ILLINOIS INSTITUTE OF TECHNOLOGY

Illinois Institute of Technology

Department of Computer Science

CS487 - Section: 02

Software Engineering

Team Final Report

Authors:	*Student Number:*
Xiaoqian YANG	A20525824
Hongbo WANG	A20524770
Yiqi WANG	A20519771
Jackie MCANINCH	A20436746
Khiem TRUONG	A20437990

April 28, 2023

1 Updates on Requirements and Design

Based on the user feedback, we determined the update requirements for the Student Assist App. The updated requirements include the additional UI interface, system performance improvements, and security policy.

1.1 System Performance Improvements

As the system performance improved, we refactored our whole system. We renamed variables, splatted large functions into smaller ones, removed duplicated code, and improved the code's overall design.

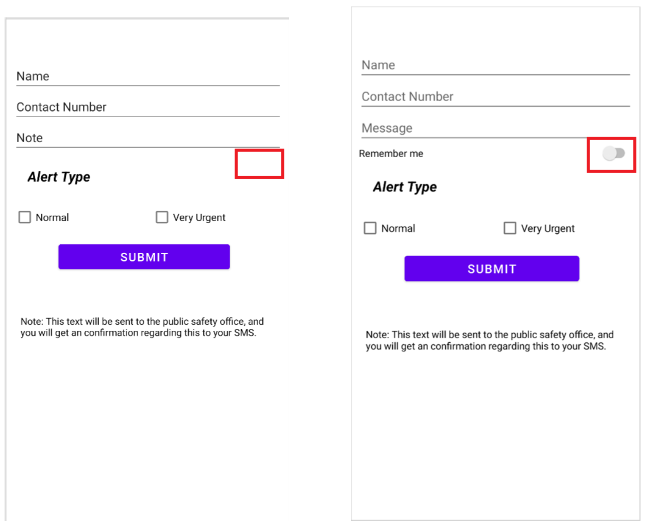
1.2 Additional User Interface

In this part, we create more user interfaces and optimized our old previous interface.

1.2.1 User Interface Update

(1) Campus Safety Page: We collected the result from the user and found out that people didn't like to enter the detail message every time since the user may be in an urgent situation and have no time to fill in the information.

Therefore, we add the "Remember me" switch to the campus safety page.



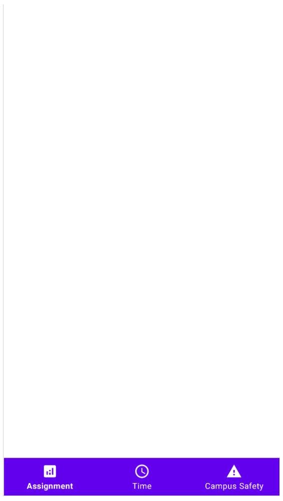
Note: This text will be sent to the public safety office, and you will get an confirmation regarding this to your SMS.

Note: This text will be sent to the public safety office, and you will get an confirmation regarding this to your SMS.

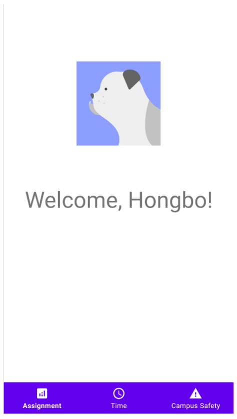
(a) old interface (b) new interface

Figure 1: campus safety interface

(2) Main Page: User feedback on the main page is too monotonous. So, we add the user icon and user name to the main page.



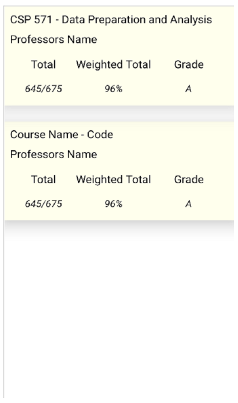
(a) old interface



(b) new interface

Figure 2: main page interface

(3) Assignment Page: The user feedback information displayed on the page is too sparse. So, we add a button on each course and highlight the course which might fail.



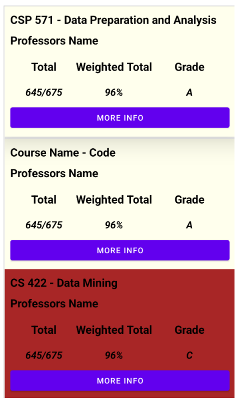
CSP 571 - Data Preparation and Analysis
Professors Name

