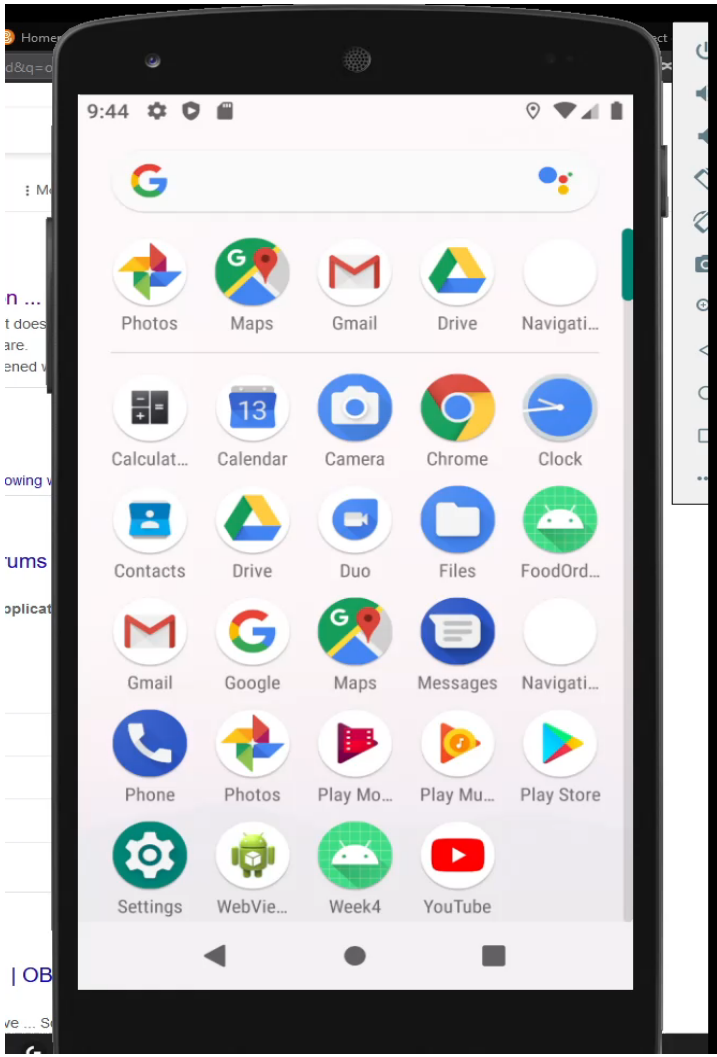
CMP309 Finn's Fare

Finlay Reid (1904629)

Overview of talk

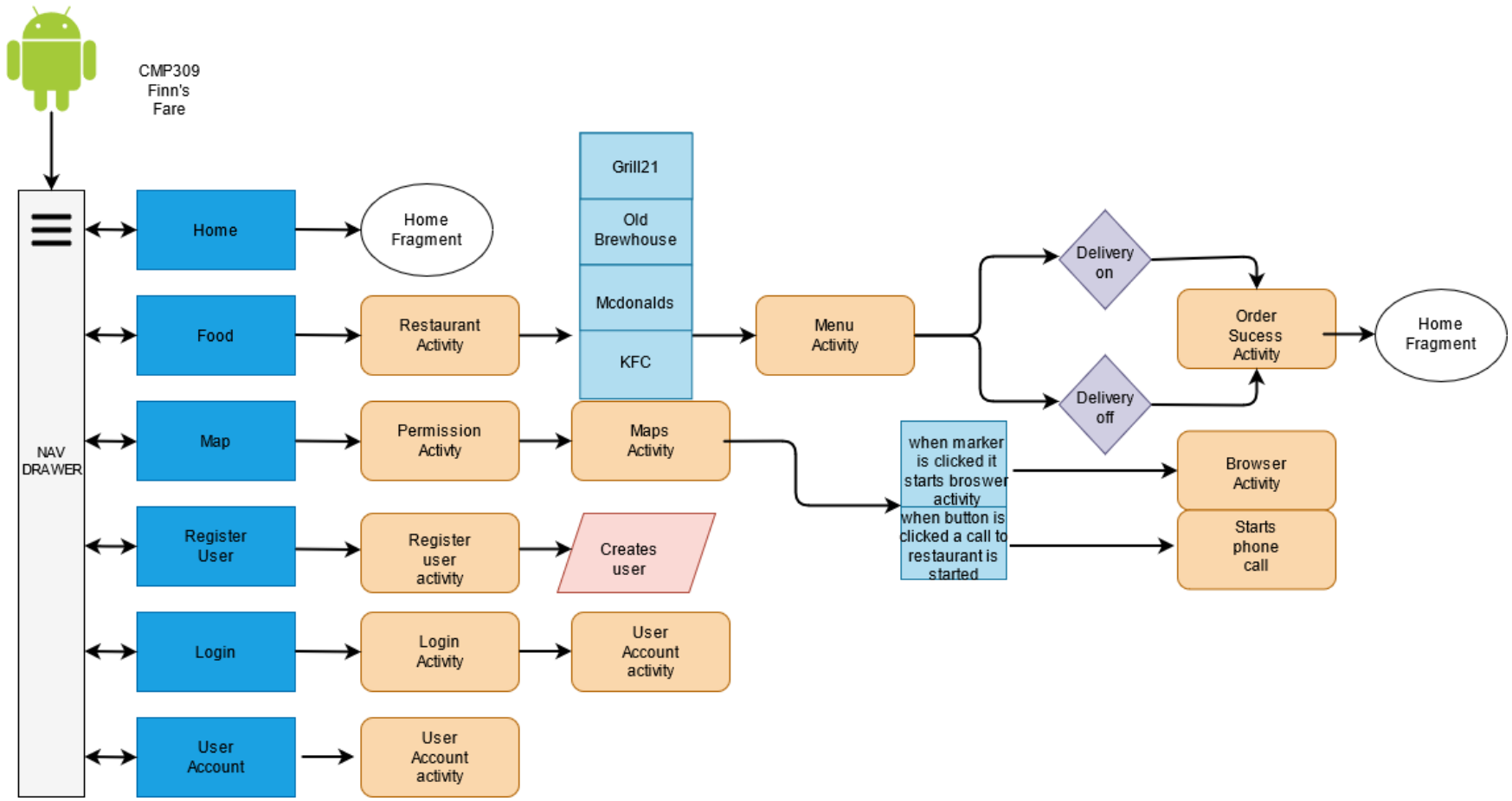
- This presentation will thoroughly describe the Finn's fare application and its functionality.
- Some of the particulars covered include:
- Reflecting on software design and the development process
- Video demonstration that presents all features included in the app
- Flow diagram detailing how activities interact and how users move through the application.
- Explanation of code relating to the most important functionality
- Critical analysis describing how the app handles security etc



