Gourmand

Food Industry & Restaurants

Final Project Report

CSP 586

Computer Science

Illinois Institute of Technology

Instructor: Dr. Atef Bader

Team Members

Maurya, Shailesh | smaurya@hawk.iit.edu (Team Leader)

Shethwala, Mohammed | mshethwa@hawk.iit.edu

Kumar, Aditya | akumar61@hawk.iit.edu

Sharma, Bijay | bsharma@hawk.iit.edu

Table of Contents

1. Business Statement

The Gourmand app is a perfect app for all the foodies out there who want an all-in-one app for finding restaurants, restaurant info, food ordering, home deliveries and table booking. Gourmand makes the life of foodie a lot easier by providing all the food & restaurant related services in a single mobile app. Now, the foodies can search and find their favorite restaurants on a single click and also get suggestions for new restaurants based on their search. The foodies can view all the information of a restaurant in a well-organized manner which includes the menu, review, photos and a lot more. There are a number of food ordering options like dine-in, home delivery and pick-up. Foodies who want to reserve a table can book a table using the table booking feature of Gourmand.

Gourmand is not just for the foodies. The restaurants, who love to serve and provide the best service to their customers can now manage their restaurants with Gourmand. A restaurant can add or update their menu, add events, accept orders, accept table reservations, accept deliveries, dine-in and pick-ups. The restaurants who want to get an upper hand in the competitive market can register for premium services provided by Gourmand. The premium services include market analytics, advertising and a lot more to increase their business. Premium members can display their ads for a specific type of users which are predicted by Gourmand's Foodie Predictor and thus can enhance their business by providing more coupons and services to the identified customers.

We also have a small section for all those food critics who want to express themselves. A critic can register with Gourmand and after being verified, can post reviews and blogs for all the foodies. This improves the quality of the reviews and the foodies can also get expert reviews for the restaurant they are planning to visit.

Gourmand is an app which strives to make life a lot easier for all the foodies and the restaurants and making them happy in every possible way.

2. Feature List

- There are 3 types of users in the system:

    1. **Foodie**: These are the primary users and comprise of the majority of the user base. A foodie is just a normal person who uses the app to search, order, and book restaurants. *'User' & 'Foodie' are used interchangeably in the document and both indicate the primary user of this app.*
    2. **Restaurants**:
        a) **Basic**: They get features like managing restaurant dashboard, updating events and menu.

b) **Premium**: Premium users get all the basic features plus they get foodie market analytics and can advertise themselves on the app more effectively to specific selected user base. They can update coupons also.

3. **Critics**: These are specialized and verified food critics and can register with the app. They can review any restaurant and write blogs. They get all the functionality of the Foodie users as well.

User (Foodie) Related Features

- A foodie selects a city in which he would like to search for restaurants. Then, the foodie can search for any restaurant by entering the search description (restaurant name). The app displays all the restaurants matching the search criteria. The search criteria can further be filtered by the following filters:
    1. Cuisine
    2. Rating (1 to 10)
    3. Price (Price Range)
    4. Distance (From Zip code entered)
    5. Feature (Coupons Available, Order Tracking, Newly Opened, Open Now, Free Delivery, Express Delivery, Rapid Pickup)
- After the search has been modified, the sorting of the list of the restaurants can also be done by the foodie based on the following sort criteria:
    1. Restaurant Name
    2. Price (Ascending & Descending)
    3. Rating
    4. Distance
    5. Pickup Estimate
- A foodie can also select from a several categorized restaurants under a specific category list. Different categories include:
    1. Cuisines
    2. Establishments (Food Court, Fine Dining, Newly Opened etc)
    3. Food Time (Breakfast, Brunch, Lunch, Dinner, Midnight Snack)
    4. Trending
    5. Nearest
    6. Specific Dishes
- After clicking on a restaurant, the restaurant page should be displayed in the app. The restaurant page should include the following:
    1. Phone Number
    2. Address
    3. Map (Directions)
    4. About (History and Description)
    5. Cuisines Offered
    6. Operating Hours
    7. Highlights (Lunch, Dinner, Pick-up)
    8. Cost for One/Two

