A Personal Finance Management App

1.Introduction:

Overview:

An expense tracker app allows you to monitor and categorize your expenses across different bank and investment accounts and credit cards. Some of these apps also offer budgeting tools, credit monitoring, mileage tracking, receipt keeping, and advice to grow your net worth.

Expense tracker apps vary in cost, although many are free. Paid apps are typically less than $10 per month, although if you want business features like invoice creation and management, you’re more likely to pay well above that. Before paying for an expense tracker app, check to see if there’s a free trial. A lot of apps offer 30 days to test drive all of their features.

Purpose:

Expense tracking helps your business by allowing you to identify and manage spending in an efficient time frame. It’s essential to be aware of your business’s cash flows so you can note any areas of excessive or inefficient spending. When you track your expenses, you can catch these before they get out of hand.

1. Helps you Take Control of Your Finances

Knowing how and when you spend is key to controlling your finances. When you record your expenses, it’s easier to note any spending that doesn’t fit your goals so you can manage your future spending to grow your business.

2. Gives you a Time Frame to Manage Your Finances

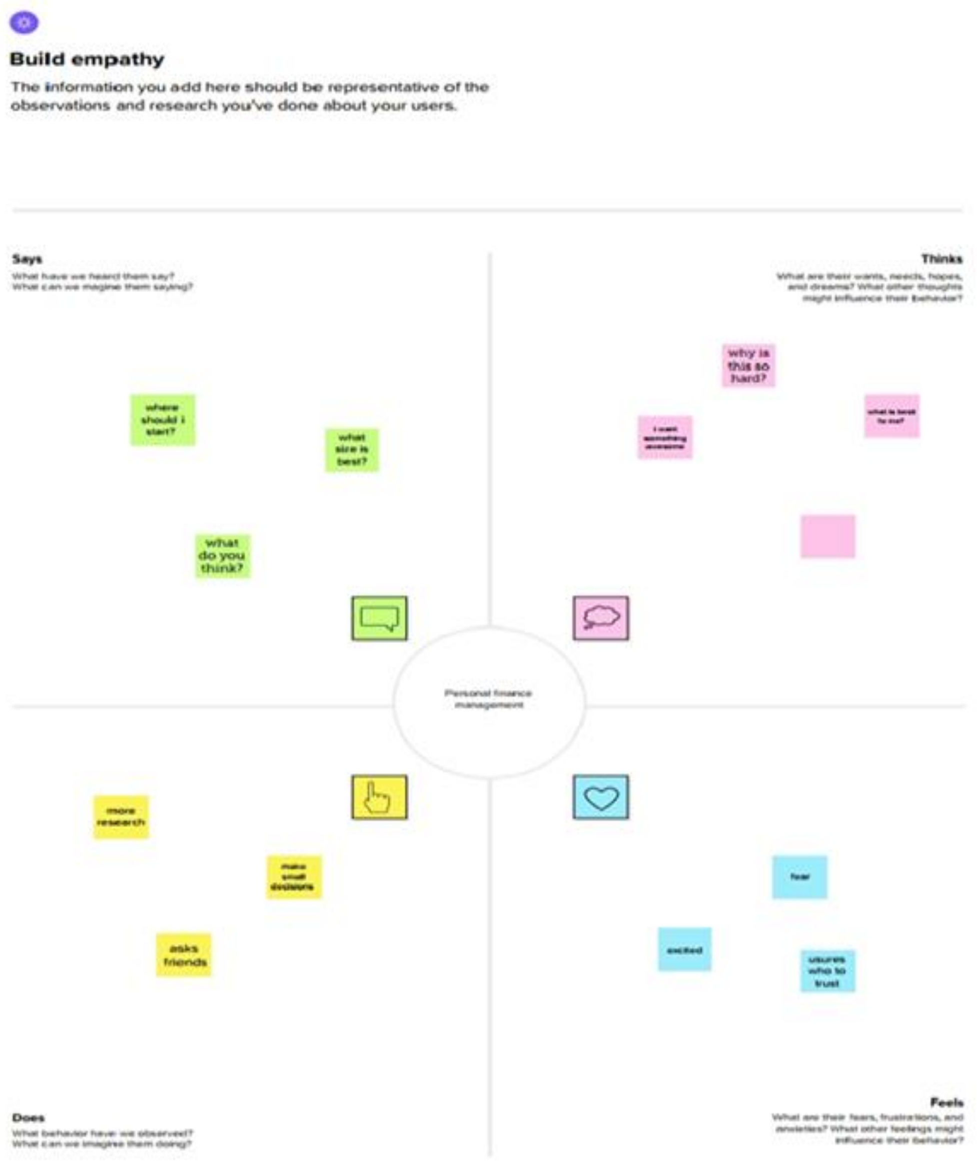
Expense tracking isn’t just for knowing where your money goes — it’s for adjusting your spending promptly to suit changing business needs. When you record your expenses within a certain time frame, it’s easy to review your spending and decide what to keep and cut.

3. Tracking Your Expenses Makes You a Better Money Manager

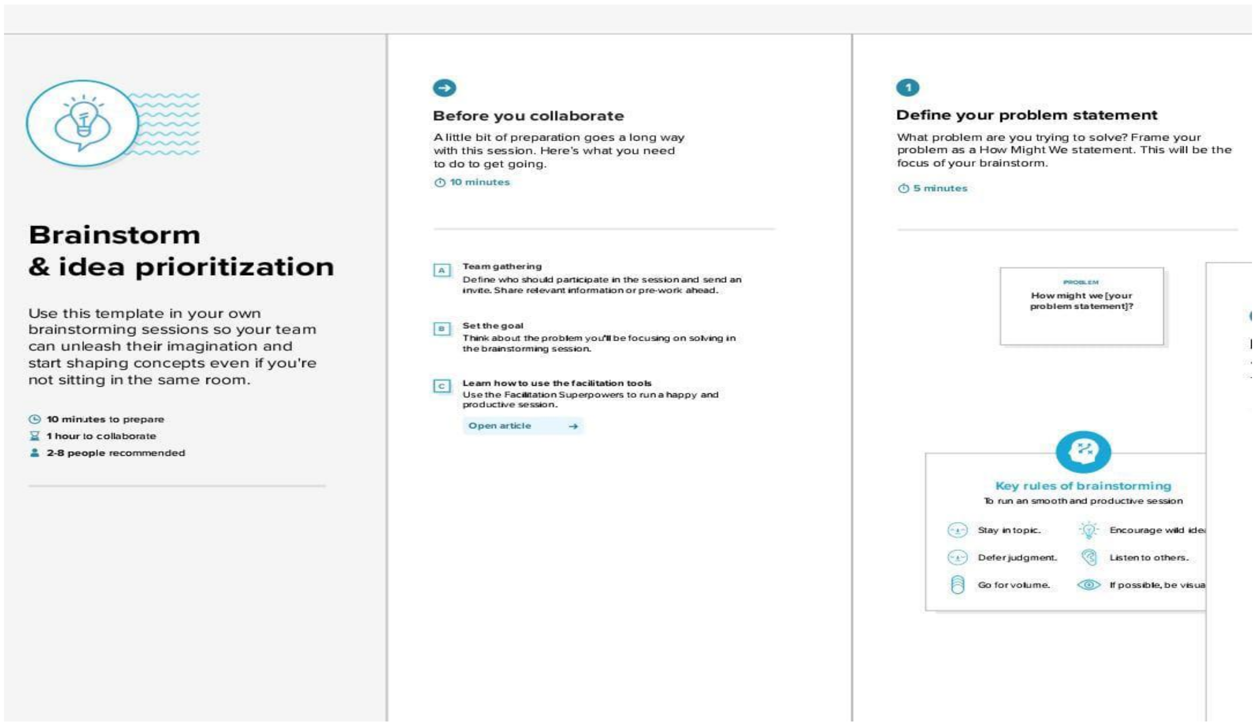
As a business owner, staying on top of your monthly budget is crucial to success. There is no better way to manage your budget and know where your money is going than expense tracking.

