FoodBuddy - Food Recommendation System

Team Number: 34

Team Members:

Vikhyat Goyal

Pavan Dhareshwar

Rishi Soni

Vision:

Our aim was to create a food and restaurant recommendation system which would help end users find restaurants that matches with their preferences, their past experiences, and that has good feedback from other users. Furthermore, the system allows hosts (restaurant owners) to add their own profile, promote their businesses, modify the services & cuisines offered by them, and finally get feedback from the users.

There are three main actors / participants in the system:

1) Admin : Has responsibility to manage, maintain and secure the whole system.
2) End users : A person looking for a food/restaurant recommendation.
3) Host (Restaurant/food product owner) : Restaurant owners looking to expand business or food product manufacturers looking to launch a new product or promote an existing product.

Our system for the project will be a website that will enable end users to find new restaurants or food product to try based on their profile, past experiences and random projection. It will also allows new restaurants to get registered and promote their business, old restaurants and food product manufacturers to add new cuisines/dishes and get feedback.

2. FEATURES IMPLEMENTED:

ID	REQUIREMENT / FEATURE	TOPIC AREA	USER	PRIORITY

UR-001	New End user should be able to create a profile	Profile	End User	High
UR-002	End user should be able to get recommendations for restaurants	Recommendation	End User	Critical
UR-003	End user should be able to get recommendations for food products	Recommendation	End User	Critical
UR-004	End user should be able to give feedback	Feedback	End User	High
UR-005	End user should be able to set filters	Database	End User	High
UR-006	New host should be able to create a profile	Profile	Host	High
UR-007	Host should be able to update dishes list	Database	Host	Critical
UR-008	Host should be able to update services list	Database	Host	High
UR-010	Host should be able to see feedbacks	Feedback	Host	Medium
UR-015	Admin should be able to add advertisements	Promotions	Admin	Medium

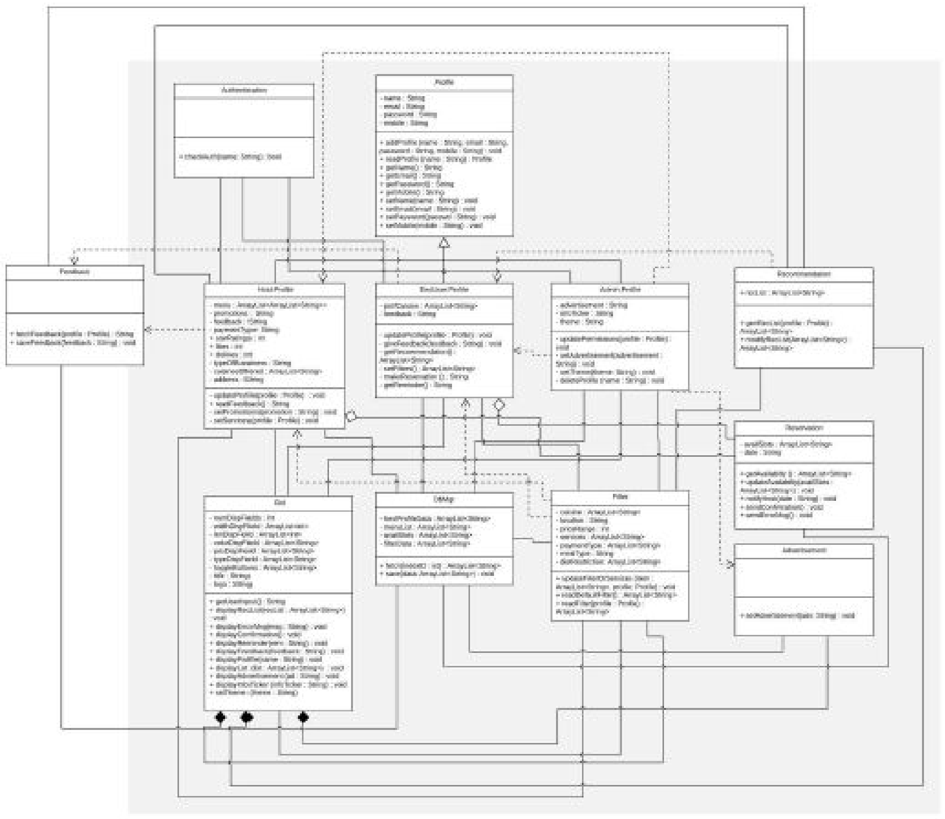
3. FEATURES NOT IMPLEMENTED:

ID	REQUIREMENT / FEATURE	TOPIC AREA	USER	PRIORITY
UR-009	Host should be able to update availability	Database	Host	Medium
UR-011	Host should be able to update promotions	Promotions	Host	Medium
UR-012	Admin should be able to authorize new host profile	Profile	Admin	Critical
UR-013	Admin should be able to delete host profile	Profile	Admin	Medium
UR-014	Admin should be able to delete an end user profile	Profile	Admin	Medium

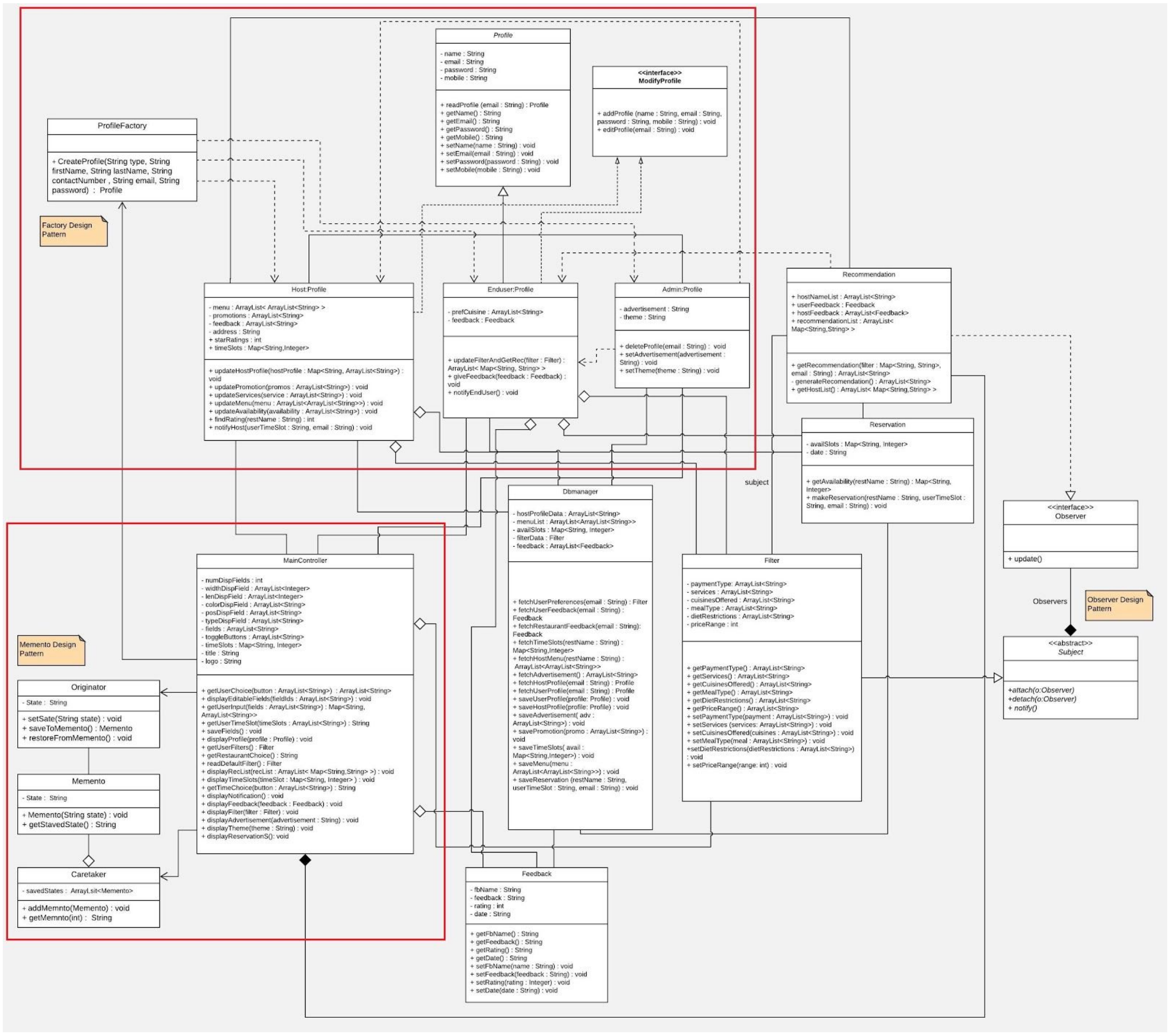
4. COMPARISON OF CLASS DIAGRAMS

Part 2 class diagram:

Class Diagram:



Final class diagram:


Profile
- name : String
- email : String
- password : String
- mobile : String
+ readProfile (email : String) : Profile
+ getName() : String
+ getEmail() : String
+ getPassword() : String
+ getMobile() : String
+ setName(name : String) : void
+ setEmail(email : String) : void
+ setPassword(password : String) : void
+ setMobile(mobile : String) : void
<<interface>> ModifyProfile
+ addProfile (name : String, email : String, password : String, mobile : String) : void
+ editProfile(email : String) : void
ProfileFactory
+ CreateProfile(String type, String firstName, String lastName, String contactNumber , String email, String password) : Profile
Factory Design Pattern
Host:Profile
- menu : ArrayList< ArrayList<String> >
- promotions : ArrayList<String>
- feedback : ArrayList<String>
- address : String
+ starRatings : int
+ timeSlots : Map<String,Integer>
+ updateHostProfile(hostProfile : Map<String, ArrayList<String>) : void
+ updatePromotion(promos : ArrayList<String>) : void
+ updateServices(service : ArrayList<String>) : void
+ updateMenu(menu : ArrayList<ArrayList<String>>) : void
+ updateAvailability(availability : ArrayList<String>) : void
+ findRating(restName : String) : int
+ notifyHost(userTimeSlot : String, email : String) : void
Enduser:Profile
- prefCuisine : ArrayList<String>
- feedback : Feedback
+ updateFilterAndGetRec(filter : Filter) : ArrayList< Map<String, String> >
+ giveFeedback(feedback : Feedback) : void
+ notifyEndUser() : void
Admin:Profile
- advertisement : String
- theme : String
+ deleteProfile(email : String) : void
+ setAdvertisement(advertisement : String) : void
+ setTheme(theme : String) : void
Recommendation
+ hostNameList : ArrayList<String>
+ userFeedback : Feedback
+ hostFeedback : ArrayList<Feedback>
+ recommendationList : ArrayList< Map<String,String> >
+ getRecommendation(filter : Map<String, String>, email : String) : ArrayList<String>
- generateRecomendation() : ArrayList<String>
+ getHostList() : ArrayList< Map<String,String> >
Reservation
- availSlots : Map<String, Integer>
- date : String
+ getAvailability(restName : String) : Map<String, Integer>
+ makeReservation(restName : String, userTimeSlot : String, email : String) : void
subject
<<interface>> Observer
+ update()
Observers
Observer Design Pattern
<<abstract>> Subject
+attach(o:Observer)
+detach(o:Observer)
+ notify()
Dbmanager
- hostProfileData : ArrayList<String>
- menuList : ArrayList<ArrayList<String>>
- availSlots : Map<String, Integer>
- filterData : Filter
- feedback : ArrayList<Feedback>
+ fetchUserPreferences(email : String) : Filter
+ fetchUserFeedback(email : String) : Feedback
+ fetchRestaurantFeedback(email : String): Feedback
+ fetchTimeSlots(restName : String) : Map<String,Integer>
+ fetchHostMenu(restName : String) : ArrayList<ArrayList<String>>
+ fetchAdvertisement() : ArrayList<String>
+ fetchHostProfile(email : String) : Profile
+ fetchUserProfile(email : String) : Profile