Total	Weighted Total	Grade
645/675	96%	A

Course Name - Code
Professors Name

Total	Weighted Total	Grade
645/675	96%	A

(a) old interface



CSP 571 - Data Preparation and Analysis
Professors Name

Total	Weighted Total	Grade
645/675	96%	A

MORE INFO

Course Name - Code
Professors Name

Total	Weighted Total	Grade
645/675	96%	A

MORE INFO

CS 422 - Data Mining
Professors Name

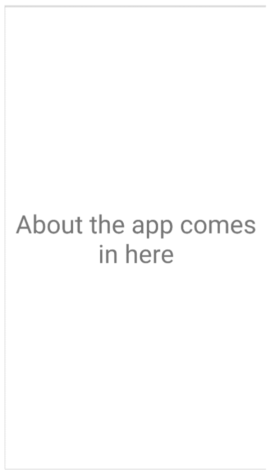
Total	Weighted Total	Grade
645/675	96%	C

MORE INFO

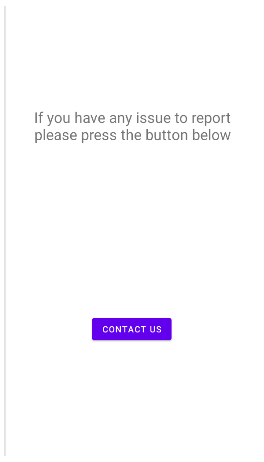
(b) new interface

Figure 3: assignment page interface

(4) About Page: We complete the About page with a button under the text.



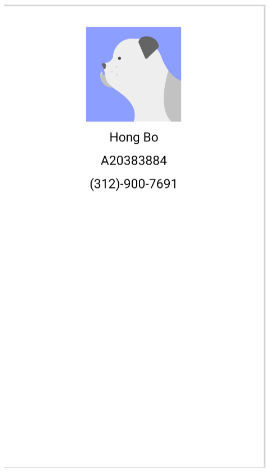
(a) old interface


If you have any issue to report please press the button below
CONTACT US

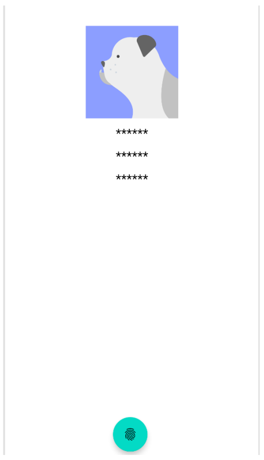
(b) new interface

Figure 4: about page interface

(5) Profile Page: We have added a user information verification page before jumping to the user profile page



(a) old interface



(b) new interface

Figure 5: profile page interface

1.2.2 New User Interface

(1) Start Page: Before the main page, we added a start page to verify user identity.

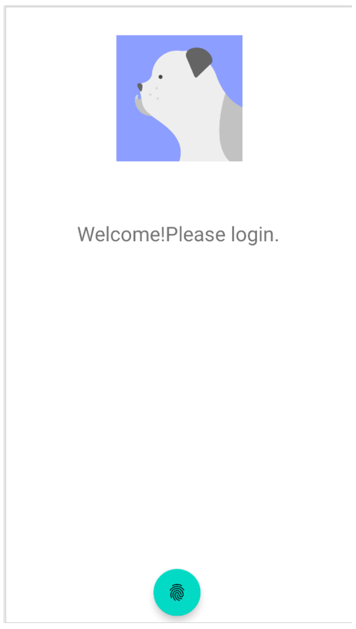
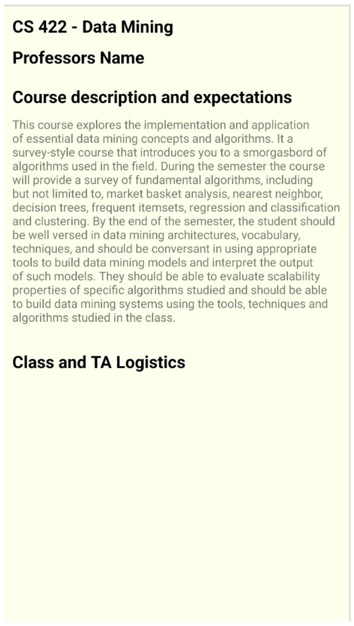


Figure 6: add a start page

(2) Assignment Detail Page: When the user press the more detail button on the assignment page, it will jump to the assignment detail page. It will show more information about the course.



CS 422 - Data Mining

Professors Name

Course description and expectations

This course explores the implementation and application of essential data mining concepts and algorithms. It a survey-style course that introduces you to a smorgasbord of algorithms used in the field. During the semester the course will provide a survey of fundamental algorithms, including but not limited to, market basket analysis, nearest neighbor, decision trees, frequent itemsets, regression and classification and clustering. By the end of the semester, the student should be well versed in data mining architectures, vocabulary, techniques, and should be conversant in using appropriate tools to build data mining models and interpret the output of such models. They should be able to evaluate scalability properties of specific algorithms studied and should be able to build data mining systems using the tools, techniques and algorithms studied in the class.

