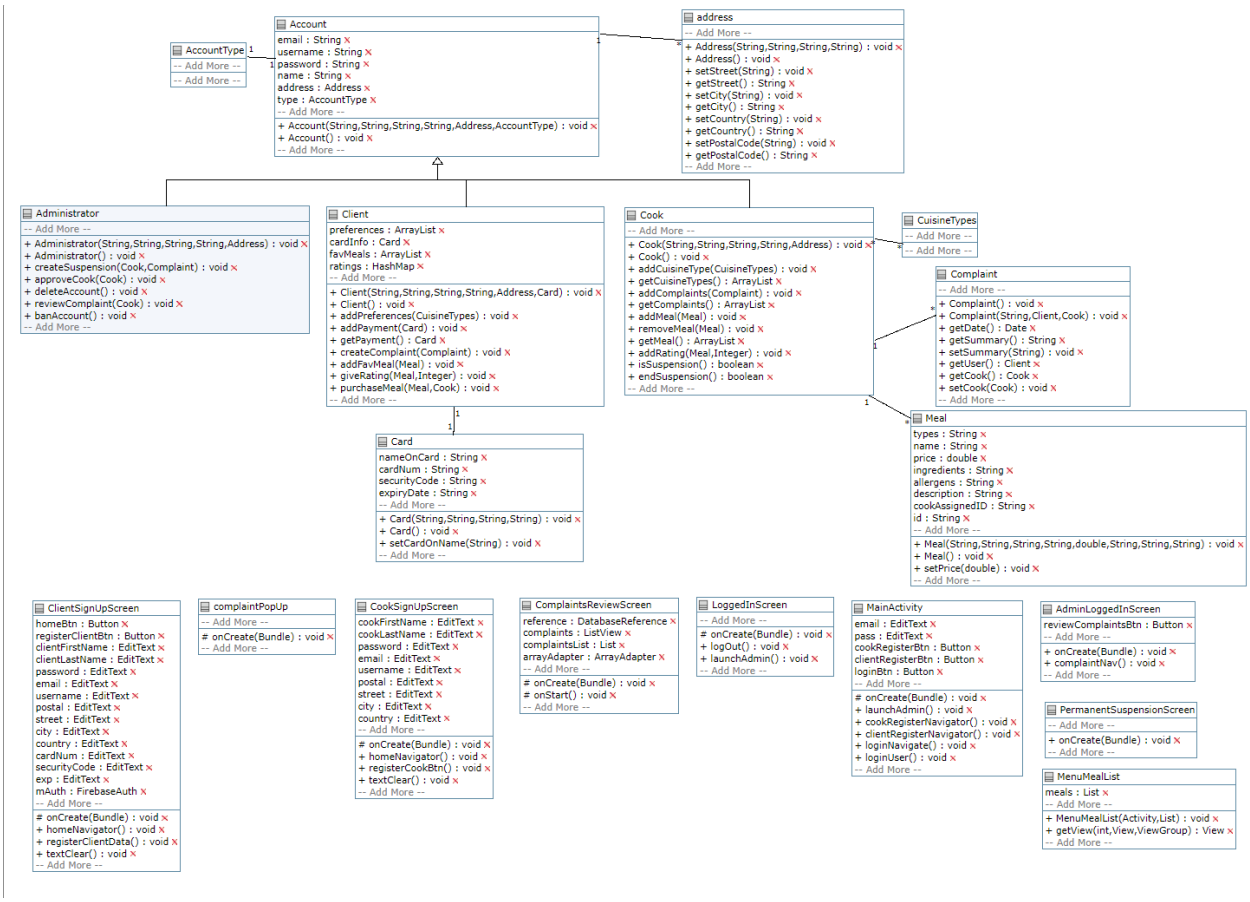
0




```
class Account {

  String email;

  String username;

  String password;

  String name;

  Address address;
```

```
  AccountType type;
  1 -- 1 AccountType;
  1 -- * address;
Account(String email, String username, String password, String name, Address address, AccountType
type) {
    this.email = email;
    this.username = username;
    this.password = password;
    this.name = name;
    this.address = address;
    this.type = type;
  }
  Account(){}
  void setEmail(String email) {}
  String getEmail() {}
  void setUsername(String username) {}
  String getUsername() {}
  String getPassword() {}
  void setPassword(String password) { }
  String getName(){ }
  void setName(String name) {}
  public Address getAddress() {}
  void setAddress(Address address){}
  void setType(AccountType type) {}
  AccountType getType(){}
}
class AccountType {}

class Administrator {
```

```
   isA Account;
Administrator(String email, String username, String password, String name, Address address) {}
  Administrator(){}
  void createSuspension(Cook cook , Complaint c) {}
  void approveCook(Cook cook) {}
  void deleteAccount() {}
  void reviewComplaint(Cook cook) {}
  void banAccount() {}

}
class address{
  Address(String street, String city, String country, String postalCode) {}
  Address(){    }
  void setStreet(String street){}
  String getStreet(){}
  void setCity(String city){}
  String getCity(){}
  void setCountry(String country){}
  String getCountry(){}
  void setPostalCode(String postalCode){}
  String getPostalCode(){}
}

class AdminLoggedInScreen {

  Button reviewComplaintsBtn;

  void onCreate(Bundle savedInstanceState) {}
```

```
  void complaintNav() {}


}
class Card {
  String nameOnCard;
  String cardNum;
  String securityCode;
  String expiryDate;
    1 -- 1 Client;
Card(String nameOnCard, String cardNum, String securityCode, String expiryDate) {
    this.cardNum = cardNum;
    this.nameOnCard = nameOnCard;
    this.securityCode = securityCode;
    this.expiryDate = expiryDate;
  }
  Card(){}
  void setCardOnName(String nameOnCard){ }
  String getNameOnCard(){ }
  void setCardNum(String cardNum){  }
  String getCardNum(){}
  void setSecurityCode(String securityCode){}
  String getSecurityCode(){}
  void setExpiryDate(String expiryDate){}
  String getExpiryDate(){}
}
class Client {
    isA Account;
ArrayList<CuisineTypes> preferences;
  Card cardInfo;
```

```
  ArrayList<Meal> favMeals;
  HashMap<Meal, Integer> ratings;
  Client(String email, String username, String password, String name, Address address, Card card) {}
  Client(){}
  void addPreferences(CuisineTypes preference) {}
  ArrayList<CuisineTypes> getPreferences() {}
  void addPayment(Card card) {}
  Card getPayment() {}
  void createComplaint(Complaint c){}
  void addFavMeal(Meal meal) {}
  ArrayList<Meal> getFavMeals() {}
  void giveRating(Meal meal, Integer rating) {}
  HashMap<Meal, Integer> getRatings() {}
  void purchaseMeal(Meal meal, Cook cooker) {}
}
class ClientSignUpScreen {
  Button homeBtn;
  Button registerClientBtn;
  EditText clientFirstName;
  EditText clientLastName;
  EditText password;
  EditText email;
  EditText username;
  EditText postal;
  EditText street;
  EditText city;
  EditText country;
  EditText cardNum;
  EditText securityCode;
```

```
  EditText exp;
  FirebaseAuth mAuth;
  protected void onCreate(Bundle savedInstanceState) {}
  public void homeNavigator(){}
