



# **ITCS424: Wireless and Mobile Computing**

**Dr. Dolvara Guna-Tilaka**

**Lect. Snit Sanghlao**

**Xuchen guo 6188137**

**Wei tong Deng 6188141**

**Faculty of Information and Communication Technology**

**Mahidol University 2021/1**

**January 4<sup>th</sup>, 2022**

# PROPOSAL

What is your project for? Project Name and Problem statement.

We are going to build a School Selector System; There are two login methods for teachers and students. The functions available after login are different learning pages. Teachers can write questions and modify student scores, and students can view the results and analyze the results and do the questions.

Developed for students who wish to enroll in the most suitable school, to help the school attract more students to be admitted to the school

What is value or contribution of the project?

1. There are two login methods for teachers and students, and the functions that can be used after login are different [Study] The page says that the teacher can write questions and modify students' scores, etc., and students can only view the results and analysis of the results and do the questions
2. There are certain restrictions when teachers register and log in (preparing the mobile phone number that can be registered in advance, the teacher will have more mobile phone SMS verification restrictions than students when registering), and then students need to perform identity authentication after logging in. After the authentication is successful, you can Enjoy all the features, otherwise only the [plan] page is available
3. [Learning] page has a graph analysis of students' grades, as well as all their grade queries (and the grades of the whole class), and then there will be a function that can do questions on this page (the teacher can make questions on this page, then students do the questions on this page)
4. [Plan] This page is for everyone to write down their own plans (can be set to be visible to themselves and to all people)
5. If the ability is met, improve the [Forum] page, connect to the cloud database, and everyone can post some content in it

## *School System*

How to finish before final examination, Grant Chart or Project Plan?

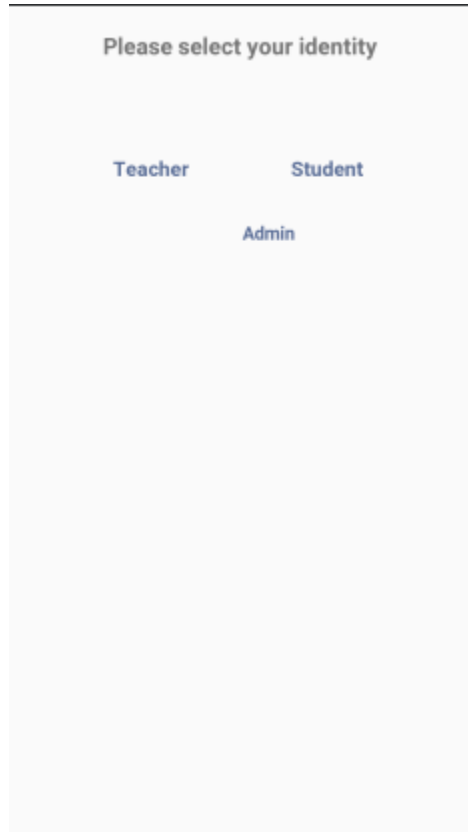
- ① Initial page
- ② Login page
- ③ Registration page
- ④ Forgot password page
- ⑤ The selection page and management page display of the administrator login successful

## *Interface and function part*

*The interface can be divided into initial page (welcome page), login page (administrator, teacher, student), registration page (teacher, student), administrator management page (class teacher, teacher, student, class), forgot password page, login The main page after success (divided into four: my, plan, study, forum)*

Of course, after clicking the corresponding function of each page, it will jump to a new page. we will show it by type:

