A report about the login system

APP name:***Intersteller***

Student ID:20213802071

Student Name:赖华溢

Completion time:2023/4/24

Catalog

1、 **Abstract** 3
2、 **Key code description** 3
2.1 **Data transition between pages(use of intent)** 3
2.2 **Build a database** 4
2.3 **Insert string in database** 5
2.4 **Query in database** 5
3、 **Implementation of tasks** 6
3.1 **Interfaces description** 6
3.1.1 **Intro interface** 6
3.1.2 **Login interface** 7
3.1.3 **Register interface** 8
3.1.4 **Welcome interface** 8
3.2 **Task implementation** 9
3.2.1 **TASK1** 9
3.2.2 **TASK2** 9
3.2.3 **TASK3** 9
3.2.4 **TASK4** 9
3.2.5 **TASK5** 10
4、 **Problems and difficulties** 10
4.1 **Problems and difficulties** 10
5、 **Shortcomings and future improvements** 10
5.1 **Shortcomings** 10
5.2 **Future improvements** 11
6、 **Appendix** 11

1、 Abstract

In this project, I mainly made 4 interfaces, which are **introduction interface, login page, registration page and welcome interface**, combined with SQLite database to build a complete login and registration system, combined with a good-looking interface design, which can provide users with a good using experience. Here you can see the structure of the four pages

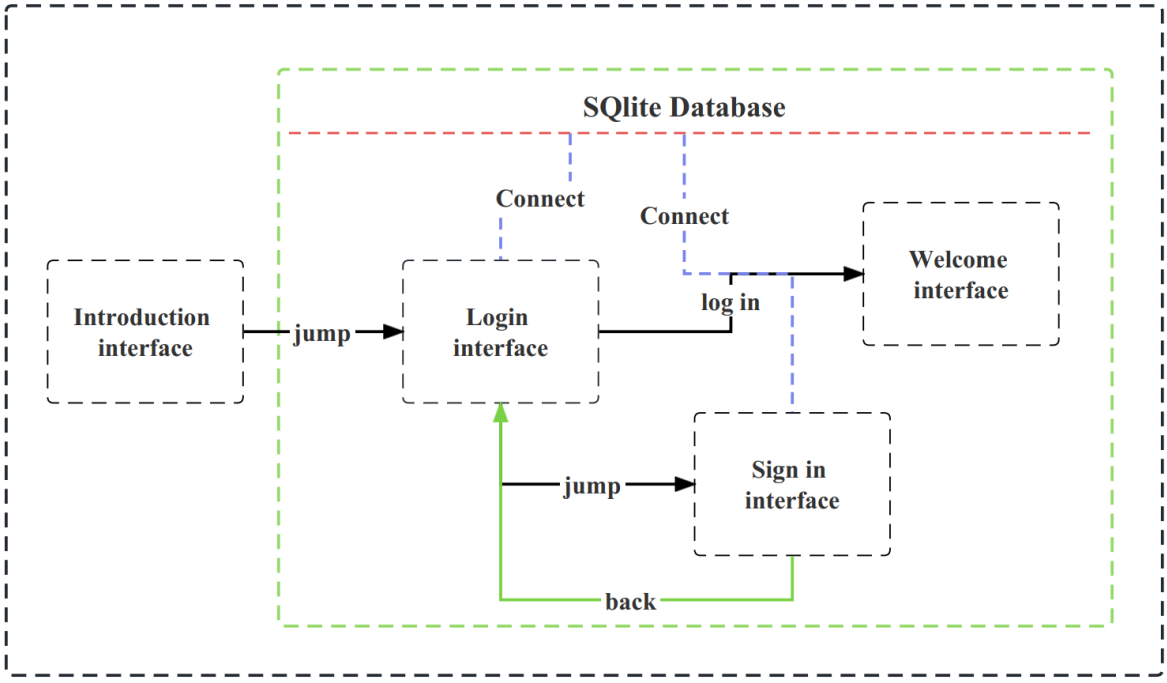

Figure 1:The structure of the total system

2、 Key code description

2. 1 Data transition between pages(use of intent)

Intent is a basic mechanism for passing messages and performing actions between different components of an application. Intent is a basic mechanism for passing messages and performing actions between different components of an application.