2.Problem Definition & Design Thinking:

Empathy Map:


Build empathy
The information you add here should be representative of the observations and research you've done about your users.
Says
What have we heard them say? What can we imagine them saying?
where should i start?
what size is best?
what do you think?
Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?
why is this so hard?
I want something awesome
what is best for me?
Personal finance management
more research
make small decisions
asks friends
fear
excited
unsures who to trust
Does
What behavior have we observed? What can we imagine them doing?
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

Ideation & Brainstorming Map:



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 10 minutes to prepare
- 1 hour to collaborate
- 2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1 **Define your problem statement**

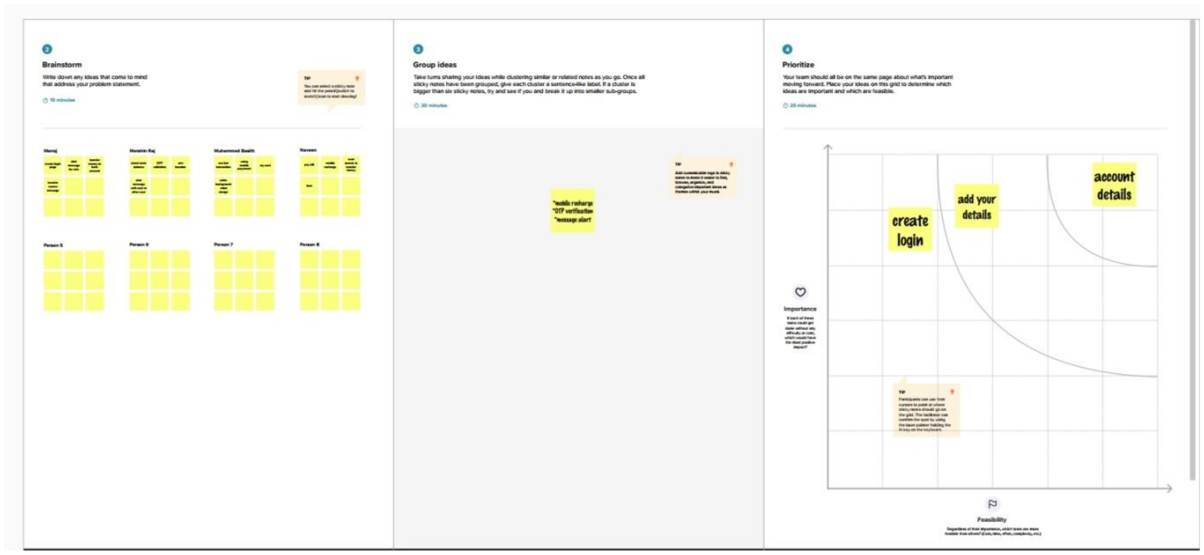
What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM
How might we [your problem statement]?

Key rules of brainstorming
To run an smooth and productive session

Stay in topic.	Encourage wild idea
Defer judgment.	Listen to others.
Go for volume.	If possible, be visua



2 **Brainstorm**

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Manoj · Merahin Raj · Muhammed Basith · Naveen · Person 5 · Person 6 · Person 7 · Person 8

3 **Group ideas**

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

*mobile recharge
*OTP verification
*message alert

TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

4 **Prioritize**

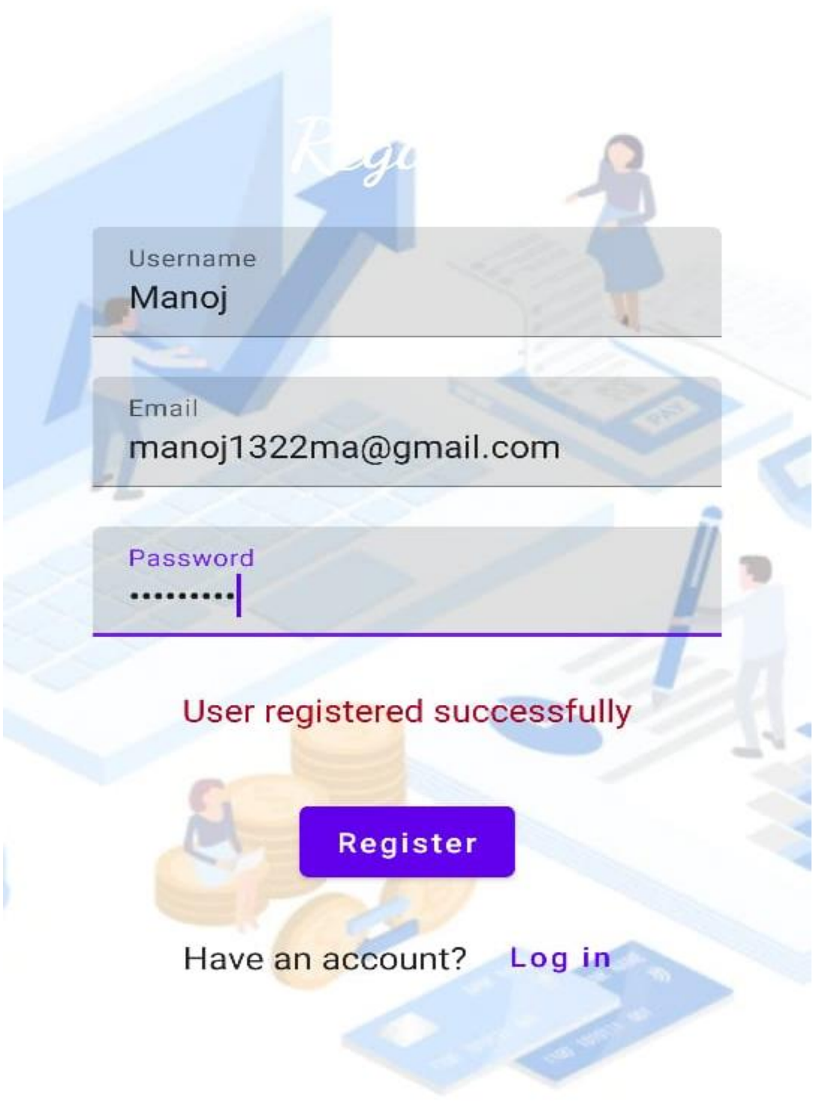
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