### ***Initial page:***



Key code of the initial page Activity:

```
//Initial login interface, select the teacher, student or administrator
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_init);
```

```
    RelativeLayout chooseTeacher = (RelativeLayout) findViewById(R.id.ChooseTeacher);
```

```
    chooseTeacher.setOnClickListener(new JumpTeacher()); //Teacher login page jump
```

```
RelativeLayout chooseStudent = (RelativeLayout) findViewById(R.id.ChooseStudent);
chooseStudent.setOnClickListener(new JumpStudent()); //Student login page jump

RelativeLayout chooseAdmin = (RelativeLayout) findViewById(R.id.ChooseAdmin);
chooseAdmin.setOnClickListener(new JumpAdmin()); //Administrator page jump
}
```

```
private class JumpTeacher implements View.OnClickListener {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent();
        intent.setClass(InitActivity.this, TLoginActivity.class);
        startActivity(intent);
    }
}
```

```
private class JumpStudent implements View.OnClickListener {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent();
        intent.setClass(InitActivity.this, SLoginActivity.class);
        startActivity(intent);
    }
}
```

```
private class JumpAdmin implements View.OnClickListener {
    @Override
```

```

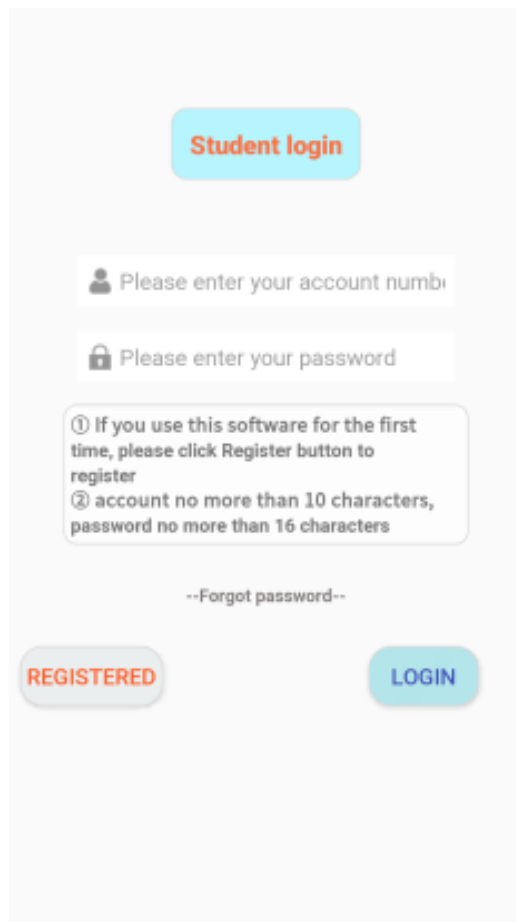
public void onClick(View view) {
    customClick(view); //Call custom Dialog
}
}

```

*As above, the role selection is actually monitored by Relative. In this way, users can monitor the jump when they click on the picture, text or the outer border. The following is the xml code in the middle of this page, because it was found during debugging. It can't adapt well on some phones with short screens, so we added Scroll View to make it slide up and down to enhance the adaptation effect of different models:*

## *log in page*

*Here we will introduce it with the student login page:*



The image shows a mobile app login screen for students. At the top, there is a light blue button with the text "Student login" in orange. Below this, there are two input fields: the first is preceded by a person icon and the text "Please enter your account number"; the second is preceded by a lock icon and the text "Please enter your password". Below the input fields is a rounded rectangle containing two lines of instructional text: "① If you use this software for the first time, please click Register button to register" and "② account no more than 10 characters, password no more than 16 characters". Underneath this box is a link that says "--Forgot password--". At the bottom of the screen, there are two buttons: a light blue button on the left with the text "REGISTERED" in orange, and a light blue button on the right with the text "LOGIN" in blue.

***The input box of the login page has monitoring and restrictions on the length of the input content. The account we set is up to 10 digits and the password is up to 16 digits. The digits here are closely related to registration. The monitoring and restriction methods are as follows:***

```
/**
 * Monitor the text in the account input box
 **/

private void nameContentListener() {
    usernameEditText2.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence charSequence, int start, int count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            Editable editable = usernameEditText2.getText();
            int len = editable.length();//The length of the input text
            if (len > 10) {
                int selEndIndex = Selection.getSelectionEnd(editable);
                String str = editable.toString();
                //Intercept the new string
                String newStr = str.substring(0, 10);
                usernameEditText2.setText(newStr);
                editable = usernameEditText2.getText();
                //New string length
                int newLen = editable.length();
                //The old cursor position exceeds the length of the new string
                if (selEndIndex > newLen) {
```

```

        selEndIndex = editable.length();
    }

    //Set the position of the new cursor
    Selection.setSelection(editable, selEndIndex);

    if (counter1 < 3) { //Nothing but three hahaha
        Toast.makeText(SLoginActivity.this, "The account number is up to 10 acridine!",
            Toast.LENGTH_SHORT).show();

        counter1++;
    }
}

@Override
public void afterTextChanged(Editable editable) {

}

});
}

/**
 * Monitor the text in the password input box
 */
private void passwordContentListener() {
    passwordEditText2.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence charSequence, int start, int count, int after) {

        }
    }