In this project, there are a total of four pages that need to implement jumping and information transfer between pages in the page jumping. First I need to create an Intent object, and then pass the name of the target activity into the Intent, and if I don't need to do data transfer, I can call the $startActivity()$ function directly. Since Intent is an asynchronous operation, there is no need to consider whether it will affect the execution of other d code. If I need to pass data between pages, I need to call the $putExtra()$ function, put the data I need to pass into the function, and finally accept it in the target activity. Here are two key code paragraphs.

Here are two code snippets to demonstrate the use of Intent in this project(Figure 2 and Figure 3)

```
val jumpButton3 = findViewById<ImageView>(R.id.back_login2)
jumpButton3.setOnClickListener { it: View!
    val intent = Intent( packageContext: this, Login::class.java)
    startActivity(intent)
}
```

Figure 2: click the ImgaeView to jump to login interface(no data transition)

```
val intent = Intent( packageContext: this, Welcome::class.java)
intent.putExtra(EXTRA_USERNAME, username)//put some data in intent
startActivity(intent)
```

```
val username = intent.getStringExtra(Login.EXTRA_USERNAME)//get data from intent
```

Figure 3:use intent to transfer data and jump to Welcome interface(including data transition)

2. 2 Build a database

In this project, I **use the SQlite database provided by AndroidStuido to store and read data.**

SQLite is a lightweight relational database management system (RDBMS). It is an embedded database that can be used without a separate server process and system. It is suitable for small applications and devices as it requires very few resources to run. In Android development, SQLite is a popular database management system that is widely used to store and manage local data.

The following steps(Figure 4,5,6) need to be followed when using SQLite to create a database:

1. Add SQLite dependencies: Add SQLite dependencies in the project's *build.gradle* file. You can use the following code to add SQLite dependencies:

```
    implementation 'androidx.sqlite:sqlite-ktx:2.2.0'
```

Figure 4: add dependency in *build.gradle*

2. Create a *SQLiteOpenHelper* class: Create a class that inherits from the *SQLiteOpenHelper* class. This class will be used to create the database and tables, and provide access to the database.
   It is obvious that the name of my database is" *myapp* **",the table I use is "users"**

```
class DBHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_VERSION) {

    companion object {
        const val DATABASE_NAME = "myapp.db"
        const val DATABASE_VERSION = 1

        const val TABLE_NAME = "users"
        const val COLUMN_ID = "_id"
        const val COLUMN_ACCOUNT = "account"
        const val COLUMN_PASSWORD = "password"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        //create database
        val createTable =
            "CREATE TABLE $TABLE_NAME ($COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, $COLUMN_ACCOUNT TEXT, $COLUMN_PASSWORD TEXT);"
        db?.execSQL(createTable)
    }
    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {
        // update the database
    }
}
```

Figure 5: create a class to inherits SQLiteOpenHelper

3. Create a database: I create a database instance in the application

```
dbHelper = DBHelper( context: this)//build the database
```

Figure 6: create an instance of the database

2. 3 Insert string in database

When implementing the registration function of the project, **I choose to use the data insertion method provided by SQLite to insert data into the database.**

First, use the is$NotBlank()$ method to check whether the account and password are both empty. Then, get a writable SQLite database instance db from the $dbHelper$ object, encapsulate the account number and password information into a $Content\ Values$ object values, and use the $insert()$ method to insert the data into the $DBHelper.TABLE\_NAME$ table. Finally, close the database connection and display a Toast prompt indicating successful registration.

```
if (account.isNotBlank() && password.isNotBlank()) {
    val db = dbHelper.writableDatabase
    val values = ContentValues().apply { this: ContentValues
        put(DBHelper.COLUMN_ACCOUNT, account)
        put(DBHelper.COLUMN_PASSWORD, password)
    }
    db.insert(DBHelper.TABLE_NAME, nullColumnHack: null, values)//insert data into the database
    db.close()
    Toast.makeText( context: this, text: "Registered successfully", Toast.LENGTH_SHORT).show()
} else {
    Toast.makeText( context: this, text: "Please enter account and password,both account and password should not be empty", Toast.LENGTH_SHORT).show()
}
```

Figure 7: insert data into the database

2. 4 Query in database

In the login interface, **I used the query function provided by SQlite to check whether the account number matches the page**, and combined with the query results to determine whether the login is successful.

