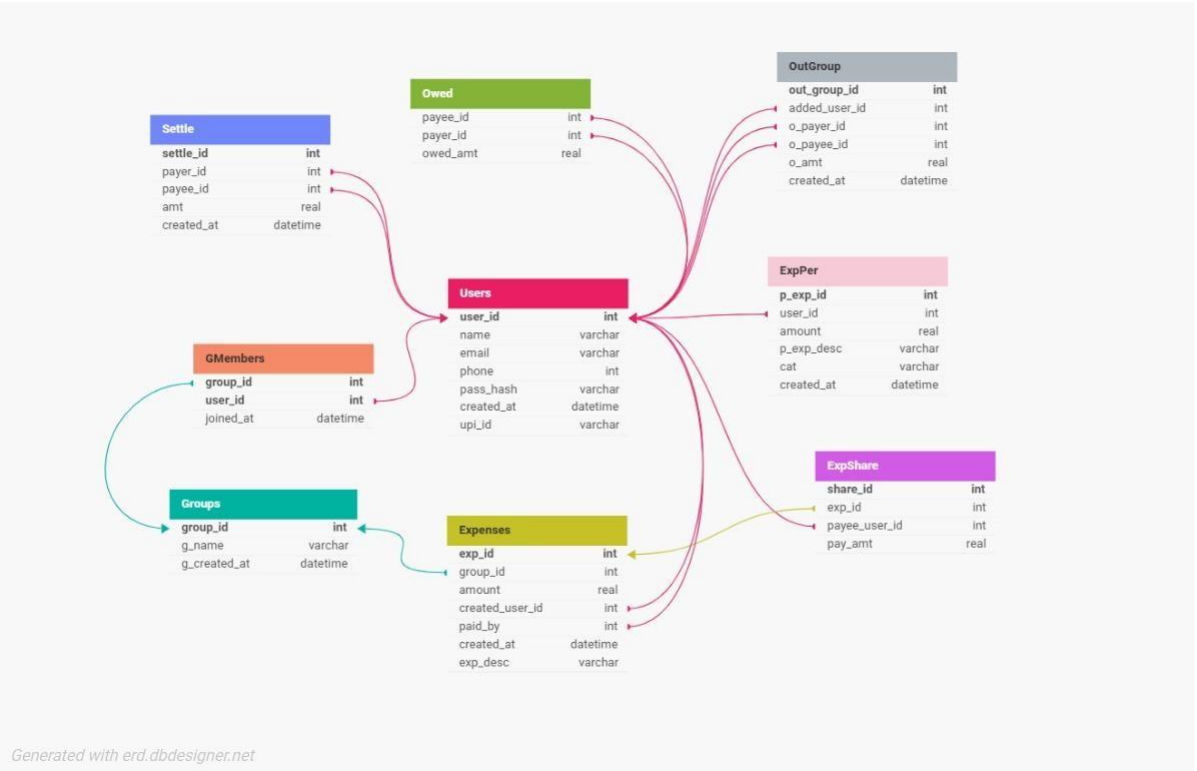


Figure 7: Database Schema

Sample Data in each table

- **Users**

User_id	Name	Email	Phone	Pass_hash	Created_at	Upi_id
1	Vansh	Vanshjogani@gmail.com	9886338115	Vj1	2025-04-18 10:00	Vanshjogani@okicici
2	Vedanh	vedanshmakharia@gmail,.com	9090881556	Vm1	2025-04-19 9:00	vedanshmakharia@okicici

- **Groups**

Group_id	G_name	G_created_at
1	Goa	2025-04-19 11:00

- ## Gmembers

Group_id	User_id	Joined_at
1	1	2025-04-19 11:00
1	2	2025-04-19 12:00

- ## Expenses

Exp_id	Group_id	Amount	Created_user_id	Paid_by	Created_at	Exp_desc
1	1	1500.0	1	1	2025-04-19 11:00	hotel
2	1	600	2	2	2025-04-19 12:00	food

- ## ExpShare

Share_id	Exp_id	Payee_user_id	Pay_amt
1	1	2	500
2	1	3	500

- ## ExpPer

P_exp_id	User_id	Amount	P_exp_desc	Cat	Created_at
1	1	250.0	Groceries	Food	2025-04-19 12:00
2	2	1200.0	Groceries	Food	2025-04-19 12:00

- ## Owed

Payee_id	Payer_id	Owed_amt
1	2	500
1	3	500

- ## Settle

Settle_id	Payer_id	Payee_id	Amt	Created_at
1	2	1	500	2025-04-19 12:00
2	3	1	500	2025-04-19 12:00

- ## Outgroup

Out_group_id	Added_user_id	O_payer_id	O_payee_id	O_amount	Created_at
1	1	1	2	200.0	2025-04-19 14:00
2	2	3	2	150.0	2025-04-19 15:00

9. Testing and Output

How we tested the app:

- We used manual testing on real Android devices to test all the major flows:
    - User registration and login
    - Adding friends and creating groups
    - Adding and splitting transactions
    - Running the debt simplification logic
    - EasyOCR scanning functionality
    - Visualizing expenses by category in bar charts
    - UPI redirection to Google Pay
- We also verified data integrity by checking SQLite database entries after each transaction and group update.