Class and TA Logistics

Figure 7: add an assignment detail page

1.3 security policy

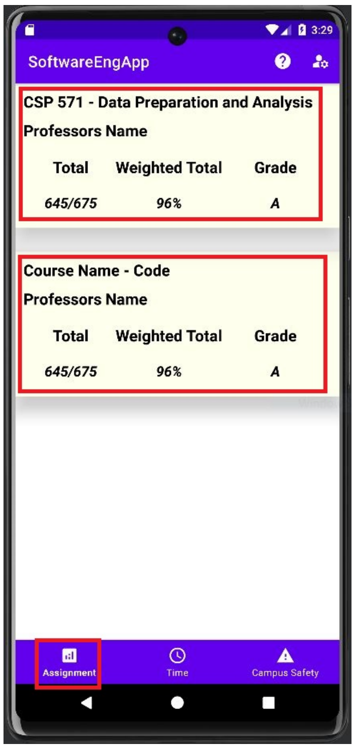
(2) Assignment Detail Page: As the most important part, we have upgraded the security level of the program. When the user is trying to get the sensitive data, the program will identify the user's identity every time.

2 Test Results

In the APP, we have a total of four parts (assignment, time, and campus safety), and we have tested the buttons and displays in them. In the process of testing, the text can be displayed correctly in the text box, and the button will not cause excessive system usage or system crashes during repeated tests.

2.1 Assignment Function

In this part, there are two text boxes and one button are tested.



CSP 571 - Data Preparation and Analysis
Professors Name

Total	Weighted Total	Grade
645/675	96%	A

Course Name - Code
Professors Name

Total	Weighted Total	Grade
645/675	96%	A

Figure 8: Assignment management interface testing area

2.2 Time Function

In this part, there is one text box and three buttons are tested.

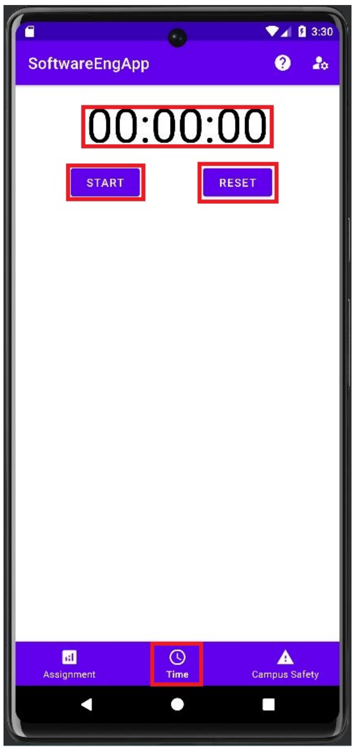
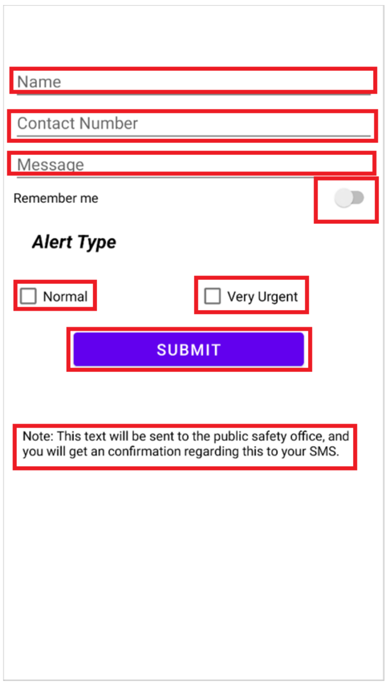


Figure 9: time management interface testing area

2.3 Campus Safety Function

In this part, there are three text boxes, two check-boxes, one button, and one text box are tested.



Note: This text will be sent to the public safety office, and you will get an confirmation regarding this to your SMS.

Figure 10: campus safety interface testing area

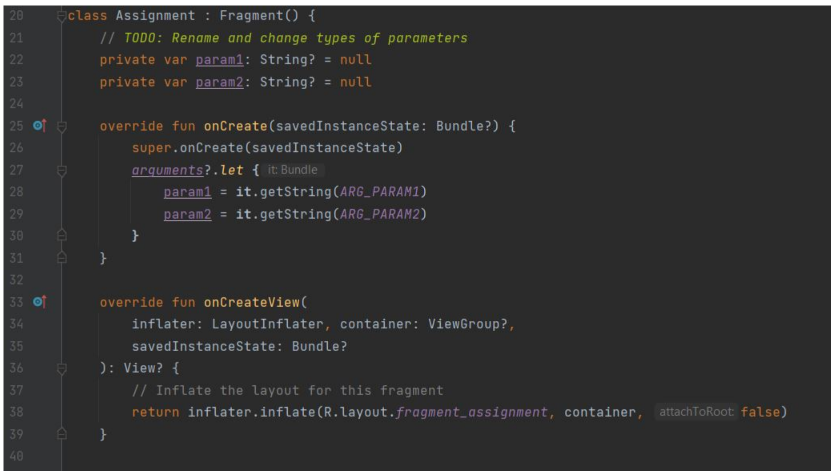
3 Core Code Fragment

In this part, we will introduce some core codes, such as the processing of click time, processing of input events, and how to realize some basic display information.

