Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: Outletter

Low Level Design Report

Muhammad Bilal Bin Khalid, Muhammad Saboor, Mian Usman Naeem Kakakhel, Daniyal Khalil, Muhammad Arham Khan

Supervisor: Hamdi Dibeklioğlu

Progress Report
Feb 4, 2021

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

1 Introduction

In today's capitalistic age, shopping is one of the most time-consuming activities that people from all walks of life participate in regularly. From shopping for some clothing articles, to a new pair of shoes to even buying the latest electronic gadget, people spend hours trying to validate their shopping decisions and confirming that they're getting the best possible bargain. But considering the inherent information gap in the manual store-by-store checking process [1], we spend many hours only to buy what we got the best deal on at the three stores we visited.

So seeing the potential for improvement in the process, we decided to introduce Outletter, A one-stop shopping companion mobile application that allows users to point their mobile phones towards items like fashion articles, electronics, etc. Our app would automatically recognize the product in realtime and render metadata like price comparison with nearby stores, product reviews, and other available variants in nearby stores, right in the user's field of view using Augmented Reality. This way, whenever a user is out shopping, they can rely on Outletter to help them make the most reliable decision about a product and be sure to get the best deal in a very easy and fast way. Considering that Outletter will be able to identify almost any product that is available online and that there aren't any steps with manual data entry or button clicks (the user can access all information just by pointing the mobile phone at a product), the application would revolutionize the shopping process for the mass user-base including the elderly.

Apart from this, Outletter will also provide the user with the latest deals and offers going on in a particular store as soon as they are near that store. This way, users will never regret missing out on a good deal from a store just because they didn't watch an ad campaign and, as promised by our app, will always be making the most well-informed shopping decision possible.

In this report, you will find the design trade-offs in several subsections, followed by the high-level explanations of the packages that will be put into the 3-tier structure of the system. Finally, the class interfaces will be given and explained in detail.

1.1 Object design trade-offs

During the development of Outletter, while choosing a certain feature or design, another design or feature needed to be sacrificed. Every decision made had a possible trade-off which we discussed thoroughly and are discussed below.

1.1.1 Response Time vs Computing Performance

In the application, after the user takes the picture, the image is processed by multiple models. Due to the limitations of the software devices in terms of computation power, performing these processes on the device would result in poor performance. Due to this reason, Google Cloud service will be used as a cloud computing service to do all the computation-intensive tasks for the application. However, since the communication between the physical device and the cloud server, that will be doing the processing, will take time and this increases the response time of the application when displaying the results. Outletter will not utilize the device's computing power for processing the image in exchange for the response time taken to get the data from the Google cloud server.

1.1.2 Compatibility vs Extensibility

As stated in the previous reports, Outletter is available for both Android and iOS platforms. This is why we chose React Native for the development. However, as our application will have AR features implemented using AR Core, it will only be compatible with devices using Androids 7.0 or higher and iOS 11.0 or higher. We also decided not to use only Android or iOS-specific features to increase the compatibility of the application on both platforms. This decision reduces the compatibility of the application based on their operating system but increases compatibility based on their platforms. It also increases programmability as AR Core provides a lot of possibilities for new features.

1.1.3 Cost vs Performance

As our application uses Google Cloud services for computation power to perform computation-intensive tasks, most of the performance of our application depends on the user's

device rather than the server. As mentioned above, the user needs an AR compatible smartphone, and depending on the smartphone specifications, the performance will vary. We plan to optimize our application in a way that produces the best performance possible. As for server-side performance, the response time depends on the bandwidth allocated to us by Google. A better bandwidth will have more costs but as the response time is not too much, we decided not to get additional services from Google.

1.1.4 Robustness vs Cost

Our expectations of Outletter is to provide reliable results and be fully available for users whenever they access the application. This requires the application to use a cloud server maintained by Google rather than using a server maintained by one of our developers. Using Google Cloud is more costly but provides better services compared to a local server. Currently, we are using the free credits received upon creating a new account but eventually, it will start costing us money. However, we will be valuing robustness over cost to provide the best experience possible.

1.2 Interface documentation guidelines

Class Sample
This Sample class represents...
Properties
private String name
private int age
Methods
public String getName(): Gets the name of the person.
public int getAge(): Gets the age of the person.
public int setAge(int age): sets the age of the person.

1.3 Engineering standards

In this report, two different engineering standards are followed. For the diagrams given in the report, UML design principles are followed [2]. As for the citations IEEE citation standards are followed [3].

1.4 Definitions, Acronyms, and Abbreviations

AR: Augmented reality (AR) is a technology that lets people superimpose digital content (images, sounds, text) over real-life scenes.

OS: Operating system is system software that manages computer hardware, software resources, and provides common services for computer programs.

SDK: Software Development Kit. An SDK is a collection of software used for developing applications for a specific device or operating system.