+ saveUserProfile(profile: Profile) : void
+ saveHostProfile(profile: Profile) : void
+ saveAdvertisement(adv : ArrayList<String>) : void
+ savePromotion(promo : ArrayList<String>) : void
+ saveTimeSlots(avail : Map<String,Integer>) : void
+ saveMenu(menu : ArrayList<ArrayList<String>>) : void
+ saveReservation (restName : String, userTimeSlot : String, email : String) : void
Filter
- paymentType: ArrayList<String>
- services : ArrayList<String>
- cuisinesOffered : ArrayList<String>
- mealType : ArrayList<String>
- dietRestrictions : ArrayList<String>
- priceRange : int
+ getPaymentType() : ArrayList<String>
+ getServices() : ArrayList<String>
+ getCuisinesOffered() : ArrayList<String>
+ getMealType() : ArrayList<String>
+ getDietRestrictions() : ArrayList<String>
+ getPriceRange() : ArrayList<String>
+ setPaymentType(payment : ArrayList<String>) : void
+ setServices (services: ArrayList<String>) : void
+ setCuisinesOffered(cuisines : ArrayList<String>) : void
+ setMealType(meal : ArrayList<String>) : void
+setDietRestrictions(dietRestrictions : ArrayList<String>) : void
+ setPriceRange(range: int) : void
MainController
- numDispFields : int
- widthDispField : ArrayList<Integer>
- lenDispField : ArrayList<Integer>
- colorDispField : ArrayList<String>
- posDispField : ArrayList<String>
- typeDispField : ArrayList<String>
- fields : ArrayList<String>
- toggleButtons : ArrayList<String>
- timeSlots : Map<String, Integer>
- title : String
- logo : String
+ getUserChoice(button : ArrayList<String>) : ArrayList<String>
+ displayEditableFields(fieldIds : ArrayList<String>) : void
+ getUserInput(fields : ArrayList<String>) : Map<String, ArrayList<String>>
+ getUserTimeSlot(timeSlots : ArrayList<String>) : String
+ saveFields() : void
+ displayProfile(profile : Profile) : void
+ getUserFilters() : Filter
+ getRestaurantChoice() : String
+ readDefaultFilter() : Filter
+ displayRecList(recList : ArrayList< Map<String,String> >) : void
+ displayTimeSlots(timeSlot : Map<String, Integer>) : void
+ getTimeChoice(button : ArrayList<String>) : String
+ displayNotification() : void
+ displayFeedback(feedback : Feedback) : void
+ displayFilter(filter : Filter) : void
+ displayAdvertisement(advertisement : String) : void
+ displayTheme(theme : String) : void
+ displayReservationS(): void
Memento Design Pattern
Originator
- State : String
+ setSate(String state) : void
+ saveToMemento() : Memento
+ restoreFromMemento() : void
Memento
- State : String
+ Memento(String state) : void
+ getStavedState() : String
Caretaker
- savedStates : ArrayLsit<Memento>
+ addMemnto(Memento) : void
+ getMemnto(int) : String
Feedback
- fbName : String
- feedback : String
- rating : int
- date : String
+ getFbName() : String
+ getFeedback() : String
+ getRating() : String
+ getDate() : String
+ setFbName(name : String) : void
+ setFeedback(feedback : String) : void
+ setRating(rating : Integer) : void
+ setDate(date : String) : void