3.1 Assignment Function

3.1.1 Click Event

The processing function and code when a click event occurs on the assignment management interface.



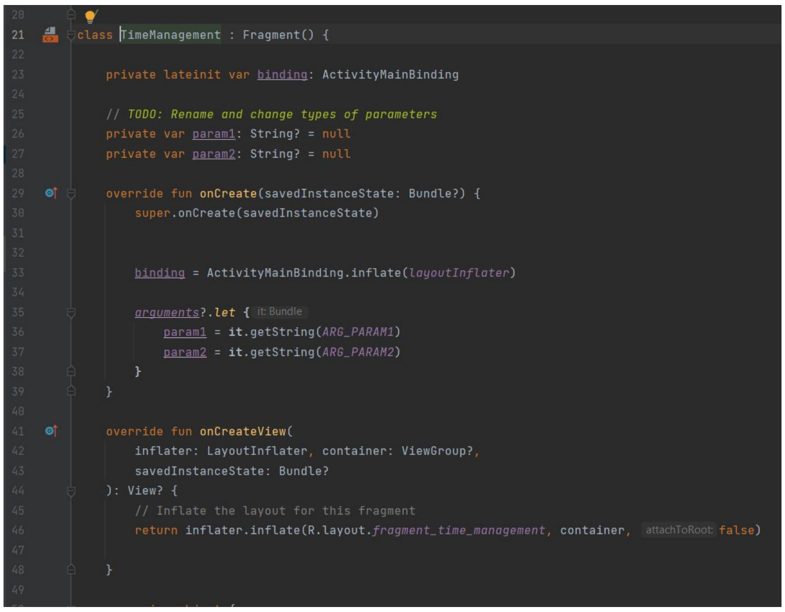
```kotlin
class Assignment : Fragment() {
    // TODO: Rename and change types of parameters
    private var param1: String? = null
    private var param2: String? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        arguments?.let { it: Bundle
            param1 = it.getString(ARG_PARAM1)
            param2 = it.getString(ARG_PARAM2)
        }
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_assignment, container, attachToRoot: false)
    }

```

Figure 11: click event code fragment

3.2 Time Function

3.2.1 Click Event

The processing function and code when a click event occurs on the time management interface.

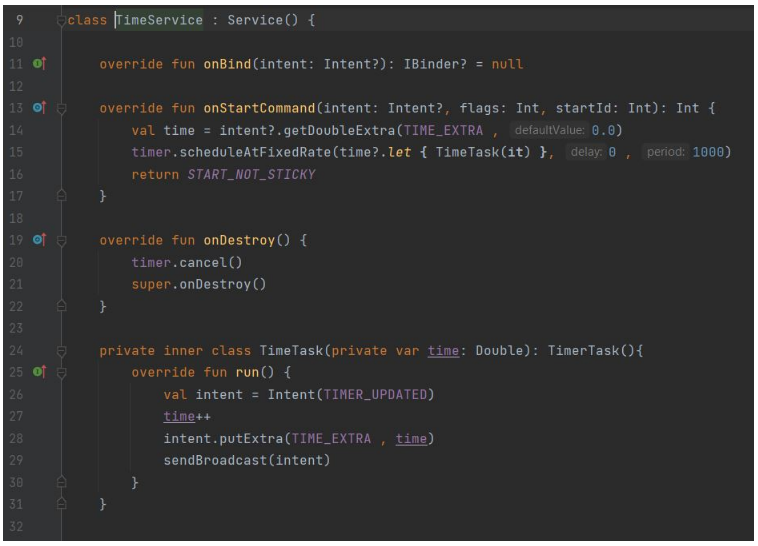


```
class TimeManagement : Fragment() {

    private lateinit var binding: ActivityMainBinding

    // TODO: Rename and change types of parameters
    private var param1: String? = null
    private var param2: String? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)


        binding = ActivityMainBinding.inflate(layoutInflater)

        arguments?.let { it: Bundle
            param1 = it.getString(ARG_PARAM1)
            param2 = it.getString(ARG_PARAM2)
        }
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_time_management, container, attachToRoot: false)

    }

```

Figure 12: click event code fragment

3.2.2 Calculate Time Function

The function and code snippet of how the time counts when the user clicks the button