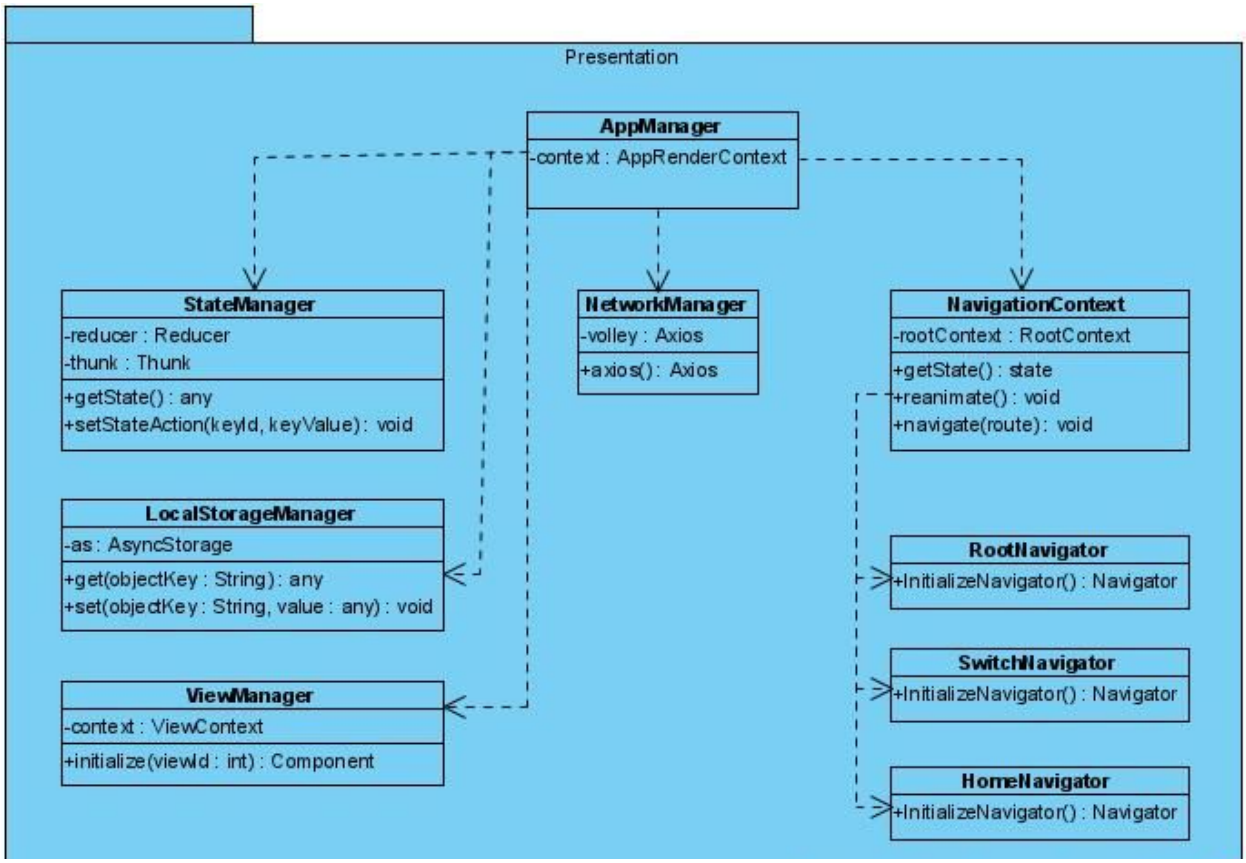
UI: User Interface. The user interface (UI) is the point of human-computer interaction and communication in a device.

UML: Unified Modelling Language. A standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems [2].