  public void registerClientData(){}
  public void textClear(){}
}
class Complaint {
    * -- 1 Cook;
private Date date;
  private String summary;
  private Client user;
  private Cook cook;

  public Complaint() {

  }

  public Complaint(String sum, Client user, Cook cook) {
    date = new Date();
    this.setSummary(sum);
    this.user = user;
    this.setCook(cook);
    this.setSummary(sum);
  }
  public Date getDate() {
    return date;
  }
  public String getSummary() {
```

```
    return summary;
  }
  void setSummary(String summary) {}
  Client getUser() {}
  Cook getCook() {}
  void setCook(Cook cook) {}
}
class complaintPopUp {

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

  DatabaseReference reference;
  ListView complaints;
  List<Complaint> complaintsList;
  ArrayAdapter arrayAdapter;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_complaints_review_screen);
    reference = FirebaseDatabase.getInstance().getReference("complaints");

    complaintsList = new ArrayList<>();
```

```
  //TO BE IMPLEMENTED
  addComplaint();
  onItemLongClick();

 }

 protected void onStart(){
  super.onStart();
  reference.addValueEventListener(new ValueEventListener() {
   @Override
   public void onDataChange(@NonNull DataSnapshot snapshot) {
    complaintsList.clear();
    for(DataSnapshot postSnapshot: dataSnapshot.getChildren()){
     Complaint complaint = postSnapshot.getValue(Complaint.class);
     complaintsList.add(complaint);
    }
    ComplaintList complaintAdapter = new ComplaintList(ComplaintsReviewScreen.this,
complaintsList);
    complaints.setAdapter(complaintAdapter);
   }

   @Override
   public void onCancelled(@NonNull DatabaseError error) {

   }
  }
  );
 }
```

```
}
class Cook {


   isA Account;
  * -- * CuisineTypes;
  1 -- * Meal;
/**
    * holds the list of complaints that are against a specific cook
  */
  private ArrayList<Complaint> complaintsAgainst;
  /**
    * hold the list of meals that a cook offers
  */
  private ArrayList<Meal> menu;
  private HashMap<Meal, Integer> ratings;
  /**
    * list of the types of cuisine that a cook can offer
  */
  private ArrayList<CuisineTypes> cuisine;
  /**
    * this is to determine if an account is suspended. an Admin will be able to modify this with their
    * access.
  */
  private boolean suspension;


  /**
    *
    * @param username
    * @param password
```

```
   * @param name
   * @param address
   *
*/

public Cook(String email, String username, String password, String name, Address address) {
  super(email, username, password, name, address, AccountType.COOK);
  this.suspension = true;
}

public Cook(){

}

public void addCuisineType(CuisineTypes type) {
  cuisine.add(type);
}

public ArrayList<CuisineTypes> getCuisineTypes() {
  return cuisine;
}

public void addComplaints(Complaint complaintMsg) {
  complaintsAgainst.add(complaintMsg);
}

public ArrayList<Complaint> getComplaints() {
  return complaintsAgainst;
}
```

```
public void addMeal(Meal meal) {
  menu.add(meal);
}

public void removeMeal(Meal meal) {
  menu.remove(meal);
}

public ArrayList<Meal> getMeal() {
  return menu;
}

public void addRating(Meal meal, Integer rating) {
  if (!ratings.containsKey(meal)) {
    ratings.put(meal, rating);
  }
  else {
    ratings.replace(meal, rating);
  }
}
public boolean isSuspension() {return (suspension = true);}

public boolean endSuspension() {return (suspension = false);}