9. Minimum Order (if any)
10. Delivery Fees (if any)
11. Order Food (Pick-up, Dine-in or Home Delivery)
12. Book Table
13. Menu
14. Photos
15. Reviews

- The Order Food section has 3 types of orders:
    1. **Pick-up**: Foodie selects the items from the menu and the estimated time to pick-up is displayed. Foodie selects type of payment and checks out. Order is sent to the restaurant and Foodie then selects the order from the restaurant.
    2. **Dine-in**: App displays if dine-in option available at selected time. If available, book table is shown and foodie books a table and orders food and checks out after selecting payment type and paying the bill.
    3. **Home Delivery**: If delivery available at restaurant, then only next step is shown otherwise, no delivery available is shown. Foodie orders the food and selects the type of payment and checks out. Order is sent to the restaurant and delivery is done.
- Foodies can track their order via the order tracking functionality.
- The book table feature can also be done explicitly apart from the dine-in option. The foodie selects a restaurant and selects the date and time to book a table. The app shows availability. Foodie selects any available table and books the table. Booking table is sent to the restaurant and a table is booked for the foodie.
- There are 3 types of payment options:
    1. Credit/Debit Card
    2. PayPal Card
    3. Cash on Delivery
- There are 2 types of delivery options offered by the restaurants:
    1. Normal Delivery
    2. Rapid Delivery
- Every foodie can view the profile of another foodie in the app. The profile of a foodie contains the following things:
    1. Name
    2. Profile Pic
    3. Follow (Click to follow a foodie)
    4. Description of Foodie
    5. Foodie Rating (done by foodies & Gourmand app)
    6. Reviews
    7. Photos
    8. Followers
    9. Gourmand Foodie Rank
    10. Rewards Gained

- Every foodie can view the profile of any critic and vice-versa. The profile of the critic is same as the foodie but adds the verified mark and also has the following extra things:
    1. Food Posts & Photos
    2. Expert Reviews
- A review of any restaurant can be done by a foodie and critic. Before reviewing, a rating is also submitted by the foodie. A review can have comments and likes as well.
- Photos can be added by the foodie or any critic for any particular restaurant.
- There is a Rewards functionality for the foodie. Every order type has some points associated with it. Whenever a foodie orders food for dine-in, pick-up or delivery, points are added to the foodie's profile. After 1 month of successful ordering and using the app, the foodie can apply the rewards points to the next order and get discounts based on the points.
- Foodie can apply coupon codes to any type of order to get further discounts. There are 2 types of coupons:
    1. Gourmand Promo Coupons
    2. Restaurant Provided Coupons

Restaurant Related Features

- Every registered restaurant has a dashboard which can be managed by the restaurant manager. Following things can be done by the restaurant:
    1. Basic:
        a) Update Menu
        b) Add & Update Events
        c) Change Operating Hours
        d) Add Photos
    2. Premium:
        a) All the Basic Features
        b) View Foodie Analytics
        c) Advertise
- After the order has been placed by the customer, the order is sent to the restaurant. If the restaurant supports any of the order options, then order tracking is supported by the app for the foodie to track their order. The restaurant app manager updates the different types of orders from within the app, which is then updated to the foodies.

Admin Related Features

- The admin has the right to verify the critics. All the verified critics and verified foodies are done by the admin.
- The admin can update (add, update, delete) restaurants.
- The admin can verify reviews. Flagged reviews are verified by the admin and submitted by the permission of admin only.

3. List of Requirements

3.1 User Related Requirements

1. User shall be able to register with the app by entering the following information: first name, last name, email address, password, and phone number.

2. User shall be able to login to the app by entering valid username and password.

3. The app shall be able to verify the username and password of the user entered at the time of login. The app shall login the user if username and password is correct, else 'invalid username/password' warning shall be displayed to the user.

4. The app shall give the option of 'forget password' to the user at the time of login.

5. User shall be able to enter his/her email address in the forget password section and the app shall send the one time password to the user.

6. User shall be able to search for the restaurants by entering the restaurant name in the app.

7. User shall be able to filter the search using filter options like: cuisine, rating, price range, distance from user, coupons available, order tracking available, newly opened or not, open now or not, free delivery offered, express delivery offered and rapid pickup available.

