Project:

Team Members: Abdulrahman Alaraj, Chirag Kamat, Ganesh Byrandurga Gopinath.

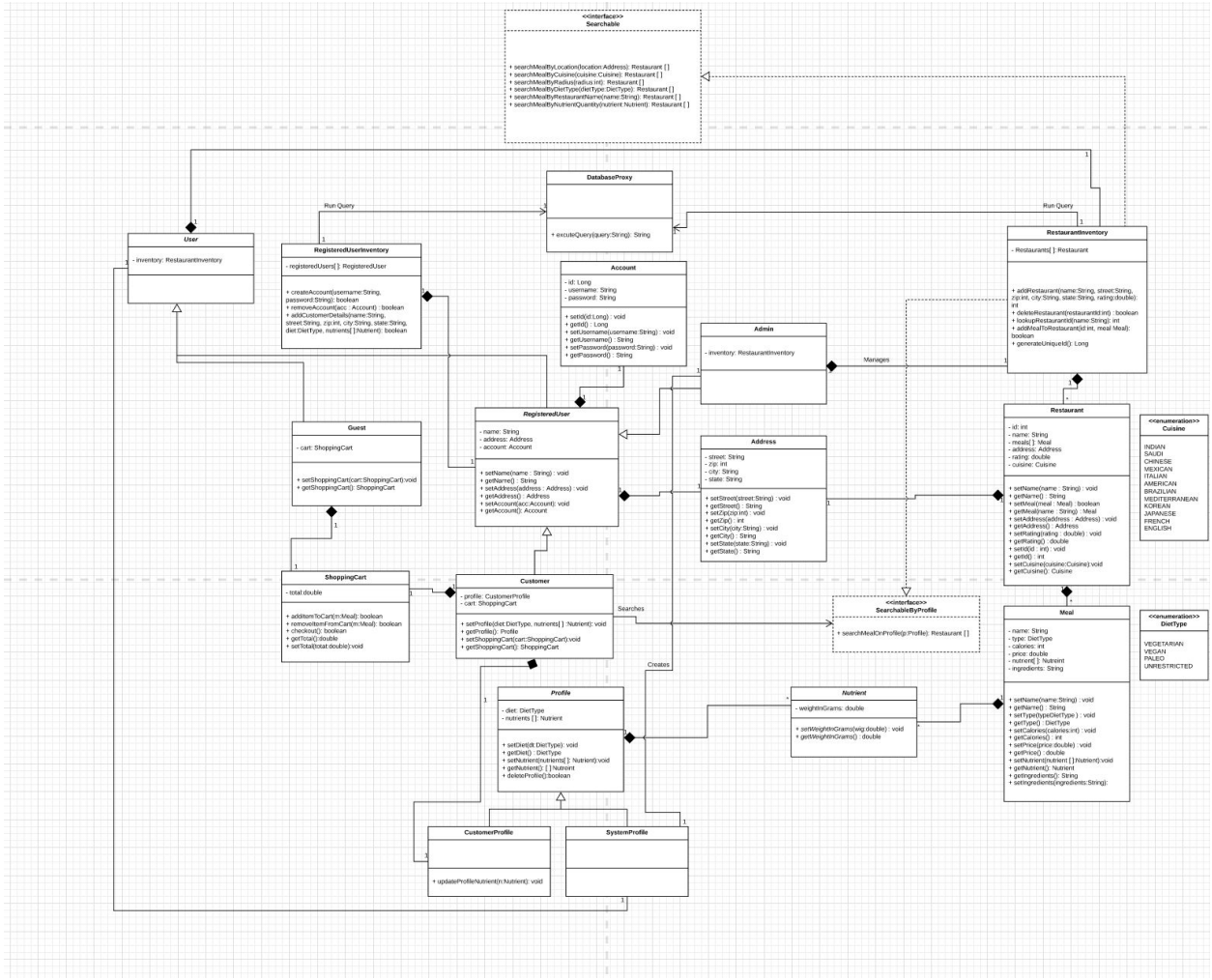
Team Number: 40

Title: FitFoodie