public HashMap<Meal, Integer> getRatings() {
  return ratings;
}
```

```
}
class CookSignUpScreen {

  private Button homeBtn;
  private Button registerCookBtn;

  EditText cookFirstName;
  EditText cookLastName;
  EditText password;
  EditText email;
  EditText username;

  EditText postal;
  EditText street;
  EditText city;
  EditText country;

  private FirebaseAuth mAuth;

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

    homeBtn = findViewById(R.id.homeButton);
    mAuth = FirebaseAuth.getInstance();
    registerCookBtn = findViewById(R.id.registerationCompleteButton);
    cookFirstName = findViewById(R.id.inputNameCook);
    cookLastName = findViewById(R.id.inputSurnameCook);
```

```
  password = findViewById(R.id.inputPasswordCook);
  email = findViewById(R.id.inputEmailCook);
  postal = findViewById(R.id.inputPostalAddressCook);
  street = findViewById(R.id.inputStreetAddressCook);
  city = findViewById(R.id.inputCityAddressCook);
  country = findViewById(R.id.inputCountryAddressCook);
  username = findViewById(R.id.inputUsernameCook);

  homeBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
      homeNavigator();
    }
  }
  );

  registerCookBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
      registerCookBtn();
    }
  }
  );
}

public void homeNavigator(){
  Intent intent = new Intent(this, MainActivity.class);
  startActivity(intent);
}
```

```
  public void registerCookBtn(){
    String emailReg = email.getText().toString();
    String name = cookFirstName.getText().toString() +" "+ cookLastName.getText().toString();
    Address address = new Address(street.getText().toString(), city.getText().toString(),
country.getText().toString(), postal.getText().toString());
    String user = username.getText().toString();
    String pass = password.getText().toString();

    if(cookFirstName.getText().toString().isEmpty()){
      cookFirstName.setError("Full name is required");
      cookFirstName.requestFocus();
      return;
    }
    if(cookLastName.getText().toString().isEmpty()){
      cookLastName.setError("Last name is required");
      cookLastName.requestFocus();
      return;
    }
    if(username.getText().toString().isEmpty()){
      username.setError("Username name is required");
      username.requestFocus();
      return;
    }
    if(password.getText().toString().isEmpty()){
      password.setError("Password is required");
      password.requestFocus();
      return;
    }
```

```
if(pass.length() < 6){
  password.setError("Password needs to be longer than 6 characters");
  password.requestFocus();
  return;
}
if(email.getText().toString().isEmpty()){
  email.setError("Email is required");
  email.requestFocus();
  return;
}
if(!(emailReg).matches("[a-zA-Z0-9._-]+@[a-z]+\\.+[a-z]+")){
  street.setError("Not a valid email type eg. user_name@gmail.com");
  street.requestFocus();
  return;
}
if(street.getText().toString().isEmpty()){
  street.setError("Street name is required");
  street.requestFocus();
  return;
}
if(city.getText().toString().isEmpty()){
  city.setError("City name is required");
  city.requestFocus();
  return;
}
if(country.getText().toString().isEmpty()){
  country.setError("Country name is required");
  country.requestFocus();
  return;
```

```
    }
    if(postal.getText().toString().isEmpty()){
      postal.setError("Postal code is required");
      postal.requestFocus();
      return;
    }



    Cook cook = new Cook(emailReg, user, pass, name, address);