Similar in Figure 6, First, I create a $DBHelper$ object, then, get a readable SQLite database instance from the $dbHelper$ object.

Then,in Figure 8, I use the $query()$ method to query from the SQLite database whether there is a record with the same account number and password entered by the user. Among them, $DBHelper.TABLE\_NAME$ is the table name, $DBHelper.COLUMN\_ACCOUNT$ and $DBHelper.COLUMN\_PASSWORD$ represent the column names of the account and password columns respectively. "${$ $DBHelper.COLUMN\_ACCOUNT$ $}$ = ? AND ${$ $DBHelper.COLUMN\_PASSWORD$ $}$ = ?" is the WHERE clause in the SQL query statement, used to specify the query conditions, $arrayOf(username, password)$ is an array, used to replace the WHERE clause The placeholder ? in the sentence.

```
val cursor = db.query(
    DBHelper.TABLE_NAME,
    arrayOf(DBHelper.COLUMN_ACCOUNT, DBHelper.COLUMN_PASSWORD),
    selection: "${DBHelper.COLUMN_ACCOUNT} = ? AND ${DBHelper.COLUMN_PASSWORD} = ?",
    arrayOf(username, password),
    groupBy: null,
    having: null,
    orderBy: null
)
```

Figure 8: query sentence in SQLite

Then, in Figure 9 check whether the number of records in the query result set is greater than 0 to determine whether the account and password entered by the user are valid. If the number of records is greater than 0, it means that the record with the same account number and password entered by the user has been queried, and return true at this time; otherwise, it means that no eligible records have been queried, and false is returned.

Finally, close the cursor and database connection, and return the judged value.

```
val isValid = cursor.count > 0

cursor.close()
db.close()

return isValid
```

Figure 9:end of the valid function

Eventually, here is the database that can be seen in AndroidStudio(Figure 10):

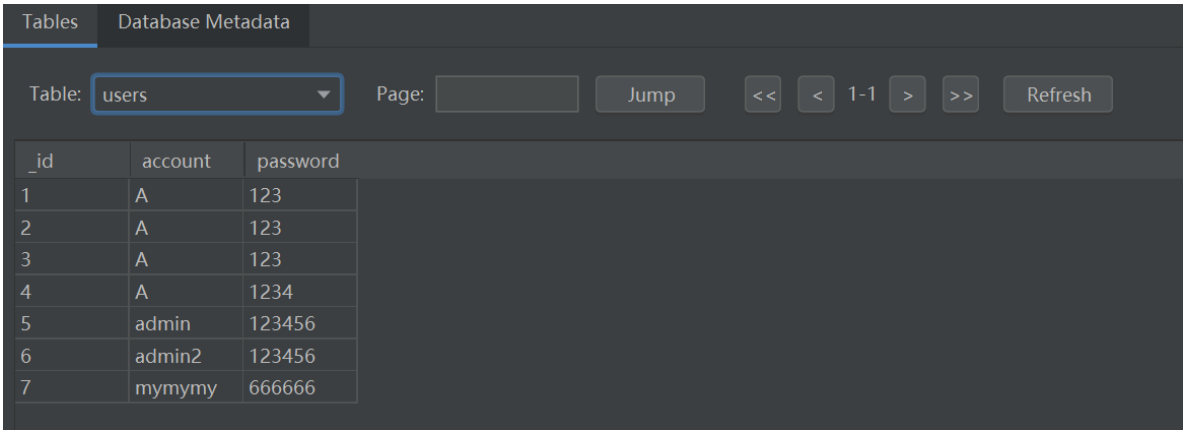

Figure 10: the database that I use

3、 Implementation of tasks

3. 1 Interfaces description

3. 1. 1 Intro interface

The background of this project is a game, this game is called ***"Interstellar "***, players can register and log in to the account. This game will provide a way to explore the planet,

and related functions will be improved in the future.

The following is the introduction page of the project. When the user opens the APP, he will enter this interface. In this interface**, I use two fonts. The main title uses "poppins", and the subtitle uses "***reemkufi***",** which are used in the APP title and title respectively. The following introduction. This introduction is a line from the movie "Interstellar", which is used to inspire players to explore unknown areas.