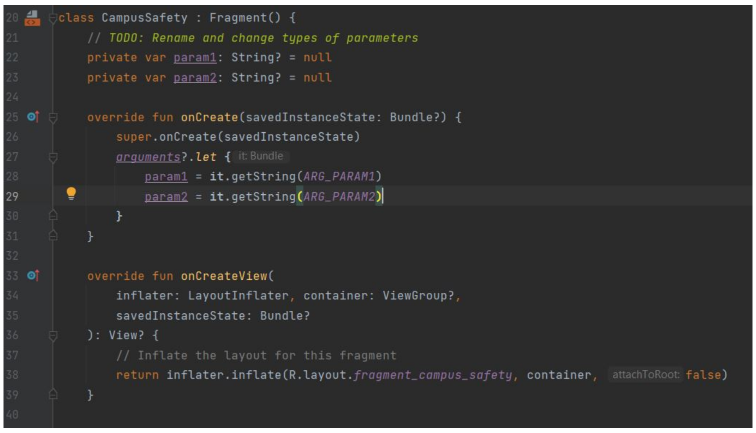
```
class TimeService : Service() {

    override fun onBind(intent: Intent?): IBinder? = null

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
        val time = intent?.getDoubleExtra(TIME_EXTRA , defaultValue: 0.0)
        timer.scheduleAtFixedRate(time?.let { TimeTask(it) }, delay: 0 , period: 1000)
        return START_NOT_STICKY
    }

    override fun onDestroy() {
        timer.cancel()
        super.onDestroy()
    }

    private inner class TimeTask(private var time: Double): TimerTask(){
        override fun run() {
            val intent = Intent(TIMER_UPDATED)
            time++
            intent.putExtra(TIME_EXTRA , time)
            sendBroadcast(intent)
        }
    }

```

Figure 13: time counts code fragment

3.3 Campus Safety Function

3.3.1 Click Event

The processing function and code when a click event occurs on the campus safety interface.

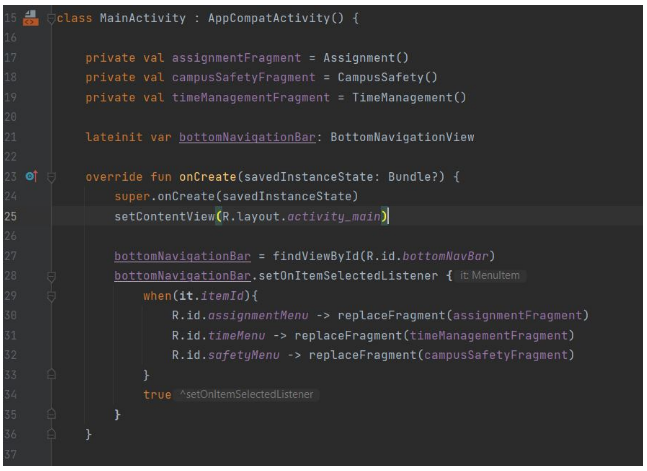


```kotlin
class CampusSafety : Fragment() {
    // TODO: Rename and change types of parameters
    private var param1: String? = null
    private var param2: String? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        arguments?.let { it: Bundle
            param1 = it.getString(ARG_PARAM1)
            param2 = it.getString(ARG_PARAM2)
        }
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_campus_safety, container, attachToRoot: false)
    }
```

Figure 14: click event code fragment

3.3.2 User Input Event

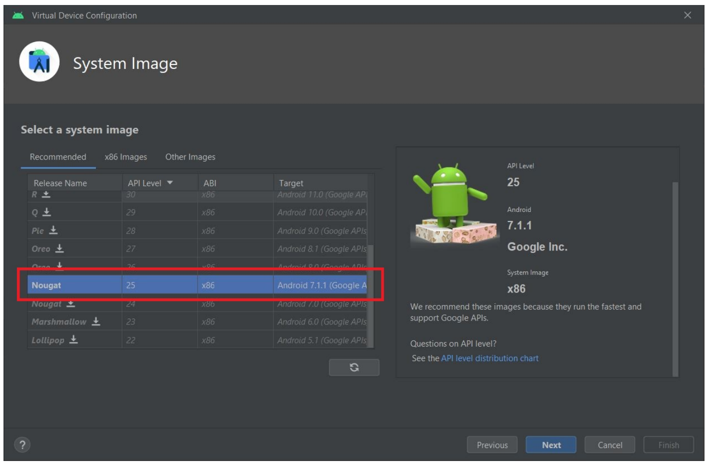
The code fragment of the user enters text in the input box and records the code snippet in the background.



```kotlin
class MainActivity : AppCompatActivity() {

    private val assignmentFragment = Assignment()
    private val campusSafetyFragment = CampusSafety()
    private val timeManagementFragment = TimeManagement()

    lateinit var bottomNavigationBar: BottomNavigationView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        bottomNavigationBar = findViewById(R.id.bottomNavBar)
        bottomNavigationBar.setOnItemSelectedListener { it: MenuItem
            when(it.itemId){
                R.id.assignmentMenu -> replaceFragment(assignmentFragment)
                R.id.timeMenu -> replaceFragment(timeManagementFragment)
                R.id.safetyMenu -> replaceFragment(campusSafetyFragment)
            }
            true ^setOnItemSelectedListener
        }
    }
```