    mAuth.createUserWithEmailAndPassword(emailReg, pass).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
      @Override
      public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
          Cook cook = new Cook(emailReg, user, pass, name, address);


          FirebaseDatabase.getInstance().getReference("accounts")
                .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
          .setValue(cook).addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
              if (task.isSuccessful()){
                Toast.makeText(CookSignUpScreen.this, "Congrats! You have been registered. Please go to the
home screen and login.", Toast.LENGTH_SHORT).show();
                textClear();
              }
            }
          }
```

```
        );
       }
     }
   }
   );
  }

  public void textClear(){
   cookFirstName.getText().clear();
   cookLastName.getText().clear();
   street.getText().clear();
   city.getText().clear();
   country.getText().clear();
   postal.getText().clear();
   username.getText().clear();
   password.getText().clear();
  }
}
class CuisineTypes {
  // Holds the types of dishes that our app supports

  ITALIAN, INDIAN, BURGERS, SANDWHICHES, VEGETARIAN, MEAT, DRINKS, BAKERY, VEGAN, CHINESE,
MEXICAN, JAPANESE,
  VIETNAMESE, THAI, KOREAN, CARRIBEAN, FRENCH, BREAKFAST, GREEK;
}
class LoggedInScreen {

  private FirebaseUser user;
  private DatabaseReference reference;
```

```
private String userID;
private Button logOutButton;

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

 user = FirebaseAuth.getInstance().getCurrentUser();
 reference = FirebaseDatabase.getInstance().getReference("accounts");
 userID = user.getUid();


 final TextView userRole = findViewById(R.id.roleSpecifier);

 logOutButton = findViewById(R.id.logOutButton);
 logOutButton.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
   logOut();
  }
 }
 );

 reference.child(userID).addListenerForSingleValueEvent(new ValueEventListener() {
  @Override
  public void onDataChange(@NonNull DataSnapshot snapshot){
   Account userProfile = null;
   if (snapshot.getValue(Client.class) != null) {
```

```
    userProfile = snapshot.getValue(Client.class);
   }
   else if (snapshot.getValue(Cook.class) != null) {
    userProfile = snapshot.getValue(Cook.class);
   }
   else if (snapshot.getValue(Administrator.class) != null){
    userProfile = snapshot.getValue(Administrator.class);
   }

   if (userProfile != null) {
    if (userProfile.getType() == AccountType.CLIENT){
     userRole.setText("You are signed in as a client");
    }
    else if (userProfile.getType() == AccountType.COOK){
     userRole.setText("You are signed in as a cook");
    }
    else if (userProfile.getType() == AccountType.ADMIN){
     launchAdmin();
    }
   }
   else{
    userRole.setText("Uh Oh! Something went wrong!");
   }

  }

  @Override
  public void onCancelled(@NonNull DatabaseError error) {
```

```
    }
   }
   );
  }

  public void logOut(){
   Intent intent = new Intent(this, MainActivity.class);
   startActivity(intent);
  }

  public void launchAdmin(){
   Intent intent = new Intent(this, AdminLoggedInScreen.class);
   startActivity(intent);
  }

}
class MainActivity {

  EditText email;
  EditText pass;

  Button cookRegisterBtn;
  Button clientRegisterBtn;
  Button loginBtn;

  private FirebaseAuth mAuth;

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