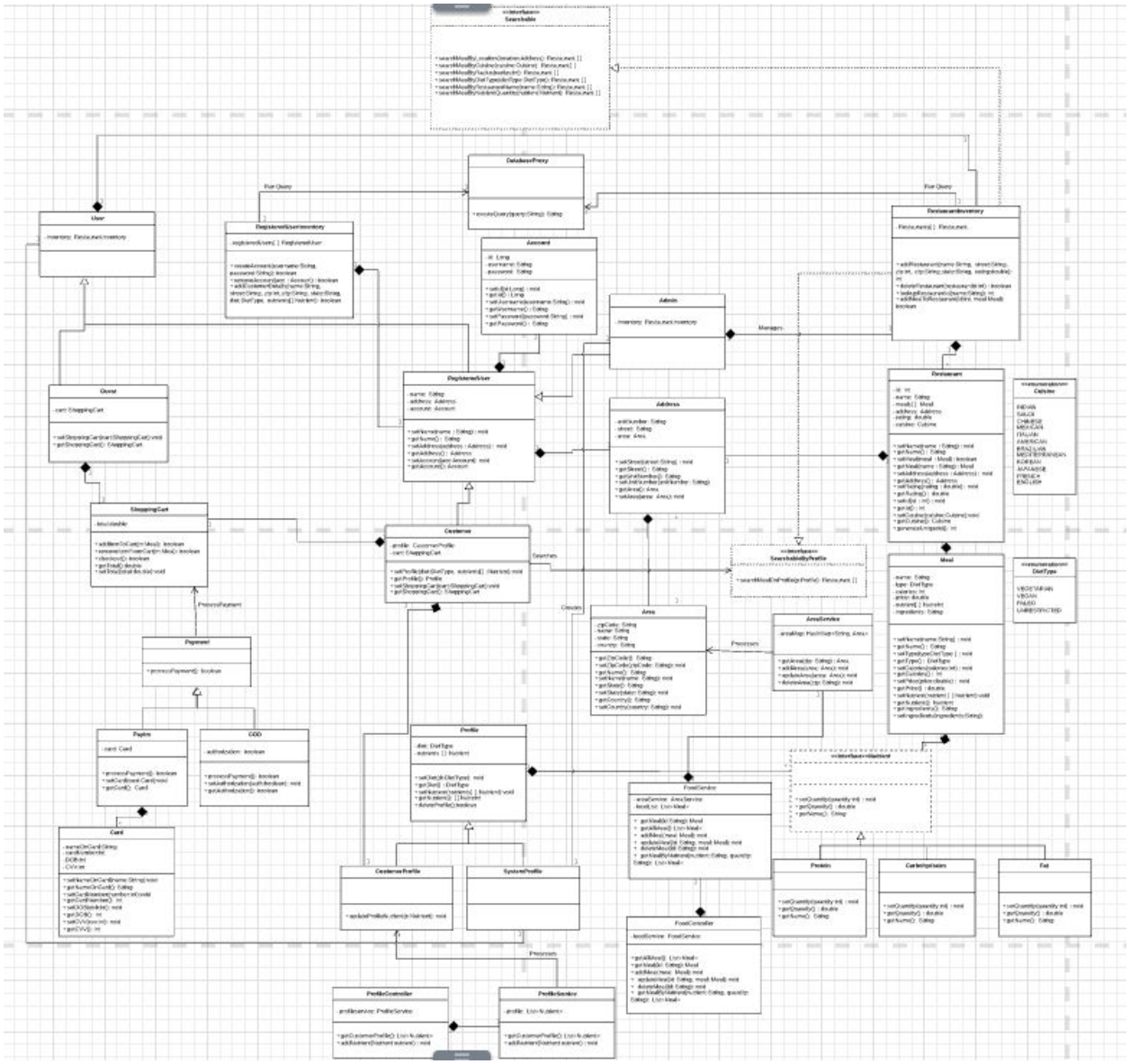
Description: A web based system which recommends restaurants and allows the user to order food based on nutrition preferences\requirements.