Sample outputs:

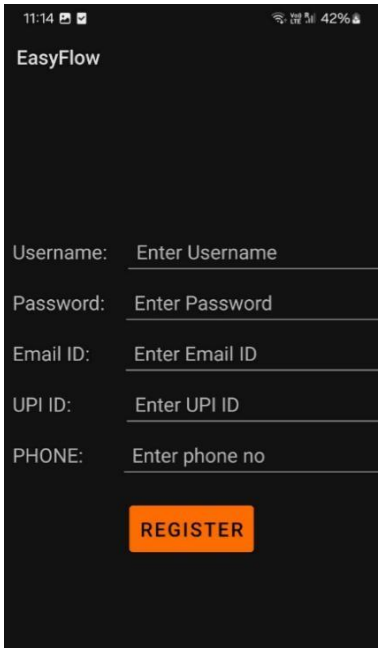


Figure 8: Login page

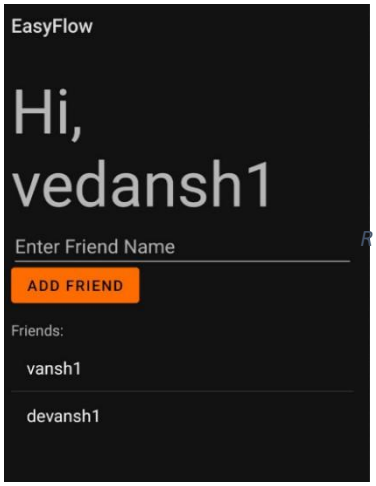


Figure 9: Making friends

Figure 10: Text Extracted from OCR

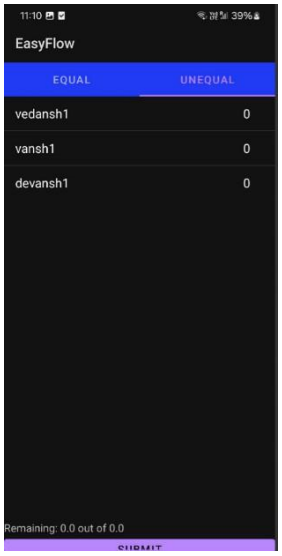


Figure 11: Selecting the custom number of people in a transaction

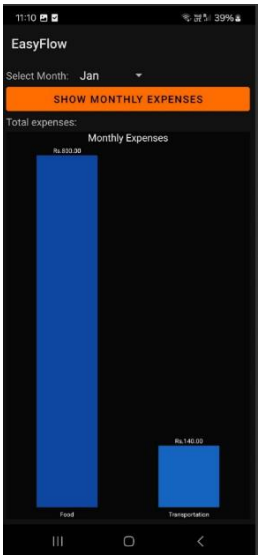


Figure 12: Bar chart showing monthly expenses

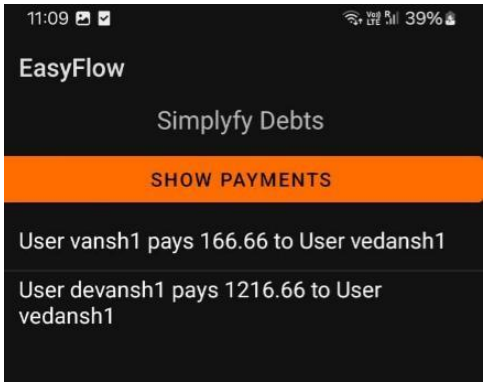


Figure 13: Payments to be made after the simplification algorithm is used

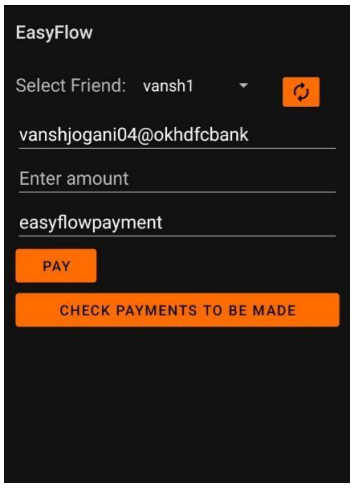


Figure 14: Payment portal that directly redirects users to GPay

Known issues:

- OCR sometimes picks up incorrect values in poorly lit or blurry receipts.
- UPI redirection works best with Google Pay; other UPI apps may not open automatically.
- No real-time sync across multiple devices as it currently uses local SQLite DB.

10. Conclusion

What we learned:

- Hands-on experience with Android development using Java/Kotlin.
- How to implement SQLite databases for structured storage.
- Integrating OCR libraries (EasyOCR) in Android apps.
- Implementing algorithms for debt simplification in group transactions.
- Creating data visualizations with bar charts to represent category-wise spending.

Challenges faced:

- Implementing efficient and accurate debt simplification logic.
- Parsing text from OCR results and reliably identifying the largest amount.
- UI handling for dynamic group-based expense sharing.
- UPI integration without deep-linking issues.

Future Scope:

- Move to Firebase or Room DB for cloud syncing and real-time updates.
- Implement push notifications for expense updates or payment reminders.
- Add receipt history and image storage.
- Introduce monthly budgeting and limit warnings.
- Improve OCR by training a custom model or using Google ML Kit.

11. References

- Android Developer Documentation
- SQLite in Android