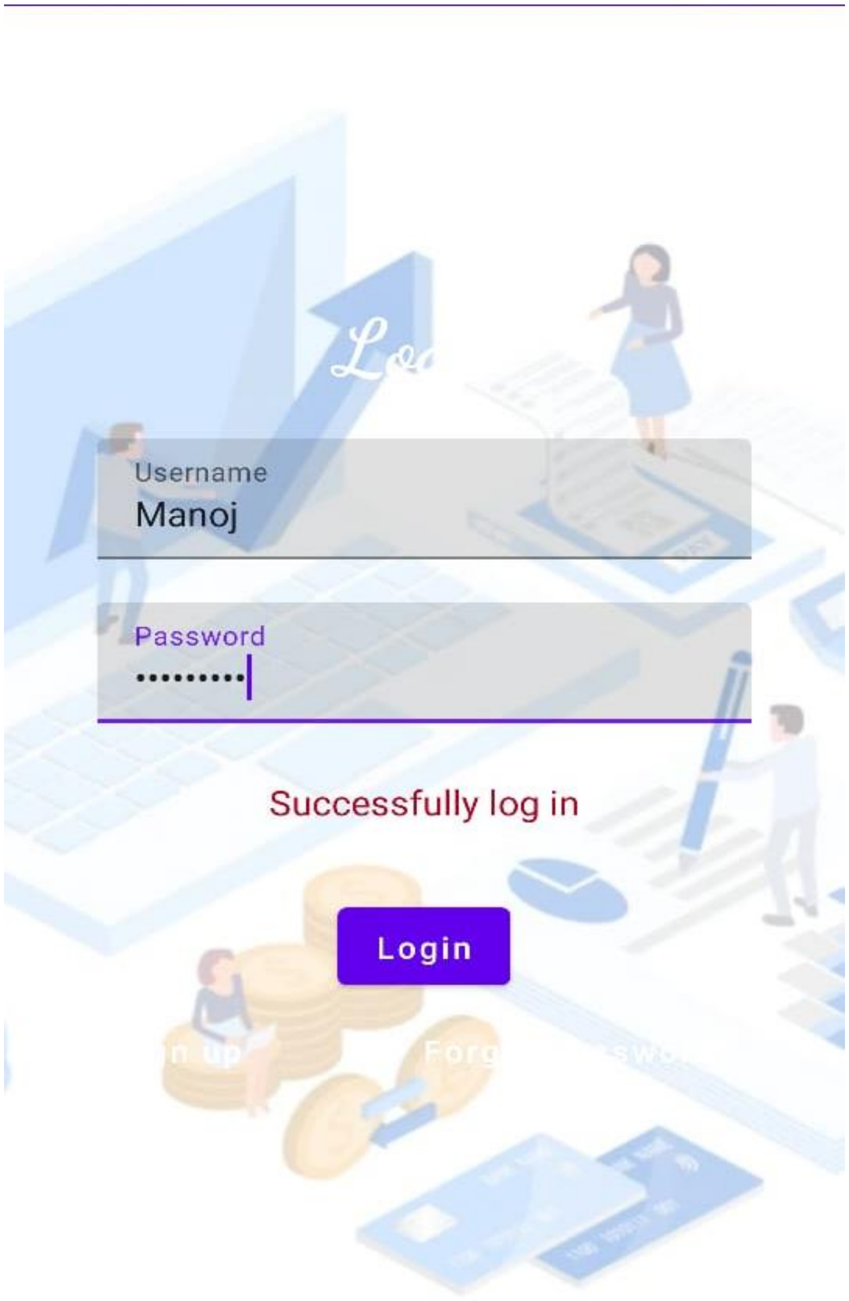
20 minutes

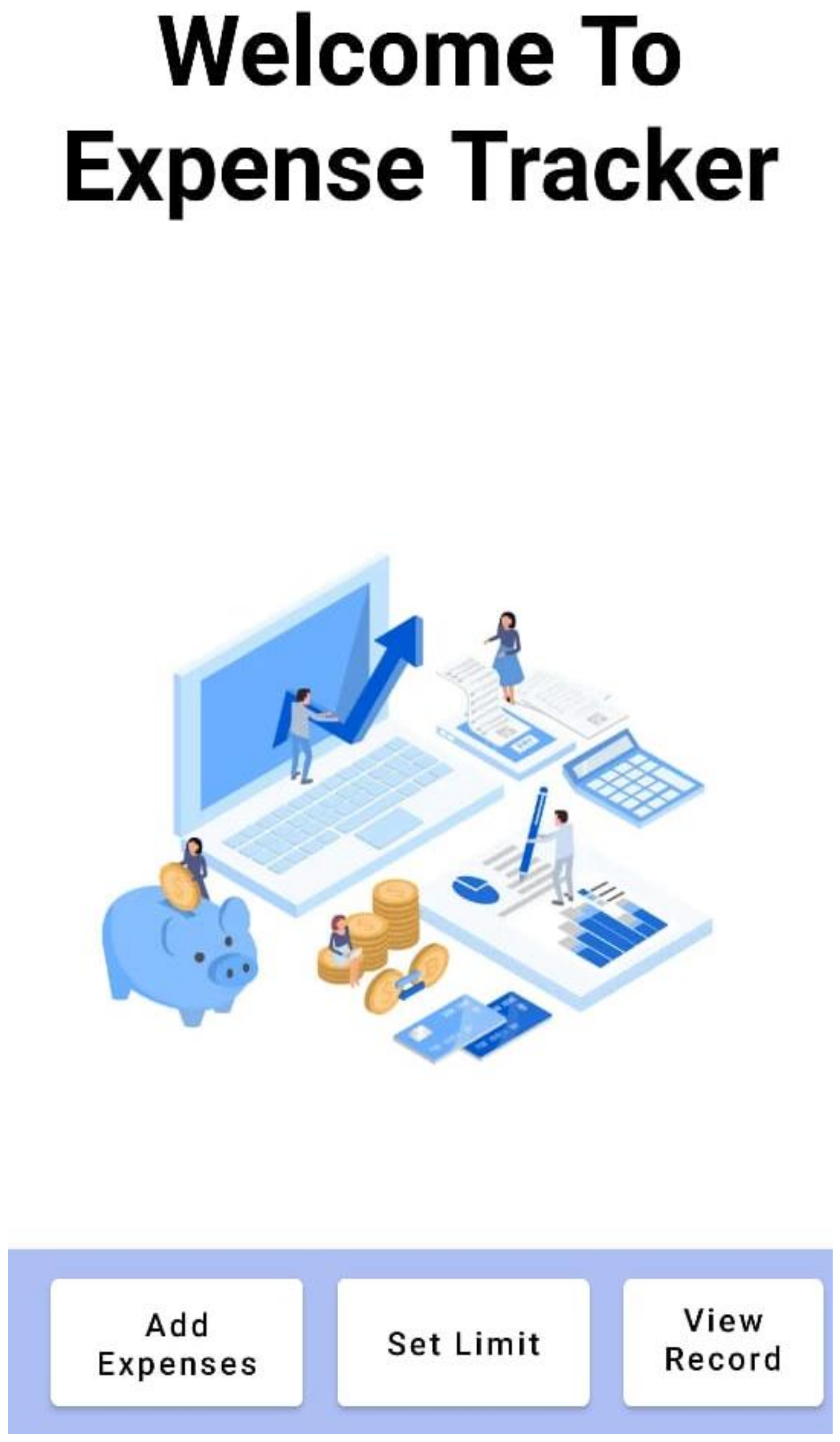
create login
add your details
account details
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

TIP
Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.

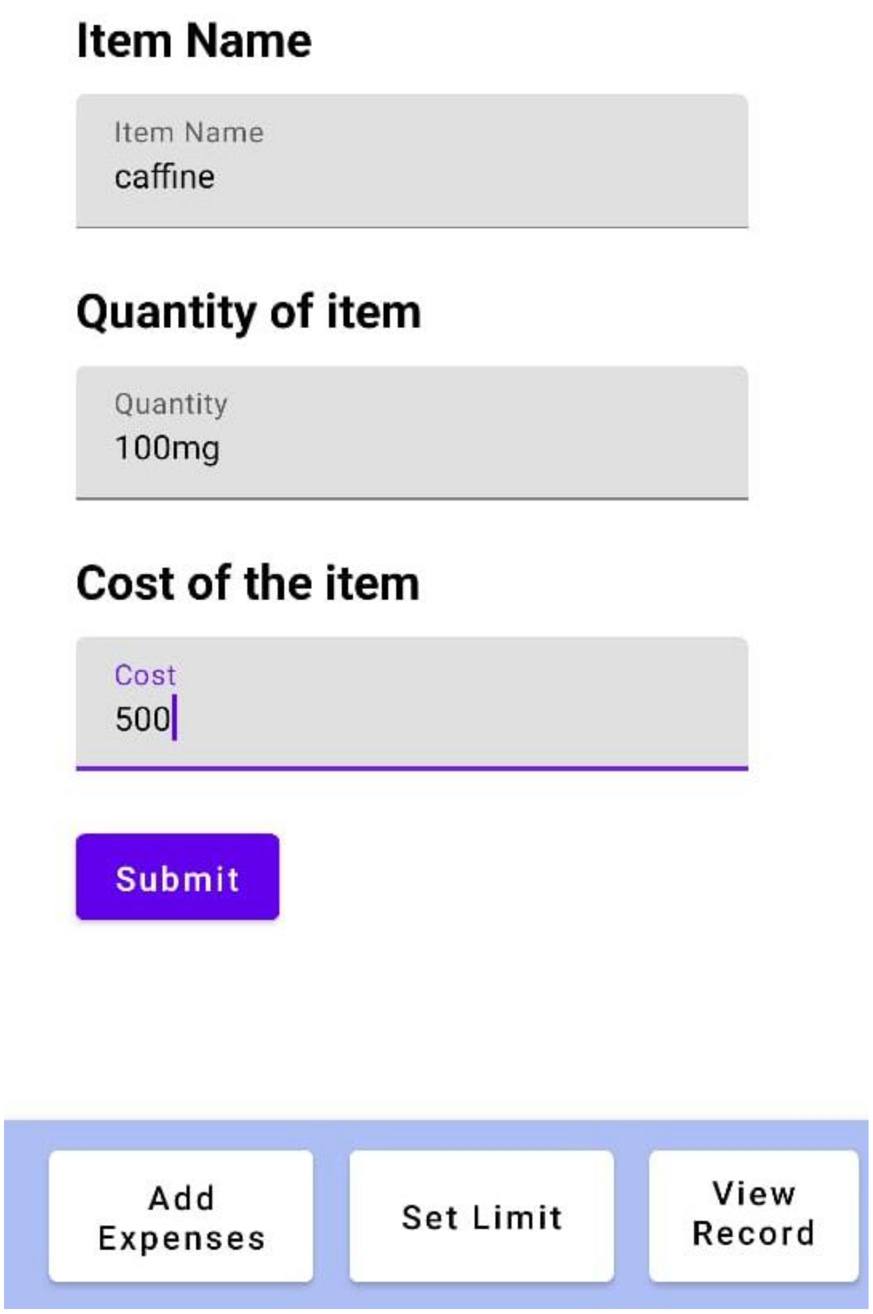
3.Result:


Username
Manoj
Email
manoj1322ma@gmail.com
Password
•••••••••
User registered successfully