8. User shall be able to sort the list of searched restaurants using the following search options: restaurant name, price, rating, distance and pick-up estimate.

9. The app shall display the list of restaurant based on the filter options and sort options selected by the user.

10. User shall be able to select different types of restaurant category like cuisines, establishments, food time, trending, nearest and specified dishes. The app shall display all the restaurants which come under the selected category by the user.

11. User shall be able to view all the details of a particular restaurant which include: phone number, address, map, restaurant description, cuisines offered operating hours, highlights such as lunch, dinner and breakfast, cost for one/two, minimum order, delivery fees, order food, book table, menu, photos and reviews. The app shall display all the related information to the user when a particular restaurant is selected.

12. User shall be able to order food by selecting any one of the following techniques: pick-up, dine-in and home delivery.

13. User shall be able to select the items from the menu in the restaurant one by one. The app shall display all the selected items in a list and also display the total for the items selected from the menu.

14. User shall be able to select the pick-up order type from the order types available. The app shall display the estimated pick-up time after all the menu items are selected by the user.

15. User shall be able to select the dine-in order type from the order types available. The app shall display the book table option to the user. The user shall select the book table option and be able to book a table after specifying the number of persons. App shall book a table if the table is available at the restaurant or else display the next available slot for booking a table. App shall display the wait time for the particular booking of a table.

16. User shall be able to select the menu items from the dine-in option. App shall add the items to the list of items ordered and display the total items with the cost for all the items.

17. User shall select the home delivery option in the order type options. The app shall display the different delivery options like normal delivery and rapid delivery. The app shall display the total cost after the selection of the menu and selection of delivery type based on the delivery type selected.

18. The app shall display the order number after a particular order has been finalized and been paid for.

19. User shall be able to track his/her order by entering the order id given to them after an order is completed and paid for.

20. The user shall be able to book a table apart from selecting to book a table from the types of order. User shall be able to select the date, time and number of persons to book a table. The app shall display the availability and wait time if available for that particular restaurant.

21. User shall be able to pay for any order through any of the following payment methods: credit card, debit card, paypal and cash. The app shall display these options when the user has finished selecting items and selects to pay for the order.

22. User shall be able to pay for any order by his/her credit/debit card by entering details like card holder’s name, card number, expiry date, cvv number and selecting the type of the card as master card, visa or discover. The app shall capture the card information and verify the information with the respective bank and successfully complete and deduct the amount of the order from the card. Receipt of the payment shall be displayed to the user after the payment is completed.

23. User shall be able to update their own profile by editing the following information from their profile: name, profile pic, self-description and adding photos.

24. The app shall add reward points when a user order foods with any type of order. The reward points shall be displayed in the user profile.

25. User shall redeem the reward points anytime while placing the order. The redeem points shall be applied by the app to the user's current order and discount shall be calculated for the current order after the application of reward points.

26. User shall be able to view all the rewards for a particular selected restaurants in a list fashion. User shall be able to add a review for a restaurant and also select a rating out of 10 for the restaurant. The app shall be able to save the review and add it to the list of reviews for that restaurant.

27. User shall be able to add photos for the restaurant selected. User shall upload multiple photos at a time for a particular restaurant. The app shall upload the photos and display them in user photos as well as in the restaurant photos.

28. User shall be able to apply any coupon (restaurant coupon or app coupon) to any order by adding coupon code to the order. The app shall capture and validate the coupon code and apply discount if available to the order.

29. User shall be able to subscribe to events which are added by the restaurant. Whenever a new event is added by the restaurant, the user shall be notified in the app.

30. User shall be able to view a profile of any other user who uses the app. The user profile visible to the user shall include the following things: name, profile pic, link to follow the user, description of user, user rating, reviews, photos, followers, gourmand rank and rewards earned.

3.2 Restaurant Related Requirements

1. Restaurant shall be able to register with the app by providing the information like restaurant name, contact number, email address, address and password. The app shall save the information and register a restaurant with the app.

2. Restaurant shall be able to view their restaurant dashboard in the app. The dashboard shall display all the information of the restaurant and also the events and menu of the restaurants.

