Quik Closet

7TH ISSUE
JULY 2021

FINAL REPORT

SUMMER 2021
COMP 380L
GROUP 5
JULY 15

MARMAR TAHERY
MICHAEL BUCKLEY
KRISTINA LAUE
OSHER BOUDARA
CAYLIN OCHOA

Table of Contents

Project Description

Project Overview

Quik Closet is an Android application primarily used for closet organization. This application allows the user to document daily outfit choices. Each outfit is built from pictures of their items and accessories.

Purpose of the Project

The purpose of this project is to provide an organized database of a user’s closet in order to easily see and create outfits for their day. It’s intent is to make the user’s day easier rather than going through their entire closet physically; they are able to see their items and create entire outfits using the Quik Closet app. By using our app, the user will be able to quickly access their closet, create a personalized outfit for themselves, and even save the selection to the database to view it again later.

Scope (refer to IEEE standard 830 5.1.2)

Our main goal with this project was to create an Android application that was capable of creating an organized closet for the user. Creating different personalized outfits based on their mood, weather or occasion. The app will accept pictures of clothing taken by the user and at the users option. The application will also require interfaces to photograph clothing, buttons to categorize it, and options to tell the app what clothes you’re taking or request an outfit from it, and it should only require that the phone possess a functioning camera, and a touchscreen.

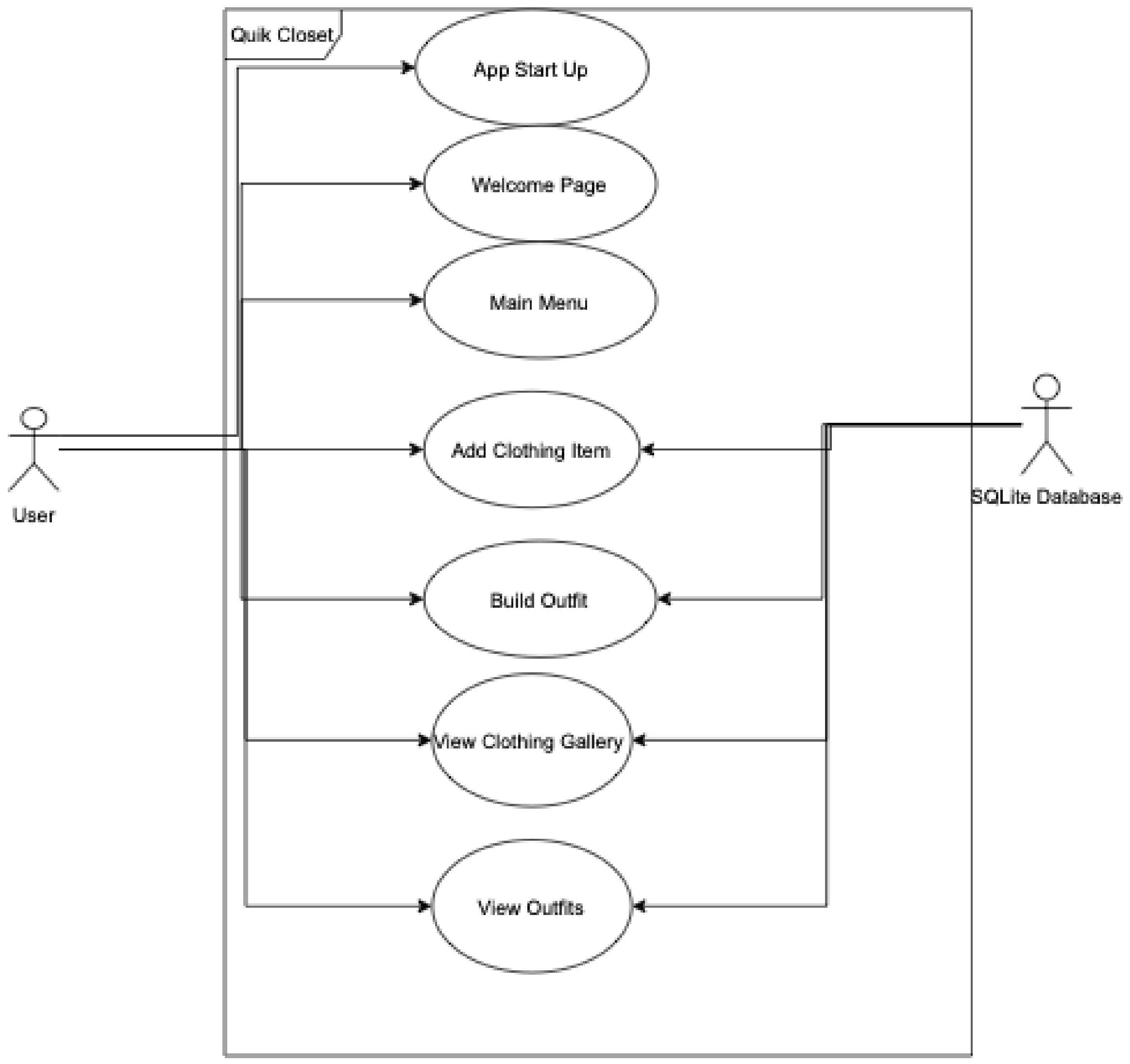
Stakeholders

- User

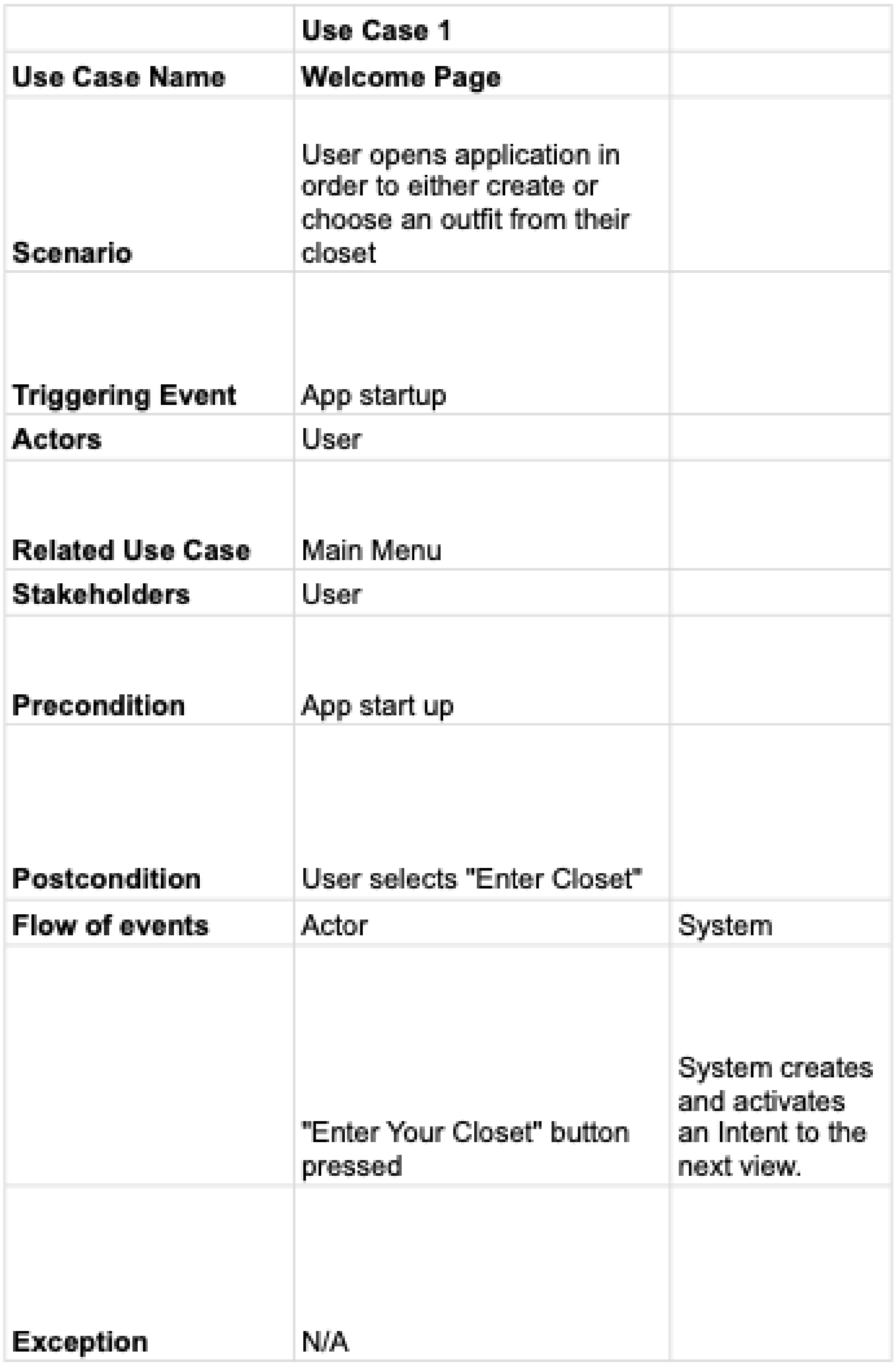
Requirements and Specification

Use Case Diagram and Descriptions

Use Case Diagram



Use Case Descriptions



	Use Case 1	
Use Case Name	**Welcome Page**	
Scenario	User opens application in order to either create or choose an outfit from their closet	
Triggering Event	App startup	
Actors	User	
Related Use Case	Main Menu	
Stakeholders	User	
Precondition	App start up	
Postcondition	User selects "Enter Closet"	
Flow of events	Actor	System
	"Enter Your Closet" button pressed	System creates and activates an Intent to the next view.
Exception	N/A	

	Use Case 2	
Use Case Name	**Main Menu**	
Scenario	User must choose which action to take next - View Clothing Gallery, Add Clothing Item, View Outfit Gallery, Build Outfit	
Triggering Event	"Enter Your Wardrobe" button is pressed	
Actors	User	
Related Use Case	View Clothing Gallery, Add Clothing Item, View Outfit Gallery, Build Outfit	
Stakeholders	User	
Precondition	User selects "Enter Closet"	
Postcondition	User must choose which action to take next - View Clothing Gallery, Add Clothing Item, View Outfit Gallery, Build Outfit	
Flow of events	Actor	
	User chooses which action to take next.	System completes request, taking the user to the desired Activity View.
Exception	N/A	

	Use Case 3	
Use Case Name	**Build Outfit**	
Scenario	User wants to build a new outfit.	
Triggering Event	User presses button in main menu that takes user to screen that allows user to build outfits	
Actors	User	
Related Use Case	Main Menu	
Stakeholders	User	
Precondition	The user would have items uploaded already to database	
Postcondition	User selects Save Outfit	
Flow of events	Actor	System
	User pushes the "Build Outfit" button	Displays screen where you choose tops, bottoms, shoes, accessories and attributes. User then presses Save Outfit and then outfit is stored into database
Exception	User has no clothes stored in database and cannot build outfit	

	Use Case 4	
Use Case Name	**Add Clothing Item**	
Scenario	User wants to add an item to the database	
Triggering Event	User presses a button to add an article of clothing, either from the main menu, or from the outfit builder	
Actors	User	
Related Use Case	Main Menu	
Stakeholders	User	
Precondition	N/A	
Postcondition	User selects Save Clothing	
Flow of events	Actor	System
	User selects Add Clothing Button	Brings up add clothing camera interface, and attributes and images are saved into database to be used later.
Exception	User must upload an image of item and select attributes, otherwise user will not be able to save item	

	Use Case 5	
Use Case Name	**View Clothing Gallery**	
Scenario	User wants to see an item they've already added	
Triggering Event	User presses a button from the closet	
Actors	User	
Related Use Case	Main Menu	
Stakeholders	User	
Precondition	User must have items stored in database	
Postcondition	User may choose item and view attributes OR select Return to Main Menu	
Flow of events	Actor	System
	User selects View Clothing Gallery	System displays clothing item with attributes.
Exception	No items were added to database, therefore leaving this as an empty closet and no gallery can be displayed	