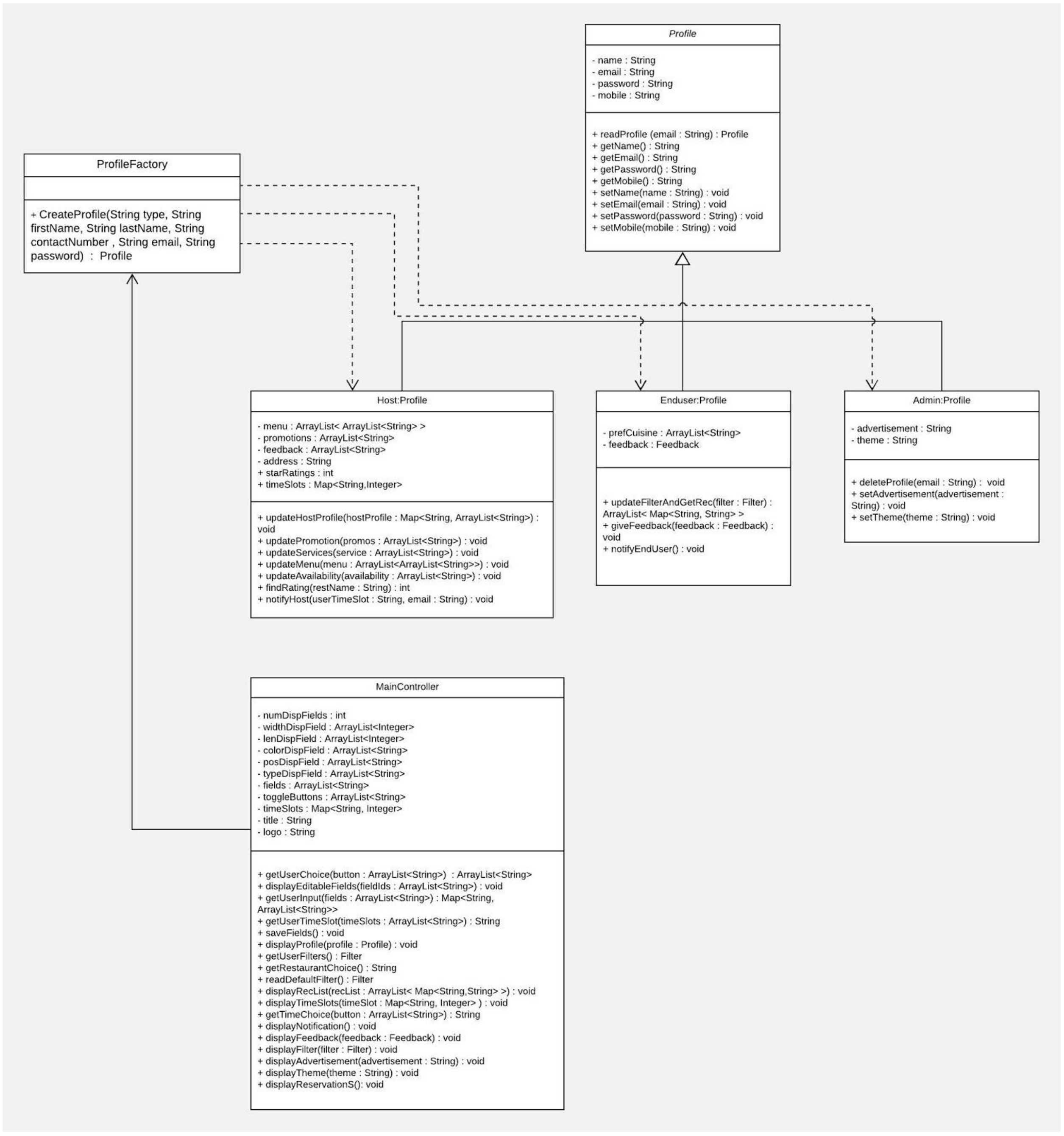
We made a lot of improvements and corrected the mistakes committed in Part 2 in our subsequent class diagram:

- Classes that were missing or misnamed were corrected in the class diagram
- Corrected a few instances of incorrect notation in the class diagram
- Identified and added design patterns that can be used in our project
- Removed the “Authentication” class since it was not necessary for the project
- Created an interface called “ModifyProfile”, which is used to create multiple end users (multiple instances) of the end user, by the “EndUser” class, which inherits from a abstract class called “Profile”.
- Modified and added methods & attributes to the Recommendation class for being the class using which recommendations can fetched from the database and request the algorithm to generate recommendations based on the filter parameters for the end user.

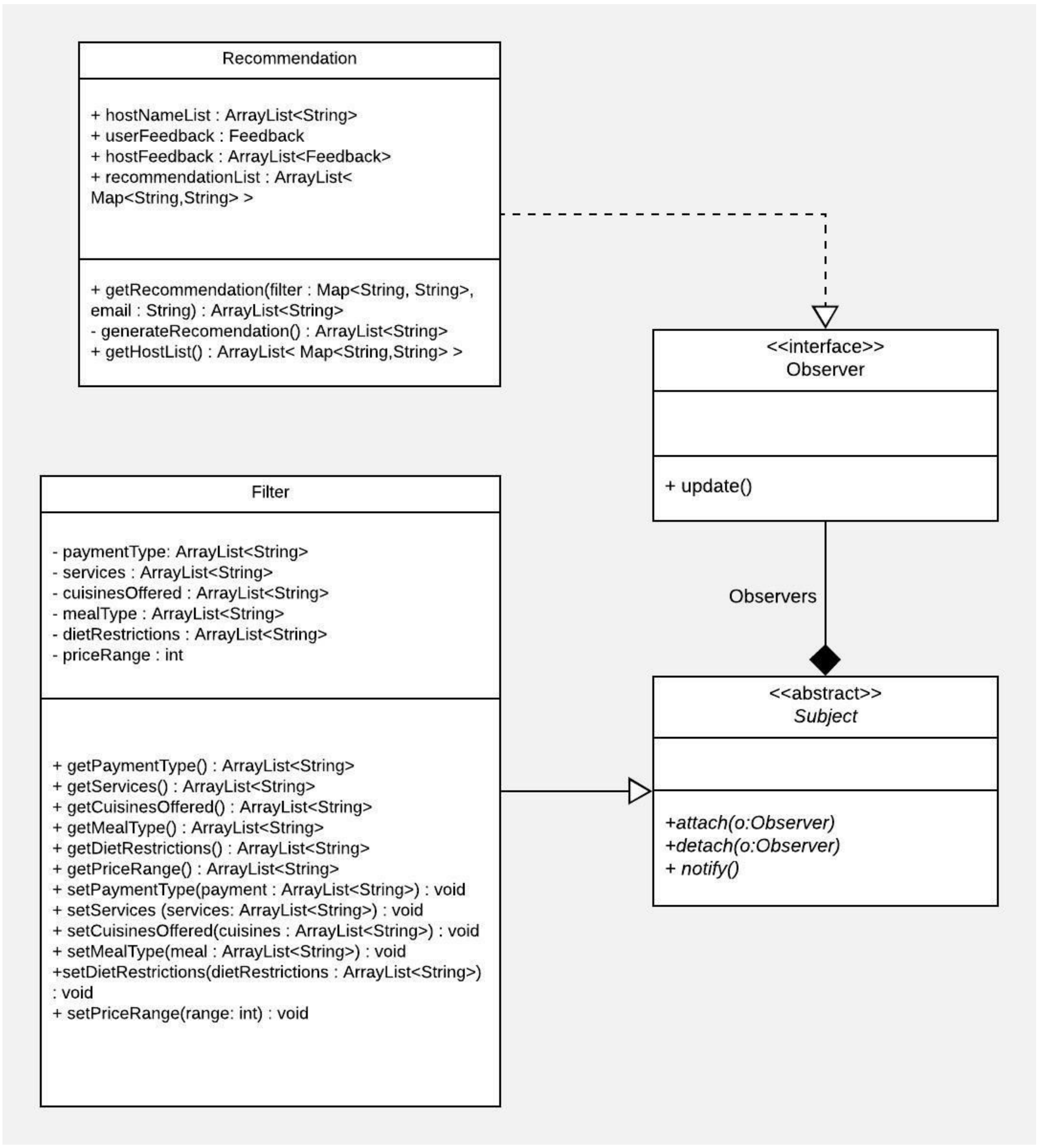
- Used to the java.utils.Observer package to implement the Observer interface for the Observer design pattern.
- Merged the functionality of the “Advertisement” class into the “Admin” class, where the advertisements are stored as an ArrayList of strings. The “Advertisement” class was therefore deleted.
- Added a number of new methods to the “Filter” class to get and set multiple attributes of the host. For example, getters and setters were used and created for the following class attributes: “paymentType”, “services”, “cusinesOffered”, “mealType”, “dietRestrications” and “priceRange”.
- Added and modified the database manager class (“DbManager”) with a lot more specific fetch and save methods, corresponding to the implemented MySQL database
- Added attributes to the “Feedback” class: “fbname”, “feedback”, “rating” & “date”. Added their corresponding getters & setters.
- Modified the “Host” class with methods & attributes to update the menu, read feedback & read ratings.
- Restricted the “Admin” class to set advertisements & delete profile (end user or host)
- Created the “ProfileFactory” class to facilitate the implementation of the Factory design patterns.
- Replaced the “GUI” class with the “MainController” since we changed the method of accessing the classes through the Hibernate and Spring MVC framework.