```



```

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {

    Editable editable = passwordEditText2.getText();

    int len = editable.length();//The length of the input text

    if (len> 16) {

        int selEndIndex = Selection.getSelectionEnd(editable);

        String str = editable.toString();

        String newStr = str.substring(0, 16);

        passwordEditText2.setText(newStr);

        editable = passwordEditText2.getText();

        int newLen = editable.length();

        if (selEndIndex> newLen) {

            selEndIndex = editable.length();

        }

        Selection.setSelection(editable, selEndIndex);

        if (counter2 <3) { //Nothing but three hahaha

            Toast.makeText(SLoginActivity.this, "The password is up to 16 digits!",
Toast.LENGTH_SHORT).show();

            counter2++;

        }

    }

}

@Override

public void afterTextChanged(Editable editable) {

}

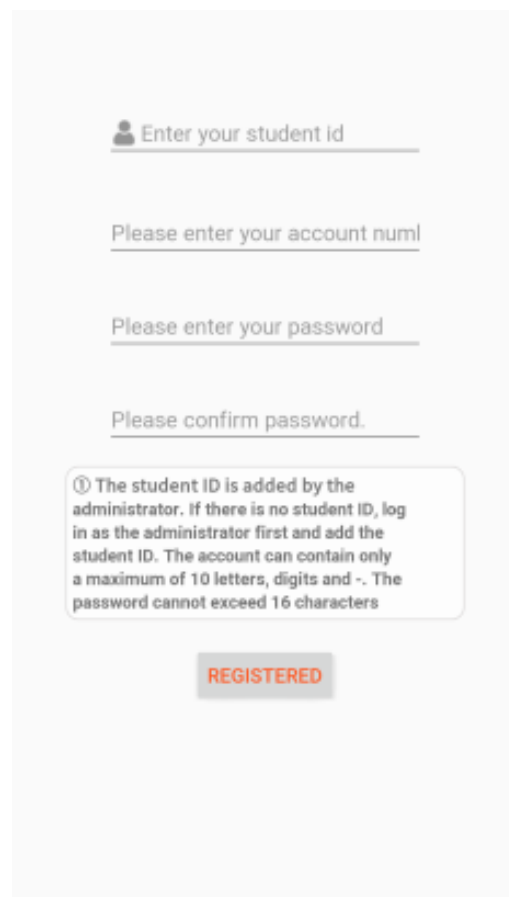
});

```

}

## ***registration page***

***The registration page on the teacher side and the student side is different. The teacher registers through the mobile phone number (send SMS verification code), and the students register through the student number:***



The registration form for students is displayed on a light gray background. It consists of four input fields, each with a placeholder text and a small icon on the left. The first field is for the student ID, the second for the account number, the third for the password, and the fourth for the password confirmation. Below the input fields is a rounded rectangular box containing a warning message. At the bottom of the form is a gray button with the text 'REGISTERED' in red.

Enter your student id

Please enter your account numl

Please enter your password

Please confirm password.

① The student ID is added by the administrator. If there is no student ID, log in as the administrator first and add the student ID. The account can contain only a maximum of 10 letters, digits and -. The password cannot exceed 16 characters

REGISTERED

*The filled phone number and student number are added at the same time when the user is added by the administrator. Therefore, here, the filled content will be queried and compared in the database. If not, a pop-up window will remind the user to bind it from the administrator first. Here takes the function of student registration as an example to show the java code:*

```
public class SRegisterActivity extends AppCompatActivity {

    private Button buttonRegister;

    private EditText editTextStudentNum, editTextUserName2, editTextPassword2,
    editTextConfirmPassword2;

    private String studentNum, userName2, password2, confirm2;

    private int counter1 = 0, counter2 = 0; //Limit the number of pop-up windows to prevent users
    from always not complying with the rules and pop-up windows constantly

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sregister);
        buttonRegister = findViewById(R.id.buttonRegister2);
        editTextStudentNum = findViewById(R.id.editStudentNum);
        editTextUserName2 = findViewById(R.id.add_username2);
        editTextPassword2 = findViewById(R.id.add_password2);
        editTextConfirmPassword2 = findViewById(R.id.confirm_password2);

        //Modify the font of the registration button
        Typeface customFont = Typeface.createFromAsset(this.getAssets(), "fonts/Coca-Cola.TTF");
        buttonRegister.setTypeface(customFont);

        nameCntentListener();
    }
}
```

```
passwordContentListener();
```

```
/**
```

```
 * register
```

```
 **/
```

```
buttonRegister.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View v) {
```

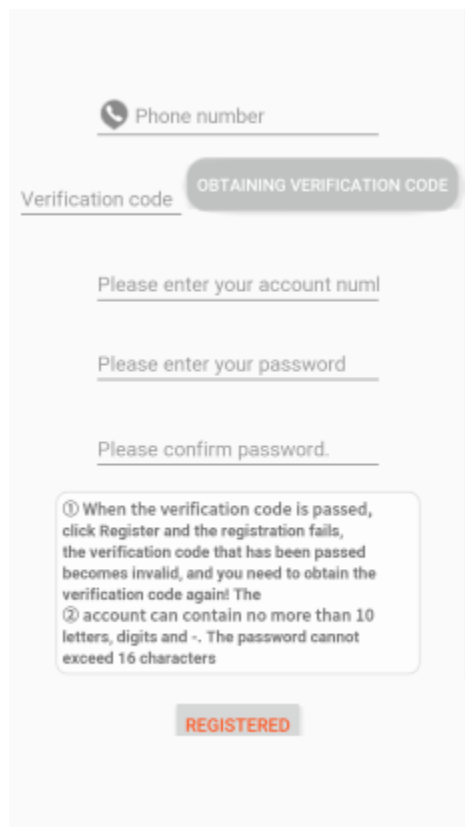
```
        studentNum = editTextStudentNum.getText().toString().trim();
```

```
        userName2 = editTextUserName2.getText().toString().trim();
```

```
        password2 = editTextPassword2.getText().toString().trim();
```

```
        confirm2 = editTextConfirmPassword2.getText().toString().trim();
```