2 Packages

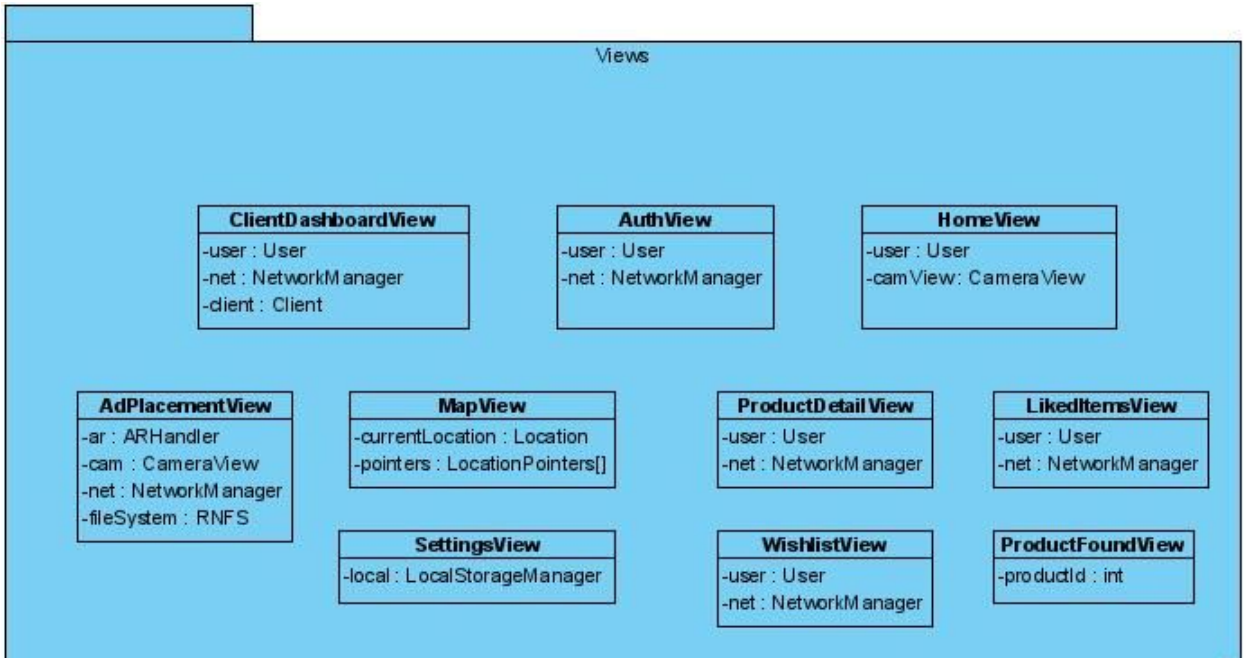
2.1 Presentation Package

2.1.1 Presentation Package



This package deals with different functional components of the front-end application. It consists of several navigator classes and different component managers. Navigator classes are responsible for navigation between screens of the application.

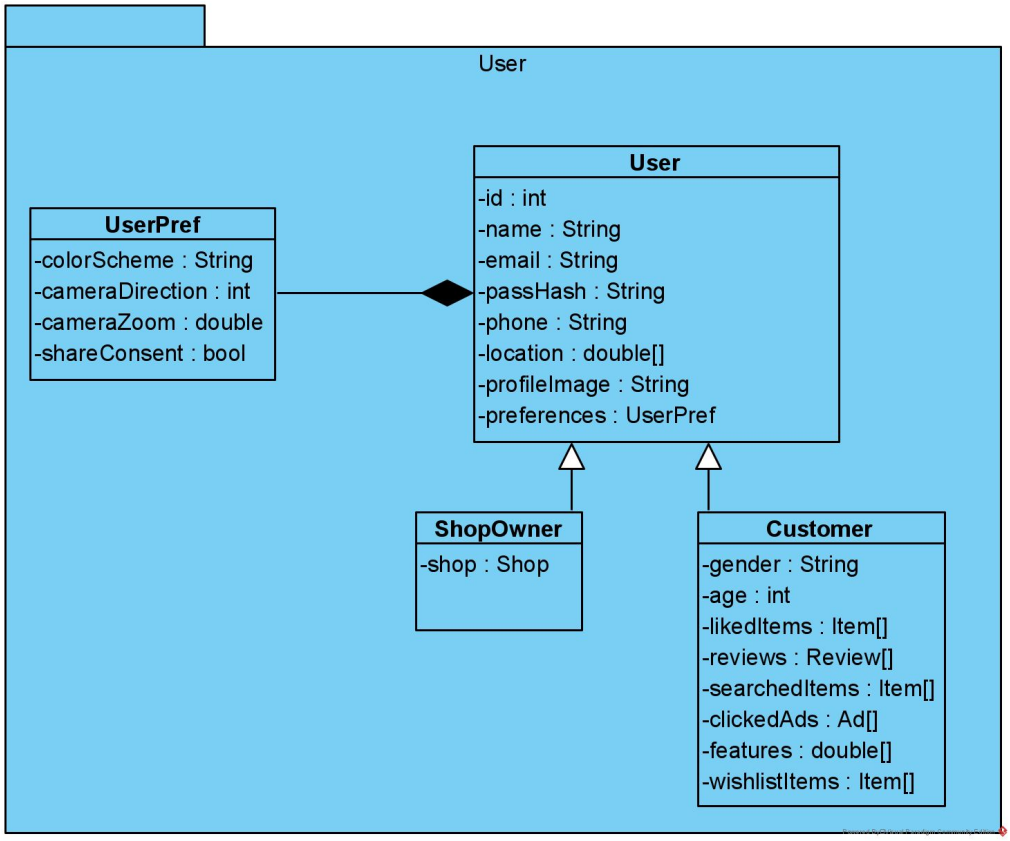
2.1.2 View Package



The View package deals with the front-end views of the application. The user interacts with these views to perform different tasks. Each view corresponds to a single screen in the application.