3. Restaurant shall be able to create an event by adding the information like event name, event date and time of event. The app shall save the information and add the event to the list of events offered by the restaurant. This event shall be displayed in the restaurant profile which is viewed by the user.

4. Restaurant shall select the type of membership at the time of registration with the app like: Basic & Premium. If the membership type is Basic, the app shall display the following features in the dashboard menu: update menu, add & update events, change of operating hours and add photos. If the membership type is Premium, the app shall display all the features available in the basic plus the following features: view analytics and an option to advertise.

5. Restaurant shall view all the analytics like total number of views, total number of advertised views, total number of orders and total number of table bookings. The app shall calculate all these analytics whenever the restaurant selects to view the analytics.

6. Restaurant shall be able to view the analytics in three different ways which are: bar chart, line chart and pie chart.

7. Restaurant shall be able to update the status of an order through the app. The app shall capture the status at every time and update it to the user whenever a user wants to check the order status.

8. Restaurant shall be able to update the dashboard and restaurant profile by editing the following things: location, address, contact number, operating hours, menu, table options, delivery options and cuisines offered.

9. Restaurants shall be able to advertise themselves on the app. The restaurants who pay for advertising shall be displayed as featured restaurants to a particular category of users.

10. Restaurant shall be able to add their own coupon on the app by adding the following information related to the coupon: coupon name, coupon code, start date and end date. The app shall save the coupon and the coupon shall be displayed on the restaurant profile which is visible to all the users.

3.3 Admin Related Requirements:

1. The admin shall be able to add/delete/update any restaurant by adding or editing the following information about a restaurant: restaurant name, address, phone number, operating hours and all other restaurant related information.

2. The admin shall be able to verify a critic by either accepting or rejecting a critic's registration request. The app shall sent the verification notification to the critic once a decision has been made by the admin.

3. The admin shall be able to verify reviews written by the user and the critics. The app shall request the admin to verify a review if any obscene language is found in the review by the app. The admin shall check and allow the review to be updated or deleted to the list of reviews.

3.4 Critic Related Requirements:

1. A critic shall be able to register as a critic by giving the reference names in the information required for a critic. The app shall transfer the request to the admin. After verification from admin, the result should be passed on to the critic if approved or not.

2. A critic shall be able to write critic reviews for any restaurant. The app shall capture the review and display it to the critics review section of the restaurant reviews.

3. A critic shall be able to also write reviews for any dishes from the menu of any restaurant. This reviews shall be updated by the app and displayed in the dish reviews section of the restaurant reviews.

4. List of Use Cases

4.1 List of Actors

1. User (Foodie)
2. Restaurant
3. Critic
4. Admin

4.2 User (Foodie) Use-Cases

1. User Register: The user opens the Gourmand app and selects to register with the app. The system displays a register form to be filled up by the user. The user enters the first name, last name, email address, password and phone number in the form and selects to register with the app. The system accepts and saves the user data to the database and takes the user to dashboard.

2. User Login: The user is already registered user and opens the Gourmand app. System displays the Login Screen to the user with Username and Password to be entered by the user. User enters username (email address) and password and clicks on Login button. The system verifies the username and password and if correct, logs the user into the system and displays the dashboard to the user.

3. Search Restaurant: System displays a search bar to the user for entering the city name. User enters city name in the search bar. Now, System displays the search bar for the user to enter name of the restaurant. User enters name of the restaurant and clicks the search button. System displays all the restaurants matching the search criteria and also displays below that some featured restaurants list based on the search criteria of the user. System also displays search filter options. User selects the filter options from the filter bar and filters the search based on cuisine, rating, price range, distance, and other features like coupons available, order tracking, newly opened, open now, free delivery, express delivery and rapid pick-up. System refines the list and shows a new list of restaurant based on the new filter criteria entered by the user.

4. Sort Restaurants: User selects the sort option in the app. System displays the different types of sorting criteria like Restaurant Name, Price, Rating, Distance and Pick-Up Estimate. User selects one or more type of sort options. System displays the sorted restaurant list to the user.