	Use Case 6	
Use Case Name	**View Outfit Gallery**	
Scenario	User can access previously saved outfits built by user	
Triggering Event	User presses button from main menu that allows viewing of previous outfits.	
Actors	User	
Related Use Case	Main Menu	
Stakeholders	User	
Precondition	User must have built a previous outfit.	
Postcondition	After user is satisfied, user selects back through Android software which takes them back to the outfit gallery	
Flow of events	Actor	System
	User selects View Outfit Gallery	System displays outfit gallery. If outfit is selected, outfit attribute is displayed
Exception	No previous outfits have been saved to gallery	

User Requirements

- QSA is targeted towards users with little to no technical experience. The ability of an unskilled user to use QSA is critical to the software's success.
- User must possess an Android phone in order to use
- Basic comprehension skills
- Users are only required to press buttons and know how to operate a camera.
  - This is to avoid app complexity

Functional Requirements

- Start Up
  - User presses icon on their phone to load the application
- Welcome Page
  - Welcome page with app log displayed
  - Users can proceed to the main menu and closet from here by clicking the "Enter your closet" button.
- Display Menus
  - Displays a menu to the user with 4 buttons: Add Clothing Item, Build Outfit, View Clothing Gallery, View Outfit Gallery.
  - If the user presses a button, they are brought to a visual interface
- Add Clothing Item
  - The application displays an interface where the user will press buttons to classify the article of clothing being uploaded.
  - Each of these buttons hold an attribute such as type and category of clothing, mood, weather (temperature and precipitation), task and color. Users can set values to NULL by pressing the specified button that declares no attribute be saved. Users can select and deselect attributes if modifications are needed.
  - Users will be prompted to upload an image of clothing either by camera view or by uploading from the user's photo gallery. Users can select and deselect images if modifications are needed which will clear the image and the user can then reupload the image.
  - Attributes and the image of the clothing article are stored in SQLite Database. They can be viewed in the gallery or used to build outfits later.
  - Users can return to the main menu.
- Build Outfit
  - Page displays with options such as top, bottom, shoes and accessories where User can click on these options.
  - When the user clicks on either of the options above, a scrollable table is displayed where the user picks which clothing item they want in the outfit. This is done for tops, bottoms, shoes and accessories. Users can also deselect items by clicking on the image.
  - Users can also tie attributes such as mood, temperature, precipitation, color and task to the outfit that they are building. These can also be selected and deselected if the user would like to make modifications.

    - ➢ After the user selects a complete outfit, they can save the outfit into the database to be viewed later in the outfits portion of the application.
    - ➢ Users can return to the main menu.
- ❖ View Clothing Gallery
    - ➢ Gallery of clothing is displayed for the user to view. It is displayed as a scrollable table.
    - ➢ The page will have four rows: tops in first row, bottoms in second, shoes in third and accessories in fourth row.
    - ➢ Users can press on a clothing article and it will display that clothing article’s attributes.
    - ➢ Users can step back into the clothing gallery using the Android return button.
    - ➢ Users can return to the main menu.
- ❖ View Outfit Gallery
    - ➢ The gallery of outfits is loaded from SQLite DB as a scrollable table. Each row is an outfit containing the images of the clothing that comprise it.
    - ➢ The user can select an outfit, and the information tied to the outfit is brought up as a view.
    - ➢ User can step back into outfit gallery using the Android return button
    - ➢ Users can return to the main menu using the created button.

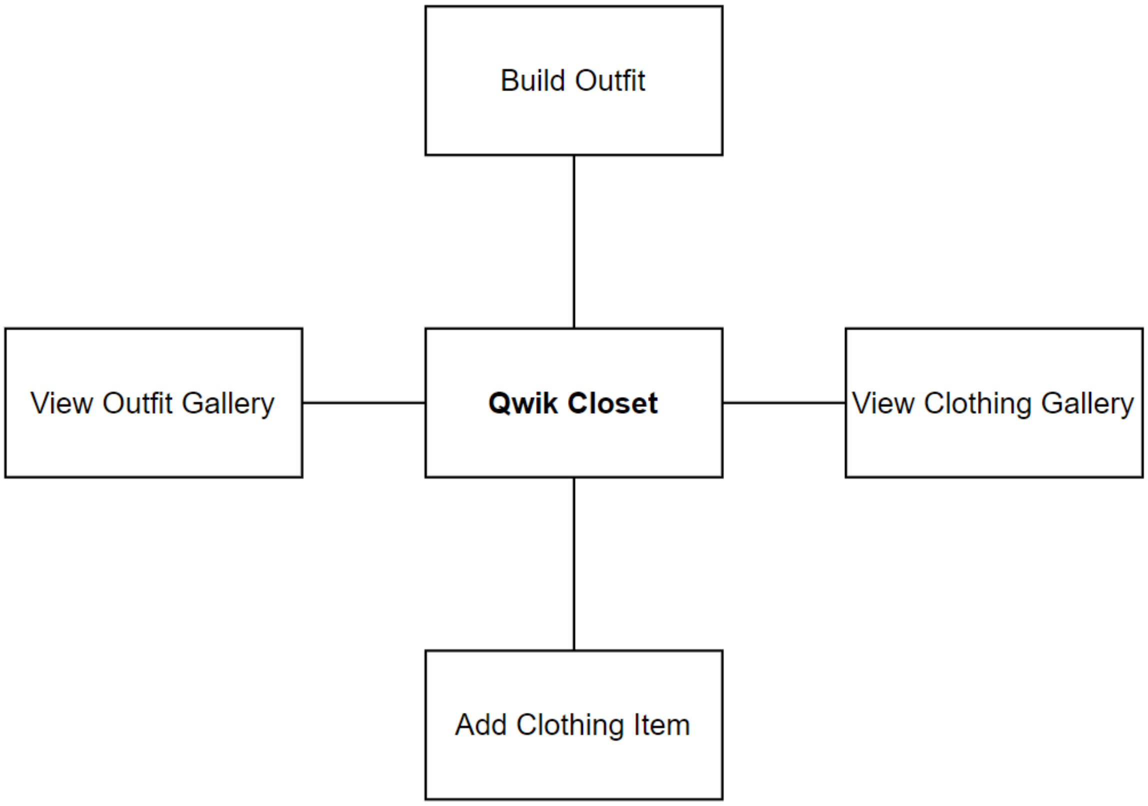
Non-functional Requirements

- ❖ Performance Requirements
    - ➢ The app should be able to store a few hundred pictures and clothing articles.
    - ➢ The app should be able to retrieve any pictures and clothing articles from the database.
    - ➢ The app should be able to store a few dozen outfits in the database.
    - ➢ The app should be able to retrieve outfits.
    - ➢ The above mentioned actions should not take more than 5 seconds to perform on a moderate strength android phone, even while containing hundreds of clothes.
- ❖ Security Requirements
    - ➢ The app does not require security features to achieve its function. The app is stored locally on the Android device, which, by default, encrypts its internal applications to protect them.
    - ➢ The app, at no point, connects to the internet; therefore, the app does not require additional encryption functionality to be attached to internet downloads.
- ❖ Usability Requirements
    - ➢ The user is only required to make button presses, and operate a camera. Button presses are segmented off to avoid complexity, and no more than 10 button presses should be required to do anything.

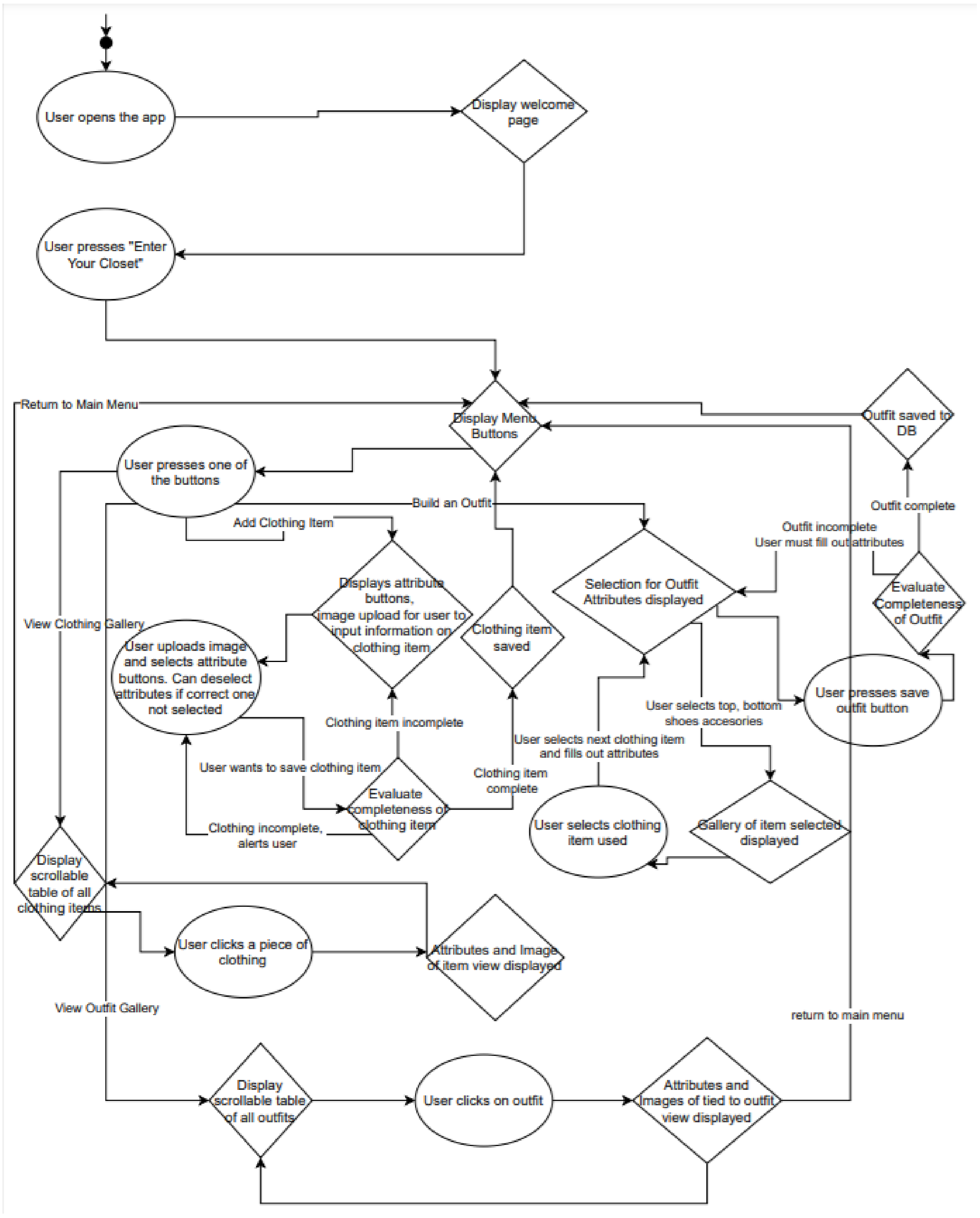
Design

System Modeling (five system models)