## *Teacher registration*



The image shows a registration form for teachers. It includes a phone number input field with a phone icon, a verification code input field with a button labeled 'OBTAINING VERIFICATION CODE', and three password input fields with prompts: 'Please enter your account number', 'Please enter your password', and 'Please confirm password'. A text box contains two instructions: ① When the verification code is passed, click Register and the registration fails, the verification code that has been passed becomes invalid, and you need to obtain the verification code again! The ② account can contain no more than 10 letters, digits and -. The password cannot exceed 16 characters. At the bottom is a red button labeled 'REGISTERED'.

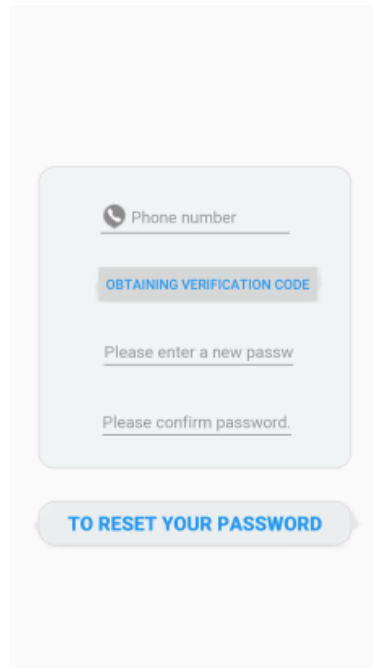
```
eh = new EventHandler() {  
    @Override  
    public void afterEvent(int event, int result, Object data) {  
        if (result == SMSSDK.RESULT_COMPLETE) {  
            //Callback completed  
            if (event == SMSSDK.EVENT_SUBMIT_VERIFICATION_CODE) {  
                //Submit the verification code successfully  
                runOnUiThread(new Runnable() {  
                    @Override  
                    public void run() {  
                        userName = editTextUserName.getText().toString().trim();
```

```
password = editTextPassword.getText().toString().trim();
phoneNum = editTextPhoneNum.getText().toString().trim();
confirm = editConfirmPassword.getText().toString().trim();

//If the username or password is empty
if (userName.equals("") || password.equals("")) {
    Toast.makeText(TRegisterActivity.this, "The account or password is empty,
please fill in again", Toast.LENGTH_SHORT).show();
} else {
    if (!LimitName.limitName(userName)) {
        Toast.makeText(TRegisterActivity.this, "The account only allows the
combination of English letters, numbers and -", Toast.LENGTH_SHORT).show();
    } else {
        nameCntentListener();
        passwordCntentListener();
        if (!password.equals(confirm)) { //The two entered passwords are inconsistent
            Toast.makeText(TRegisterActivity.this, "The password entered twice is
inconsistent, please re-enter", Toast.LENGTH_SHORT).show();
            editConfirmPassword.setText(""); //Clear the content of the input box
            editTextPassword.setText("");
        }
    }
}
```

***Forgot password page:***

***Student retrieve password:***



A screenshot of a 'Forgot password' form. The form is a light blue rounded rectangle centered on a light gray background. It contains a phone number input field with a telephone icon, a button labeled 'OBTAINING VERIFICATION CODE', two password input fields labeled 'Please enter a new password' and 'Please confirm password.', and a button labeled 'TO RESET YOUR PASSWORD'.

Phone number

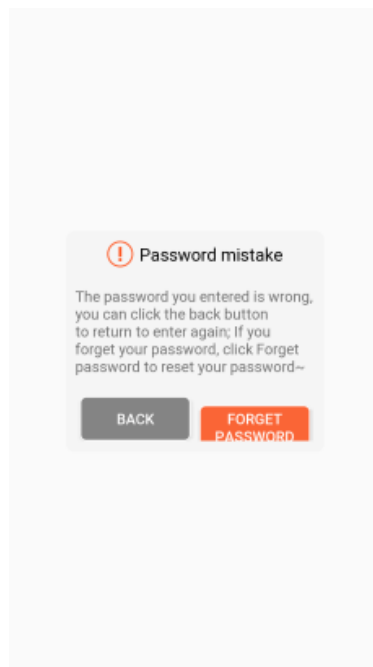
OBTAINING VERIFICATION CODE

Please enter a new password

Please confirm password.

