2CB107

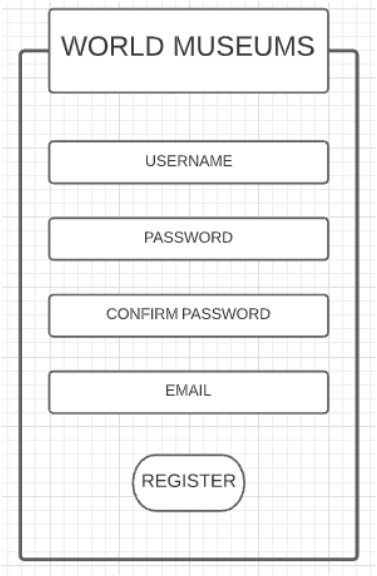
Mobile Application Development

Summative Assessment

Student Number: 199026342

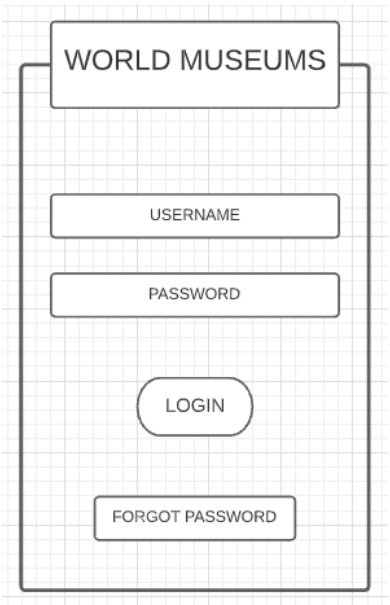
Design Prototypes

Registration Page



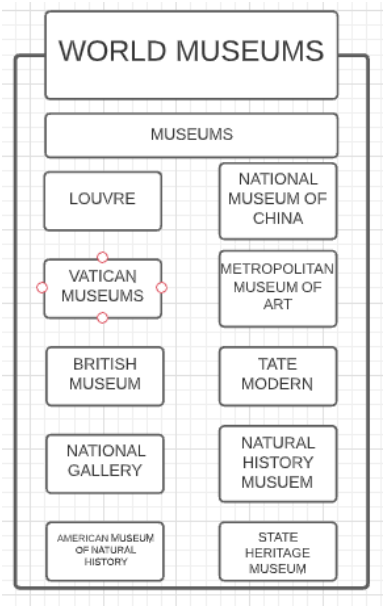
Above is a prototype design for the registration page to the app. A user must have account to use the app, users who already have an account will be redirect to the login page.

Login Page



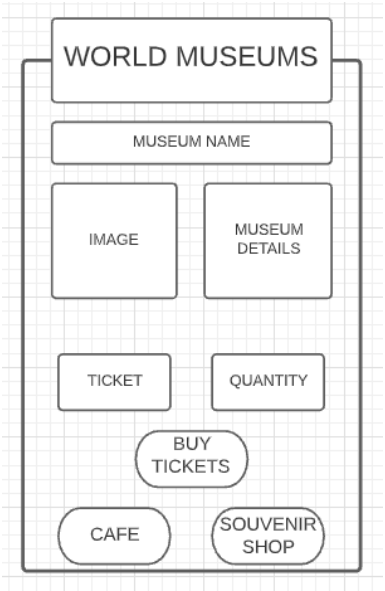
The login page, above, will allow users to login to user their account. There is also a button that allows them to login even if they have forgot their password.

List Of Museums Page



Above is possibly the most important page, the list of museums. This page consists of buttons that when pressed take the user to that museums page so they can get more info or book tickets.

Museum Page

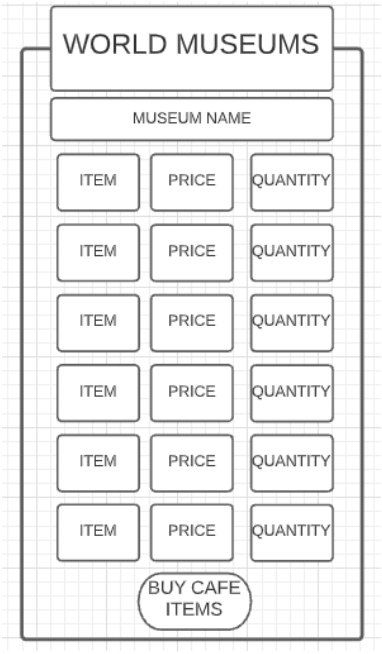


The above page is one that each museum will have. This will be where there is extra info about each museum as well as the place too book tickets and go to wither the shop or café for the museum.