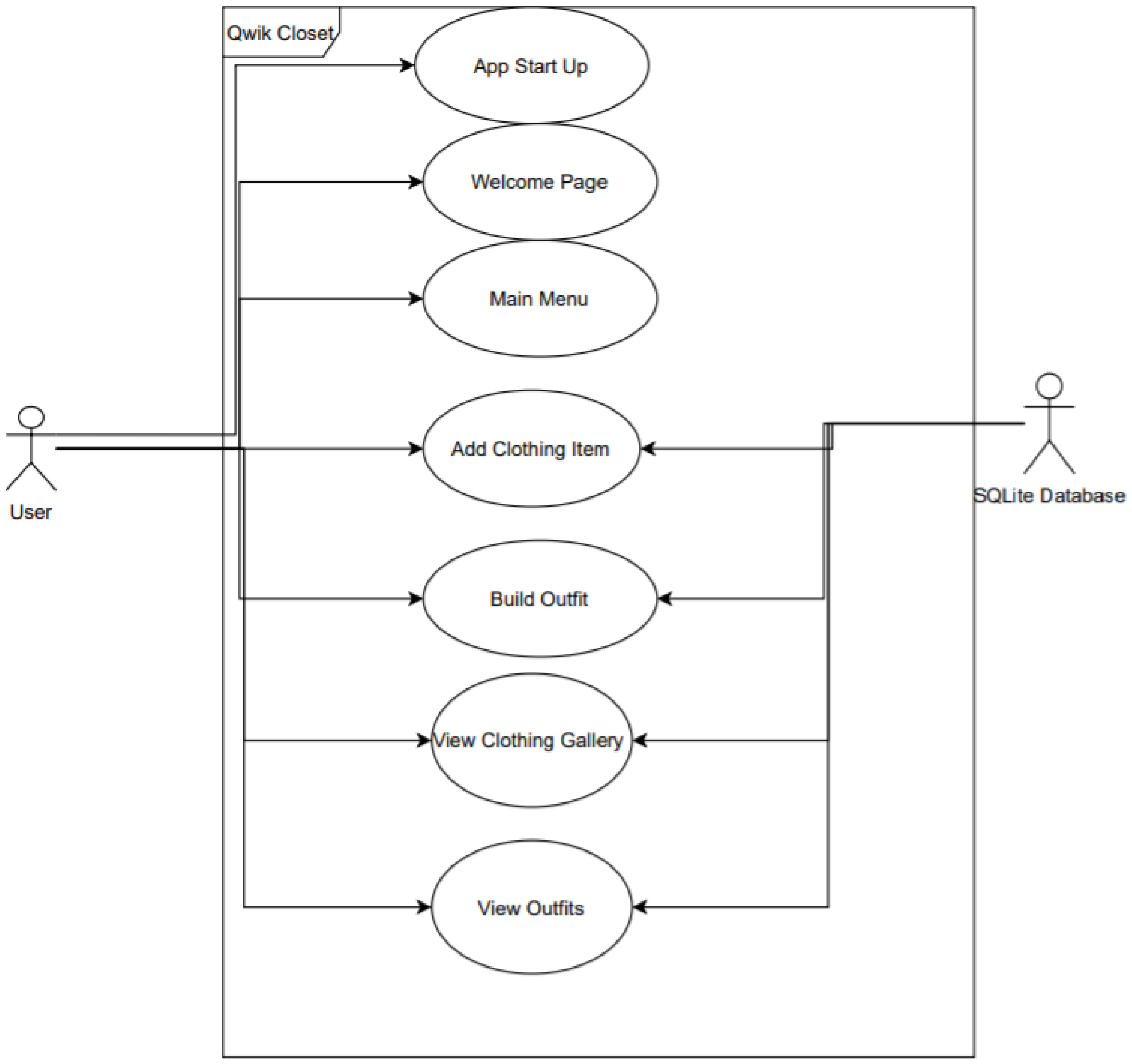
CONTEXT MODEL



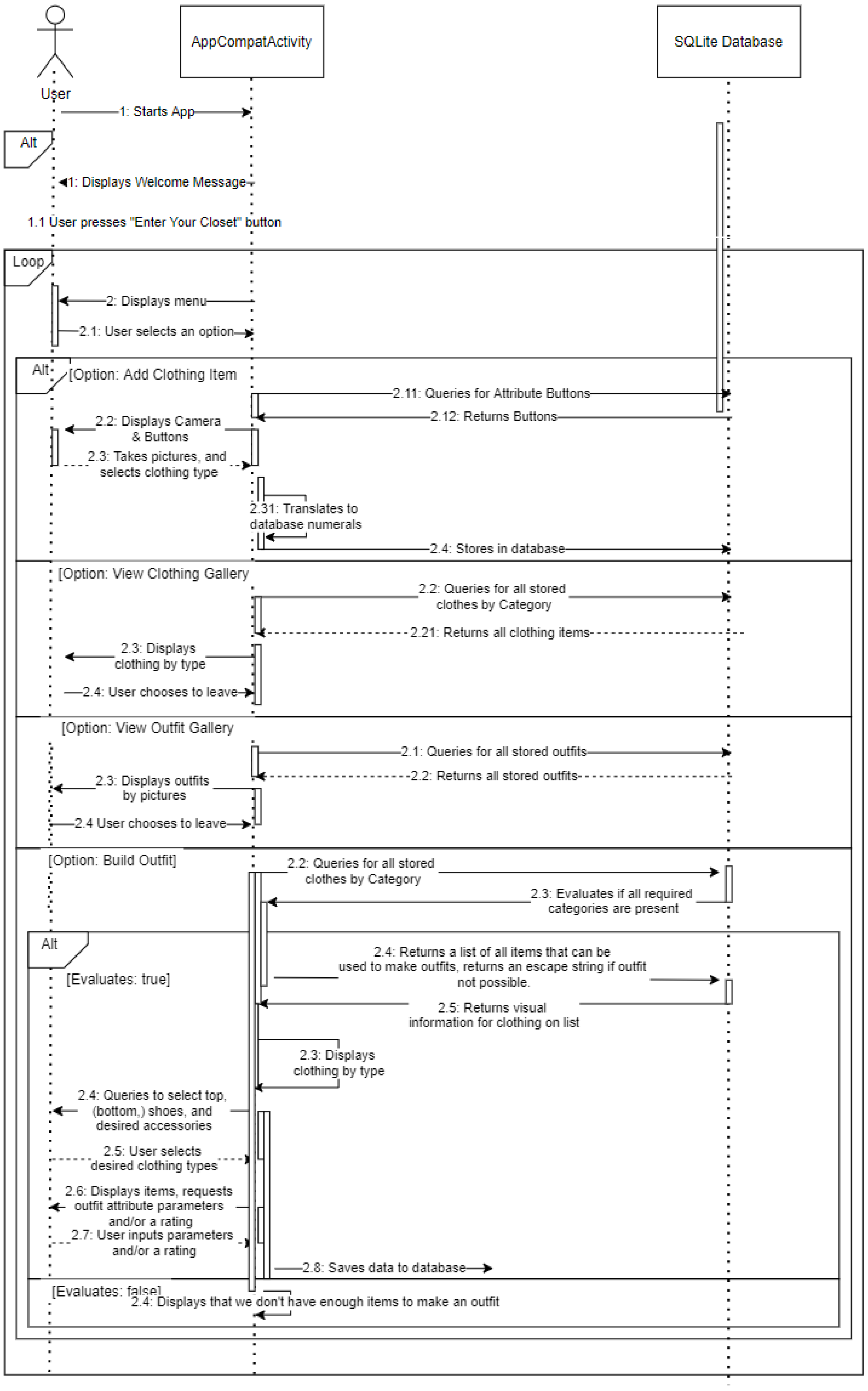
PROCESS MODEL


User opens the app
Display welcome page
User presses "Enter Your Closet"
Display Menu Buttons
Return to Main Menu
User presses one of the buttons
Build an Outfit
Add Clothing Item
View Clothing Gallery
Displays attribute buttons, image upload for user to input information on clothing item
User uploads image and selects attribute buttons. Can deselect attributes if correct one not selected
Clothing item saved
Clothing item incomplete
User wants to save clothing item
Evaluate completeness of clothing item
Clothing incomplete, alerts user
Clothing item complete
Selection for Outfit Attributes displayed
Outfit incomplete User must fill out attributes
Outfit complete
Outfit saved to DB
Evaluate Completeness of Outfit
User presses save outfit button
User selects top, bottom shoes accesories
User selects next clothing item and fills out attributes
User selects clothing item used
Gallery of item selected displayed
Display scrollable table of all clothing items
User clicks a piece of clothing
Attributes and Image of item view displayed
View Outfit Gallery
return to main menu
Display scrollable table of all outfits
User clicks on outfit
Attributes and Images of tied to outfit view displayed

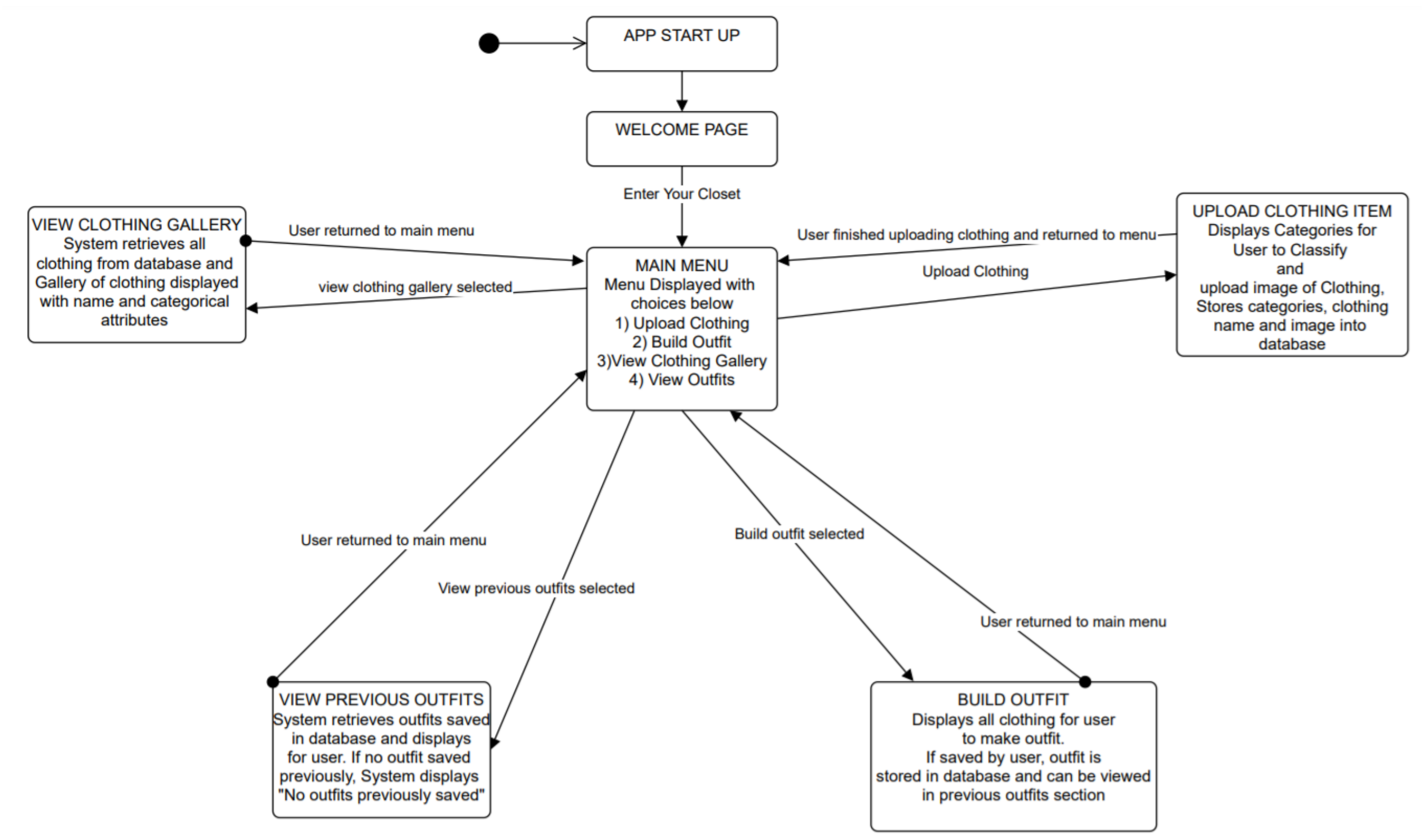
INTERACTION MODEL



INTERACTION MODEL (cont)