Figure 11: picture of introduction interface

After that, when you click the **purple button** in the middle of intro interface, you jump to the login interface

3. 1. 2 Login interface

In this interface, the user can click LOGIN to log in to the account, but if the user does not have an account, the user needs to click "Don't have an account?" or "Register" to register the account. The Attempts left below this interface also uses *poppins* font. Also contains security registration number 3. The user have two choice when he is in this interface, one is that he can log in and another is that he can jump to regiser.

The following figure(figure 12)shows the appearance of login interface.

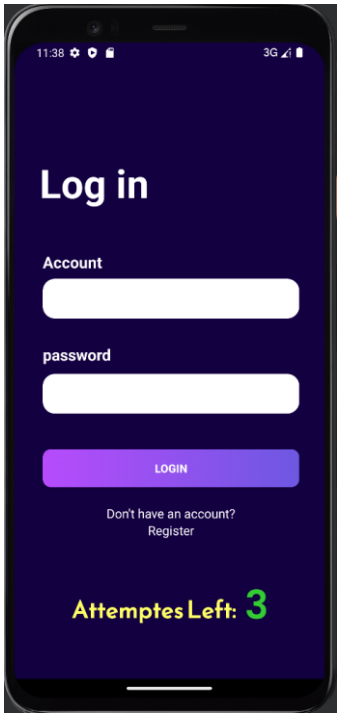

Figure 12: picture of log in interface

3. 1. 3 **Register interface**

If you do not have an account, you can enter the registration page for account registration, enter the user name and password, and click the REGISTER button to complete the account registration. After registration, you can **click the purple round button** below to go to the login interface to log in.

The following figure(figure 13)shows the appearance of login interface.

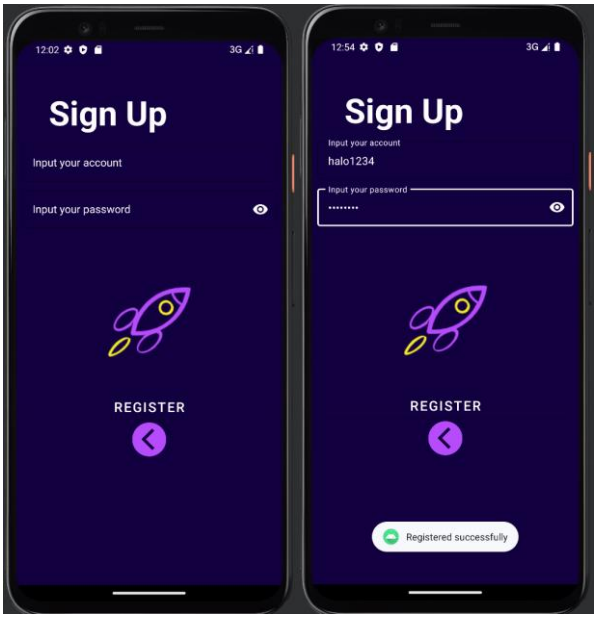

Figure 13: picture of sign in interface

3. 1. 4 **Welcome interface**

If you have entered the correct account and password, you can successfully log in to the

account and log in to the welcome page(Figure 14).

Figure 14: picture of sign in interface（username:admin password:123456）

3. 2 Task implementation

I have completed all the tasks successfully

3. 2. 1 TASK1

When you first open this project, there are no data in database, so if you click the login button, it will hint you that you have not input texts, it's time to register(Figure 17)

3. 2. 2 TASK2

Click the **register textview(whose text is *Don't have an account Register*)** and jump to the register interface

3. 2. 3 TASK3

In the register interface, instead of only let us click the register can we go back to the login interface, I made another button(**circle purple button at the bottom of register button**) to jump back to the login interface

3. 2. 4 TASK4

After you login, it will welcome you(**account:user1 password:user1**) in the welcome interface. (Figure 15)

Figure 15: the username is transmitted to welcome interface

3. 2. 5 **TASK5**

Every time you fail to log in, the number of left times will decrease, and the color will change at the same time(Figure 16).

Here are some details about it:

1、If you do not input your account or you input wrong accout, it will raise an alert

2、When you input wrong account or wrong password the left times will decrease to **2**,what's more, t**he color of the number 2 will change to orange.**

3、if you input wrong account and wrong password again, the times will decrease again, and **the color will jump from orange to red,** what's more, it will also alert you.