Souvenir Page

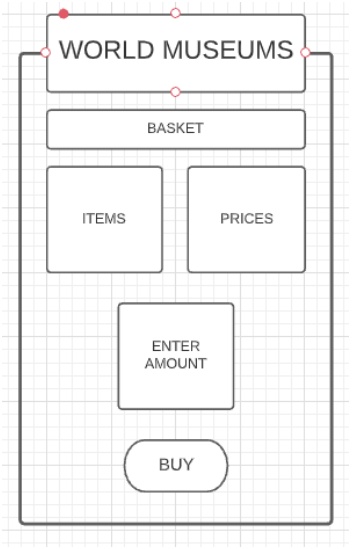
This is the souvenir shop page; it consists of a table were users select what they want to buy and then a button to buy them.

Café Page



This is the cafe page; it consists of a table were users select what they want to buy and then a button to buy them.

Payment Page



This is the page used for payment. A booking number will be displayed to the user for confirmation of bookings.

Principles of Design

In mobile application design there are two main principle of design ideas, these are Norman's fundamentals and Shneiderman principles. Norman's fundamentals consist of 6 elements to ensure good design in interfaces, Shneiderman principles however have 8 golden rules of interface design.

One of the main parts to both fundamental ideas is consistency. In the prototype designs that I have created are all based off the same basic layout. This is to ensure the design and layout for each page of the app will be similar. Another way in which I will ensure consistency in the app interface will be using the same font as well as sticking to the same colour theme throughout the app. By doing this all the pages will look as though they belong to the same app and not just randomly joined together. The reason why consistency is such a key factor in interface design is because without it an app will look untidy, unorganised and unprofessional as well as make it harder for users trying to use it.

An element of Norman's fundamentals is visibility. A function must be visible for users to see and use them, the more visible a function is the more likely it is to be used. The apps main functions will be buttons, and these will all be made visible by having background colours that contrast the rest of the app which will make them stand out. Another reason why these

buttons need to stand out is people won’t buy tickets, souvenirs or anything from the café if they aren't able to find them so its key the buttons stand out to avoid a loss in revenue.

Another element of Norman’s fundamentals that the app needs to meet is feedback. Feedback is a response to the user's actions; users should experience feedback as a natural consequence of their actions. Feedback on an app could be anything from a beep or a change in colour. The app I have designed uses feedback using pop-up boxes to confirm that payment has been made as well as confirming booking of any tickets.

One principle of Shneiderman principles is error handling. This principle is designed to add failsafe into the system so the user can’t make a serious error. If an error is made, the system will be able to detect it and allow the user to fix it. Error handling will be mainly used in the registration and logging on the app. This is because these pages have fields that require user input, which is where most errors will occur. Error handling will be used to ensure the user inputs a username, email in the correct format and a password, by doing this it ensures all the accounts created are made with the right details to allow users to login whenever they want.

Another principle from Shneiderman principles is shortcuts. This principle is used to reduce the number of interactions and to increase the pace of interactions. Although it isn’t directly clear from the prototype designs how this is included in the app, it will be implemented through having a minimal number of buttons used to change the pages as well as only linking pages together that are directly related to one another.

User Concerns

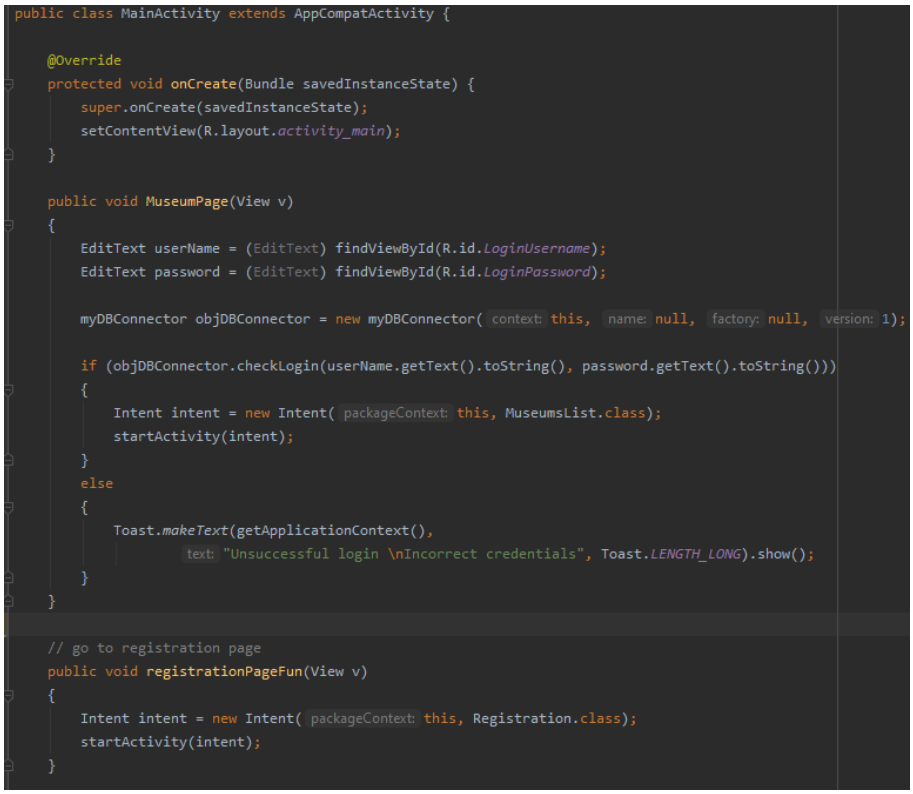
For the app to be considered a success it must be easy to learn and effective to use. To do this the user requirements and user concerns must be considered.