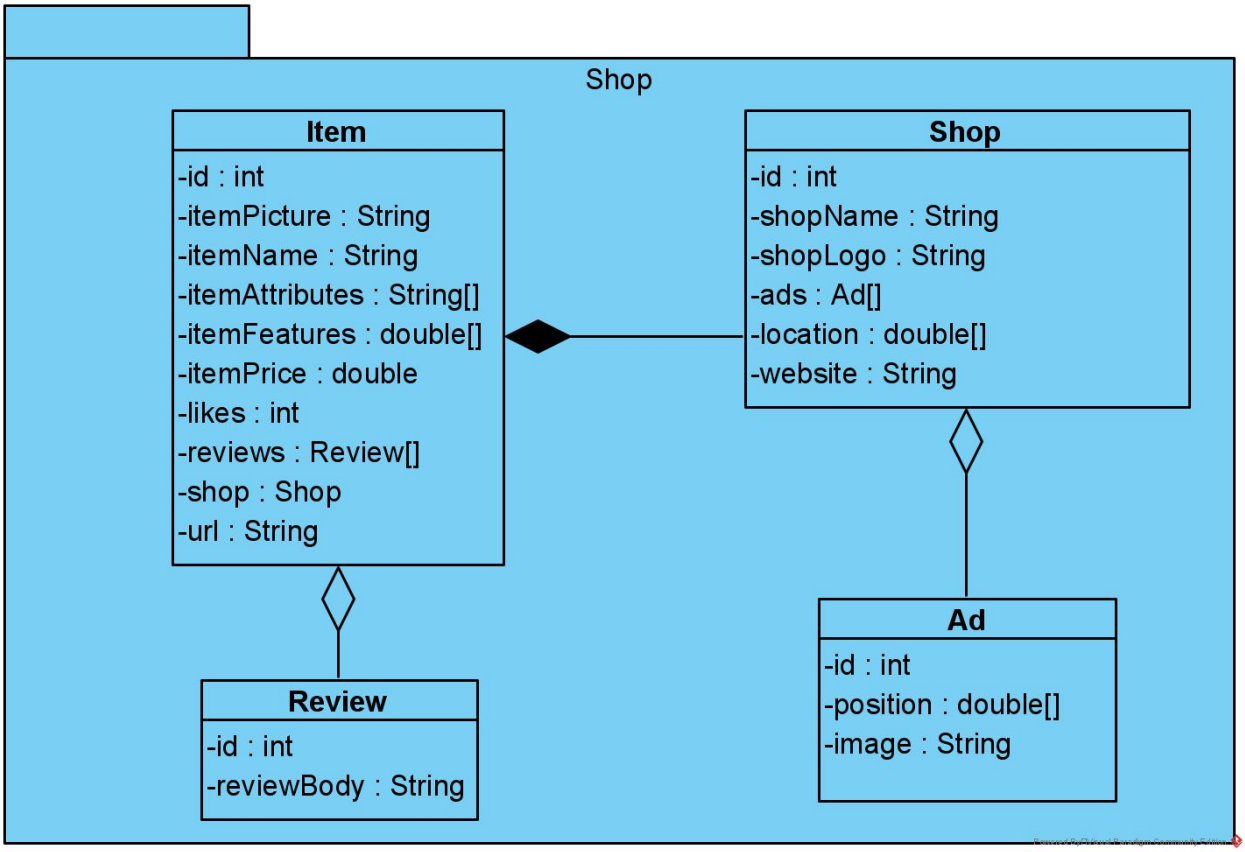
2.2 Application Package

2.2.1 User Package



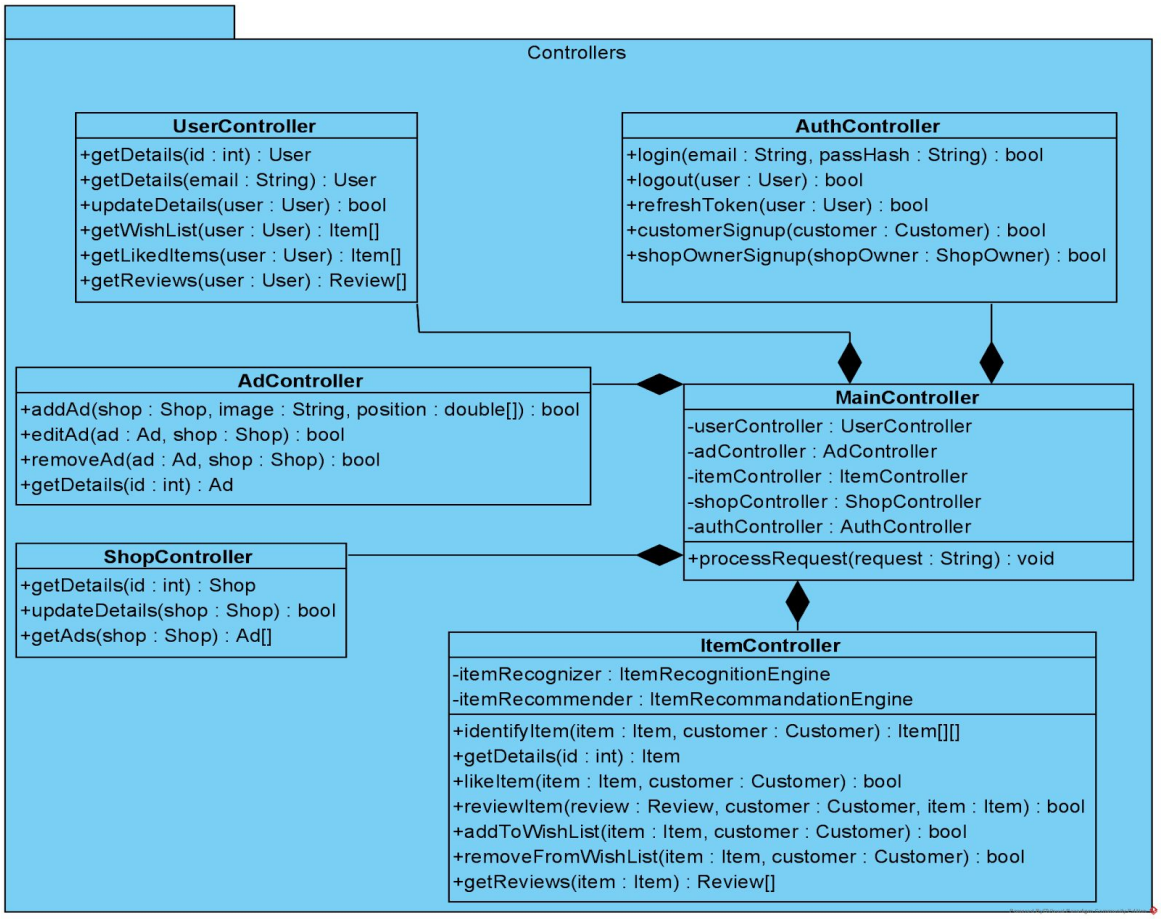
The user package deals contains the models for both the users, ShopOwners and Customers. This package is used for managing the user data and their preferences used for item finding.