Username
Manoj
Password
•••••••••
Successfully log in
Login



Welcome To
Expense Tracker



Item Name

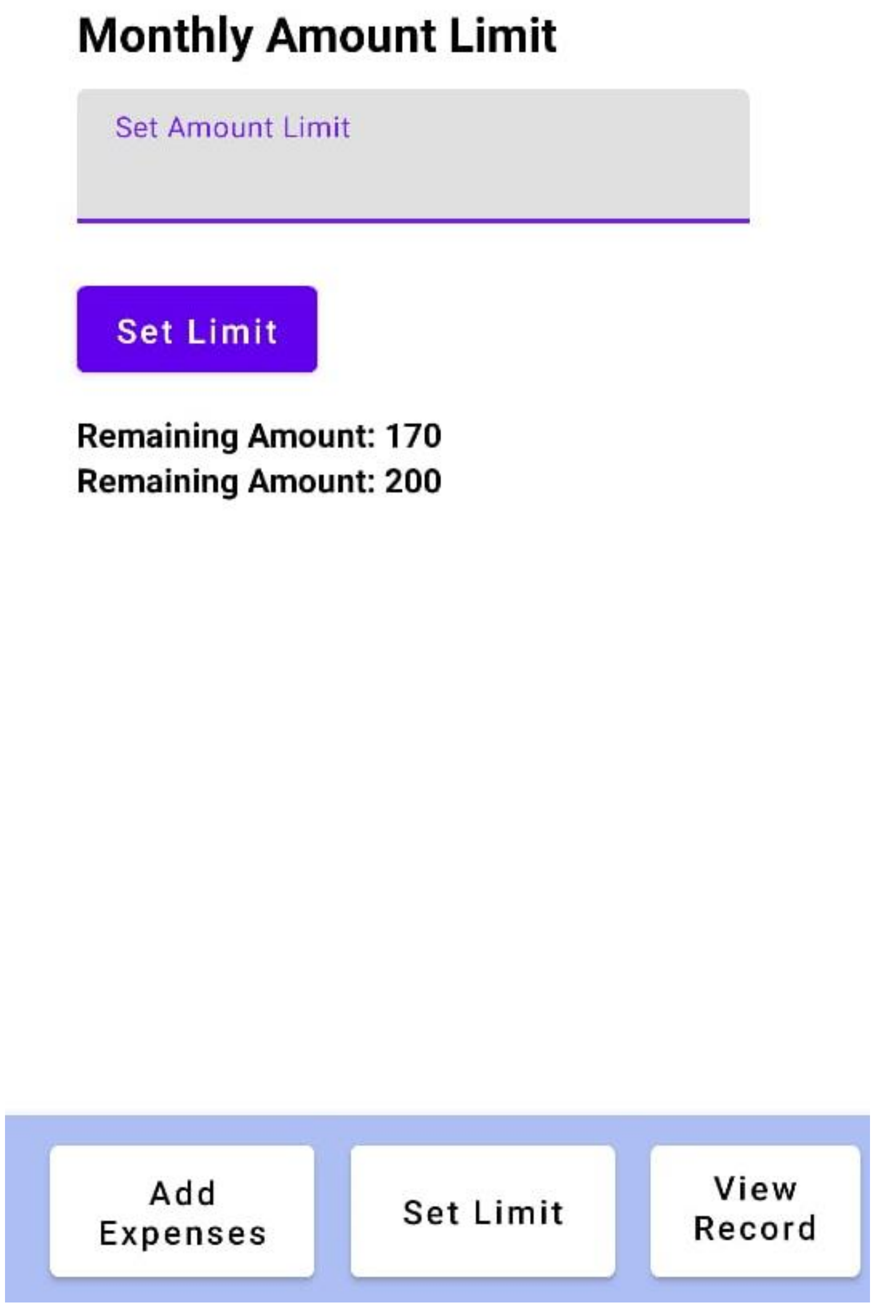
Item Name
caffine

Quantity of item

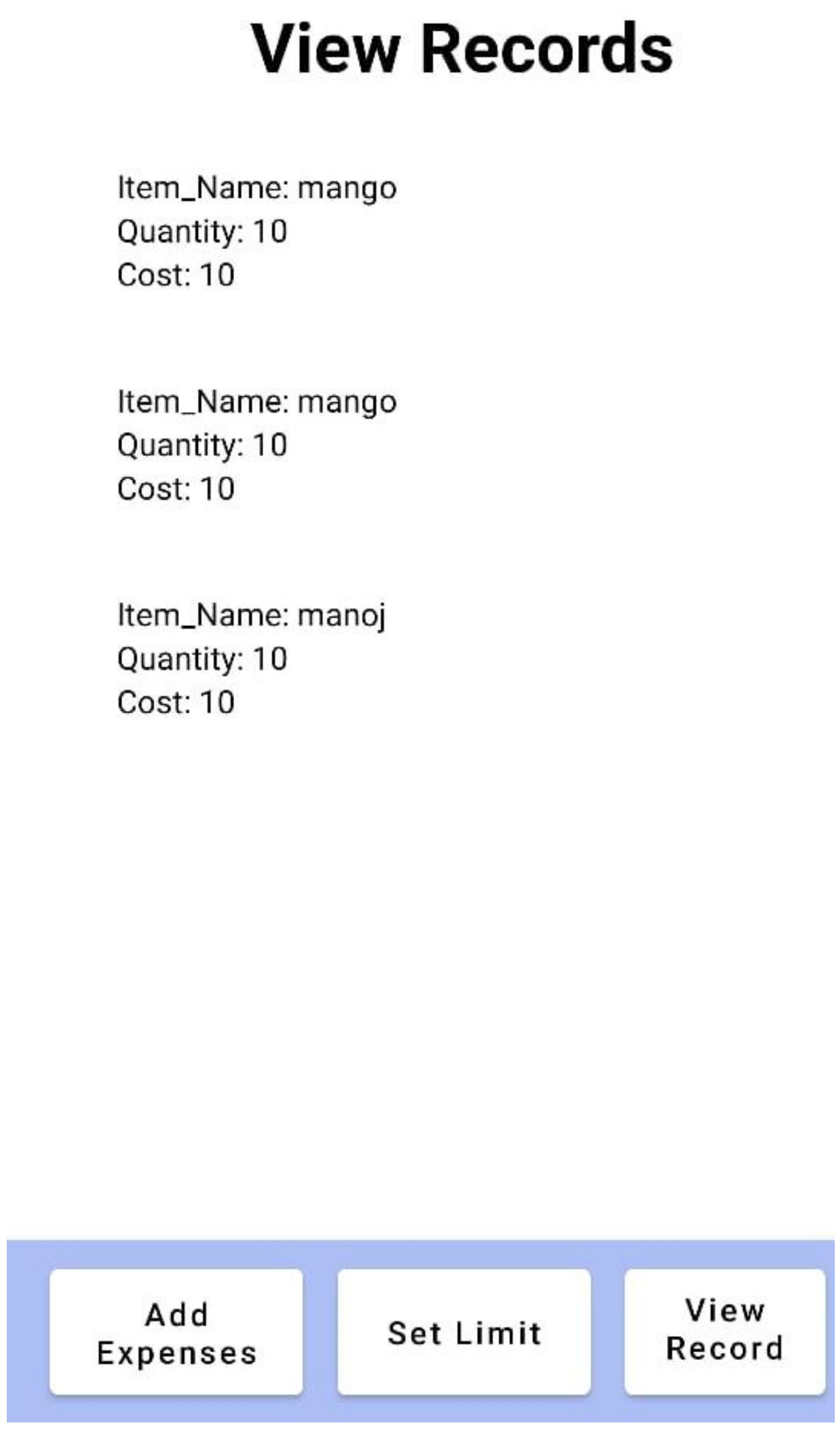
Quantity
100mg

Cost of the item

Cost
500


Monthly Amount Limit

Remaining Amount: 170
Remaining Amount: 200


View Records

Item_Name: mango
Quantity: 10
Cost: 10

Item_Name: mango
Quantity: 10
Cost: 10

Item_Name: manoj
Quantity: 10
Cost: 10

4.Advantages & Disadvantage:

*This can help you understand how much money you can spend for the rest of the project while staying within your budget. Allows for budgeting on future projects: If you record your expenses for a project, you can use that information to budget for similar future projects

*A con with any system used to track spending is that one may start doing it then taper off until it's forgotten about all together. Yet, this is a risk for any new goal such as trying to lose weight or quit smoking.

5.Applications:

*Consultants:

Solution to help consultants manage their expenses.

*Education:

Educational institutions need not worry about their expenses.

*Healthcare:

Healthcare manufacturers can expenses easily.

*Manufacturing:

Manufacturing expenses are no longer a pain.

*Non-profits:

Go on your mission and don’t worry about expense reporting.

*Marketing :

Market your offering and leave your expense reporting to us.

*Self-employed:

Be your own boss,we'll handle your expenses.

*Small Business:

Grow your business, we'll handle your expense management.

6.Conclusion:

Tracking your expenses daily can save your amount, but it can also help you set financial goals for the future. If you know exactly where your amount is going every month, you can easily see where some cutbacks and compromises can be made.

7.Future Scope:

Changing face of expense management software

Year-on-year, modern expense management software undergone a continuous evolution from traditional back-office function to strategic internal set of processes. But would it be sufficient to meet the needs of next-gen companies? Have you ever thought how the next-generation software should look like? As the requirements of companies evolve continuously, the software should undergo a series of changes to meet the growing needs of next generation companies.

The next-generation travel and expense (T & E) management apps should not only just accelerate the expense management process but also should come with mobile and cloud integration capabilities that add tremendous value to the business bottom line. Future T & E management software should be able to provide greater visibility into spending and standardise critical procedures.

Next-gen expense management

A recent T & E study unveiled that visibility and intelligence are the two key aspects that companies look forward to understanding spending associated with business travel. Analytics is also on the priority list of the best-in-class organisations. The motto is not just to enhance the existing process but also to leverage analytical capabilities and visibility that can help companies drive efficiency, forecast and plan better for corporate finances.

Apparently, as per research, integration, analytics and mobile apps are the three key factors that can help companies succeed at a faster pace. When incorporated, these factors add edge and value to the businesses.

Integration

Integration between corporate cards and expense management software increases transparency and makes the process effortless throughout the expense report cycle.

Analytics

Increased intelligence provides you with an unheralded level of visibility into travel spending and enhances overall T & E intelligence. Companies can measure the true performance of any business trip by evaluating the ROI. Efficiency complimented by intelligence proves to be a great way to take the business to new heights.