Figure 15: user input event code fragment

4 Documentation



Select a system image

Release Name	API Level	ABI	Target
R	30	x86	Android 11.0 (Google API
Q	29	x86	Android 10.0 (Google API
Pie	28	x86	Android 9.0 (Google APIs
Oreo	27	x86	Android 8.1 (Google APIs
Oreo	26	x86	Android 8.0 (Google APIs
Nougat	25	x86	Android 7.1.1 (Google A
Nougat	24	x86	Android 7.0 (Google APIs
Marshmallow	23	x86	Android 6.0 (Google APIs
Lollipop	22	x86	Android 5.1 (Google APIs

API Level
25
Android
7.1.1
Google Inc.
System Image
x86

We recommend these images because they run the fastest and support Google APIs.

Questions on API level?
See the API level distribution chart

Figure 16: The fifth step of running code

Name: Student Assistance Application

Team Member:

- Hongbo Wang
- Jachie McAninch
- Yiqi Wang
- Xiaoqian Yang
- Khiem Truong

Environment:

- Android Studio 11.0.15
- JDK: 19.0.1
- Android SDK: 7.1.1

Details:

- Type: term project
- Professor: Dennis Hood
- Developing Language: Java
- Project Name: Student Assist App

- Time: 2023/04/27
- Description: Student Assist App is a mobile phone software developed through Android Studio. This software mainly has three functions: homework management, time management, and safety assistance.

Dependencies:

- None

Install package:

- None

Run Program:

Method:

- Download JDK and Android Studio (Environment)
- Using Android Studio, open SoftwareEngApp (Figure 17)

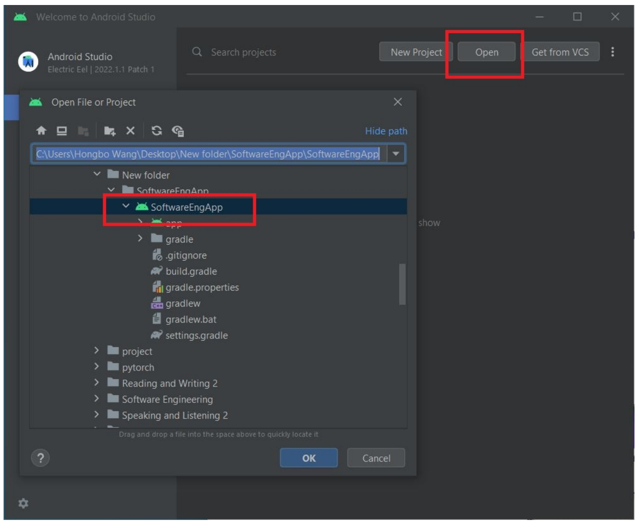


Figure 17: The first step of running code

- Wait for the Gradle build yourself environment (Figure 18)

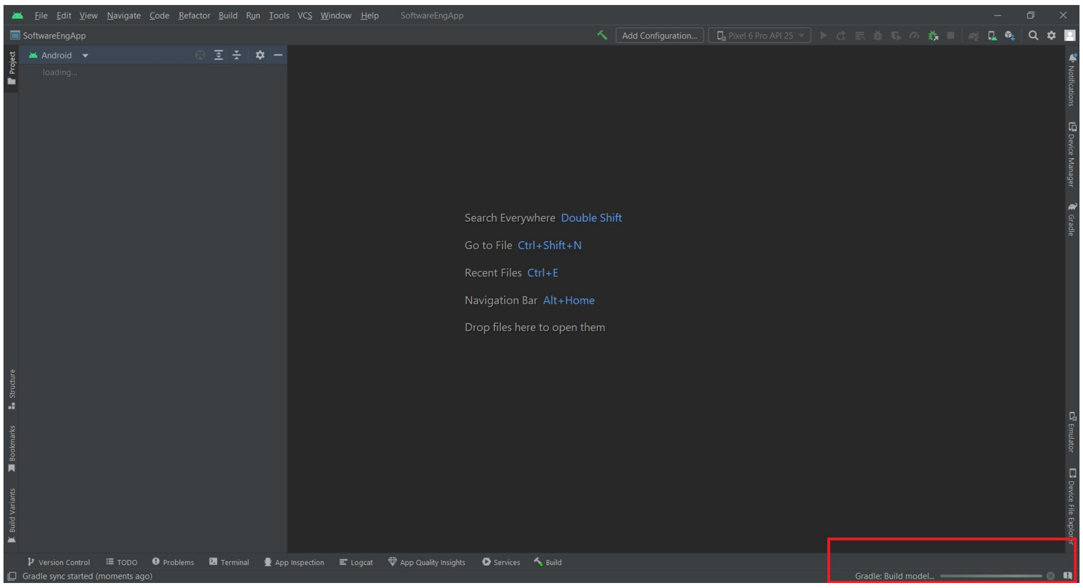


Figure 18: The second step of running code

- Click the run button (Figure 19)