Vision: Building a web based system by leveraging object oriented frameworks and applying object oriented design patterns.

Previous class



Completed Class diagram:



Please follow this link to view a clearer version of our current class diagram - https://www.lucidchart.com/invitations/accept/c08a2889-d494-466e-91c9-267f7a8dab51

The class diagram can be found under “class-diagram-part4” tab.

Summary: We had 3 team meetings, on the first one, we discussed on the framework and design patterns that we need to implement. On the second meeting, we discussed and shared information on the spring setup and started following spring tutorials online. We also implemented very basic application of our project. We distributed the work among ourselves. On the third meet, we worked on the changed class diagram, reviewed each other's work and worked on the project 4 report.

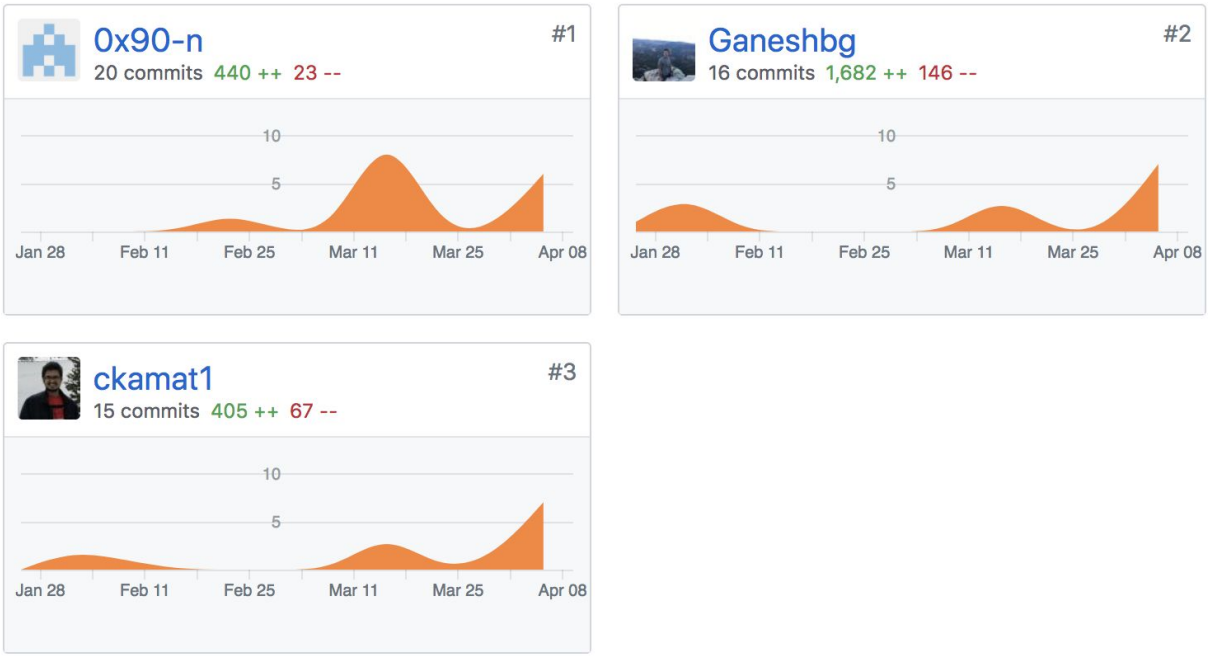
Breakdown:

Abdul: Springboot setup, Implemented profile service and profile controller so the customers can initialize their own profiles. Extended the payment options to include cash on delivery. Applied strategy design pattern to encapsulate the algorithm for processing customers payments.

Chirag: Springboot setup, Implemented and made changes in various services required by the system such as AreaService, FoodService. Implemented Area class. Updated Address class to now be composed of Area. Implemented flyweight design pattern on the AreaService class which creates Area objects and helps in reusing them.