Need for mobile applications

Mobile apps provide employees with the opportunity to manage expenses on the go. It gives both the companies and employees the flexibility that they need in managing the expense related activities. In fact, it increases accuracy as the power of technology is put into the hands of both employees and employers.

On a final note, the next generation software should be something that gives users an unprecedented experience in expense management and allows organisations to emphasise on what’s really important to make their businesses better.

8.Appendix:

Create User data class:

```
Package
com.example.expensestracker
                                  import androidx.room.ColumnInfo
                                  import androidx.room.Entity
                                  import androidx.room.PrimaryKey
                                  @Entity(tableName = "user_table")
```

```
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?,
    @ColumnInfo(name = "email") val email: String?,
    @ColumnInfo(name = "password") val password: String?,
    )
```

Create an UserDao interface:

```
package
com.example.expensestracker
import androidx.room.*
@Dao
interface UserDao {
    @Query("SELECT * FROM user_table WHERE email = :email")
    suspend fun getUserByEmail(email: String): User?
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)
    @Update
    suspend fun updateUser(user: User)
    @Delete
    suspend fun deleteUser(user: User)
}
```

Create an UserDatabase class:

```
package
com.example.expensestracker
import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao
    companion object {
        @Volatile
        private var instance: UserDatabase? = null
        fun getDatabase(context: Context): UserDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    UserDatabase::class.java,
                    "user_database"
                ).build()
                instance = newInstance
                newInstance
            }
        }
    }
}
```

Create an UserDatabaseHelper class:

```
package
com.example.expensestracker
import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
class UserDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null,
DATABASE_VERSION) {
    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "UserDatabase.db"
        private const val TABLE_NAME = "user_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_FIRST_NAME = "first_name"
        private const val COLUMN_LAST_NAME = "last_name"
        private const val COLUMN_EMAIL = "email"
        private const val COLUMN_PASSWORD = "password"
    }
    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "$COLUMN_FIRST_NAME TEXT, " +
                "$COLUMN_LAST_NAME TEXT, " +
                "$COLUMN_EMAIL TEXT, " +
                "$COLUMN_PASSWORD TEXT" +
                ")"
        db?.execSQL(createTable)
    }
    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion:
Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }
    fun insertUser(user: User) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }
    @SuppressLint("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME
WHERE $COLUMN_FIRST_NAME = ?", arrayOf(username))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getUserById(id: Int): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME
WHERE $COLUMN_ID = ?", arrayOf(id.toString()))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getAllUsers(): List<User> {
        val users = mutableListOf<User>()
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME",
null)
        if (cursor.moveToFirst()) {
            do {
                val user = User(
                    id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                    lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                    email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                    password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
                )
                users.add(user)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return users
```

```
  }
}
```

Create User data class:

```
package
com.example.expensestracker
import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?,
    @ColumnInfo(name = "email") val email: String?,
    @ColumnInfo(name = "password") val password: String?,
    )
```

Create an UserDao interface:

```
package
com.example.expensestracker
import androidx.room.*
@Dao
interface UserDao {
    @Query("SELECT * FROM user_table WHERE email = :email")
    suspend fun getUserByEmail(email: String): User?
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)
    @Update
    suspend fun updateUser(user: User)
    @Delete
    suspend fun deleteUser(user: User)
}
```

Create an UserDatabase class:

```
package
com.example.expensestracker
import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao
    companion object {
        @Volatile
        private var instance: UserDatabase? = null
        fun getDatabase(context: Context): UserDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    UserDatabase::class.java,
                    "user_database"
                ).build()
                instance = newInstance
                newInstance
            }
        }
    }
}
```

Create an UserDatabaseHelper class:

```
package
com.example.expensestracker
import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
class UserDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null,
DATABASE_VERSION) {
    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "UserDatabase.db"
        private const val TABLE_NAME = "user_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_FIRST_NAME = "first_name"
        private const val COLUMN_LAST_NAME = "last_name"
        private const val COLUMN_EMAIL = "email"
        private const val COLUMN_PASSWORD = "password"
    }
    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "$COLUMN_FIRST_NAME TEXT, " +
                "$COLUMN_LAST_NAME TEXT, " +
                "$COLUMN_EMAIL TEXT, " +
                "$COLUMN_PASSWORD TEXT" +
                ")"
        db?.execSQL(createTable)
    }
    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion:
Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }
    fun insertUser(user: User) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }
    @SuppressLint("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME
WHERE $COLUMN_FIRST_NAME = ?", arrayOf(username))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getUserById(id: Int): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME
WHERE $COLUMN_ID = ?", arrayOf(id.toString()))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getAllUsers(): List<User> {
        val users = mutableListOf<User>()
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME",
null)
        if (cursor.moveToFirst()) {
            do {
                val user = User(
                    id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                    lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                    email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                    password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
                )
                users.add(user)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return users
    }
}
```

Create Items data class:

```
package
com.example.expensestracker
import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
@Entity(tableName = "items_table")
data class Items(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "item_name") val itemName: String?,
    @ColumnInfo(name = "quantity") val quantity: String?,
    @ColumnInfo(name = "cost") val cost: String?,
)
```

Create ItemsDao interface:

```
package
com.example.expensestracker
import androidx.room.*
@Dao
interface ItemsDao {
    @Query("SELECT * FROM items_table WHERE  cost= :cost")
    suspend fun getItemsByCost(cost: String): Items?
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertItems(items: Items)
    @Update
    suspend fun updateItems(items: Items)
    @Delete
    suspend fun deleteItems(items: Items)
}
```

Create ItemsDatabse class:

```
package
com.example.expensestracker
import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
@Database(entities = [Items::class], version = 1)
abstract class ItemsDatabase : RoomDatabase() {
    abstract fun ItemsDao(): ItemsDao
    companion object {
        @Volatile
        private var instance: ItemsDatabase? = null
        fun getDatabase(context: Context): ItemsDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    ItemsDatabase::class.java,
                    "items_database"
                ).build()
                instance = newInstance
                newInstance
            }
        }
    }
}
```

Create ItemsDatabaseHelper class:

```
package
com.example.expensestracker
import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
class ItemsDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME,
null,DATABASE_VERSION){
    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "ItemsDatabase.db"
        private const val TABLE_NAME = "items_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_ITEM_NAME = "item_name"
        private const val COLUMN_QUANTITY = "quantity"
        private const val COLUMN_COST = "cost"
    }
    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "${COLUMN_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "${COLUMN_ITEM_NAME} TEXT," +
                "${COLUMN_QUANTITY} TEXT," +
                "${COLUMN_COST} TEXT" +
                ")"
        db?.execSQL(createTable)
    }
    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion:
Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }
    fun insertItems(items: Items) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_ITEM_NAME, items.itemName)
        values.put(COLUMN_QUANTITY, items.quantity)
        values.put(COLUMN_COST, items.cost)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }
    @SuppressLint("Range")
    fun getItemsByCost(cost: String): Items? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME
WHERE $COLUMN_COST = ?", arrayOf(cost))
        var items: Items? = null
        if (cursor.moveToFirst()) {
            items = Items(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                itemName =
cursor.getString(cursor.getColumnIndex(COLUMN_ITEM_NAME)),
                quantity =
cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),
                cost = cursor.getString(cursor.getColumnIndex(COLUMN_COST)),
            )
        }
        cursor.close()
        db.close()
        return items
    }
    @SuppressLint("Range")
    fun getItemsById(id: Int): Items? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME
WHERE $COLUMN_ID = ?", arrayOf(id.toString()))
        var items: Items? = null
        if (cursor.moveToFirst()) {
            items = Items(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                itemName =
cursor.getString(cursor.getColumnIndex(COLUMN_ITEM_NAME)),
                quantity =
cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),
                cost = cursor.getString(cursor.getColumnIndex(COLUMN_COST)),
            )
        }
        cursor.close()
        db.close()
        return items
    }
    @SuppressLint("Range")
    fun getAllItems(): List<Items> {
        val item = mutableListOf<Items>()
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME",
null)
        if (cursor.moveToFirst()) {
            do {
                val items = Items(
                    id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    itemName =
cursor.getString(cursor.getColumnIndex(COLUMN_ITEM_NAME)),
                    quantity =
cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),
          cost = cursor.getString(cursor.getColumnIndex(COLUMN_COST)),
        )
        item.add(items)
      } while (cursor.moveToNext())
    }
    cursor.close()
    db.close()
    return item
  }
}
```

Create Expense data class:

```
package
com.example.expensestracker
import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
@Entity(tableName = "expense_table")
data class Expense(
  @PrimaryKey(autoGenerate = true) val id: Int?,
  @ColumnInfo(name = "amount") val amount: String?,
)
```

Create ExpenseDao interface:

```
package
com.example.expensestracker
import androidx.room.*
@Dao
interface ExpenseDao {
  @Query("SELECT * FROM expense_table WHERE  amount= :amount")
  suspend fun getExpenseByAmount(amount: String): Expense?
  @Insert(onConflict = OnConflictStrategy.REPLACE)
  suspend fun insertExpense(items: Expense)
  @Update
  suspend fun updateExpense(items: Expense)
  @Delete
  suspend fun deleteExpense(items: Expense)
}
```