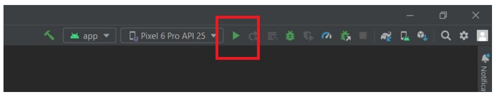
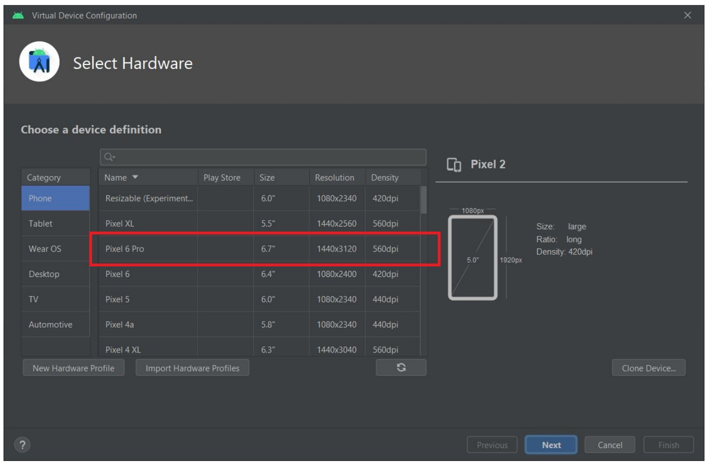


Figure 19: The third step of running code

- Select Pix 6 Pro and click the Next button (Figure 20)



Virtual Device Configuration

Select Hardware

Choose a device definition

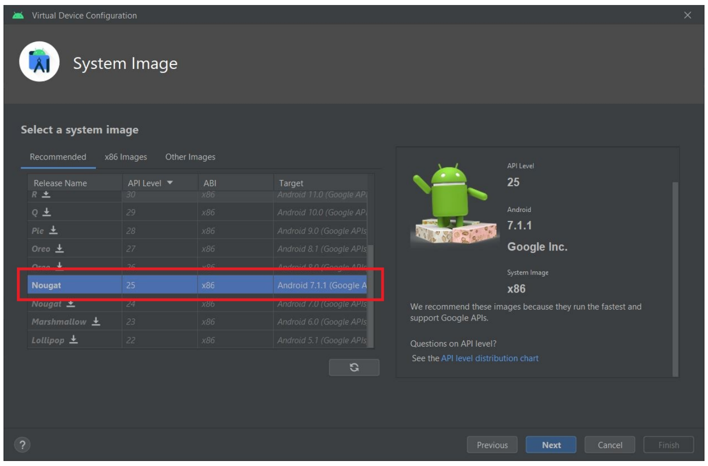
Category	Name	Play Store	Size	Resolution	Density
Phone	Resizable (Experiment...		6.0"	1080x2340	420dpi
Tablet	Pixel XL		5.5"	1440x2560	560dpi
Wear OS	Pixel 6 Pro		6.7"	1440x3120	560dpi
Desktop	Pixel 6		6.4"	1080x2400	420dpi
TV	Pixel 5		6.0"	1080x2340	440dpi
Automotive	Pixel 4a		5.8"	1080x2340	440dpi
	Pixel 4 XL		6.3"	1440x3040	560dpi

Pixel 2

Size: large
Ratio: long
Density: 420dpi

Figure 20: The forth step of running code

- Click Nougat and click next button (Figure 21)



Virtual Device Configuration

System Image

Select a system image

Recommended | x86 Images | Other Images

Release Name	API Level	ABI	Target
R	30	x86	Android 11.0 (Google AP
Q	29	x86	Android 10.0 (Google AP
Pie	28	x86	Android 9.0 (Google APIs
Oreo	27	x86	Android 8.1 (Google APIs
Oreo	26	x86	Android 8.0 (Google APIs
Nougat	25	x86	Android 7.1.1 (Google A
Nougat	24	x86	Android 7.0 (Google APIs
Marshmallow	23	x86	Android 6.0 (Google APIs
Lollipop	22	x86	Android 5.1 (Google APIs

API Level
25

Android
7.1.1

Google Inc.

