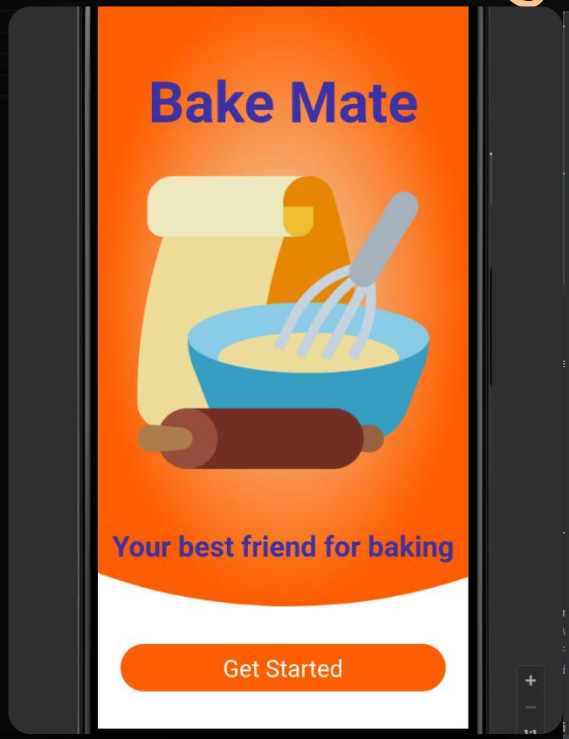
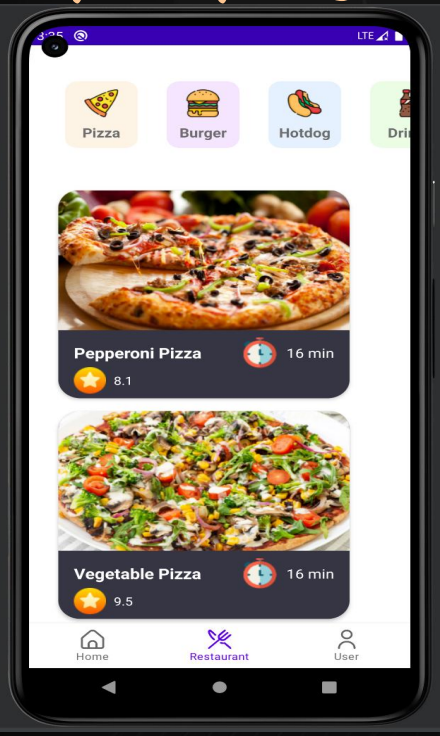
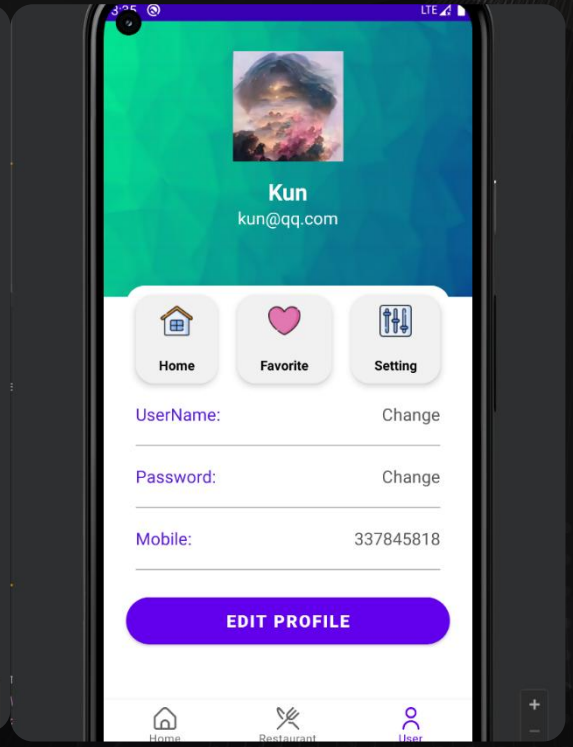
BAKE MATE