2.2.2 Shop Package



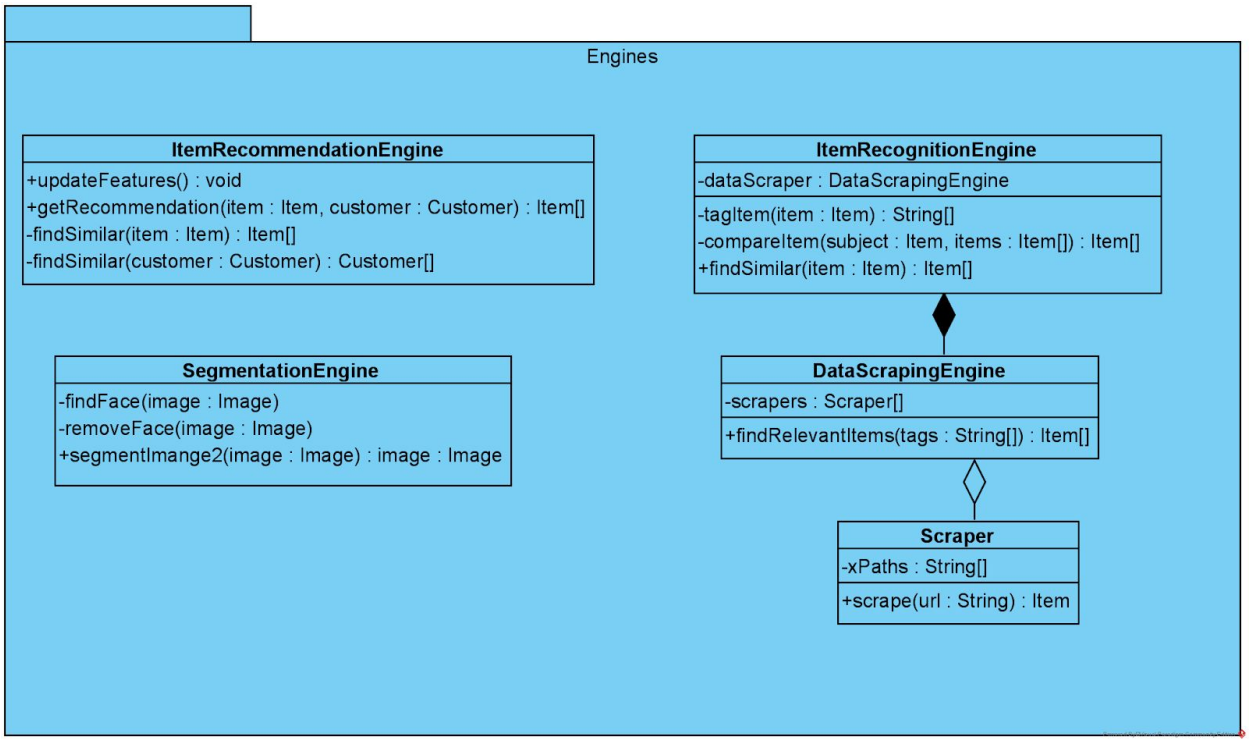
The Shop Package deals with all the attributes of a shop. This package contains the Shop, Item, and Ad models, along with their Reviews. The shop can have Ads and Items are supposed to belong to a shop, meanwhile items can have reviews.