TO RESET YOUR PASSWORD

***wrong password***



A screenshot of a 'Wrong password' error message. It features a red warning icon and the text 'Password mistake'. Below this, a message explains that the password is wrong and provides instructions on how to proceed. At the bottom, there are two buttons: 'BACK' and 'FORGET PASSWORD'.

! Password mistake

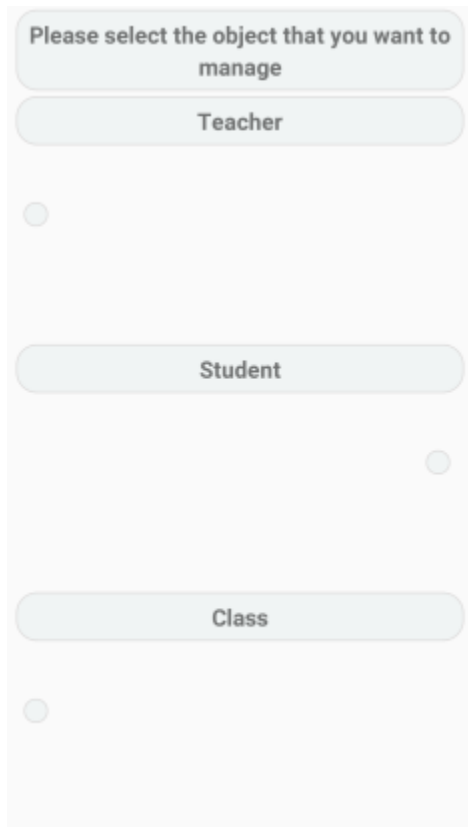
The password you entered is wrong, you can click the back button to return to enter again; If you forget your password, click Forget password to reset your password~

BACK FORGET PASSWORD

*This page can be directly clicked to jump from the [Login] page or my customized AlertDialog pops up because of the wrong password. If you select [Forgot Password], you will be redirected to this page*

### *Administrator page*

*Selection page and management page display for successful administrator login*



The image shows a vertical selection interface for an administrator. It features a light gray background with a thin black vertical line on the right side. At the top, a rounded rectangle contains the text "Please select the object that you want to manage". Below this, there are three main selection options, each consisting of a light gray rounded rectangle with a small circle to its left. The first option is labeled "Teacher" and has its circle selected. The second option is labeled "Student" and has its circle unselected. The third option is labeled "Class" and has its circle unselected.

Please select the object that you want to manage

Teacher

Student

Class



## *Manage student pages*



For the user management page, I use List View and SimpleCursorAdapter to achieve real-time refresh and display. When using these two, it is important to have "\_id integer primary key autoincrement" in the table.

```
private void initView() {  
    listView = (ListView) findViewById(R.id.studentListview);  
    btn_insert = (Button) findViewById(R.id.btn_insert);  
    btn_search = (Button) findViewById(R.id.btn_search);  
    et_ID = (EditText) findViewById(R.id.et_studentID);  
    et_name = (EditText) findViewById(R.id.et_studentName);  
    et_phone = (EditText) findViewById(R.id.et_studentPhone);  
    et_search = (EditText) findViewById(R.id.et_searchStudent);  
}
```

```
}
```

```
private void initEvent() {
```

```
    btn_insert.setOnClickListener(this);
```

```
    btn_search.setOnClickListener(this);
```

```
    openHelper = new DBOpenHelper(this);
```

```
    mDbWriter = openHelper.getWritableDatabase();
```

```
    mDbReader = openHelper.getReadableDatabase();
```

```
    simpleCursorAdapter = new SimpleCursorAdapter(StudentActivity.this, R.layout.user_item,  
    null,
```

```
        new String[]{"userID", "realName_u", "phoneNumber_u"}, new int[]{R.id.user_id,  
R.id.user_name, R.id.user_phone},  
CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
```

```
    listView.setAdapter(simpleCursorAdapter); //Set the adapter for ListView
```

```
    refreshListview(); //Custom method used to refresh ListView when the data list changes
```

```
}
```

```
//Refresh the data list
```