5. View Restaurant: From the list of all the restaurants, the user selects any one restaurant. System displays the profile of the selected restaurant. User views the name, address, contact details, open hours, description of the restaurant, cuisines offered, highlights (lunch, dinner or breakfast), cost for two, minimum order (if any), delivery fees. User selects to see the restaurant on the map. System displays the Map and the user can enter the start point of the location. System displays the estimated time to reach the restaurant. User selects to view the photos, menu and reviews & comments. System displays the already uploaded photos, reviews and menu of the restaurant.

6. Place Order: The user selects the order food option on the restaurant profile. The system displays three order food options: Pick-up, Dine-in and Home Delivery. The user selects any one of the options from the three options. System displays the selected page to the user in the app. This use-case is divided into three sub use-cases based on the selection of the order type by the user. The three sub-cases are described below as Pick-up, Book Table and Home Delivery.

7. Pick-up (Order Food Type): System displays the pick-up page to the user. User selects the items from the menu. System displays the items in the cart and the running total to the user when any new item is added to the cart. User selects to make order. System displays the items and total cost for the items. User confirms and selects to pick-up the order from the restaurant. System takes the user to payment options (another use-case). System displays the estimated time and order id to the user.

8. Dine-In (Order Food Type): System displays the Dine-In page to the user. System displays the book table view to the user. User selects the number of persons to book a table. System displays the table availability to the user. User selects the table and books the table. System displays the confirmation of table being booked and also displays the confirmation number to the user.

9. Home Delivery (Order Food Type): User selects the items from the menu one by one. System displays the items and the running total. User moves on to next step. System displays the total items and total cost for the food to the user. User confirms the order and moves on to the next step. System displays the 2 types of delivery options to the user: Normal Delivery and Rapid Delivery. User selects any one of the delivery types and moves on to the next step. System displays the new total if any charges of rapid delivery are applied to the order. User confirms the order. System takes the user to payment options (another use-case). System displays the order id to the user

10. Track Order: User selects the Track Order feature in the system. System displays the track order screen to the user. User enters the order id in the order id field. System displays the current order status to the user.

11. Pay for Order: User confirms all the order and moves on to the next step. The system displays three types of payment options: credit card, debit card and cash on delivery. User selects the type of payment and moves to next step. Depended on the selection type, the system prompts the user to enter credit or debit card details or displays

the confirmation for cash on delivery type. User enters the card details and selects next. System verifies the card details and accepts the payment and displays the confirmation to the user.

11.1. Pay with Credit/Debit: User selects the payment with credit type as the payment type options. The system displays the pay with credit form and the user enters the card number, card holder's name, and expiry date and cvv information in the form and selects to pay. System authorizes the payment information and displays the payment receipt to the user.

11.2. Pay with Cash: User selects to pay with cash at the payment type. System displays the total amount for the order and displays the order id to the user. System updates the payment and confirms it once the cash is given by the user to the delivery driver.

11.3. Pay with PayPal: User selects to pay with the PayPal account. System displays a form where user enters the PayPal username and password. System authorizes and confirms the payment with the user. User confirms the payment and the payment is deducted from the users PayPal account.

12. View User Profile: User selects the View Foodie functionality in the app. System displays the list of users (foodie) around the selected location of the user. System also displays the top rated users. User selects to view any one user profile. System displays the foodie profile to the user. User views the name, profile picture, description, foodie rating and reviews by the foodie, photos, followers, gourmand foodie rank and rewards earned. User also selects to follow the foodie. System increases the followers of the foodie by one.

13. Rate & Review: User selects to Rate & review any restaurant from the restaurant profile of the restaurant. System displays the rate & review screen to the user. User selects the rating out of 10 and writes a review in the review area. User submits the review. System accepts the review and goes to the admin for the verification of review. Admin verifies the review and the review is then visible to the user in the reviews screen.

14. Add Photos: User selects to add photos for a restaurant. System displays the screen to add photos. User selects to add photos from gallery or take a new picture. Based on the selection, the system opens the gallery or opens the camera, User selects the photo and uploads the photo. System saves the photo and adds it to the list of photos for a restaurant.

15. Show Rewards: System displays the rewards gained by the user until the current date for ordering food from the app. User can redeem rewards by selecting Redeem My Rewards option. System accepts the reward and adds the rewards to the user's next order.