2.2.3 Controller Package



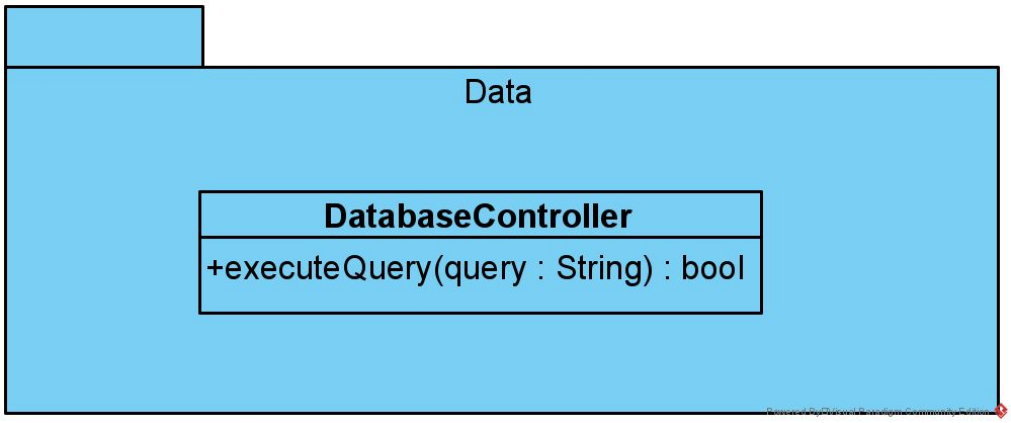
The controller package contains all the controllers that are used in the application. All the controllers are a part of the MainController and all the controllers interacts with their own models/packages to perform activities related to those models.

2.2.4 Engine Package



This package contains all our Machine Learning and Vision related classes. The ItemRecommendationEngine is used for recommending to generate the item recommendations for the users. The ItemRecognitionEngine is used to find the most similar item that we have available. The DataScrappingEngine is used to scrape data from the websites using the Scraper class.

2.3 Data Package



The DatabaseController is the only class that we have in the Data Package, since all the interactions that involve sql injections go through the DatabaseController.

3 Class Interfaces

3.1 Model

3.1.1 User

Class User
This class represents the users of Outletter application
Properties
private int id
private String name
private String email
private String passHash
private String phone
private double[] location

private String profileImage
private UserPref preferences
Methods
public void updateUserPreferences(String colorScheme, int cameraDirection, double CameraZoom, bool shareConsent): This method updates the preferences of this user.

3.1.2 ShopOwner

Class ShopOwner
This class represents a type of user of Outletter application
Properties
private Shop shop

3.1.3 Customer

Class Customer
This class represents a type of user of Outletter application
Properties
private String gender
private int age
private Item[] likedItems
private Review[] reviews
private Item[] searchedItems
private Ad[] clickedAds
private double[] features
private Item[] wishListItems
Methods

public void emptyLikedItems(): Delete all the liked items of the customer.
public void clearSearchedHistory(): Delete all the search history related with the customer
public void clearWishList(): Delete all the items from the wish list of the customer.

3.1.4 UserPref

Class UserPref
This class represents the preferences of each user
Properties
private String colorScheme
private int cameraDirection
private double cameraZoom
private bool shareConsent

3.1.5 Review

Class Review
This class represents the reviews made by the customers
Properties
private int id
private String reviewBody

3.1.6 Ad

Class Ad
This class represents the ads published in the shop by shop owners.
Properties

private int id
private double[] position
private String image

3.1.7 Shop

Class Shop
This class represents the shop owned by the shop owner
Properties
private int id
private String shopName
private String shopLogo
private Ad[] ads
private double[] locations
private String website
Methods
public void deleteAds(): deletes all the ads displayed in the shop

3.1.8 Item

Class Item
This class represents a clothing item
Properties
private int id
private String itemPicture
private String itemName
private String[] itemAttributes
private double[] itemFeatures

private double itemPrice
private int likes
private Review[] reviews
private Shop shop
private String url
Methods
public bool validateUrl(): This method validates the URL of the current item by visiting the URL and comparing the item details on both sides.

3.2 Controller

3.2.1 MainController

Class MainController
This class is responsible for routing all incoming requests to the responsible controller of the backend of the application
Properties
private UserController userController
private AdController adController
private ItemController itemController
private ShopController shopController
private AuthController authController
Methods
public void processRequest(String request): This method receives a request from the front-end of the application and utilizes one of the controllers to perform the required request.

3.2.2 UserController

Class UserController

This class is responsible for handling all the requests regarding user model attributes.
Methods
public User getDetails(int id): This method retrieves the User using its unique ID from the database.
public User getDetails(String email): This method retrieves the User using its unique email from the database.
public bool updateDetails(User user): This method is responsible for updating the details of a user in the backend database.
public Item[] getWishlist(User user): This method retrieves the wishlist of a specific user from the database.
public Item[] getLikedItems(User user): This method retrieves the liked items of a specific user from the database.
public Review[] getReviews(User user): This method retrieves the reviews of a specific user from the database.