Ganesh: Springboot setup, created the classes Meal, Restaurant, Address, Nutrients, Protein, Fat and Carbohydrates. Implemented respective functionalities in the classes. Implemented and tested POST, GET, UPDATE and DELETE functionalities on Meal class. Implemented strategy design pattern on Nutrients class and its sub-classes Fat, Carbohydrates and Protein.

GitHub Graph:



Estimate remaining Effort: We have implemented searchMealByNutrient, adding, deleting and updating meal in the list. We need to extend the same functionality for searchMealByLocation, searchMealByRadius, searchMealByCuisine, searchMealByRestaurantName. We have implement user payment. We have to link user login and user payment, We have implemented searchByProfile and user payment needs to be integrated with user login. We are confident that we can complete the project in the remaining time.

Design Patterns:

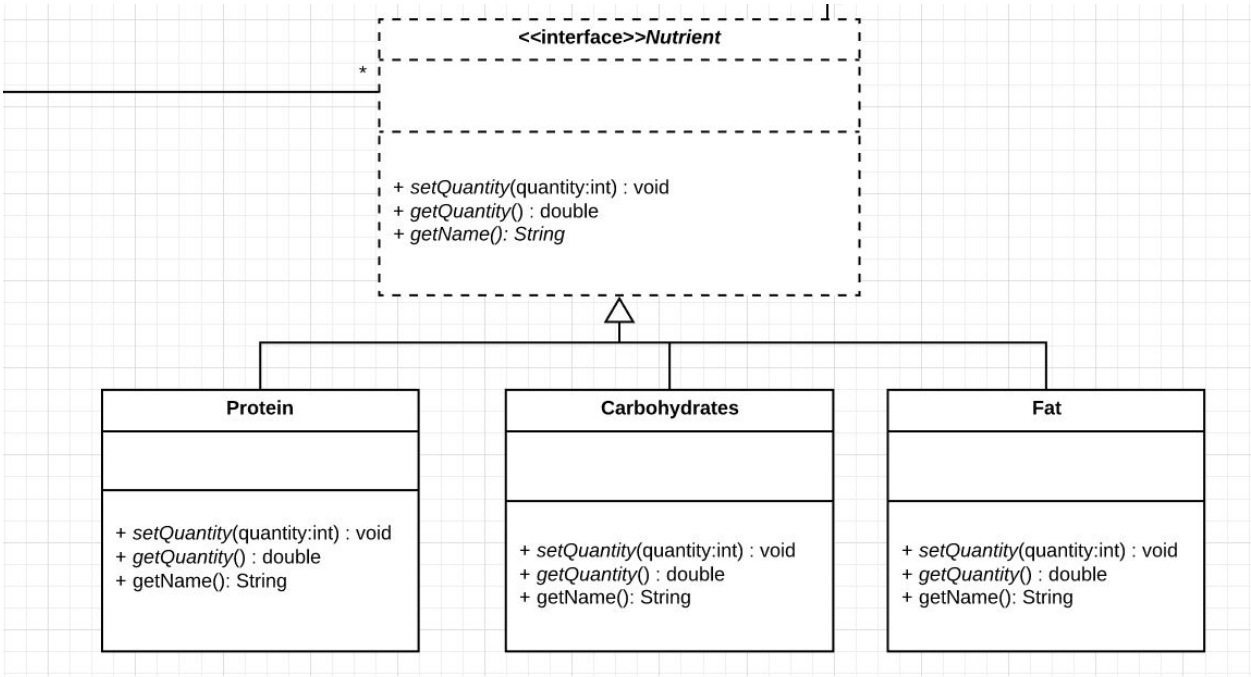


Figure 3: Class Diagram showing Strategy design pattern

Ganesh: Strategy Design Pattern has been implemented in the above class diagram. The code is implemented according to the above diagram.