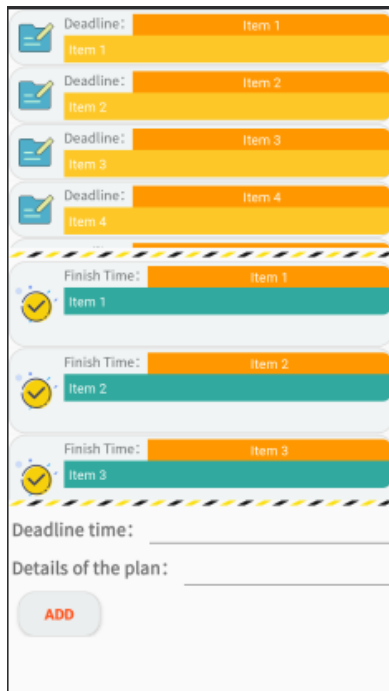
```
public void refreshListview() {
```

```
    Cursor cursor = mDbWriter.query(DBOpenHelper.USER_INFO, null, "identity=?", new  
String[]{"1"}, null, null, "userID");
```

```
    simpleCursorAdapter.changeCursor(cursor);
```

```
}
```

## *Planning interface*



The planning interface is a vertical form with a light gray background. It features a list of four items, each with a 'Deadline:' label and a yellow bar containing 'Item 1' through 'Item 4'. Below this is a section with a yellow and black striped border, containing three items with 'Finish Time:' labels and teal bars containing 'Item 1' through 'Item 3'. At the bottom, there are two text input fields labeled 'Deadline time:' and 'Details of the plan:', followed by a red 'ADD' button.

Deadline: Item 1  
Item 1

Deadline: Item 2  
Item 2

Deadline: Item 3  
Item 3

Deadline: Item 4  
Item 4

Finish Time: Item 1  
Item 1

Finish Time: Item 2  
Item 2

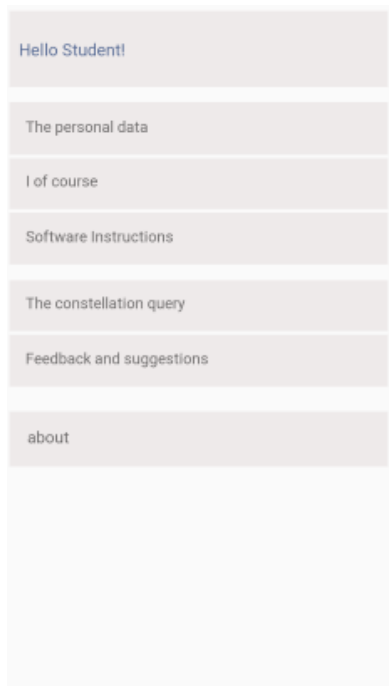
Finish Time: Item 3  
Item 3

Deadline time: \_\_\_\_\_

Details of the plan: \_\_\_\_\_

ADD

## *Personal interface*



The personal interface is a vertical list of seven light gray rectangular buttons. The buttons contain the following text from top to bottom: 'Hello Student!', 'The personal data', 'I of course', 'Software Instructions', 'The constellation query', 'Feedback and suggestions', and 'about'.

Hello Student!

The personal data

I of course

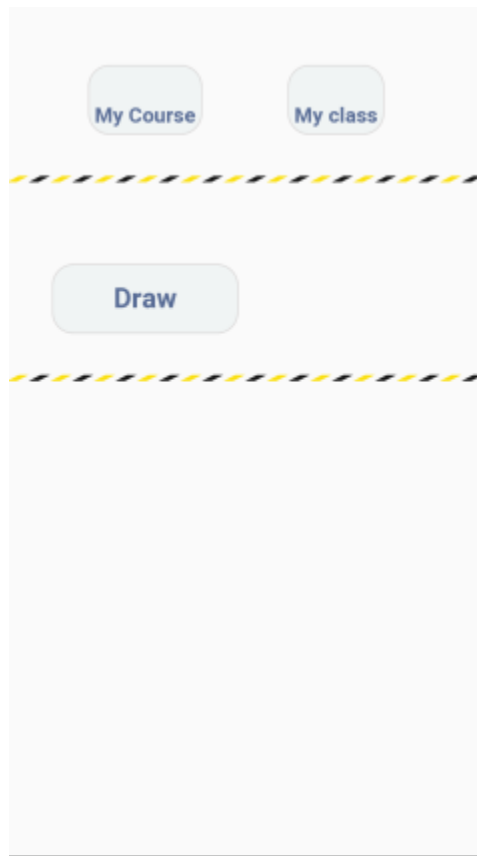
Software Instructions

The constellation query

Feedback and suggestions

about

## *Learning interface*



*Functionally, a listener is added to each Image Button on the right, as follows:*