3.2.3 AuthController

Class AuthController
This class is responsible for handling all incoming requests related to authentication functions of the user of the system.
Methods
public bool login(String email, String passHash): This method is responsible for generating a new session token for a user aftering validating the credentials.
public bool logout(User user): This method is responsible for marking the user as logged out from the system and invalidating the respective token.
public bool refreshToken(User user): This method is responsible for extending the duration of the user's token.
public bool validateToken(User user): This method is responsible for validating the user's token.
public bool customerSignup(Customer customer): This method is responsible for creating a new customer in the database.
public bool shopOwnerSignup(ShopOwner shopOwner): This method is responsible for creating a new shop owner in the database.

3.2.4 ShopController

Class ShopController
This class is responsible for handling all incoming requests related to administration of the shops.
Methods
public Shop getDetails(int id): This method retrieves the shop details using its unique ID from the database.
public bool updateDetails(Shop shop): This method is responsible for updating the details of a shop in the backend database.
public Ad[] getAds(Shop shop): This method retrieves all the ads in a specific shop from the database.

3.2.5 AdController

Class AdController
This class is responsible for handling all incoming requests related to administration of the ads in a shop.
Methods
public bool addAd(Shop shop, String image, double[] position): This method is responsible for adding an advertisement in a shop at the specified position.
public bool editAd(Shop shop, Ad ad): This method is responsible for updating the details of an advertisement in a shop.
public bool removeAd(Shop shop, Ad ad): This method is responsible for removing an advertisement from a shop.
public Ad getDetails(int id): This method retrieves the advertisement details using its unique ID from the database.

3.2.6 DatabaseController

Class DatabaseController
This class is responsible for executing the database queries on behalf of the application
Methods
public bool executeQuery(String query): This method is responsible for executing the specified query in the database and return the required results to the application

3.2.7 ItemController

Class ItemController
This class is responsible for handling all the incoming requests related to the administration of an item.
Properties
private ItemRecognitionEngine itemRecognizer
private ItemRecommendationEngine itemRecommender
private ItemSegmentationEngine itemSegmenter
Methods
public Item getDetails(int id): This method retrieves the item details using its unique ID from the database.
public Item[][] identifyItem(Item item, Customer customer): This method gets a query item from a customer and finds the related and recommended items for the customer.
public bool likeItem(Item item, Customer customer): This method is responsible for marking an item liked by a customer.
public bool reviewItem(Item item, Review review, Customer customer): This method is responsible for leaving a review on an item by a customer.
public bool addToWishlist(Item item, Customer customer): This method is responsible for adding an item in a customer's wishlist.
public bool removeFromWishlist(Item item, Customer customer): This method is responsible for removing an item in a customer's wishlist.
public Review[] getReviews(Item item): This method is responsible for retrieving all the reviews for an item from the backend database.

3.3 Engines

3.3.1 ItemSegmentationEngine

Class ItemSegmentationEngine
This class is responsible for segmenting an item's image
Methods
public Item[] segmentItems(Item[] items): This method is responsible for using a trained model to segment the item's images in a list.

3.3.2 ItemRecognitionEngine

Class ItemRecognitionEngine
This class is responsible for Computer Vision tasks related to item's image
Properties
private DataScrapingEngine dataScraper
Methods
private String[] tagItems(Item item): This method is responsible for annotating an item's image using a trained computer vision model.
private Item[] compareItems(Item subject, Item[] items): This method is responsible for sorting the given item list according to their similarity with the subject image.
public Item[] findSimilar(Item item): This method is responsible for finding and returning the similar images in a sorted manner.

3.3.3 ItemRecommendationEngine

Class ItemRecommendationEngine
This class is responsible for recommendation tasks for the application
Methods

private Item[] findSimilar(Item item): This method is responsible for retrieving the recommended items for the query item by comparing its features with all the other items in the database.
private customer[] findSimilar(Customer customer): This method is responsible for retrieving the recommended items for the query customer by comparing its features with all the other customers in the database.
public void updateFeatures(): This method is responsible for updating the features of all the users and items regularly.
public Item[] getRecommendation(Item item, Customer customer): This method is responsible for retrieving the recommended items according to the query image and customer.

3.3.4 DataScrapingEngine

Class DataScrapingEngine
This class is responsible for scraping tasks for the application
Methods
public Item[] findRelevantItems(String[] tags, String gender, String website): This method is responsible for finding all the relevant items according to the given tags and gender on the given website by web scraping techniques.

3.4 Presentation Layer

3.4.1 AppManager

Class AppManager
This class is responsible to load and instantiate the native base code for the app and initialize the JS application code
Properties
private AppRenderContext context

3.4.2 NavigationContext

Class NavigationContext
This class is responsible to instantiate and hold the current navigators and their state across the application
Properties
private RootContext rootContext
Methods
public state getState(): This method returns the current navigator state and active navigator details public void reanimate(): This method resets the navigator to the default context public void navigate(route): This method navigates the application screens to a particular screen name across the app

3.4.3 NetworkManager

Class NetworkManager
This class is responsible to initialize and intercept all API calls across the app and implement features like header authentication, error handling and response redirections across the app
Properties
private Axios volley
Methods
public Axios axios(): This method returns the default axios network call handler to allow request calls like POST, GET etc.