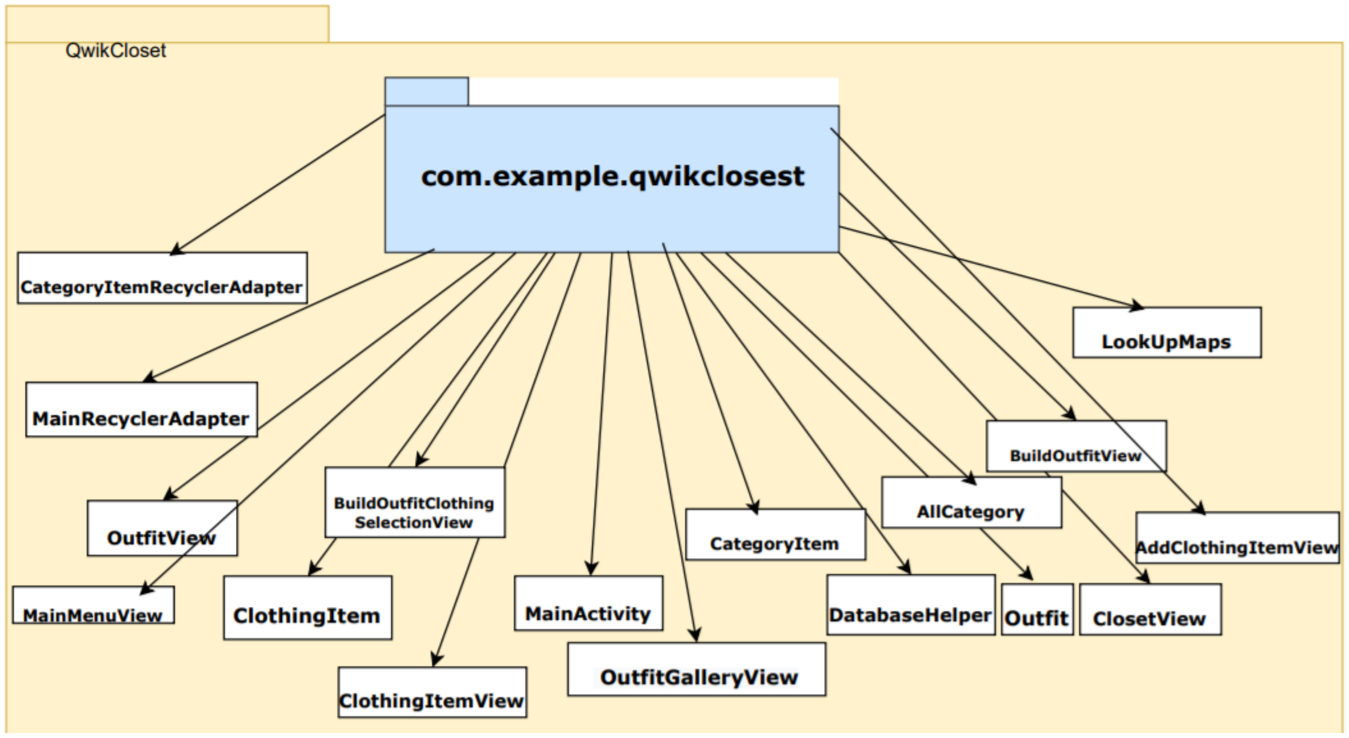

User
AppCompatActivity
SQLite Database
1: Starts App
Alt
1: Displays Welcome Message
1.1 User presses "Enter Your Closet" button
Loop
2: Displays menu
2.1: User selects an option
Alt
[Option: Add Clothing Item
2.11: Queries for Attribute Buttons
2.12: Returns Buttons
2.2: Displays Camera & Buttons
2.3: Takes pictures, and selects clothing type
2.31: Translates to database numerals
2.4: Stores in database
[Option: View Clothing Gallery
2.2: Queries for all stored clothes by Category
2.21: Returns all clothing items
2.3: Displays clothing by type
2.4: User chooses to leave
[Option: View Outfit Gallery
2.1: Queries for all stored outfits
2.2: Returns all stored outfits
2.3: Displays outfits by pictures
2.4 User chooses to leave
[Option: Build Outfit]
2.2: Queries for all stored clothes by Category
2.3: Evaluates if all required categories are present
Alt
[Evaluates: true]
2.4: Returns a list of all items that can be used to make outfits, returns an escape string if outfit not possible.
2.5: Returns visual information for clothing on list
2.3: Displays clothing by type
2.4: Queries to select top, (bottom,) shoes, and desired accessories
2.5: User selects desired clothing types
2.6: Displays items, requests outfit attribute parameters and/or a rating
2.7: User inputs parameters and/or a rating
2.8: Saves data to database
[Evaluates: false]
2.4: Displays that we don't have enough items to make an outfit

BEHAVIORAL MODEL

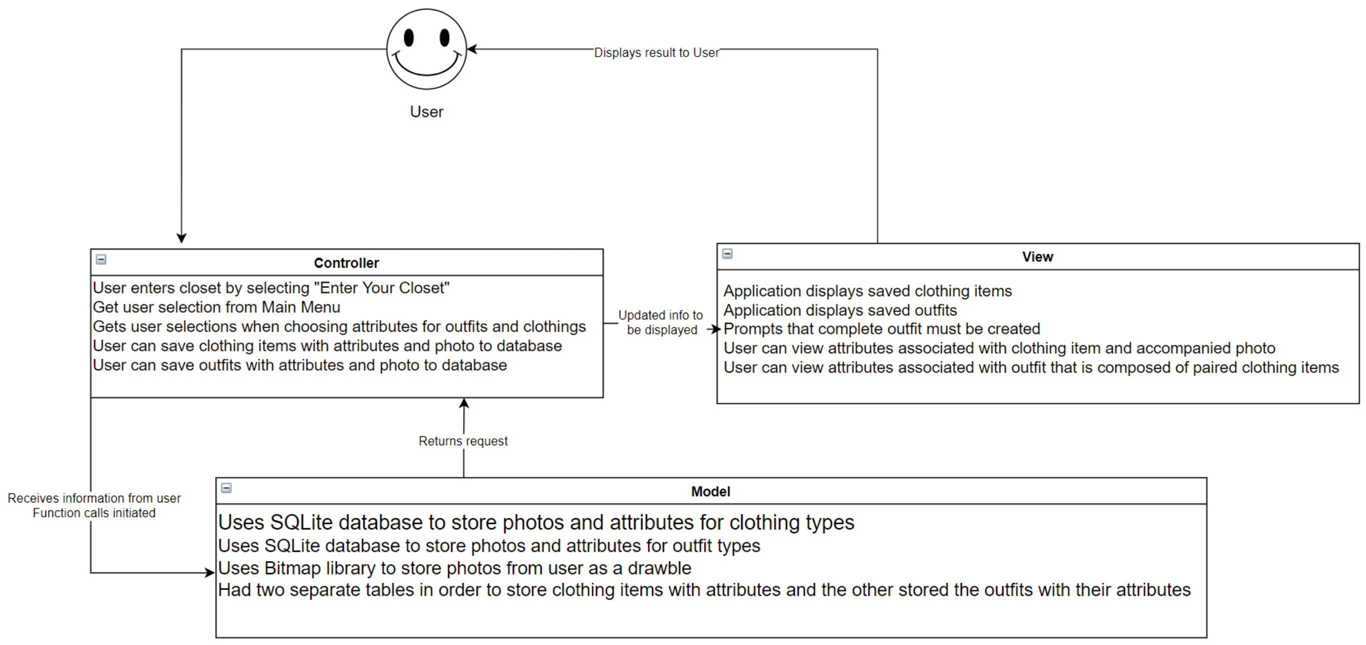


APP START UP
WELCOME PAGE
Enter Your Closet
MAIN MENU Menu Displayed with choices below 1) Upload Clothing 2) Build Outfit 3)View Clothing Gallery 4) View Outfits
VIEW CLOTHING GALLERY System retrieves all clothing from database and Gallery of clothing displayed with name and categorical attributes
User returned to main menu
view clothing gallery selected
UPLOAD CLOTHING ITEM Displays Categories for User to Classify and upload image of Clothing, Stores categories, clothing name and image into database
User finished uploading clothing and returned to menu
Upload Clothing
User returned to main menu
View previous outfits selected
Build outfit selected
User returned to main menu
VIEW PREVIOUS OUTFITS System retrieves outfits saved in database and displays for user. If no outfit saved previously, System displays "No outfits previously saved"
BUILD OUTFIT Displays all clothing for user to make outfit. If saved by user, outfit is stored in database and can be viewed in previous outfits section