CONTENTS

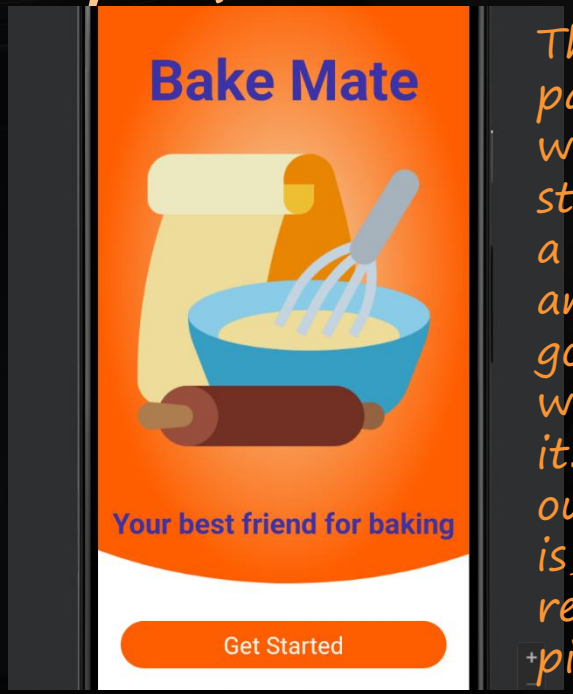
This is our group's page



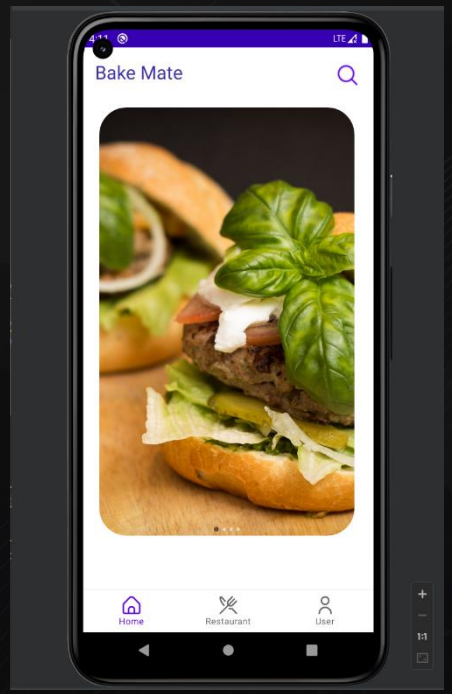




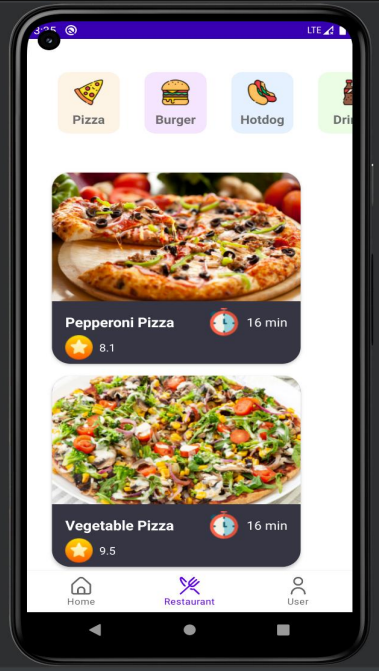
Our creativity with the ui



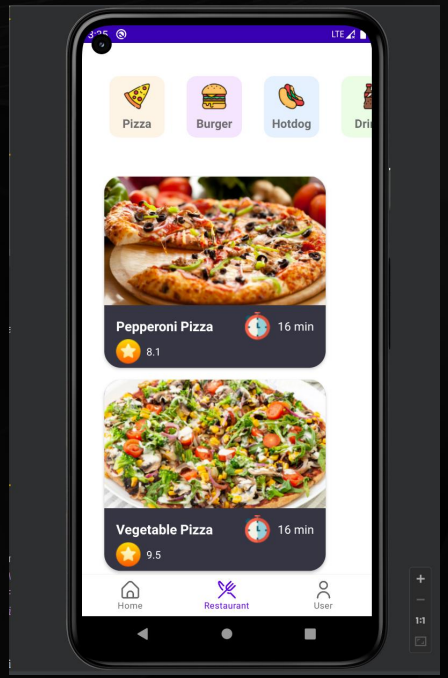
This is our home page, and when we click "get started," there's a slow animation that goes to the page with the cat on it. This is where our app "home" is, and we can replace the picture with a pastry picture.