3.4.4 ViewManager

Class ViewManager
This class is responsible for the base instance of all view initializers and maintains the index routes of currently initialized views to avoid redundant reloads on the UI thread.

Properties
private ViewContext context
Methods
public JSX.Component initialize(int viewid): This method returns a cached and indexed JSX component that can be tracked by the ViewManager for lifecycle events

3.4.5 StateManager

Class StateManager
This class is responsible to hold the current states by utilizing the app-wide redux reducer in the application
Properties
private Reducer reducer private Thunk thunk
Methods
public any getState(): This method returns the current app-wide state body public void setStateAction(keyId, keyValue): This value sets the state of a relevant state name in the app and triggers the relevant action (if any).

3.4.6 LocalStorageManager

Class LocalStorageManager
This class is responsible to load, save and retrieve all data stored locally on the device such as auth tokens, login credentials etc.
Properties
private AsyncStorage as
Methods
public any get(String objectKey): This method retrieves the values relevant to the objectKey public void set(String objectKey, any value): This method stores a value for a specified objectValue in the local storage

3.4.7 SwitchNavigator

Class SwitchNavigator
This class is responsible to load the switchNavigators of the app and initialize them with the NavigationContext. This class basically returns a switch navigator that runs the app state logic to see if user is logged in and show the auth views or whether to show the home navigators
Methods
public Navigator InitializeNavigator(): This method returns a new SwitchNavigator that references updates to the app-wide navigation context

3.4.8 HomeNavigator

Class HomeNavigator
This class is the navigator of the app once the user has logged in. This returns a StackNavigatorComponent that can be used for navigation in the app.
Methods
public Navigator InitializeNavigator(): This method returns a new StateNavigator that references updates to the app-wide navigation context

3.4.9 RootNavigator

Class RootNavigator
This class returns the rootNavigator (a switch navigator) as the root navigation component in the app which further redirects to other navigators. Again, the JSX.Component navigator returned references the app-wide navigation context for state updates
Methods
public Navigator InitializeNavigator(): This method returns a new SwitchNavigator that references updates to the app-wide navigation context

3.4.10 AppManager

3.5 Views

3.5.1 ProductFoundView

Class ProductFoundView
This JSX.Component class instance is the view that shows the details about the product that was found
Properties
private int productId

3.5.2 HomeView

Class HomeView
This JSX.Component class instance is view that is rendered once the user logs in and contains the root camera view, navigation instance and other relevant components
Properties
private User user private CameraView camView

private Navigator nav private Networkmanager net

3.5.3 SettingsView

Class SettingsView
This JSX.Component class instance is the view that shows the settings in the app and lets the user change preferences
Properties
private LocalStorageManager local

3.5.4 LikedItemsView

Class LikedItemsView
This JSX.Component class instance is the view that shows all the liked items by the user.
Properties
private User user private NetworkManager net

3.5.5 AuthView

Class AuthView
This JSX.Component class instance is the view that shows the component views for authentication like the login view, the register view or the forgot password view inside a wrapped parent component for easier transition
Properties
private User user private NetworkManager net

3.5.6 ProductDetailView

Class ProductDetailView
This JSX.Component class instance that shows the details of a product such as similar items, top items and product meta information etc
Properties
private int productId private NetworkManager net

3.5.7 ClientDashboardView

Class ClientDashboardView
This JSX.Component class instance is the root view that clients see that shows details about their store and previous items
Properties
private User user private NetworkManager net private Client client

3.5.8 MapView

Class MapView
This JSX.Component class instance is the extended object from GMapView class that shows the map with relevant location pointers and the users current location
Properties
private Location currentLocation private LocationPointers[] pointers

3.5.9 AdPlacementView

Class AdPlacementView
This JSX.Component class instance that wraps the ARCores interface library to allow users to place their ad content in real-time in their field-of-view
Properties
private ARHandler ar private CameraView cam private NetworkManager net private RNFS fileSystem

3.5.9 WishlistView

Class WishlistView
This JSX.Component class instance is the view that shows the items stored in the wishlist stored by the user
Properties
private User user private NetworkManager net

4 Glossary

React-Native: An open-source mobile application framework. It is used to develop applications for Android, iOS and more.

ARCore: A software development kit developed by Google that allows for augmented reality applications to be built.

Cloud: Cloud refers to servers that are accessed over the Internet, and the software and databases that run on those servers.

Google Cloud Platform: Google Cloud Platform, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally

Machine Learning Model: It can be defined as a black box system that embodies a mathematical representation of a real-world process.

Bandwidth: The maximum data transfer rate of a network or Internet connection.

5 References

[1] Mohsin, Maryam. “10 Online Shopping Statistics You Need to Know in 2020 [Infographic].” [Infographic]. Oberlo, Accessed February 4, 2021. https://www.oberlo.com/blog/online-shopping-statistics

[2] “What is Unified Modeling Language (UML)?” Accessed February 4, 2021. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/.

[3] “Citation Styles: APA, MLA, Chicago, Turabian, IEEE: IEEE Style.” LibGuides. Accessed February 4, 2021. https://pitt.libguides.com/citationhelp/ieee.