STRUCTURAL MODEL

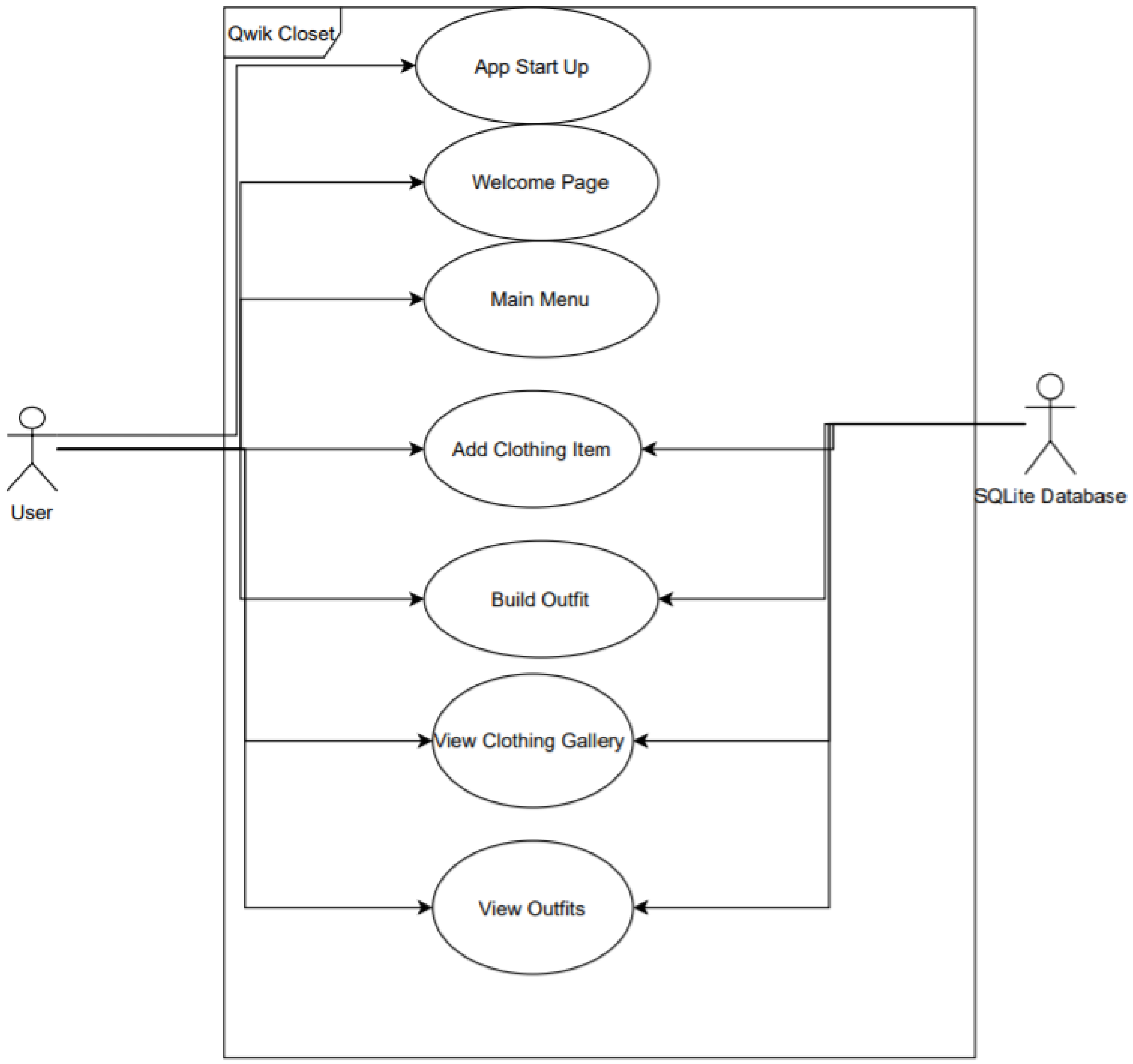


System Architecture and Patterns

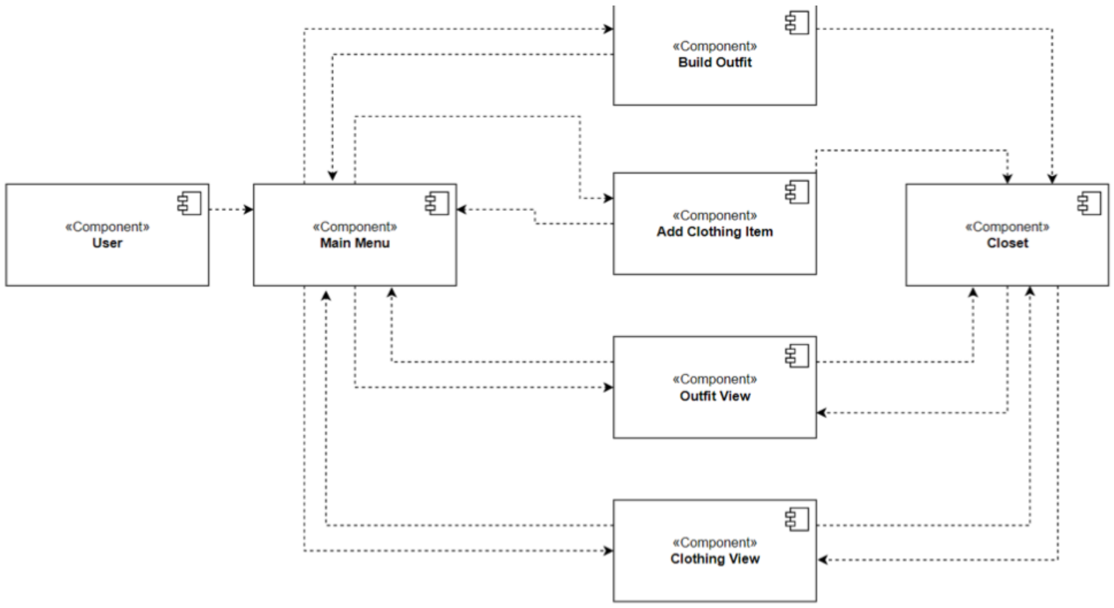

User
Displays result to User
Controller
User enters closet by selecting "Enter Your Closet"
Get user selection from Main Menu
Gets user selections when choosing attributes for outfits and clothings
User can save clothing items with attributes and photo to database
User can save outfits with attributes and photo to database
Updated info to be displayed
View
Application displays saved clothing items
Application displays saved outfits
Prompts that complete outfit must be created
User can view attributes associated with clothing item and accompanied photo
User can view attributes associated with outfit that is composed of paired clothing items
Returns request
Receives information from user
Function calls initiated
Model
Uses SQLite database to store photos and attributes for clothing types
Uses SQLite database to store photos and attributes for outfit types
Uses Bitmap library to store photos from user as a drawble
Had two separate tables in order to store clothing items with attributes and the other stored the outfits with their attributes

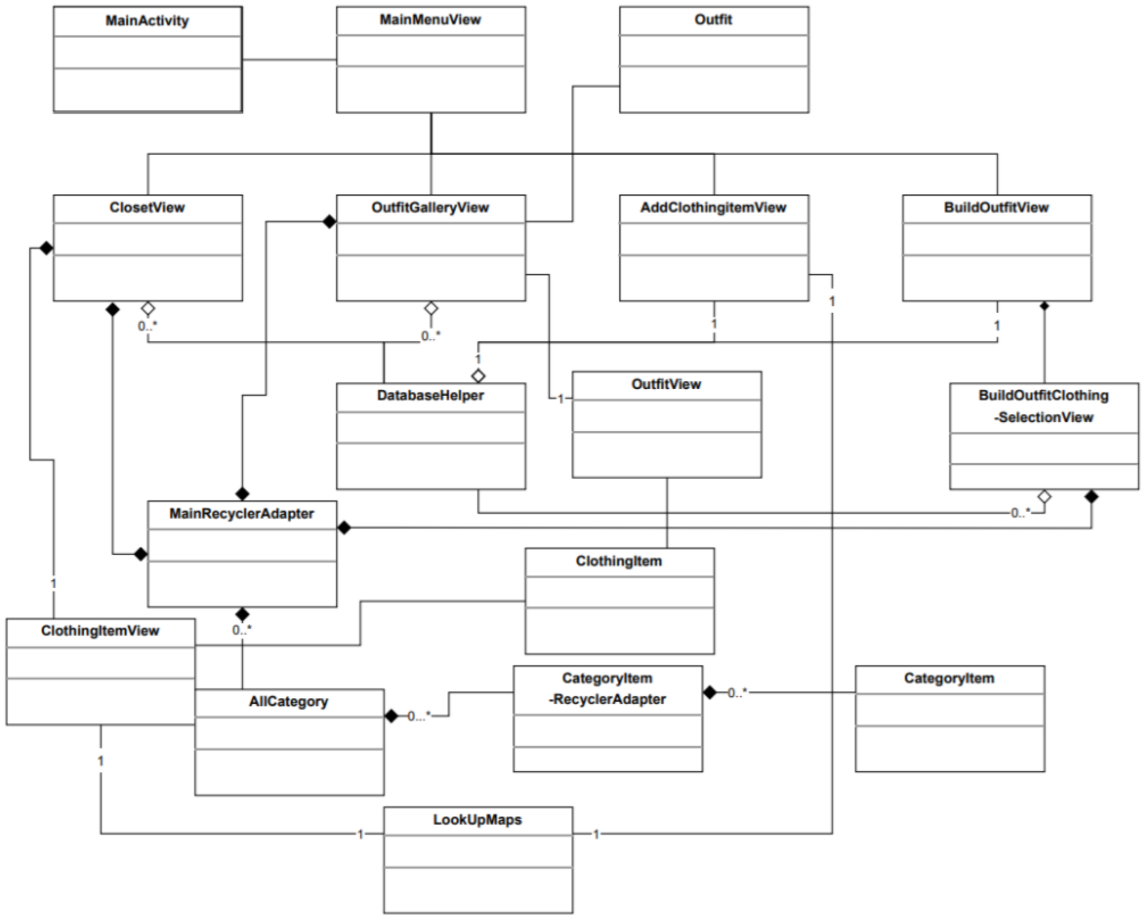
4+1 views (using UML Diagrams)