One user concern could be the app doesn’t meet the needs of the users. To try and overcome this issue would be to make sure the app has all the required features that has been set out, by doing this the app will be better suited to the needs of the users as well as function in the desired way. The key to developing an app is to make sure all the requirements are meet so it is what is wanted by the user, and they therefore won’t have an issue with the final product.

Other user concerns could be language barriers, not understanding different currencies or not being as tech able as others. In order to make sure the site over comes potential language barriers it will have an option to select the language you want. By including this feature people can translate the app to their native language, making it accessible for everyone. To make sure users can understand the prices used on the site there will be one of two potential features. The first could be an option to select a currency that they wish to use on the app, the other feature could be a button that brings up a pop that shows them the exchange rate so they can change the price from one currency to another.

Screenshots of Implemented Application

Below is a screenshot of code that is used to create the login page.



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void MuseumPage(View v)
    {
        EditText userName = (EditText) findViewById(R.id.LoginUsername);
        EditText password = (EditText) findViewById(R.id.LoginPassword);

        myDBConnector objDBConnector = new myDBConnector( context: this,  name: null,  factory: null,  version: 1);

        if (objDBConnector.checkLogin(userName.getText().toString(), password.getText().toString()))
        {
            Intent intent = new Intent( packageContext: this, MuseumsList.class);
            startActivity(intent);
        }
        else
        {
            Toast.makeText(getApplicationContext(),
                    text: "Unsuccessful login \nIncorrect credentials", Toast.LENGTH_LONG).show();
        }
    }

    // go to registration page
    public void registrationPageFun(View v)
    {
        Intent intent = new Intent( packageContext: this, Registration.class);
        startActivity(intent);
    }
```

Below is the code to make the registration page.

```java
public class Registration extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration);
    }

    // method for adding a new user
    public void addNewUserFun (View view)
    {
        myDBConnector dbHandler = new myDBConnector( context: this,  name: null,  factory: null,  version: 1);

        // establish binding with components in GUI, name, username and password
        EditText name = (EditText) findViewById(R.id.txtName);
        EditText userName = (EditText) findViewById(R.id.RegisterUsername);
        EditText password = (EditText) findViewById(R.id.RegisterPass);

        // add new user with name, username and password
        dbHandler.addNewUser(name.getText().toString(),
                             userName.getText().toString(),
                             password.getText().toString() );

        Intent intent = new Intent( packageContext: this, MainActivity.class);
        startActivity(intent);
    }
```

Below is the code for the museums list page, there is a function for each museum

```java
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_museums_list);
    }
    // go to louvre page
    public void LouvreFun(View v)
    {
        Intent intent = new Intent( packageContext: this, Louvre.class);
        startActivity(intent);
    }

    // go to museum of china page
    public void ChinaFun(View v)
    {
        Intent intent = new Intent( packageContext: this, China.class);
        startActivity(intent);
    }