```
super.onCreate(savedInstanceState);
requestWindowFeature(Window.FEATURE_NO_TITLE);
getSupportActionBar().hide();
setContentView(R.layout.activity_main);

mAuth = FirebaseAuth.getInstance();

email = findViewById(R.id.emailInputField);
pass = findViewById(R.id.passwordInputField);

cookRegisterBtn = findViewById(R.id.registerCookButton);
cookRegisterBtn.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
    cookRegisterNavigator();
  }
}
);

clientRegisterBtn= findViewById(R.id.registerClientButton);
clientRegisterBtn.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
    clientRegisterNavigator();
  }
}
);

loginBtn = findViewById(R.id.loginButton);
```

```
  loginBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
      loginUser();
    }
  }
  );
}
   public void launchAdmin(){
      Intent intent = new Intent(this, AdminLoggedInScreen.class);
      startActivity(intent);
  }
public void cookRegisterNavigator() {
  Intent intent = new Intent(this, CookSignUpScreen.class);
  startActivity(intent);
}

public void clientRegisterNavigator() {
  Intent intent = new Intent(this, ClientSignUpScreen.class);
  startActivity(intent);
}

public void loginNavigate(){
  Intent intent = new Intent(this, LoggedInScreen.class);
  startActivity(intent);
}

public void loginUser(){
  String emailLog = email.getText().toString();
```

```
   String password = pass.getText().toString();


   if(email.getText().toString().isEmpty()){
    email.setError("Email is required");
    email.requestFocus();
    return;
   }
   if(!(emailLog).matches("[a-zA-Z0-9._-]+@[a-z]+\\.+[a-z]+")){
    email.setError("Not a valid email type eg. user_name@gmail.com");
    email.requestFocus();
    return;
   }
   if(pass.getText().toString().isEmpty()){
    pass.setError("Password is required");
    pass.requestFocus();
    return;
   }


   mAuth.signInWithEmailAndPassword(emailLog, password).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
     if(task.isSuccessful()){
      loginNavigate();
     }
     else{
      Toast.makeText(MainActivity.this, "Wrong credentials. Input correct login info.",
Toast.LENGTH_SHORT).show();
     }
```