*//Set up monitoring*

*ib\_nickName.setOnClickListener(new EditNickName()); //nickname editing*

*ib\_sex.setOnClickListener(new EditSex()); //Sex edit*

*ib\_class.setOnClickListener(new EditClass()); //Class edit*

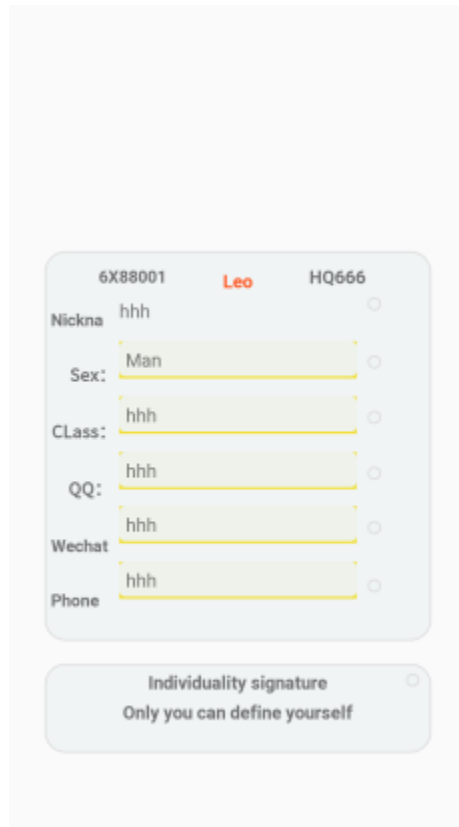
*ib\_qq.setOnClickListener(new EditQQ()); //qq edit*

*ib\_wechat.setOnClickListener(new EditWechat()); //Wechat edit*

*ib\_phoneNumber.setOnClickListener(new EditPhoneNumber()); //phone number edit*

***ib\_motto.setOnClickListener(new EditMotto()); //signature editing***

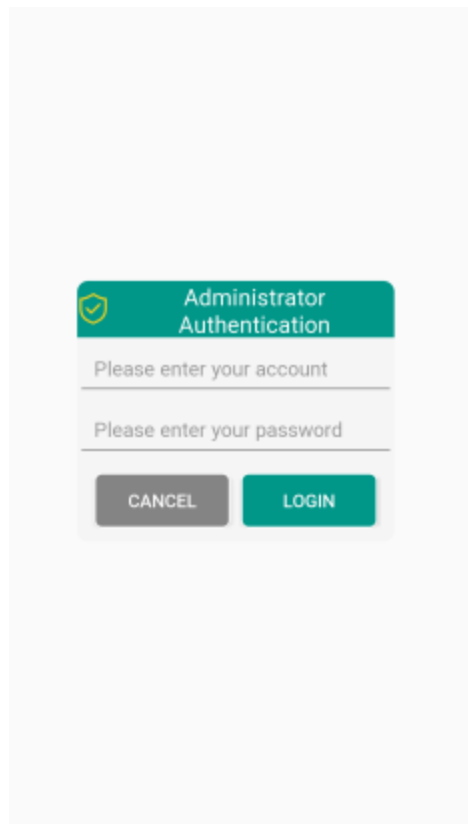
***Profile page:***



A screenshot of a user profile editing interface. At the top, there are three fields: '6X88001', 'Leo' (in red), and 'HQ666'. Below these are several input fields with labels on the left and radio buttons on the right. The labels are: 'Nickna', 'Sex:', 'CLass:', 'QQ:', 'Wechat', and 'Phone'. Each label has a corresponding input field containing 'hhh' or 'Man'. Below these is a section titled 'Individuality signature' with the text 'Only you can define yourself' and a radio button.

Field	Value	Radio Button
6X88001	6X88001	
Leo	Leo	
HQ666	HQ666	
Nickna	hhh	
Sex:	Man	
CLass:	hhh	
QQ:	hhh	
Wechat	hhh	
Phone	hhh	
Individuality signature	Only you can define yourself	

***Administrator verification:***



```
/**
```

```
* Custom login dialog
```

```
*/
```

```
private void customClick(View v) {
```

```
    AlertDialog.Builder builder = new AlertDialog.Builder(InitActivity.this);
```

```
    final AlertDialog dialog = builder.create();
```

```
    final View dialogView = View.inflate(InitActivity.this, R.layout.dialog_admin_login, null);
```

```
    //Set the dialog layout
```

```
    dialog.setView(dialogView);
```

```
    dialog.show();
```

```
    dialog.getWindow().setBackgroundDrawable(null);
```

```
    final Button btnLogin = (Button) dialogView.findViewById(R.id.btn_login);
```

```
    final Button btnCancel = (Button) dialogView.findViewById(R.id.btn_cancel);
```