5. DESIGN PATTERNS IMPLEMENTED

Factory Design Pattern:


Profile
- name : String
- email : String
- password : String
- mobile : String
+ readProfile (email : String) : Profile
+ getName() : String
+ getEmail() : String
+ getPassword() : String
+ getMobile() : String
+ setName(name : String) : void
+ setEmail(email : String) : void
+ setPassword(password : String) : void
+ setMobile(mobile : String) : void
ProfileFactory
+ CreateProfile(String type, String firstName, String lastName, String contactNumber , String email, String password) : Profile
Host:Profile
- menu : ArrayList< ArrayList<String> >
- promotions : ArrayList<String>
- feedback : ArrayList<String>
- address : String
+ starRatings : int
+ timeSlots : Map<String,Integer>
+ updateHostProfile(hostProfile : Map<String, ArrayList<String>) : void
+ updatePromotion(promos : ArrayList<String>) : void
+ updateServices(service : ArrayList<String>) : void
+ updateMenu(menu : ArrayList<ArrayList<String>>) : void
+ updateAvailability(availability : ArrayList<String>) : void
+ findRating(restName : String) : int
+ notifyHost(userTimeSlot : String, email : String) : void
Enduser:Profile
- prefCuisine : ArrayList<String>
- feedback : Feedback
+ updateFilterAndGetRec(filter : Filter) : ArrayList< Map<String, String> >
+ giveFeedback(feedback : Feedback) : void
+ notifyEndUser() : void
Admin:Profile
- advertisement : String
- theme : String
+ deleteProfile(email : String) : void
+ setAdvertisement(advertisement : String) : void
+ setTheme(theme : String) : void
MainController
- numDispFields : int
- widthDispField : ArrayList<Integer>
- lenDispField : ArrayList<Integer>
- colorDispField : ArrayList<String>
- posDispField : ArrayList<String>
- typeDispField : ArrayList<String>
- fields : ArrayList<String>
- toggleButtons : ArrayList<String>
- timeSlots : Map<String, Integer>
- title : String
- logo : String
+ getUserChoice(button : ArrayList<String>) : ArrayList<String>
+ displayEditableFields(fieldIds : ArrayList<String>) : void
+ getUserInput(fields : ArrayList<String>) : Map<String, ArrayList<String>>
+ getUserTimeSlot(timeSlots : ArrayList<String>) : String
+ saveFields() : void
+ displayProfile(profile : Profile) : void
+ getUserFilters() : Filter
+ getRestaurantChoice() : String
+ readDefaultFilter() : Filter
+ displayRecList(recList : ArrayList< Map<String,String> >) : void
+ displayTimeSlots(timeSlot : Map<String, Integer>) : void
+ getTimeChoice(button : ArrayList<String>) : String
+ displayNotification() : void
+ displayFeedback(feedback : Feedback) : void
+ displayFilter(filter : Filter) : void
+ displayAdvertisement(advertisement : String) : void
+ displayTheme(theme : String) : void
+ displayReservationS(): void

Observer Design Pattern:


Recommendation
+ hostNameList : ArrayList<String>
+ userFeedback : Feedback
+ hostFeedback : ArrayList<Feedback>
+ recommendationList : ArrayList< Map<String,String> >
+ getRecommendation(filter : Map<String, String>, email : String) : ArrayList<String>
- generateRecomendation() : ArrayList<String>
+ getHostList() : ArrayList< Map<String,String> >
<<interface>> Observer
+ update()
Observers
<<abstract>> Subject
+attach(o:Observer)
+detach(o:Observer)
+ notify()
Filter
- paymentType: ArrayList<String>
- services : ArrayList<String>
- cuisinesOffered : ArrayList<String>
- mealType : ArrayList<String>
- dietRestrictions : ArrayList<String>
- priceRange : int
+ getPaymentType() : ArrayList<String>
+ getServices() : ArrayList<String>
+ getCuisinesOffered() : ArrayList<String>
+ getMealType() : ArrayList<String>
+ getDietRestrictions() : ArrayList<String>
+ getPriceRange() : ArrayList<String>
+ setPaymentType(payment : ArrayList<String>) : void
+ setServices (services: ArrayList<String>) : void
+ setCuisinesOffered(cuisines : ArrayList<String>) : void
+ setMealType(meal : ArrayList<String>) : void
+setDietRestrictions(dietRestrictions : ArrayList<String>) : void
+ setPriceRange(range: int) : void

6. EXPERIENCES & LESSONS LEARNT

//TODO!!