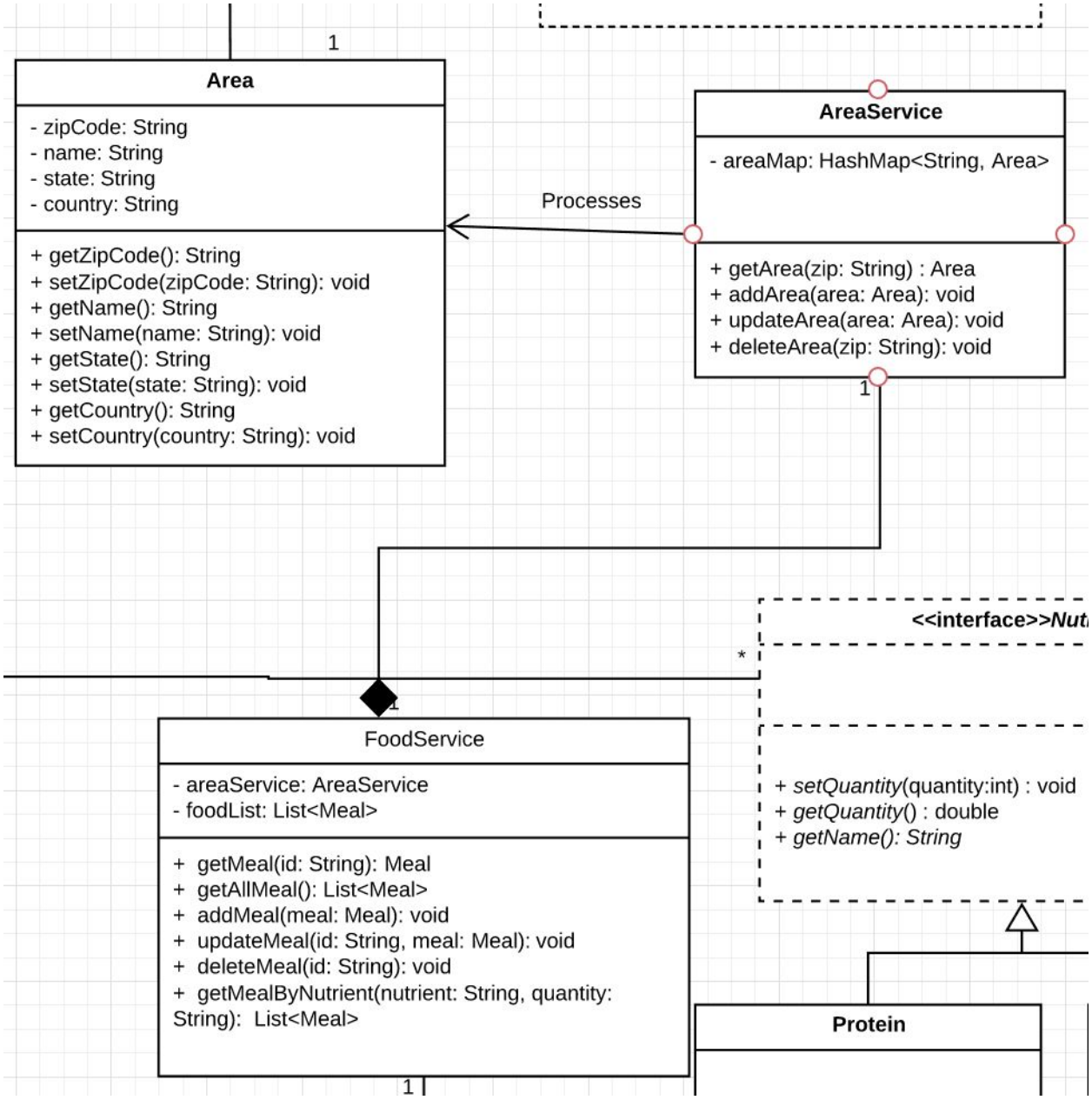
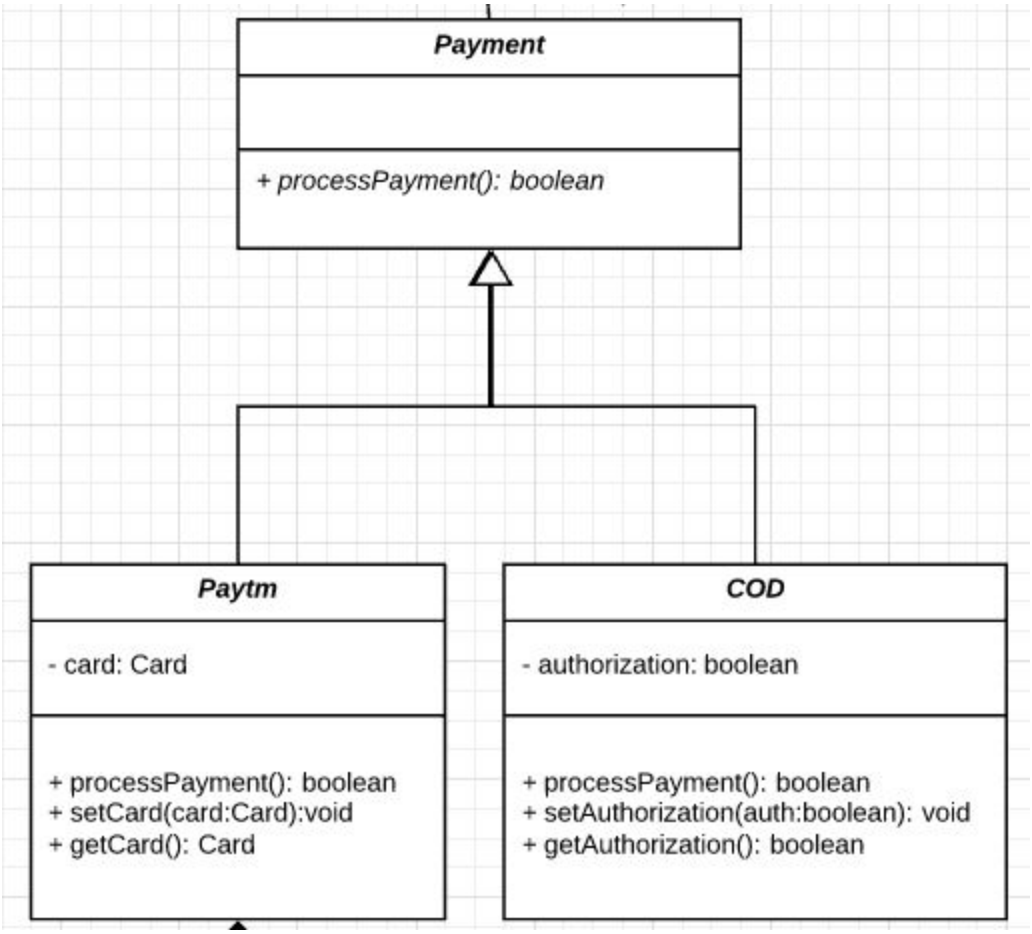


Figure 4: Class Diagram showing Flyweight design pattern

Chirag: Flyweight design pattern was used to store Area objects in the static AreaService class. Since there can be many restaurants located in the same area, area objects can be reused by more than one restaurant. Area objects are stored in a hashmap with zipcode as a unique key identifier in the hash map. When a new meal and restaurant and its corresponding area is added, we first check the areaMap provided by AreaService class. If the Area object exists in the map we just return that area. Else, we store a new area object in the map. This helps to save a lot of memory in our system.



Abdul: Strategy design pattern was used to avoid conditionals while processing customers payments. Without the strategy design pattern, we would need to verify the payment type using conditionals which, in turn, violates one of the object oriented design principles (SOLID), namely open-closed principle. Hence, the algorithm for processing customers payments is encapsulated.

Final Iteration: We will implement the user login and authentication and linking searchByProfile with specific users. User payment using Paytm and cash on delivery needs to be linked with user login. We also need to make data persistent. We will refactor and clean the code if needed.