```
      }
    }
    );
  }

}
class MenuMealList {

    private Activity context;
    List<Meal> meals;

    public MenuMealList(Activity context, List<Meal> meals){
        super(context, R.layout.activity_menu_meal_list, meals);
        this.context = context;
        this.meals = meals;
    }

    public View getView(int position, View convertView, ViewGroup parent){
        LayoutInflater inflater = context.getLayoutInflater();
        View listViewMenuMeals = inflater.inflate(R.layout.activity_menu_meal_list,null,true);

        TextView mealName = (TextView) listViewMenuMeals.findViewById(R.id.menuMealName);
        TextView mealPrice = (TextView) listViewMenuMeals.findViewById(R.id.menuMealPrice);

        Meal meal = meals.get(position);

        mealName.setText("Meal Name: " + meal.getName());

        mealPrice.setText("Price : " + meal.getPrice());
```

```
        return listViewMenuMeals;
    }

}
class Meal {

    String types;
    String name;
    double price;
    String ingredients;
    String allergens;
    String description;
    String cookAssignedID;
    String id;



    /**
     *
     * @param name holds the name of the meal as assigned to by a Cook
     * @param price holds the price as assigned to by a Cook
     * @param types holds the type the meal falls under within the dish types available in the
CuisineTypes enum
     *
     * This class is the core of the app, as this is what the Client instances will be ordering from the Cook
instances
     * Precondition: types is a parameter that is found in CuisineTypes
     *
     */
    public Meal(String name, String ingredients, String allergens,
```

```
        String description, double price, String types, String cookAssignedID, String id) {
    this.name = name;
    this.ingredients = ingredients;
    this.allergens = allergens;
    this.description = description;
    this.price = price;
    this.types = types;
    this.cookAssignedID = cookAssignedID;
    this.id = id;
}
/**
 * empty constructor, items can be added later
 */
public Meal(){

}
/**
 *
 * @return the name of the specific meal
 */
public String getName() {
    return name;
}
/**
 * sets the name of the meal, it is resonable to assume that the names of meals may be updated
 * or changed
 * @param name
 */
public void setName(String name) {
```

```
    this.name = name;
}
/**
 * prices may be updated and this method will allow for the price to be updated
 * @param price
 */
public void setPrice(double price) {
    this.price = price;
}
/**
 *
 * @return the price of a specific meal
 */
public double getPrice() {
    return price;
}
/**
 *
 * @return gets the list of cuisineTypes associated with a specific meal
 */
public String getTypes() {
    return types;
}
/**
 *
 * @return the complete list of ingredients
 */
public String getIngredients() {
    return ingredients;
```

```
}
/**
 *
 * @return the list of allergens associated with a meal
 */
public String getAllergens(){
   return allergens;
}
/**
 * adds a new allergen to a meal
 * @param aller
 * @return
 */
public void setAllergens(String aller){
   this.allergens = aller;
}

public void setDescription(String theText){
   this.description = theText;
}
/**
 *
 * @return the description of a specific meal.
 */
public String getDescription(){
   return description;
}
/**
 * adds an additional type to the list of cuisines
```

```
     * @param types
     */
    public void setTypes(String types) {
        this.types = types;
    }

    public String getCookAssignedID(){
       return this.cookAssignedID;
    }

    public void setCookAssignedID(String cookAssignedID){
       this.cookAssignedID = cookAssignedID;
    }

    public String getId(){
       return this.id;
    }

    public void setId(String id){
       this.id = id;
    }

}

class PermanentSuspensionScreen {
  void onCreate(Bundle savedInstanceState) {}
}
```