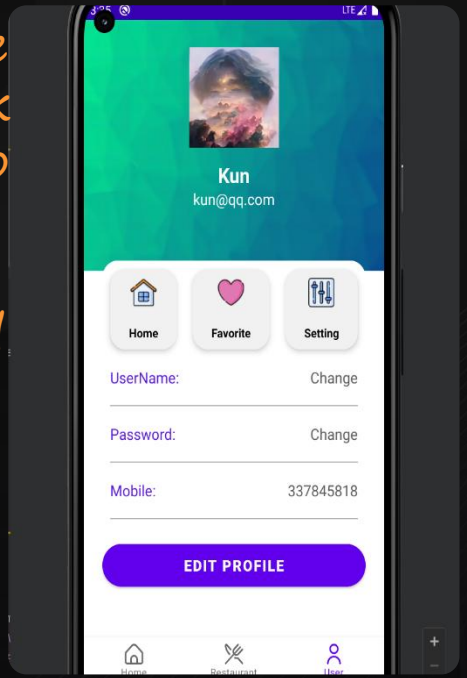
Then we click the "restaurant" button at the bottom of the app and we can go to a new page



We can see the pizza, burger and other buttons on it. Users can click relevant food according to their needs, and then the software will screen and leave the food they need. We can see the score of the relevant food, the time it takes and so on.



Then we can click "user" to access the personal user interfac e



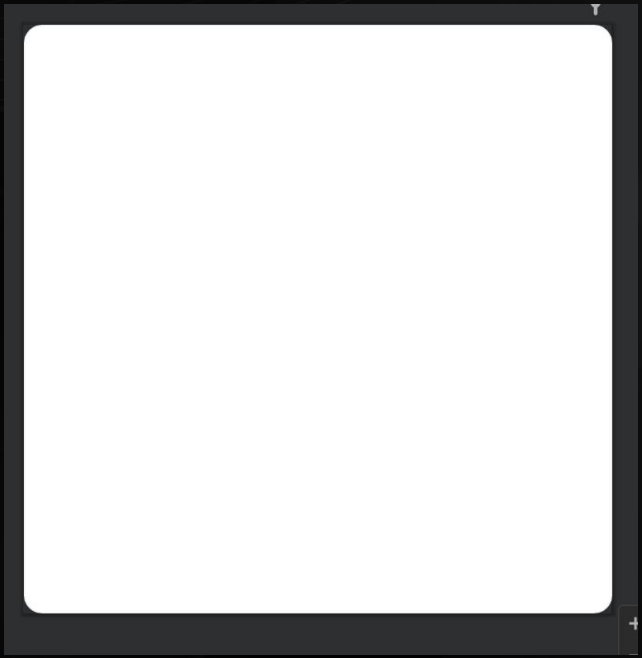
The personal user interface includes background Settings, account passwords, and phone numbers

the fluidity of the design will be shown in following viedo

Completeness, workload and difficulty of the UI design.