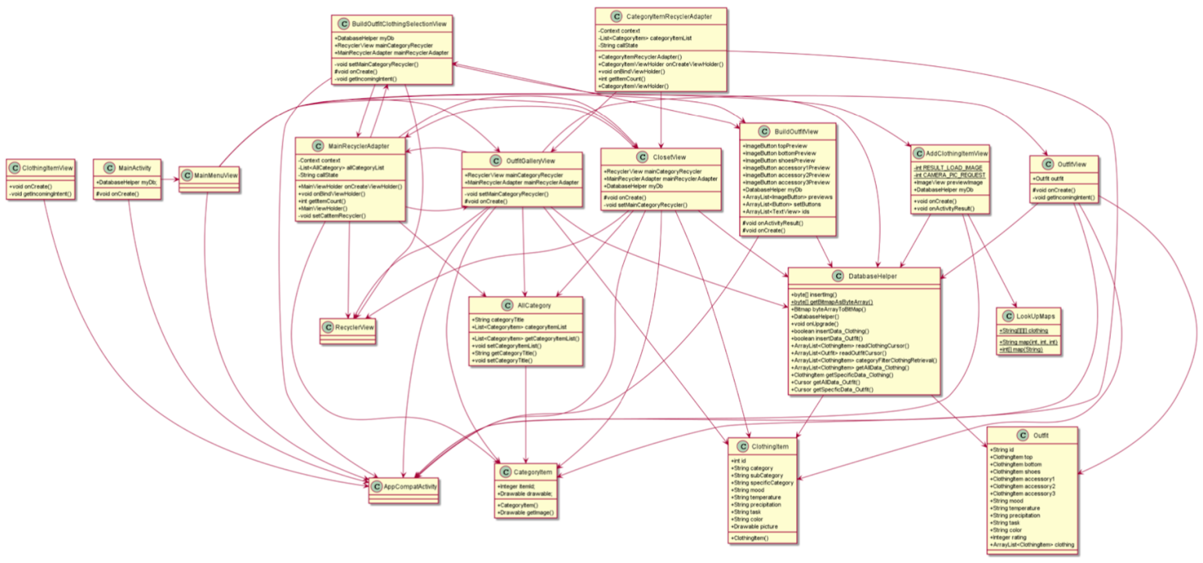
<u>USER MODEL</u>



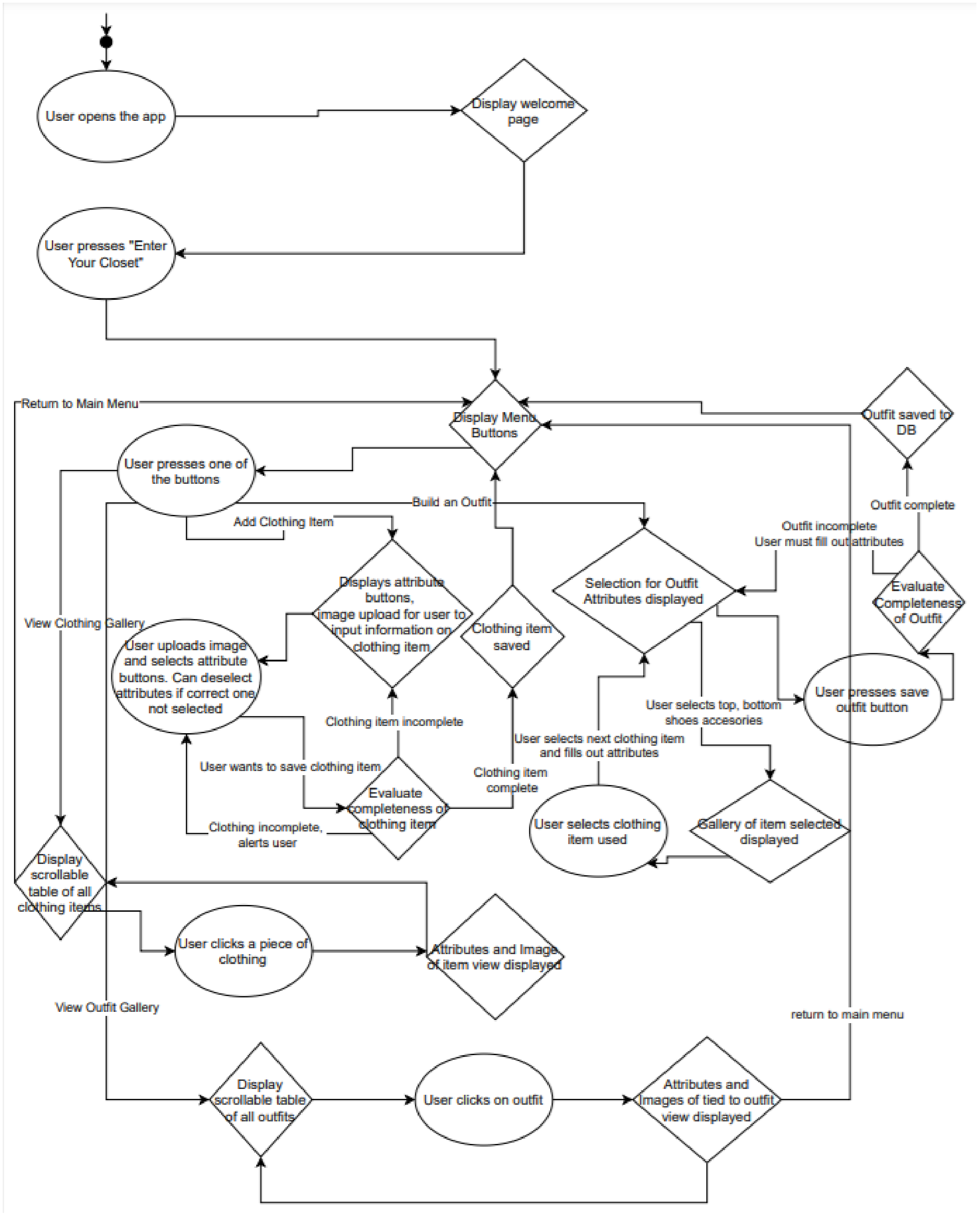
LOGICAL MODEL

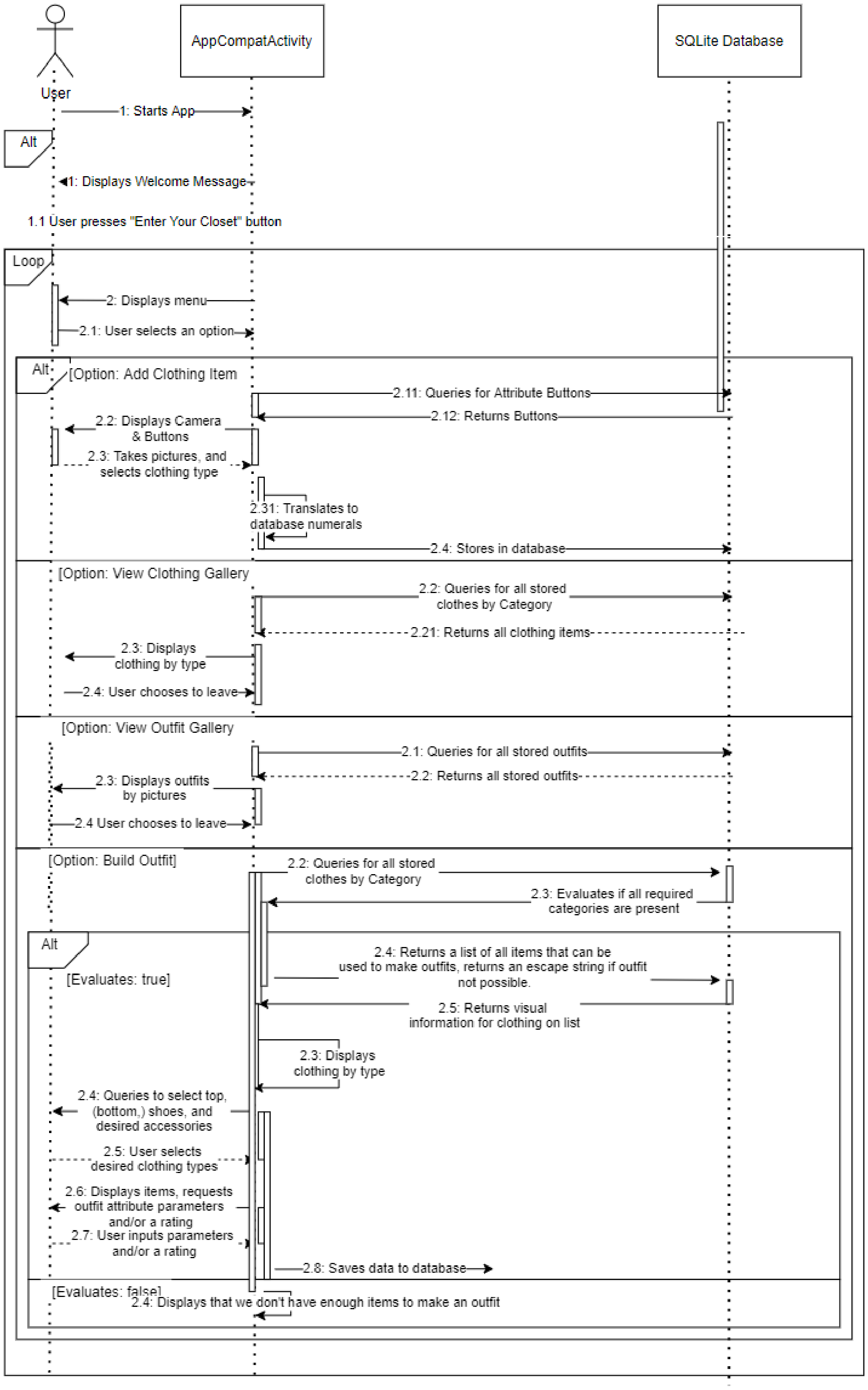





BuildOutfitClothingSelectionView
+DatabaseHelper myDb
+RecyclerView mainCategoryRecycler
+MainRecyclerAdapter mainRecyclerAdapter
-void setMainCategoryRecycler()
#void onCreate()
-void getIncomingIntent()
CategoryItemRecyclerAdapter
-Context context
-List<CategoryItem> categoryItemList
-String callState
+CategoryItemRecyclerAdapter()
+CategoryItemViewHolder onCreateViewHolder()
+void onBindViewHolder()
+int getItemCount()
+CategoryItemViewHolder()
ClothingItemView
+void onCreate()
-void getIncomingIntent()
MainActivity
+DatabaseHelper myDb;
#void onCreate()
MainMenuView
MainRecyclerAdapter
-Context context
-List<AllCategory> allCategoryList
-String callState
+MainViewHolder onCreateViewHolder()
+void onBindViewHolder()
+int getItemCount()
+MainViewHolder()
-void setCatItemRecycler()
OutfitGalleryView
+RecyclerView mainCategoryRecycler
+MainRecyclerAdapter mainRecyclerAdapter
-void setMainCategoryRecycler()
#void onCreate()
ClosetView
+RecyclerView mainCategoryRecycler
+MainRecyclerAdapter mainRecyclerAdapter
+DatabaseHelper myDb
#void onCreate()
-void setMainCategoryRecycler()
BuildOutfitView
+ImageButton topPreview
+ImageButton bottomPreview
+ImageButton shoesPreview
+ImageButton accessory1Preview
+ImageButton accessory2Preview
+ImageButton accessory3Preview
+DatabaseHelper myDb
+ArrayList<ImageButton> previews
+ArrayList<Button> setButtons
+ArrayList<TextView> ids
#void onActivityResult()
#void onCreate()
AddClothingItemView
-int RESULT_LOAD_IMAGE
-int CAMERA_PIC_REQUEST
+ImageView previewImage
+DatabaseHelper myDb
+void onCreate()
+void onActivityResult()
OutfitView
+Outfit outfit
#void onCreate()
-void getIncomingIntent()
DatabaseHelper
+byte[] insertImg()
+byte[] getBitmapAsByteArray()
+Bitmap byteArrayToBitMap()
+DatabaseHelper()
+void onUpgrade()
+boolean insertData_Clothing()
+boolean insertData_Outfit()
+ArrayList<ClothingItem> readClothingCursor()
+ArrayList<Outfit> readOutfitCursor()
+ArrayList<ClothingItem> categoryFilterClothingRetrieval()
+ArrayList<ClothingItem> getAllData_Clothing()
+ClothingItem getSpecificData_Clothing()
+Cursor getAllData_Outfit()
+Cursor getSpecficData_Outfit()
RecyclerView
AllCategory
+String categoryTitle
+List<CategoryItem> categoryItemList
+List<CategoryItem> getCategoryItemList()
+void setCategoryItemList()
+String getCategoryTitle()
+void setCategoryTitle()
LookUpMaps
+String[][][] clothing
+String map(int, int, int)
+int[] map(String)
AppCompatActivity
CategoryItem
+Integer itemId;
+Drawable drawable;
+CategoryItem()
+Drawable getImage()
ClothingItem
+int id
+String category
+String subCategory
+String specificCategory
+String mood
+String temperature
+String precipitation
+String task
+String color
+Drawable picture
+ClothingItem()
Outfit
+String id
+ClothingItem top
+ClothingItem bottom
+ClothingItem shoes
+ClothingItem accessory1
+ClothingItem accessory2
+ClothingItem accessory3
+String mood
+String temperature
+String precipitation
+String task
+String color
+Integer rating
+ArrayList<ClothingItem> clothing

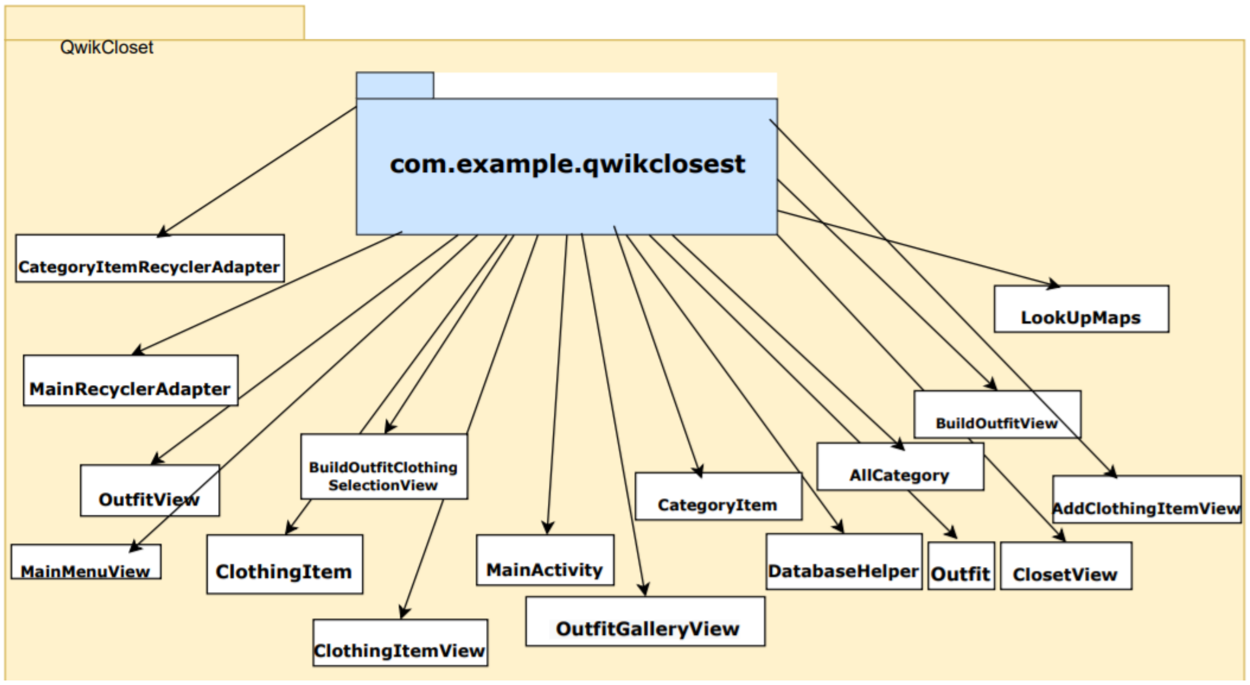
PROCESS MODEL


User opens the app
Display welcome page
User presses "Enter Your Closet"
Display Menu Buttons
Return to Main Menu
User presses one of the buttons
Build an Outfit
Add Clothing Item
View Clothing Gallery
View Outfit Gallery
Displays attribute buttons, image upload for user to input information on clothing item
User uploads image and selects attribute buttons. Can deselect attributes if correct one not selected
Clothing item saved
Clothing item incomplete
User wants to save clothing item
Evaluate completeness of clothing item
Clothing incomplete, alerts user
Clothing item complete
Selection for Outfit Attributes displayed
User selects top, bottom shoes accesories
User selects next clothing item and fills out attributes
Gallery of item selected displayed
User selects clothing item used
User presses save outfit button
Evaluate Completeness of Outfit
Outfit incomplete User must fill out attributes
Outfit complete
Outfit saved to DB
Display scrollable table of all clothing items
User clicks a piece of clothing
Attributes and Image of item view displayed
Display scrollable table of all outfits
User clicks on outfit
Attributes and Images of tied to outfit view displayed
return to main menu