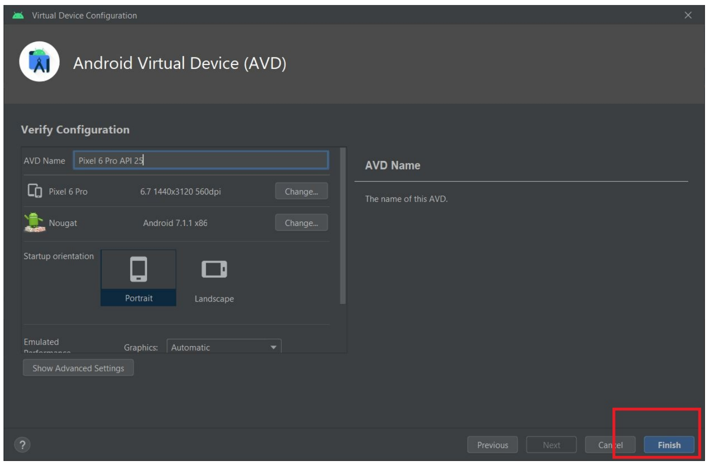
System Image
x86

We recommend these images because they run the fastest and support Google APIs.

Questions on API level?
See the API level distribution chart

Figure 21: The fifth step of running code

- Click the Finish button (Figure 22)



AVD Name: Pixel 6 Pro API 25

Pixel 6 Pro 6.7 1440x3120 560dpi

Nougat Android 7.1.1 x86

Startup orientation: Portrait

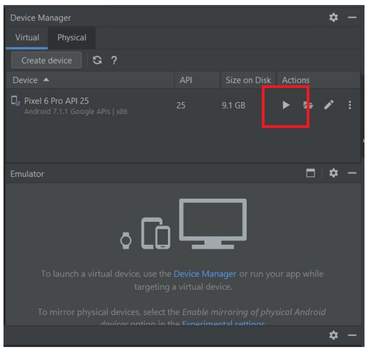
Emulated Performance Graphics: Automatic

AVD Name

The name of this AVD.

Figure 22: The sixth step of running code

- Click the run button (Figure 23)



Device	API	Size on Disk	Actions
Pixel 6 Pro API 25 Android 7.1.1 Google APIs | x86	25	9.1 GB	

To launch a virtual device, use the Device Manager or run your app while targeting a virtual device.

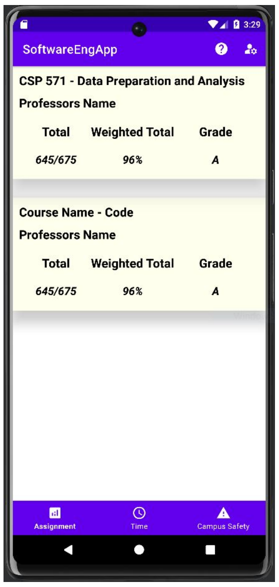
To mirror physical devices, select the *Enable mirroring of physical Android devices* option in the Experimental settings.

Figure 23: The seventh step of running code

- If there is the interface of the phone, you can click the run button again (Figure 24)
- Now you successfully open the App in Android Studio

Example:

1. The screenshot of assignment management.



SoftwareEngApp

CSP 571 - Data Preparation and Analysis
Professors Name

Total	Weighted Total	Grade
645/675	*96%*	*A*

Course Name - Code
Professors Name

Total	Weighted Total	Grade
645/675	*96%*	*A*

Figure 24: The interface of assignment management function

2. The screenshot of time management.

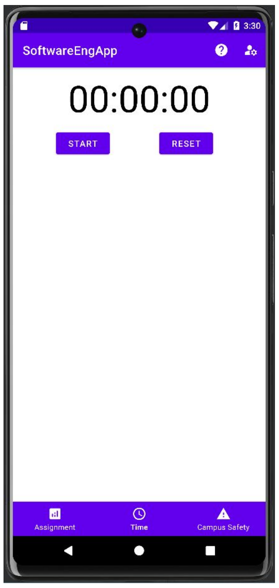
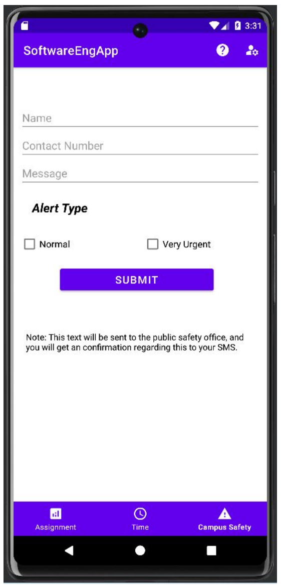


Figure 25: The interface of time management function

3. The screenshot of safety assistance.



Note: This text will be sent to the public safety office, and you will get an confirmation regarding this to your SMS.

Figure 26: The interface of safety management function