Figure 16: 3 kinds of color about the hint number

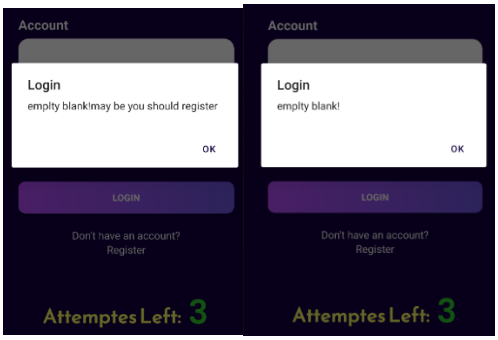

Figure 17: hints when you could not login successfully

4、 Problems and difficulties

4. 1 **Problems and difficulties**

1、When some packages are imported, they cannot be imported due to version reasons, and other methods must be found, sometimes I open AndroidStudio again to solve this problem

2、When running a virtual machine, the virtual machine sometimes crashes due to too many background apps. The solution is to **clean up** the background and re-run AndroidStudio

5、 Shortcomings and future improvements

5. 1 **Shortcomings**

1、There are no restrictions on user input during registration, which will result in irregular account formats

2、 The option to repeat the password is not provided when setting the password, which may cause users to remember the password they entered incorrectly

5. 2 **Future improvements**

1、Improve the user registration interface, provide repeated input units for the user's password, and help users remember the password

2、Add more animations for the jumps between pages to make the pages more lively

6、 **Appendix**

The relevant code can be seen in the following git repository:
https://github.com/rabbitrose/AndroidStudio_LoginHomework

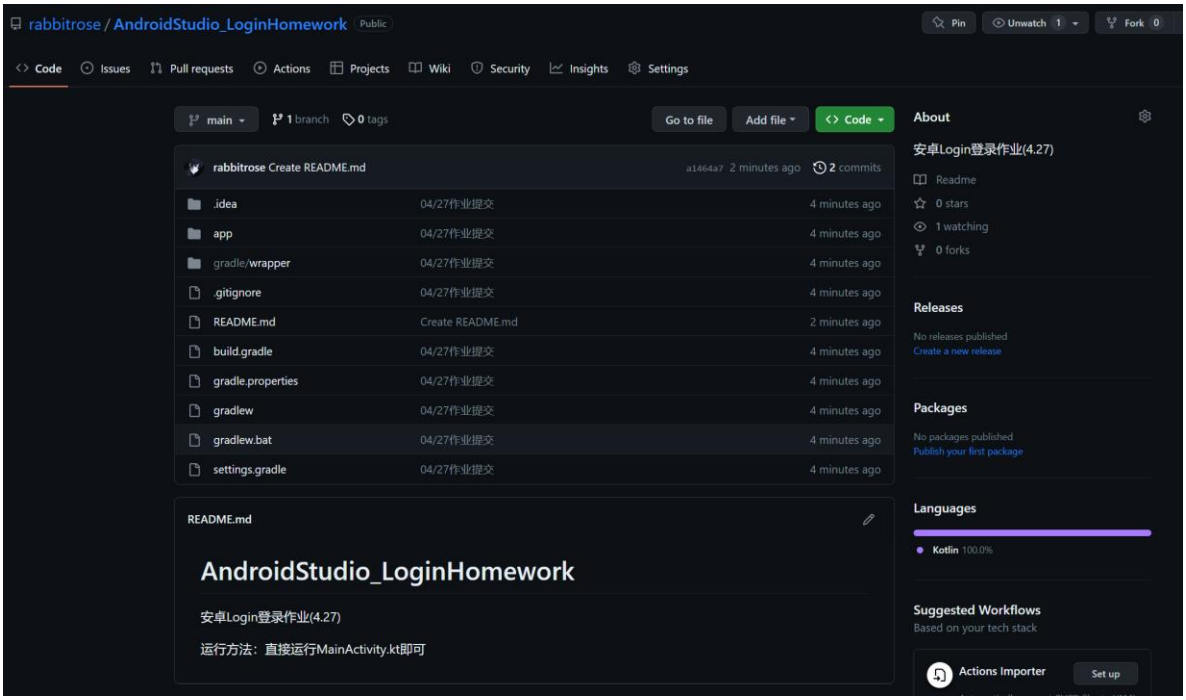

Figure 18: origin code for the homework