AppCompatActivity
SQLite Database
User
1: Starts App
Alt
1: Displays Welcome Message
1.1 User presses "Enter Your Closet" button
Loop
2: Displays menu
2.1: User selects an option
Alt
[Option: Add Clothing Item
2.11: Queries for Attribute Buttons
2.12: Returns Buttons
2.2: Displays Camera & Buttons
2.3: Takes pictures, and selects clothing type
2.31: Translates to database numerals
2.4: Stores in database
[Option: View Clothing Gallery
2.2: Queries for all stored clothes by Category
2.21: Returns all clothing items
2.3: Displays clothing by type
2.4: User chooses to leave
[Option: View Outfit Gallery
2.1: Queries for all stored outfits
2.2: Returns all stored outfits
2.3: Displays outfits by pictures
2.4 User chooses to leave
[Option: Build Outfit]
2.2: Queries for all stored clothes by Category
2.3: Evaluates if all required categories are present
Alt
[Evaluates: true]
2.4: Returns a list of all items that can be used to make outfits, returns an escape string if outfit not possible.
2.5: Returns visual information for clothing on list
2.3: Displays clothing by type
2.4: Queries to select top, (bottom,) shoes, and desired accessories
2.5: User selects desired clothing types
2.6: Displays items, requests outfit attribute parameters and/or a rating
2.7: User inputs parameters and/or a rating
2.8: Saves data to database
[Evaluates: false]
2.4: Displays that we don't have enough items to make an outfit

DEVELOPMENT MODEL



PHYSICAL MODEL

«Device»
Android Device

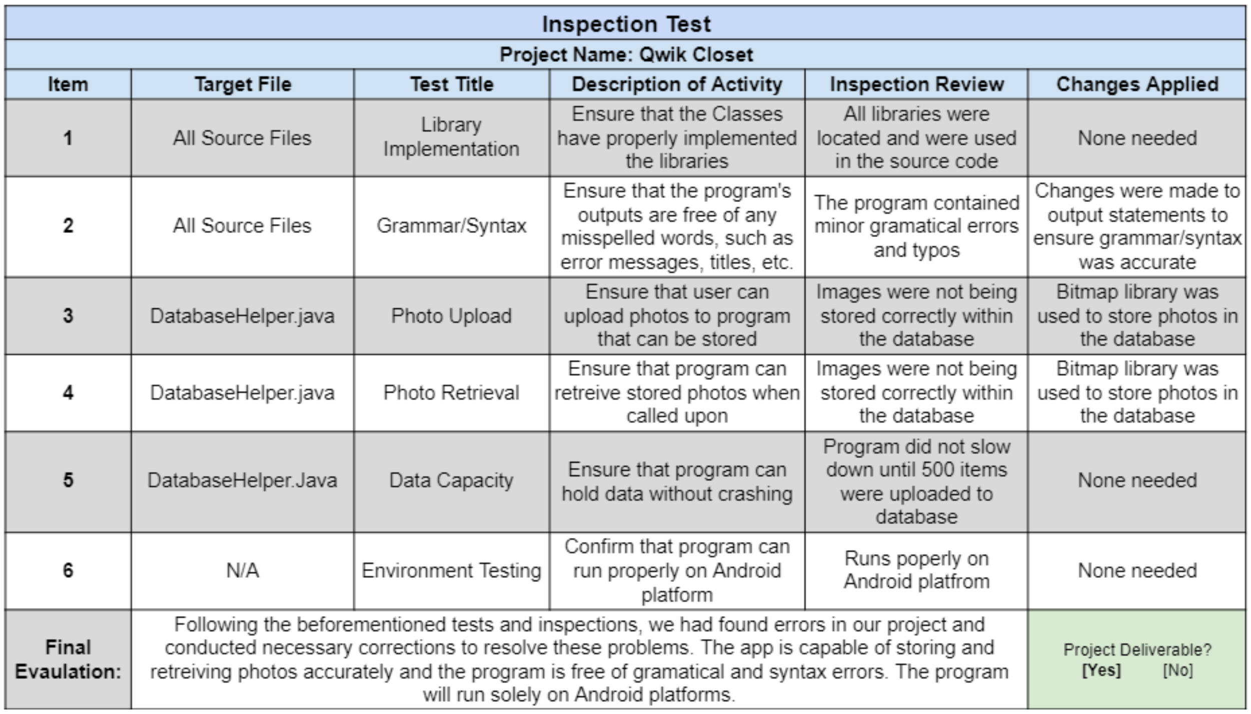
Detailed design principles (five principles)

We have chosen to retain the current implementation of our diagrams shown above. We felt that no application of SOLID principles would be capable of extending clarity or adding meaningful depth or functionality to our project. The primary reason we feel as such is: our program, though it uses classes, does not use inheritance principles. Our program interacts between its classes either through the use of aggregation within the MainActivity class, or statically through the MainActivity class. Because we do not use inheritance principles, attempting to shore up inheritance in any location within our program is futile. Additionally, attempting to limit the use of our classes to single-use likewise proves confusing and inefficient. The only classes we have that fulfill multiple functions are the MainActivity, and the DatabaseHelper. Limiting the scope of the first would cripple its functionality as the Class-Responsibility-Collaborator, which is intended to be complicated in its execution of the intended processes. Limiting the scope of the second would entail separating the processes necessary to read and write to the database, and require additional implementation to ensure that the same database is being read from and written to. Such processes are, again, more confusing than they are helpful, so we elect not to engage in such.

Testing

Non-execution testing (walk-through or inspection)

Inspection Test



Inspection Test					
Project Name: Qwik Closet					
Item	**Target File**	**Test Title**	**Description of Activity**	**Inspection Review**	**Changes Applied**
1	All Source Files	Library Implementation	Ensure that the Classes have properly implemented the libraries	All libraries were located and were used in the source code	None needed
2	All Source Files	Grammar/Syntax	Ensure that the program's outputs are free of any misspelled words, such as error messages, titles, etc.	The program contained minor gramatical errors and typos	Changes were made to output statements to ensure grammar/syntax was accurate
3	DatabaseHelper.java	Photo Upload	Ensure that user can upload photos to program that can be stored	Images were not being stored correctly within the database	Bitmap library was used to store photos in the database
4	DatabaseHelper.java	Photo Retrieval	Ensure that program can retreive stored photos when called upon	Images were not being stored correctly within the database	Bitmap library was used to store photos in the database
5	DatabaseHelper.Java	Data Capacity	Ensure that program can hold data without crashing	Program did not slow down until 500 items were uploaded to database	None needed
6	N/A	Environment Testing	Confirm that program can run properly on Android platform	Runs poperly on Android platfrom	None needed
Final Evaulation:	Following the beforementioned tests and inspections, we had found errors in our project and conducted necessary corrections to resolve these problems. The app is capable of storing and retreiving photos accurately and the program is free of gramatical and syntax errors. The program will run solely on Android platforms.				Project Deliverable? **[Yes]** [No]

Validation & Verification Test

The Quik Closet program will be tested through a walkthrough approach; the user will navigate through the application and attempt a multitude of possible choices to prove its functionality. If there are issues found during the walkthrough, then these said issues will be documented and later corrected following the test. The SRS, Software Requirements Specifications, will be heavily relied on for the validation and verification testing process. Use case descriptions, the activity diagram, and use case diagrams will be used in order to validate the accuracy of the software.

There are two types of testing that will be performed during these tests: non-executable and executable. During non-executable testing of code, GitHub will be used during the code reviews. The code reviews of the pull requests that are submitted on Github help the team collaborate on what issues need to be resolved in an organized manner. As far as executable testing, we are going to use the use case descriptions, use case diagram, system-level sequence diagram, and activity diagram to cross reference our work and ensure that the application is functioning as we expect it to.

Execution testing (black box testing)

We decided to use the Equivalence Testing technique to test our project, since our Android application is simple to use and navigate.

- ❖ Our expected outputs will either be that we successfully stored and retrieved data from the database and displayed it in the fragments of our app, or that we checked if the information was valid.
- ❖ We applied these tests to our app:
  - ➢ We uploaded 10 clothing items to the database with different clothing types such as long dresses, tank tops, pants, tennis shoes etc. under several attributes such as mood, temperature, precipitation etc. Then, we saved these clothing items and used an external software to check if our items were stored in the database. As expected, our storage routines were successful.
  - ➢ We also had to check our retrieval functions from the database. In order to do that, the items must be displayed in the clothing gallery. The clothing gallery is a scrollable table split by tops, bottoms, shoes and accessories. Therefore, each clothing image must be loaded into the correct section of the scrollable table. After uploading the clothing items, all clothing items were loaded into the gallery and were also placed into the correct sections as specified by their attributes.
  - ➢ We also must check the build outfit functionality. With the clothing items uploaded, we can now build outfits. Putting together the clothing items, we again had to check if our outfits were being stored into the outfits table of our database. When pulling up the database, we saw that it stored the tops, bottoms and shoes incorrectly (stored them in the clothing table instead of the outfit table). We were able to fix this and retested it. At this point, our outfits were safely stored into our database.
  - ➢ Now that our outfits are stored in the database correctly, we must be able to retrieve them in the view outfits portion of the code. Tests showed that this function worked perfectly, and all functionality for the application works correctly

Project/Process

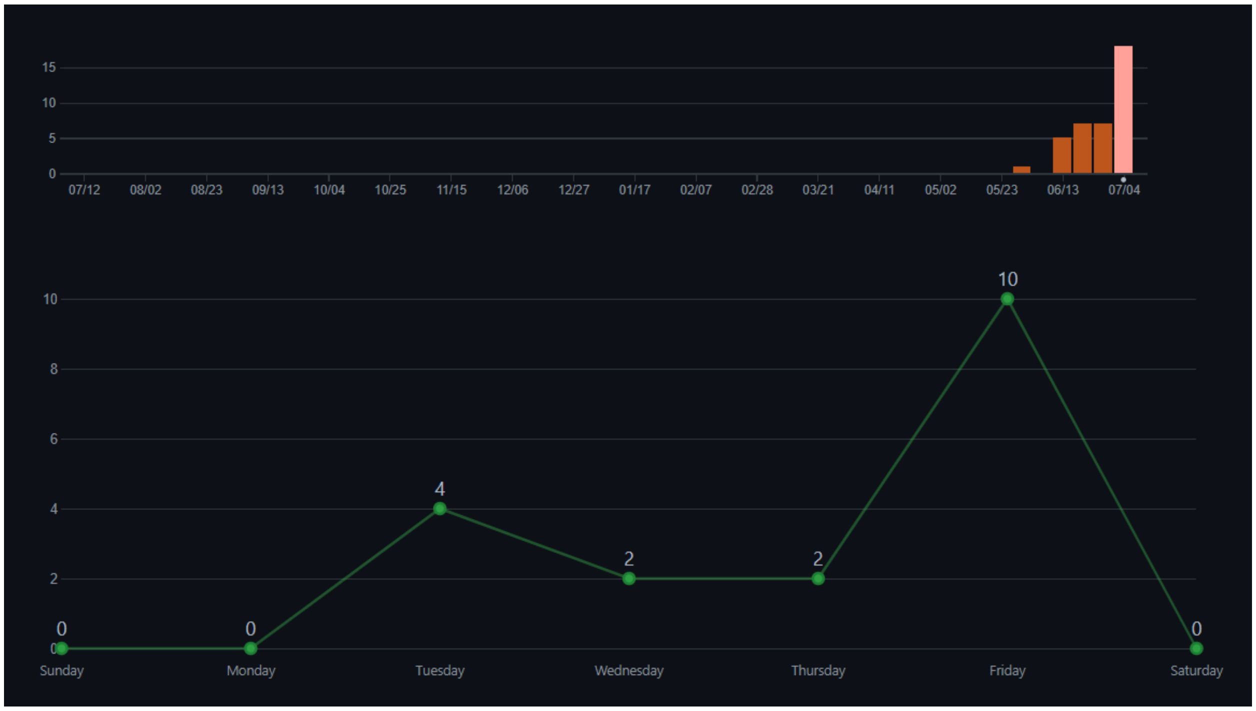
Open issues

- Some features that were discussed in the beginning of this project are implemented, but not all
  - Our original intent was for the application to make recommendations to the user, but we decided to alter our concept to be an organizational tool instead
- No additional features other than our original organizational objective were added to the application
- No Open Issues