Video Demonstration

*Missing forgot password functionality

Flow Diagram



Map activity functionality

- Layout involves screen being split horizontally with two components.
- One is a map view and the other is a recycler.
- Displays colour coded markers
- Animated key that allows users to determine what the colours correspond to.
- Clickable marker takes user to establishments web site

- Scrollable recycler view listing individual food establishments
- Camera is dynamically updated when a restaurant tab is selected.
- A call now button is available to immediately contact a food outlet.

Map Activity Code

```
public boolean onMarkerClick(@NonNull Marker marker) {
// starts browser activty when a marker is clicked on and then loads the restaurants website
    if(marker.equals(Grill21)){
        Intent intent = new Intent( packageContext: MapsActivity.this, BrowserActivity.class); //store intents
        intent.putExtra( name: "URL", URL[0]); // adds extras from URL array
        startActivity(intent); //starts the browser activty and passes the data
```

```
public void onUserClicked(int position) {
    // when a restaurant is clicked on in the recycler view it will move the camera to that marker

        if(position == 0){ // if the position is clicked on in the recycler view

            map.animateCamera(CameraUpdateFactory.newLatLng(Grill21Pos)); // moves camera to the specific marker
```

Restaurant/menu functionality

- Choice of four restaurants displayed in a recycler view.
- Presents order items neatly in a grid view
- The ability to add and remove products
- Dynamically updates of the quantity of items in basket

```
private List<RestaurantModel> getRestaurantData(){

    InputStream is = getResources().openRawResource(R.raw.restaurent);
    Writer writer = new StringWriter();
    char[] buffer = new char[1024];
    try{
        Reader reader = new BufferedReader(new InputStreamReader(is,  charsetName: "UTF-8"));
        int n;
        while(( n = reader.read(buffer)) != -1) {
            writer.write(buffer,  i: 0,n);
        }
    }catch (Exception e) {

    }

    String jsonStr = writer.toString();
    Gson gson = new Gson();
    RestaurantModel[] restaurantModels =  gson.fromJson(jsonStr, RestaurantModel[].class);
    List<RestaurantModel> restList = Arrays.asList(restaurantModels);

    return  restList;
```