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@color/white"/>
    <corners
        android:topRightRadius="15dp"
        android:bottomRightRadius="15dp"
        android:bottomLeftRadius="15dp"
        android:topLeftRadius="15dp"/>
    <stroke android:width="1dp" android:color="#eeeeee"/>
</shape>
```

At first, we wanted to implement the rounded edges of the wheel cast image, so we went online and found this string of code. But unfortunately, he couldn't do rounded corners on a wheel cast, and we've been thinking about this for a long time,



This is a rounded picture of when we failed

```
.addBannerLifecycleObserver( owner: this).setBannerRound(100.0F)
 .setIndicatorRadius(500)
 .setIndicator(CircleIndicator(requireContext()))
```

This is where the rounded corners are achieved

Finally, we found that this string of code is the key to achieve the round corners of the wheel cast map border

The animation from the main page that we clicked on “get started” got us thinking for a long time。

```
class Intro : AppCompatActivity() {
    @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_intro)
        val startBtn = findViewById<ConstraintLayout>(R.id.startbtn)
        startBtn.setOnClickListener{ it: View!
            val myintent = Intent( packageContext: this, FadeActivity::class.java)
            // 启动 Activity 并设置过渡动画
            startActivity(myintent, ActivityOptions.makeSceneTransitionAnimation( activity: this).toBundle())
        }
    }
}
```

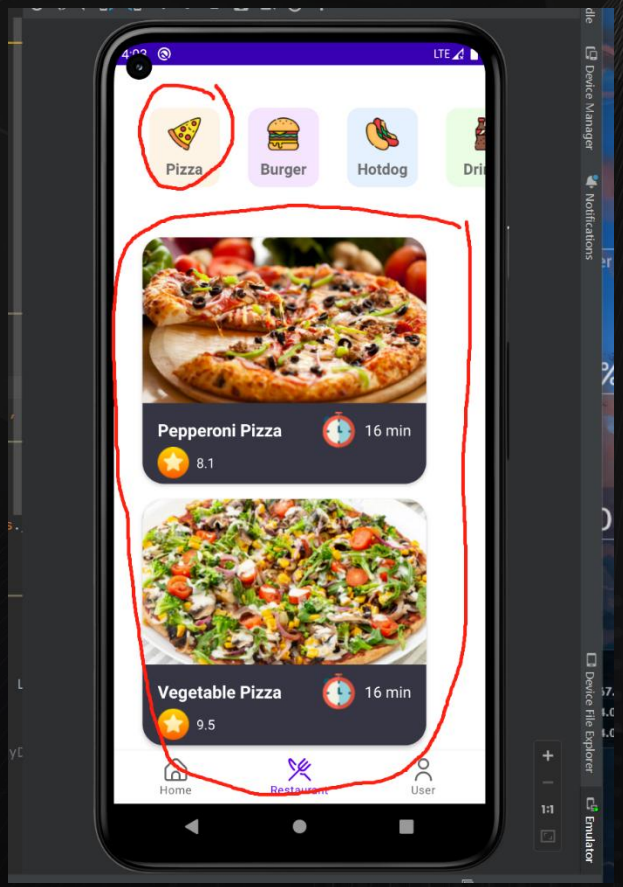
```kotlin
class FadeActivity : AppCompatActivity() {
    @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)
        val navView = findViewById<BottomNavigationView>(R.id.nav_view)
        val navController = findNavController(R.id.nav_host_fragment_activity_main)
        navView.setupWithNavController(navController)
        // 进入效果
        window.enterTransition = Fade().setDuration(2000)
        // 退出效果
        window.exitTransition = Fade().setDuration(0)
    }
}
```

Later we found that we need to implement the animation effect through this string of code. The main way to do this is "entertransition.

Because this layout uses fragments, which are different from normal activities, I thought about this for a long time. Finally, the code on the figure was used to solve the problem

In restaurant (dashboard), there are two recylerview categories. In order to realize clicking an item in horizontal recylerview, it took us a long time to change the content in vertical recylerview accordingly.