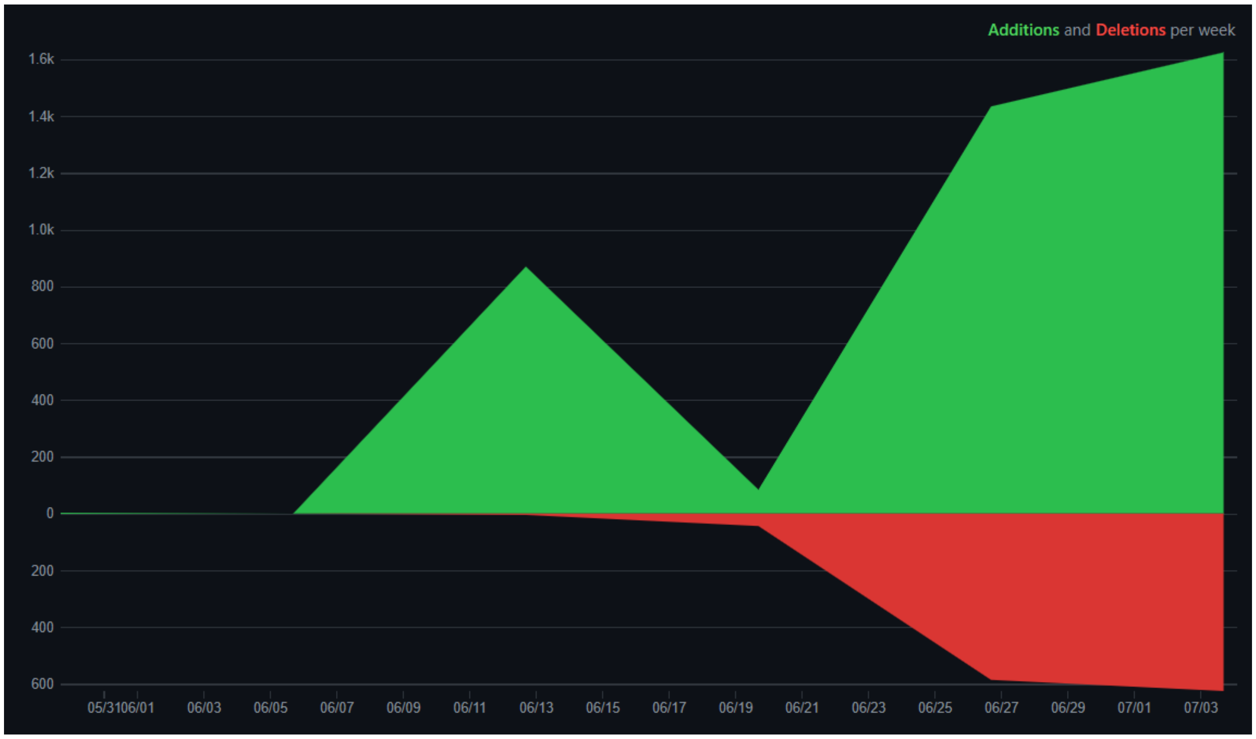
Retrospectives on project management and software development process

- This project allowed us to learn and become more familiar with Java, SQLite, and Android Studio
- We learned how to create proper program documentation when creating an application
  - SRS Report, Project Reports, GitHub, UML Diagrams, and more
- We learned how to implement certain libraries in order to create a database to store user created information and photos
  - Bitmap library
- We learned the purpose of certain diagrams, how to create them, and how to implement them within the program
  - UML diagrams, system models, 4+1 architectural views, system architectures and patterns
- We were required to collaborate together as a team and build based off of differences and similarities in order to create the application

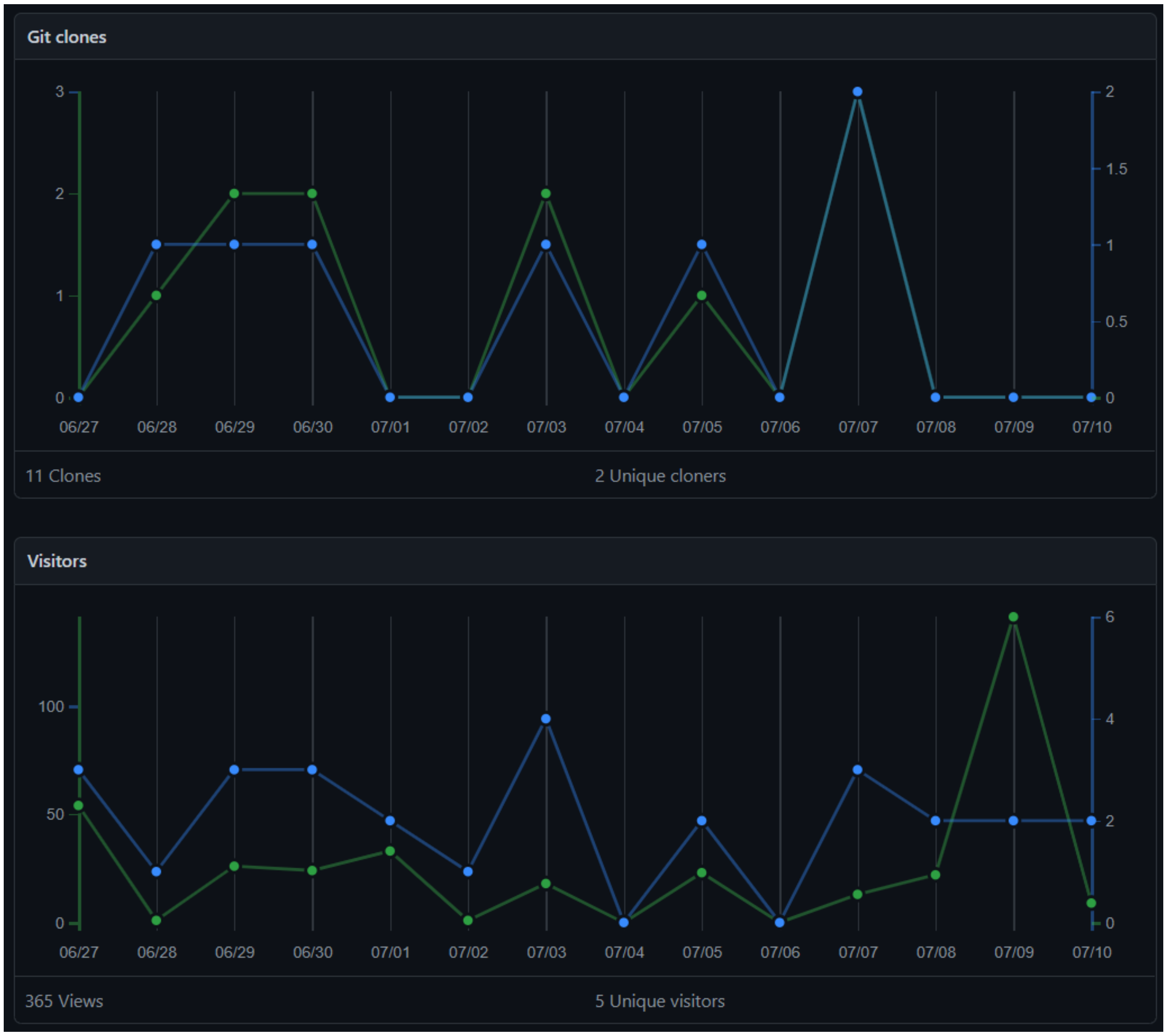
GitHub activity reports, graphs, log files



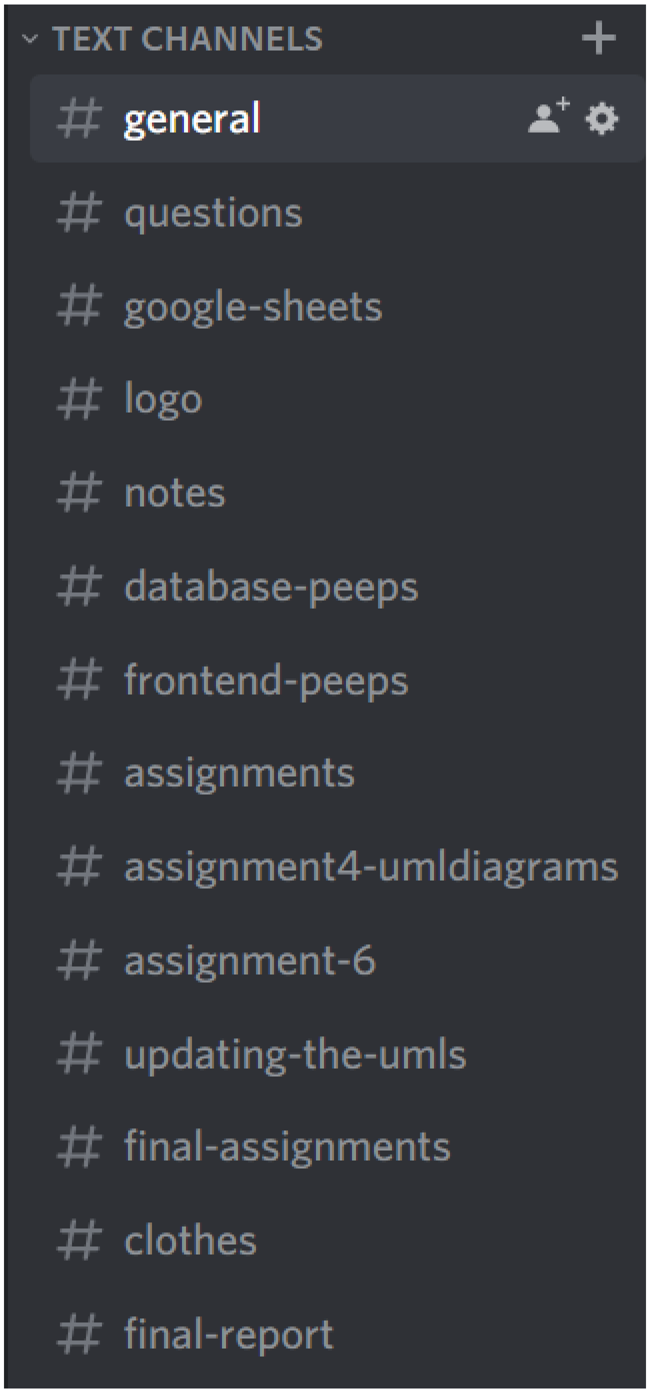
Github Commit History



Github Code Frequency



Github Traffic


TEXT CHANNELS

general
questions
google-sheets
logo
notes
database-peeps
frontend-peeps
assignments
assignment4-umldiagrams
assignment-6
updating-the-umls
final-assignments
clothes
final-report

Discord Channel Organization

Appendix

Reference Links:

Project Github Repository	https://github.com/cs380summer2021/quik-closet.git
Android Developers	developer.android.com/docs
Android - SQLite Database - Tutorialspoint	https://www.tutorialspoint.com/android/android_sqlite_database.htm
Recommender Systems · GitBook	https://apple.github.io/turicreate/docs/userguide/recommender/
Hello World · GitHub Guides	https://guides.github.com/activities/hello-world/