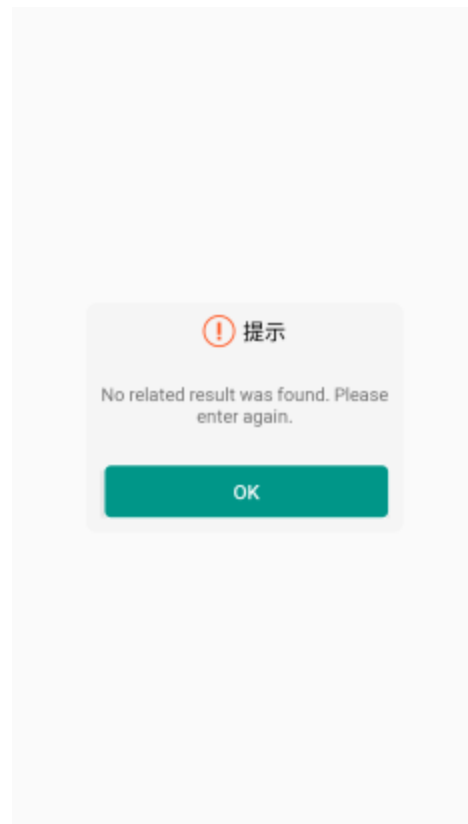
```

btnLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        et_name = (EditText) dialogView.findViewById(R.id.et_name);
        et_pwd = (EditText) dialogView.findViewById(R.id.et_pwd);
        String name = et_name.getText().toString().trim();
        String pwd = et_pwd.getText().toString().trim();
        if (name.equals("") || pwd.equals("")) {
            Toast.makeText(InitActivity.this, "The account and password cannot be empty, please fill in again", Toast.LENGTH_SHORT).show();
        } else {
            if (name.equals("Henry")) {
                if (pwd.equals("666666")) {
                    Intent intent = new Intent();
                    intent.setClass(InitActivity.this, AdminActivity.class);
                    startActivity(intent);
                    dialog.dismiss();
                } else {
                    Toast.makeText(InitActivity.this, "The password is incorrect, please fill in again", Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(InitActivity.this, "The account number is incorrect, please fill in again", Toast.LENGTH_SHORT).show();
            }
        }
    }
});

```

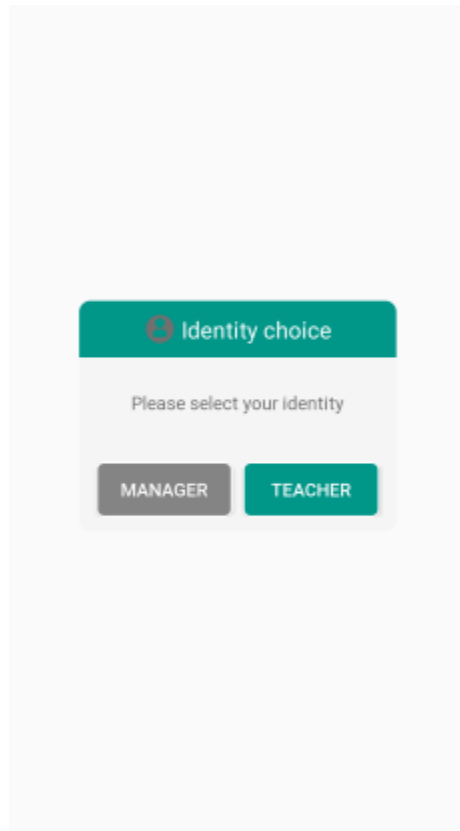
```
btnCancel.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        dialog.dismiss();  
    }  
});  
}
```

***Prompt page:***





### *Identity selection:*



```
private void selectClick(View v) {  
    AlertDialog.Builder builder = new AlertDialog.Builder(AdminActivity.this);  
    final AlertDialog dialog = builder.create();  
    final View dialogView = View.inflate(AdminActivity.this, R.layout.dialog_admin_teacher,  
null);  
    //Set the dialog layout  
    dialog.setView(dialogView);  
    dialog.show();  
    dialog.getWindow().setBackgroundDrawable(null);  
    final Button btnHead = (Button) dialogView.findViewById(R.id.btn_selectHead);  
    final Button btnTeacher = (Button) dialogView.findViewById(R.id.btn_selectTeacher);  
}
```

```

btnHead.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent();
        intent.setClass(AdminActivity.this, TeacherActivity.class);
        startActivity(intent);
        dialog.dismiss();
    }
});

btnTeacher.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent();
        intent.setClass(AdminActivity.this, Teacher2Activity.class);
        startActivity(intent);
        dialog.dismiss();
    }
});
}

```

***To sum it up, it is actually equivalent to designing this Dialog like writing an interface layout, but its size is relatively small, and attention should be paid to the matching of its content, components and Alert Dialog. You can't design it as you want. What kind, follow the method of Alert Dialog***