    // go to vatican museum page
    public void VaticanFun(View v)
    {
        Intent intent = new Intent( packageContext: this, Vatican.class);
        startActivity(intent);
    }
```

Below is the code for a museum page. In this example it is the Louvre

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_louvre);

    ticketQTY = findViewById(R.id.LouvreQTY);

    timeSlot = findViewById(R.id.LovreTimeSlots);

    LouvreTicket = new ticket();
    LouvreTicket.museum = "Louvre";

}
// go to museum page
public void MuseumListFun(View v)
{
    Intent intent = new Intent( packageContext: this, MuseumsList.class);
    startActivity(intent);
}
// go to shop
public void ShopFun(View v)
{
    Intent intent = new Intent( packageContext: this, Shop.class);
    startActivity(intent);
}
// add ticket to basket
public void TicketFun(View v)
{
    LouvreTicket.qty = Integer.parseInt(String.valueOf(ticketQTY.getText()));
    LouvreTicket.total = 10 * LouvreTicket.qty;

    System.out.println("Museum: " + LouvreTicket.museum);
    System.out.println("Ticket time slot: " + LouvreTicket.TimeSlot);
    System.out.println("Ticket quantity: " + LouvreTicket.qty);
    System.out.println("Ticket cost: " + LouvreTicket.total);

}
public void checkButton(View v)
{
    int radioId = timeSlot.getCheckedRadioButtonId();

    radioButton = findViewById(radioId);
    //objTicket.timeSlot = (String) radioButton.getText();
    LouvreTicket.TimeSlot = (String) radioButton.getText();
    System.out.println("Ticket time slot: " + LouvreTicket.TimeSlot);
}
```

Below is a screenshot of the code to make the shop, this code is repeated and modified for every item. In this example it is a shirt

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_shop);

    EditText shirtqty = findViewById(R.id.shirtQTY);
    Button addShirtBttn = findViewById(R.id.shirtBuy);


    item shirt = new item( _name: "Shirt",  _cost: 10);

    itemOrder = new order();

    itemOrder.basket.add(shirt);

    addShirtBttn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            itemOrder.basket.get(0).qty = Integer.parseInt(String.valueOf(shirtqty.getText()));
        }
    });
```

Below is the code for the café, each item is made in the same way. This example is of tea

```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cafe);
    EditText TeaQty = findViewById(R.id.teaQTY);
    Button AddTea = findViewById(R.id.teaBuy);

    item tea = new item( _name: "Tea",  _cost: 1);


    Shop.itemOrder.basket.add(tea);

    System.out.println(Shop.itemOrder.basket.get(0).name);

    AddTea.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Shop.itemOrder.basket.get(1).qty = Integer.parseInt(String.valueOf(TeaQty.getText()));
        }
    });
```

Below is a screenshot of the basket code

```
public class basket extends AppCompatActivity
{
    int moneyPaid;
    int total;
    Random rand;
    int minNum = 100000;
    int maxNum = 999999;
    double confirmationCode;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_basket);

        TextView checkoutTXT = findViewById(R.id.checkoutTXT);
        EditText inputAmount = findViewById(R.id.inputAmount);
        Button payButton = findViewById(R.id.payButton);
        TextView confirmationTXT = findViewById(R.id.confirmationCodeTXT);

        System.out.println("NAME: " + Shop.itemOrder.basket.get(0).name);
        System.out.println("QTY: " + Shop.itemOrder.basket.get(0).qty);

        checkoutTXT.setText("Museum name: " + Louvre.LouvreTicket.museum +
                "\nTime Slot: " + Louvre.LouvreTicket.TimeSlot +
                "\nAmount of ticket(s): " + Louvre.LouvreTicket.qty);

        total = 0;
```

```
for(int i=0; i < Shop.itemOrder.basket.size(); i++)
{
    if(Shop.itemOrder.basket.get(i).qty > 0)
    {
        total += (Shop.itemOrder.basket.get(i).cost * Shop.itemOrder.basket.get(i).qty);
        checkoutTXT.append("\n\n Item Name: " + Shop.itemOrder.basket.get(i).name + "       " + "Quantity: " + Shop.itemOrder.basket.get(i).qty );
    }
}

total += (Louvre.LouvreTicket.total);
checkoutTXT.append("\n\n Total: £" + total);

rand = new Random();
confirmationCode = Math.floor(Math.random()*(maxNum-minNum+1)+minNum);

payButton.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        moneyPaid = Integer.parseInt(String.valueOf(inputAmount.getText()));
        if(moneyPaid >= total)
        {
            confirmationTXT.setText("Total paid: " + total +"\n Confirmation code: " + confirmationCode);

        }

    }
});
```

Below is the code that is the layout for the ticket to the museum

```
public class ticket
{
    public String museum;
    public int qty;
    public String TimeSlot;
    public int total;

}
```

Below is the code for the layout of every item in the shop and café

```
public class item {

    public String name;
    public int cost;
    public int qty;

    public item(String _name, int _cost)
    {
        this.name = _name;
        this.cost = _cost;
    }
}
```

Below is the creation of an array to store the order of the user

```
import java.util.ArrayList;

public class order {
    ArrayList<item> basket = new ArrayList<~>();

}
```