Create ExpenseDatabase class:

```
package
com.example.expensestracker
import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
@Database(entities = [Items::class], version = 1)
abstract class ExpenseDatabase : RoomDatabase() {
    abstract fun ExpenseDao(): ItemsDao
    companion object {
        @Volatile
        private var instance: ExpenseDatabase? = null
        fun getDatabase(context: Context): ExpenseDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    ExpenseDatabase::class.java,
                    "expense_database"
                ).build()
                instance = newInstance
                newInstance
            }
        }
    }
}
```

Create ExpenseDatabaseHelper class:

```
package
com.example.expensestracker
import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
class ExpenseDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME,
null,DATABASE_VERSION){
    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "ExpenseDatabase.db"
        private const val TABLE_NAME = "expense_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_AMOUNT = "amount"
    }
    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "${COLUMN_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "${COLUMN_AMOUNT} TEXT" +
                ")"
        db?.execSQL(createTable)
    }
    override fun onUpgrade(db1: SQLiteDatabase?, oldVersion: Int, newVersion:
Int) {
        db1?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db1)
    }
    fun insertExpense(expense: Expense) {
        val db1 = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_AMOUNT, expense.amount)
        db1.insert(TABLE_NAME, null, values)
        db1.close()
    }
    fun updateExpense(expense: Expense) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_AMOUNT, expense.amount)
        db.update(TABLE_NAME, values, "$COLUMN_ID=?",
arrayOf(expense.id.toString()))
        db.close()
    }
    @SuppressLint("Range")
    fun getExpenseByAmount(amount: String): Expense? {
        val db1 = readableDatabase
        val cursor: Cursor = db1.rawQuery("SELECT * FROM
${ExpenseDatabaseHelper.TABLE_NAME} WHERE
${ExpenseDatabaseHelper.COLUMN_AMOUNT} = ?", arrayOf(amount))
        var expense: Expense? = null
        if (cursor.moveToFirst()) {
            expense = Expense(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                amount =
cursor.getString(cursor.getColumnIndex(COLUMN_AMOUNT)),
            )
        }
        cursor.close()
        db1.close()
        return expense
    }
    @SuppressLint("Range")
    fun getExpenseById(id: Int): Expense? {
        val db1 = readableDatabase
        val cursor: Cursor = db1.rawQuery("SELECT * FROM $TABLE_NAME
WHERE $COLUMN_ID = ?", arrayOf(id.toString()))
        var expense: Expense? = null
        if (cursor.moveToFirst()) {
            expense = Expense(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                amount =
cursor.getString(cursor.getColumnIndex(COLUMN_AMOUNT)),
            )
        }
        cursor.close()
        db1.close()
        return expense
    }
    @SuppressLint("Range")
    fun getExpenseAmount(id: Int): Int? {
        val db = readableDatabase
        val query = "SELECT $COLUMN_AMOUNT FROM $TABLE_NAME
WHERE $COLUMN_ID=?"
        val cursor = db.rawQuery(query, arrayOf(id.toString()))
        var amount: Int? = null
        if (cursor.moveToFirst()) {
            amount = cursor.getInt(cursor.getColumnIndex(COLUMN_AMOUNT))
        }
        cursor.close()
        db.close()
        return amount
    }
    @SuppressLint("Range")
    fun getAllExpense(): List<Expense> {
        val expenses = mutableListOf<Expense>()
        val db1 = readableDatabase
        val cursor: Cursor = db1.rawQuery("SELECT * FROM $TABLE_NAME",
null)
```

```
        if (cursor.moveToFirst()) {
            do {
                val expense = Expense(
                    id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    amount =
cursor.getString(cursor.getColumnIndex(COLUMN_AMOUNT)),
                )
                expenses.add(expense)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db1.close()
        return expenses
    }
}
```

Creating LoginActivity.kt with database:

```
package
com.example.expensestracker
import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.expensestracker.ui.theme.ExpensesTrackerTheme
```

```
class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            ExpensesTrackerTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    LoginScreen(this, databaseHelper)
                }
            }
        }
    }
}
@Composable
fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {
    Image(
        painterResource(id = R.drawable.img_1), contentDescription = "",
        alpha =0.3F,
        contentScale = ContentScale.FillHeight,
        )
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(
            fontSize = 36.sp,
            fontWeight = FontWeight.ExtraBold,
            fontFamily = FontFamily.Cursive,
            color = Color.White,
            text = "Login"
        )
        Spacer(modifier = Modifier.height(10.dp))
        TextField(
            value = username,
            onValueChange = { username = it },
            label = { Text("Username") },
            modifier = Modifier.padding(10.dp)
                .width(280.dp)
        )
        TextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Password") },
            modifier = Modifier.padding(10.dp)
                .width(280.dp),
            visualTransformation = PasswordVisualTransformation()
        )
        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }
        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty()) {
                    val user = databaseHelper.getUserByUsername(username)
                    if (user != null && user.password == password) {
                        error = "Successfully log in"
                        context.startActivity(
                            Intent(
                                context,
                                MainActivity::class.java
                            )
                        )
                        //onLoginSuccess()
                    }
                    else {
                        error =  "Invalid username or password"
                    }
                } else {
                    error = "Please fill all fields"
                }
            },
            modifier = Modifier.padding(top = 16.dp)
        ) {
```

```
            Text(text = "Login")
        }
        Row {
            TextButton(onClick = {context.startActivity(
                Intent(
                    context,
                    RegisterActivity::class.java
                )
            )}
            )
            { Text(color = Color.White,text = "Sign up") }
            TextButton(onClick = {
            })
            {
                Spacer(modifier = Modifier.width(60.dp))
                Text(color = Color.White,text = "Forget password?")
            }
        }
    }
}
private fun startMainPage(context: Context) {
    val intent = Intent(context, MainActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

Creating RegisterActivity.kt with database:

```
package
com.example.expensestracker
import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.expensestracker.ui.theme.ExpensesTrackerTheme
class RegisterActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            ExpensesTrackerTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    RegistrationScreen(this,databaseHelper)
                }
            }
        }
    }
}
@Composable
fun RegistrationScreen(context: Context, databaseHelper:
UserDatabaseHelper) {
    Image(
        painterResource(id = R.drawable.img_1), contentDescription = "",
        alpha =0.3F,
        contentScale = ContentScale.FillHeight,
        )
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var email by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
```

```
Text(
    fontSize = 36.sp,
    fontWeight = FontWeight.ExtraBold,
    fontFamily = FontFamily.Cursive,
    color = Color.White,
    text = "Register"
)
Spacer(modifier = Modifier.height(10.dp))
TextField(
    value = username,
    onValueChange = { username = it },
    label = { Text("Username") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)
)
TextField(
    value = email,
    onValueChange = { email = it },
    label = { Text("Email") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)
)
TextField(
    value = password,
    onValueChange = { password = it },
    label = { Text("Password") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp),
    visualTransformation = PasswordVisualTransformation()
)
if (error.isNotEmpty()) {
    Text(
        text = error,
        color = MaterialTheme.colors.error,
        modifier = Modifier.padding(vertical = 16.dp)
    )
}
Button(
    onClick = {
        if (username.isNotEmpty() && password.isNotEmpty() &&
email.isNotEmpty()) {
                    val user = User(
                        id = null,
                        firstName = username,
                        lastName = null,
                        email = email,
                        password = password
                    )
                    databaseHelper.insertUser(user)
                    error = "User registered successfully"
                    // Start LoginActivity using the current context
                    context.startActivity(
                        Intent(
                            context,
                            LoginActivity::class.java
                        )
                    )
                } else {
                    error = "Please fill all fields"
                }
            },
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Register")
        }
        Spacer(modifier = Modifier.width(10.dp))
        Spacer(modifier = Modifier.height(10.dp))
        Row() {
            Text(
                modifier = Modifier.padding(top = 14.dp), text = "Have an
account?"
            )
            TextButton(onClick = {
                context.startActivity(
                    Intent(
                        context,
                        LoginActivity::class.java
                    )
                )
            })
            {
                Spacer(modifier = Modifier.width(10.dp))
                Text(text = "Log in")
            }
        }
    }
}
private fun startLoginActivity(context: Context) {
    val intent = Intent(context, LoginActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