Place order functionality

- Allows users to order there requested item
- Option for a pick up or delivery
- Calculation of total price.
- Input validation for fields

```
private void  calculateTotalAmount( RestaurantModel restaurantModel){new Thread(new Runnable() {
    //calculates total amount of order
    float subTotalAmount = 0f;
    @Override
    public void run() {
        for(
                Menu m :restaurantModel.getMenus())

        {
            subTotalAmount += m.getPrice() * m.getTotalInCart();
        }

        tvSubtotalAmount.setText("£"+String.format("%.2f",subTotalAmount));
        if(isDeliveryOn)

        {
            tvDeliveryChargeAmount.setText("£" + String.format("%.2f", restaurantModel.getDelivery_charge()));
            subTotalAmount += restaurantModel.getDelivery_charge();
        }
        tvTotalAmount.setText("£"+String.format("%.2f",subTotalAmount));
    }

}).start();
}
```

```
public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked) {
    //displays fields if the switch has been clicked
    if(isChecked) {
        inputAddress.setVisibility(View.VISIBLE);
        inputCity.setVisibility(View.VISIBLE);
        inputState.setVisibility(View.VISIBLE);
        inputZip.setVisibility(View.VISIBLE);
        tvDeliveryChargeAmount.setVisibility(View.VISIBLE);
        tvDeliveryCharge.setVisibility(View.VISIBLE);
        isDeliveryOn = true;
        calculateTotalAmount(restaurantModel);
    } else {
        inputAddress.setVisibility(View.GONE);
        inputCity.setVisibility(View.GONE);
        inputState.setVisibility(View.GONE);
        inputZip.setVisibility(View.GONE);
        tvDeliveryChargeAmount.setVisibility(View.GONE);
        tvDeliveryCharge.setVisibility(View.GONE);
        isDeliveryOn = false;
        calculateTotalAmount(restaurantModel);
    }
```

User functionality

- Ability to register user creating a Finn's fare account
- Option to log in and view user profile page
- Requirement of email verification
- User forgot password functionality
- Input validation
- Creation of session to remember user when application is paused

User functionality code

```
mAuth.createUserWithEmailAndPassword(email,password)
        .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {

                if (task.isSuccessful())
                {

                    User user = new User(fullName, age, email);
                    FirebaseDatabase.getInstance().getReference( path: "Users")
                            .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
                            .setValue(user).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if(task.isSuccessful()){

                                Toast.makeText( context: RegisterUser.this,  text: "User has been registered", Toast.LENGTH_LONG).show();
                                progressBar.setVisibility(View.GONE);
                            }else{

                               Toast.makeText( context: RegisterUser.this,  text: "Failed to register! Try again", Toast.LENGTH_LONG).show();
                                progressBar.setVisibility(View.GONE);
                            }
```

```
progressBar.setVisibility(View.VISIBLE); // displays progress bar
mAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() { // uses the fire ba
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {

        if(task.isSuccessful()){
            FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser(); // gets the current user if the log in is correct

            if(user.isEmailVerified()){ // checks if email is verified
                progressBar.setVisibility(View.INVISIBLE); // stops progress bar
                startActivity(new Intent( packageContext: Login.this, UserAccount.class)); // opens user account page


            }else{ // if email is not verified a verification email is sent using a firebase function
                progressBar.setVisibility(View.INVISIBLE); // stops progress bar
                user.sendEmailVerification();
                Toast.makeText( context: Login.this,  text: "Check your email to verify your account", Toast.LENGTH_LONG).show();
            }
```

Usability of the app

- Follows a set colour design with blue, white and black.
- Follows best practice, input boxes display hints and strings are not hardcoded.
- Simple but clean UI that enables users to navigate through the different functionality
- Uses many different components to better usability and improve design.
- Only makes use of one orientation horizontal
- Code has been given numerous comments.

Critical analysis

Performance

- Firebase is used to store user details
- Have to meet a criteria for using certain functionality of application
- Application follows android lifecycle with the use of onstop etc

Security

- Input validation
- Correct request of permissions
- Uses firebases in built functions and statements.
- Checking for google play services and GPS

Future Work

- Database used to store orders, menu and restaurants.
  - More user friendly UI
  - Improved home page
- Ability to make real purchases.

Weaknesses

- No Edit or delete user profile abilities
  - Reliance on the internet
- No custom themes i.e. dark mode
  - Trouble for future expansion

References

- Youtube.com. 2021. *Before you continue to YouTube*. [online] Available at: <https://www.youtube.com/watch?v=DMkzIOLppf4&t=96s> [Accessed 13 May 2021]
- Youtube.com. 2021. *Before you continue to YouTube*. [online] Available at: <https://www.youtube.com/watch?v=2saHwKHxpyk> [Accessed 13 May 2021].
- Youtube.com. 2021. Before you continue to YouTube. [online] Available at: <https://www.youtube.com/watch?v=Cys_i-6Pu-o> [Accessed 13 May 2021].
- Medium. 2021. *How to use FirebaseRecyclerAdpater with latest Firebase Dependencies in Android*. [online] Available at: <https://medium.com/android-grid/how-to-use-firebaserecycleradpater-with-latest-firebase-dependencies-in-android-aff7a33adb8b> [Accessed 13 May 2021].
- Android Developers. 2021. *Permissions on Android | Android Developers*. [online] Available at: <https://developer.android.com/guide/topics/permissions/overview> [Accessed 13 May 2021].
- Android Developers. 2021. *Intents and Intent Filters | Android Developers*. [online] Available at: <https://developer.android.com/guide/components/intents-filters> [Accessed 13 May 2021].

- Tutorialspoint.com. 2021. *Android - Studio - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/android/android_studio.htm> [Accessed 13 May 2021].

END