```kotlin
private var listener: OnItemClickListener? = null

fun setOnItemClickListener(listener: OnItemClickListener) {
    this.listener = listener
}

interface OnItemClickListener {
    fun onItemClick(position: Int, category: String, content: String)
}
```

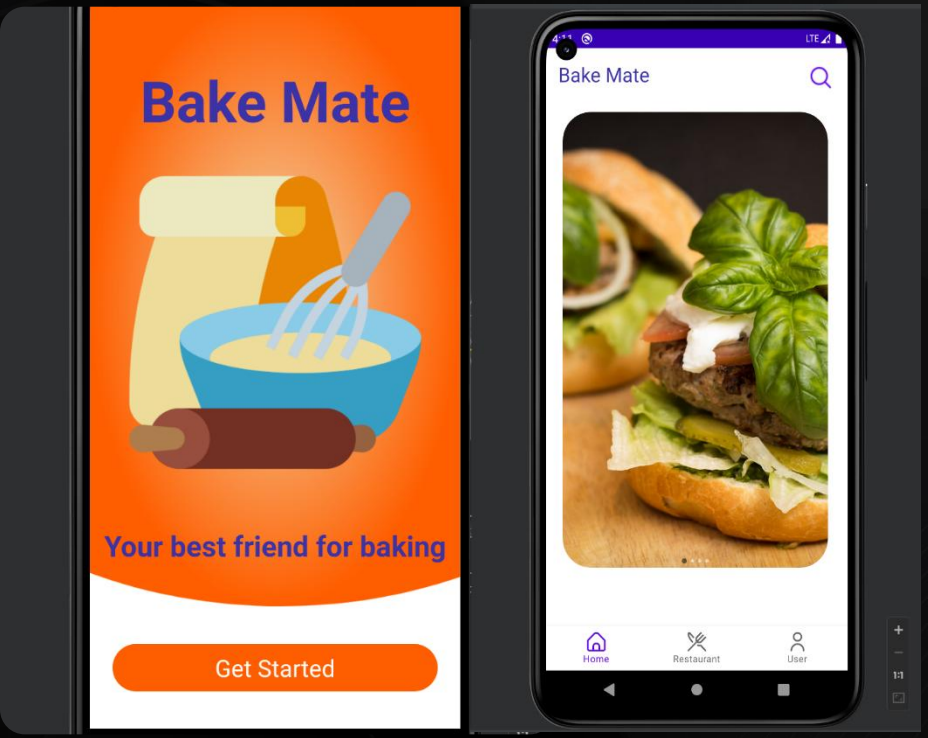
```kotlin
holder.itemView.setOnClickListener { it: View!
    listener?.onItemClick(position, categoryDomains[position].title, content: "This is ${categoryDomains[position].title} category.")
}
```

```kotlin
adapter.setOnItemClickListener(object : CategoryAdapter.OnItemClickListener{
    @SuppressLint("NotifyDataSetChanged")
    override fun onItemClick(position: Int, category: String, content: String) {
        when(position){
            0 -> {
                adapter2 = FastDeliveryAdapter(fastlist)
            }
            1 -> {
                adapter2 = FastDeliveryAdapter(fastlistBurger)
            }
            2 -> {
                adapter2 = FastDeliveryAdapter(fastlistHotdog)
            }
        }
        recyclerViewfastList.adapter = adapter2
        adapter2.notifyDataSetChanged()
    }

})
```

We realized this through the code on the figure. First, we defined onItemclickListener in the adapter and used the setonItemclickListener function to receive data. Then call this function in dashboard for vertical recyclerview content change to implement the sorting effect

Fluidity and expressiveness of the animation



in order to make the page jump (animation) more smooth, fadeactivity is created to have a fade in and fade out effect when switching between the following two pages, making the transition between the two pages more smooth

```kotlin
class FadeActivity : AppCompatActivity() {
    @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)
        val navView = findViewById<BottomNavigationView>(R.id.nav_view)
        val navController = findNavController(R.id.nav_host_fragment_activity_main)
        navView.setupWithNavController(navController)
        // 进入效果
        window.enterTransition = Fade().setDuration(2000)
        // 退出效果
        window.exitTransition = Fade().setDuration(0)
    }
}
```

This is the animation code that implements the transition between the two pages

THANKS!

汇报人：稻小壳