Creating MainActivity.kt file:

```
package
com.example.expensestracker
import android.annotation.SuppressLint
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.expensestracker.ui.theme.ExpensesTrackerTheme
class MainActivity : ComponentActivity() {
    @SuppressLint("UnusedMaterialScaffoldPaddingParameter")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Scaffold(
                // in scaffold we are specifying top bar.
                bottomBar = {
                    // inside top bar we are specifying
                    // background color.
                    BottomAppBar(backgroundColor = Color(0xFFadbef4),
                        modifier = Modifier.height(80.dp),
                        // along with that we are specifying
                        // title for our top bar.
                        content = {
                            Spacer(modifier = Modifier.width(15.dp))
                            Button(
                                onClick =
{startActivity(Intent(applicationContext,AddExpensesActivity::class.java))},
                                colors = ButtonDefaults.buttonColors(backgroundColor =
Color.White),
                                modifier = Modifier.size(height = 55.dp, width = 110.dp)
                            )
                            {
                                Text(
                                    text = "Add Expenses", color = Color.Black, fontSize =
14.sp,
                                    textAlign = TextAlign.Center
                                )
                            }
                            Spacer(modifier = Modifier.width(15.dp))

                            Button(
                                onClick = {
                                    startActivity(
                                        Intent(
                                            applicationContext,
                                            SetLimitActivity::class.java
                                        )
                                    )
                                },
                                colors = ButtonDefaults.buttonColors(backgroundColor =
Color.White),
                                modifier = Modifier.size(height = 55.dp, width = 110.dp)
                            )
                            {
                                Text(
                                    text = "Set Limit", color = Color.Black, fontSize = 14.sp,
                                    textAlign = TextAlign.Center
                                )
                            }
                            Spacer(modifier = Modifier.width(15.dp))
                            Button(
                                onClick = {
```

```
                        startActivity(
                            Intent(
                                applicationContext,
                                ViewRecordsActivity::class.java
                            )
                        )
                    },
                    colors = ButtonDefaults.buttonColors(backgroundColor =
Color.White),
                    modifier = Modifier.size(height = 55.dp, width = 110.dp)
                )
                {
                    Text(
                        text = "View Records", color = Color.Black, fontSize = 14.sp,
                        textAlign = TextAlign.Center
                    )
                }
            }
        )
    }
    ) {
        MainPage()
    }
  }
 }
}
@Composable
fun MainPage() {
    Column(
        modifier = Modifier.padding(20.dp).fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(text = "Welcome To Expense Tracker", fontSize = 42.sp, fontWeight =
FontWeight.Bold,
        textAlign = TextAlign.Center)
        Image(painterResource(id = R.drawable.img_1), contentDescription ="",
modifier = Modifier.size(height = 500.dp, width = 500.dp))
    }
}
```

Creating AddExpensesActivity.kt file:

```
package
com.example.expensestracker
import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
class AddExpensesActivity : ComponentActivity() {
    private lateinit var itemsDatabaseHelper: ItemsDatabaseHelper
    private lateinit var expenseDatabaseHelper: ExpenseDatabaseHelper
    @SuppressLint("UnusedMaterialScaffoldPaddingParameter")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        itemsDatabaseHelper = ItemsDatabaseHelper(this)
        expenseDatabaseHelper = ExpenseDatabaseHelper(this)
        setContent {
            Scaffold(
                // in scaffold we are specifying top bar.
                bottomBar = {
                    // inside top bar we are specifying
                    // background color.
                    BottomAppBar(backgroundColor = Color(0xFFadbef4),
                        modifier = Modifier.height(80.dp),
                        // along with that we are specifying
                        // title for our top bar.
                        content = {
                            Spacer(modifier = Modifier.width(15.dp))
                            Button(
                                onClick =
{startActivity(Intent(applicationContext,AddExpensesActivity::class.java))},
                                colors = ButtonDefaults.buttonColors(backgroundColor =
Color.White),
                                modifier = Modifier.size(height = 55.dp, width = 110.dp)
                            )
                            {
                                Text(
                                    text = "Add Expenses", color = Color.Black, fontSize =
14.sp,
                                    textAlign = TextAlign.Center
                                )
                            }
                            Spacer(modifier = Modifier.width(15.dp))
                            Button(
                                onClick = {
                                    startActivity(
                                        Intent(
                                            applicationContext,
                                            SetLimitActivity::class.java
                                        )
                                    )
                                },
                                colors = ButtonDefaults.buttonColors(backgroundColor =
Color.White),
                                modifier = Modifier.size(height = 55.dp, width = 110.dp)
                            )
                            {
                                Text(
                                    text = "Set Limit", color = Color.Black, fontSize = 14.sp,
                                    textAlign = TextAlign.Center
                                )
                            }
                            Spacer(modifier = Modifier.width(15.dp))
                            Button(
                                onClick = {
                                    startActivity(
                                        Intent(
                                            applicationContext,
                                            ViewRecordsActivity::class.java
                                        )
                                    )
                                },
                                colors = ButtonDefaults.buttonColors(backgroundColor =
Color.White),
                                        modifier = Modifier.size(height = 55.dp, width = 110.dp)
                                      )
                                      {
                                        Text(
                                          text = "View Records", color = Color.Black, fontSize = 14.sp,
                                          textAlign = TextAlign.Center
                                        )
                                      }
                                    }
                                  )
                                }
                              ) {
                                AddExpenses(this, itemsDatabaseHelper, expenseDatabaseHelper)
                              }
                            }
                          }
}
@SuppressLint("Range")
@Composable
fun AddExpenses(context: Context, itemsDatabaseHelper: ItemsDatabaseHelper,
expenseDatabaseHelper: ExpenseDatabaseHelper) {
                          Column(
                            modifier = Modifier
                              .padding(top = 100.dp, start = 30.dp)
                              .fillMaxHeight()
                              .fillMaxWidth(),
                            horizontalAlignment = Alignment.Start
                          ) {
                            val mContext = LocalContext.current
                            var items by remember { mutableStateOf("") }
                            var quantity by remember { mutableStateOf("") }
                            var cost by remember { mutableStateOf("") }
                            var error by remember { mutableStateOf("") }
                            Text(text = "Item Name", fontWeight = FontWeight.Bold, fontSize = 20.sp)
                            Spacer(modifier = Modifier.height(10.dp))
                            TextField(value = items, onValueChange = { items = it },
                              label = { Text(text = "Item Name") })
                            Spacer(modifier = Modifier.height(20.dp))
                            Text(text = "Quantity of item", fontWeight = FontWeight.Bold, fontSize =
20.sp)
                            Spacer(modifier = Modifier.height(10.dp))
                            TextField(value = quantity, onValueChange = { quantity = it },
                              label = { Text(text = "Quantity") })
```

```
        Spacer(modifier = Modifier.height(20.dp))
        Text(text = "Cost of the item", fontWeight = FontWeight.Bold, fontSize =
20.sp)
        Spacer(modifier = Modifier.height(10.dp))
        TextField(value = cost, onValueChange = { cost = it },
            label = { Text(text = "Cost") })
        Spacer(modifier = Modifier.height(20.dp))
        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }
        Button(onClick = {
            if (items.isNotEmpty() && quantity.isNotEmpty() && cost.isNotEmpty()) {
                val items = Items(
                    id = null,
                    itemName = items,
                    quantity = quantity,
                    cost = cost
                )
               val limit= expenseDatabaseHelper.getExpenseAmount(1)
                val actualvalue = limit?.minus(cost.toInt())
               // Toast.makeText(mContext, actualvalue.toString(),
Toast.LENGTH_SHORT).show()
                val expense = Expense(
                    id = 1,
                    amount = actualvalue.toString()
                )
                if (actualvalue != null) {
                    if (actualvalue < 1) {
                        Toast.makeText(mContext, "Limit Over",
Toast.LENGTH_SHORT).show()
                    } else  {
                        expenseDatabaseHelper.updateExpense(expense)
                        itemsDatabaseHelper.insertItems(items)
                    }
                }
            }
        }) {
            Text(text = "Submit")
        }
        }
    }
```

Creating SetLimitActivity.kt file:

```
package
com.example.expensestracker
import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.lazy.items
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.expensestracker.ui.theme.ExpensesTrackerTheme
class SetLimitActivity : ComponentActivity() {
    private lateinit var expenseDatabaseHelper: ExpenseDatabaseHelper
    @SuppressLint("UnusedMaterialScaffoldPaddingParameter")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        expenseDatabaseHelper = ExpenseDatabaseHelper(this)
        setContent {
            Scaffold(
                // in scaffold we are specifying top bar.
                bottomBar = {
                    // inside top bar we are specifying
                    // background color.
                    BottomAppBar(backgroundColor = Color(0xFFadbef4),
                        modifier = Modifier.height(80.dp),
                        // along with that we are specifying
```

```
                                // title for our top bar.
                                content = {
                                    Spacer(modifier = Modifier.width(15.dp))
                                    Button(
                                        onClick = {
                                            startActivity(
                                                Intent(
                                                    applicationContext,
                                                    AddExpensesActivity::class.java
                                                )
                                            )
                                        },
                                        colors = ButtonDefaults.buttonColors(backgroundColor =
                        Color.White),
                                        modifier = Modifier.size(height = 55.dp, width = 110.dp)
                                    )
                                    {
                                        Text(
                                            text = "Add Expenses", color = Color.Black, fontSize =
                        14.sp,
                                            textAlign = TextAlign.Center
                                        )
                                    }
                                    Spacer(modifier = Modifier.width(15.dp))
                                    Button(
                                        onClick = {
                                            startActivity(
                                                Intent(
                                                    applicationContext,
                                                    SetLimitActivity::class.java
                                                )
                                            )
                                        },
                                        colors = ButtonDefaults.buttonColors(backgroundColor =
                        Color.White),
                                        modifier = Modifier.size(height = 55.dp, width = 110.dp)
                                    )
                                    {
                                        Text(
                                            text = "Set Limit", color = Color.Black, fontSize = 14.sp,
                                            textAlign = TextAlign.Center
                                        )
                                    }
```

```
                            Spacer(modifier = Modifier.width(15.dp))
                            Button(
                                onClick = {
                                    startActivity(
                                        Intent(
                                            applicationContext,
                                            ViewRecordsActivity::class.java
                                        )
                                    )
                                },
                                colors = ButtonDefaults.buttonColors(backgroundColor =
Color.White),
                                modifier = Modifier.size(height = 55.dp, width = 110.dp)
                            )
                            {
                                Text(
                                    text = "View Records", color = Color.Black, fontSize =
14.sp,
                                    textAlign = TextAlign.Center
                                )
                            }
                        }
                    )
                }
            ) {
                val data=expenseDatabaseHelper.getAllExpense();
                Log.d("swathi" ,data.toString())
                val expense = expenseDatabaseHelper.getAllExpense()
                Limit(this, expenseDatabaseHelper,expense)
            }
        }
    }
}
@Composable
fun Limit(context: Context, expenseDatabaseHelper: ExpenseDatabaseHelper,
expense: List<Expense>) {
    Column(
        modifier = Modifier
            .padding(top = 100.dp, start = 30.dp)
            .fillMaxHeight()
            .fillMaxWidth(),
        horizontalAlignment = Alignment.Start
    ) {
```

```
    var amount by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }
    Text(text = "Monthly Amount Limit", fontWeight = FontWeight.Bold,
fontSize = 20.sp)
    Spacer(modifier = Modifier.height(10.dp))
    TextField(value = amount, onValueChange = { amount = it },
      label = { Text(text = "Set Amount Limit ") })
    Spacer(modifier = Modifier.height(20.dp))
    if (error.isNotEmpty()) {
      Text(
        text = error,
        color = MaterialTheme.colors.error,
        modifier = Modifier.padding(vertical = 16.dp)
      )
    }
    Button(onClick = {
      if (amount.isNotEmpty()) {
        val expense = Expense(
          id = null,
          amount = amount
        )
        expenseDatabaseHelper.insertExpense(expense)
      }
    }) {
      Text(text = "Set Limit")
    }
    Spacer(modifier = Modifier.height(10.dp))
    LazyRow(
      modifier = Modifier
        .fillMaxSize()
        .padding(top = 0.dp),
      horizontalArrangement = Arrangement.Start
    ) {
      item {
        LazyColumn {
          items(expense) { expense ->
            Column(
            ) {
              Text("Remaining Amount: ${expense.amount}", fontWeight =
FontWeight.Bold)
            }
          }
        }
```

```
            }
        }
    }
}
//@Composable
//fun Records(expense: List<Expense>) {
//    Text(text = "View Records", modifier = Modifier.padding(top = 24.dp, start =
106.dp, bottom = 24.dp ), fontSize = 30.sp)
//    Spacer(modifier = Modifier.height(30.dp))
//    LazyRow(
//        modifier = Modifier
//            .fillMaxSize()
//            .padding(top = 80.dp),
//
//        horizontalArrangement = Arrangement.SpaceBetween
//    ){
//        item {
//
//            LazyColumn {
//                items(expense) { expense ->
//                    Column(modifier = Modifier.padding(top = 16.dp, start = 48.dp,
bottom = 20.dp)) {
//                        Text("Remaining Amount: ${expense.amount}")
//                    }
//                }
//            }
//        }
//
//    }
//}
```

Creating ViewRecordsActivity.kt file:

```
package
com.example.expensestracker
import android.annotation.SuppressLint
import android.content.Intent
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.ScrollState
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.expensestracker.ui.theme.ExpensesTrackerTheme
class ViewRecordsActivity : ComponentActivity() {
    private lateinit var itemsDatabaseHelper: ItemsDatabaseHelper
    @SuppressLint("UnusedMaterialScaffoldPaddingParameter",
"SuspiciousIndentation")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        itemsDatabaseHelper = ItemsDatabaseHelper(this)
        setContent {
            Scaffold(
                // in scaffold we are specifying top bar.
                bottomBar = {
                    // inside top bar we are specifying
                    // background color.
                    BottomAppBar(backgroundColor = Color(0xFFadbef4),
                        modifier = Modifier.height(80.dp),
                        // along with that we are specifying
                        // title for our top bar.
                        content = {
                            Spacer(modifier = Modifier.width(15.dp))
                            Button(
                                onClick = {
                                    startActivity(
                                        Intent(
                                            applicationContext,
                                            AddExpensesActivity::class.java
                                        )
                                    )
                                },
                                colors = ButtonDefaults.buttonColors(backgroundColor
= Color.White),
                                        modifier = Modifier.size(height = 55.dp, width = 110.dp)
                                    )
                                    {
                                        Text(
                                            text = "Add Expenses", color = Color.Black, fontSize =
14.sp,
                                            textAlign = TextAlign.Center
                                        )
                                    }
                                    Spacer(modifier = Modifier.width(15.dp))
                                    Button(
                                        onClick = {
                                            startActivity(
                                                Intent(
                                                    applicationContext,
                                                    SetLimitActivity::class.java
                                                )
                                            )
                                        },
                                        colors = ButtonDefaults.buttonColors(backgroundColor
= Color.White),
                                        modifier = Modifier.size(height = 55.dp, width = 110.dp)
                                    )
                                    {
                                        Text(
                                            text = "Set Limit", color = Color.Black, fontSize =
14.sp,
                                            textAlign = TextAlign.Center
                                        )
                                    }
                                    Spacer(modifier = Modifier.width(15.dp))
                                    Button(
                                        onClick = {
                                            startActivity(
                                                Intent(
                                                    applicationContext,
                                                    ViewRecordsActivity::class.java
                                                )
                                            )
                                        },
                                        colors = ButtonDefaults.buttonColors(backgroundColor
= Color.White),
                                        modifier = Modifier.size(height = 55.dp, width = 110.dp)
```

```
                            )
                            {
                                Text(
                                    text = "View Records", color = Color.Black, fontSize =
14.sp,
                                    textAlign = TextAlign.Center
                                )
                            }
                        }
                    )
                }
            ) {
                val data=itemsDatabaseHelper.getAllItems();
                Log.d("swathi" ,data.toString())
                val items = itemsDatabaseHelper.getAllItems()
                    Records(items)
                }
            }
        }
    }
@Composable
fun Records(items: List<Items>) {
    Text(text = "View Records", modifier = Modifier.padding(top = 24.dp, start
= 106.dp, bottom = 24.dp ), fontSize = 30.sp, fontWeight = FontWeight.Bold)
    Spacer(modifier = Modifier.height(30.dp))
    LazyRow(
        modifier = Modifier
            .fillMaxSize()
            .padding(top = 80.dp),
        horizontalArrangement = Arrangement.SpaceBetween
    ){
        item {
            LazyColumn {
                items(items) { items ->
                    Column(modifier = Modifier.padding(top = 16.dp, start = 48.dp,
bottom = 20.dp)) {
                        Text("Item_Name: ${items.itemName}")
                        Text("Quantity: ${items.quantity}")
                        Text("Cost: ${items.cost}")
                    }
                }
            }
        }
```

```
                                        }
                                      }
```

Modifying AndroidManifest.xml:

```
<?xml
version="1.0"
encoding="utf-
8"?>
              <manifest xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:tools="http://schemas.android.com/tools">
                <application
                  android:allowBackup="true"
                  android:dataExtractionRules="@xml/data_extraction_rules"
                  android:fullBackupContent="@xml/backup_rules"
                  android:icon="@mipmap/ic_launcher"
                  android:label="@string/app_name"
                  android:supportsRtl="true"
                  android:theme="@style/Theme.ExpensesTracker"
                  tools:targetApi="31">
                  <activity
                    android:name=".RegisterActivity"
                    android:exported="false"
                    android:label="@string/title_activity_register"
                    android:theme="@style/Theme.ExpensesTracker" />
                  <activity
                    android:name=".MainActivity"
                    android:exported="false"
                    android:label="MainActivity"
                    android:theme="@style/Theme.ExpensesTracker" />
                  <activity
                    android:name=".ViewRecordsActivity"
                    android:exported="false"
                    android:label="@string/title_activity_view_records"
                    android:theme="@style/Theme.ExpensesTracker" />
                  <activity
                    android:name=".SetLimitActivity"
                    android:exported="false"
                    android:label="@string/title_activity_set_limit"
                    android:theme="@style/Theme.ExpensesTracker" />
                  <activity
                    android:name=".AddExpensesActivity"
                    android:exported="false"
```

```
            android:label="@string/title_activity_add_expenses"
            android:theme="@style/Theme.ExpensesTracker" />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.ExpensesTracker">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```