16. Add Coupon to Order: System displays the final bill amount to the user. System also displays a place to add coupons. User selects available coupon options from Gourmand Promo Coupons or Restaurant Provided Coupons. System allows user to enter

coupon code to the order. User enters the coupon code and moves on to the next. System applies the coupon to the order and discounted final rate is displayed to the user.

17. Subscribe to Restaurant for Events: User selects to subscribe with the restaurant. System confirms the subscription and sends email notification to the user any time a new event is added by the restaurant.

18. Redeem Rewards: User selects the option of Redeem Rewards at the time of order checkout and payment options. System calculates the discount based on the redeem rewards and applies it to the total and displays the discounted total to the user.

4.3 Restaurant Use-Cases

1. Restaurant Register: A Restaurant Admin selects to register with the app. System displays two type of register options: Basic & Premium. Restaurant selects the registration type and moves on to next step. System displays the registration form. Restaurant fills out the information like restaurant name, address, phone number, email address and password and submits. If the selected type was basic, then the restaurant dashboard is displayed. If type was premium, payment options are provided to the restaurant. Restaurant selects the payment type and pays for the premium package. System verifies the payment and takes the user to the dashboard.

2. Update Menu: Restaurant selects the update menu option from the dashboard. System display the menu in the editable fashion. Restaurant adds or deletes or updates any item from the menu and submits. System displays the updated menu to the restaurant.

3. Add & Update Events: Restaurant selects the Events tab from the dashboard. System displays the events screen to the restaurant with all the upcoming, completed and today's events to the restaurant. Restaurant selects to add a new event. New event screen is displayed by the system. Restaurant enters event name, date, time and location of the event and submits the event. System saves the event and updates the events on the dashboard of the restaurant. Users who are subscribed are also notified about the restaurant via email.

4. Add Restaurant Photos: Restaurant selects to upload pictures. System displays the gallery in the phone. Restaurant selects the total number of photos it wants to upload and submits. System uploads the photos and saves it. Now the new pictures are visible in the restaurant dashboard and also to the restaurant profile which is viewed by the user.

5. View Analytics: Restaurant clicks on the View Analytics functionality on the dashboard. System runs the analytics and displays the analytics page to the restaurant. Restaurant can then select to view analytics in different forms like bar chart, pie chart and line chart. System displays the selected chart to the user.

6. Advertise: Restaurant selects to advertise themselves on the app for certain category of users. System displays the different advertising options to the restaurant. Restaurant selects any one package from the list and moves on to next step. System displays

payment options and total bill for the selected advertising package. User selects the payment type and completes the payment. System verifies the payment and advertises the restaurant based on the selected package. Users get to see the advertised restaurants as the featured restaurants.

7. Update Order: Restaurant accepts an order from the app. Restaurant updates the status of the order as the order is being prepared and in the process of being ready to serve, deliver or ready to take-away. System updates the status of the order in the system.

8. Add Restaurant Coupon: Restaurant select to add a new coupon. System displays the new coupon page. Restaurant enters the coupon fields like coupon name, start date, end date, discount and description and submits the coupon. System saves the coupon and updates the coupon in the system. The restaurant profile will now display this coupon which can be viewed by all the users.

9. Update Restaurant Details: Restaurant selects to update the restaurant information from the dashboard. System displays the edit details page to the restaurant. Restaurant edits and updates the details and saves the updated details. System saves the details and displays in the restaurant dashboard.

4.4 Critic Use-Cases:

1. Register as Critic: Critic selects to register with an app as a critic. System displays the registration form to the critic. Critic enter information like name, contact information, email address, password and references. System saves the form and sends the form to the app admin. Admin verifies the critic. After the verification is done. Critic is notified of being successfully registered with the app.

2. Write Critic Review: Critic selects to write a review from the critic dashboard. System displays the critic review screen. Critic writes the review and submits the review. System saves the review and updates it to the critic reviews section on the restaurant profiles.

4.5 Admin Use-Cases:

1. Verify Critic: System sends a critic registration request to the admin. Admin sees the critic information and contacts the critic over phone and email. Admin verifies the critic by provided references. Admin takes a decision and selects to verify the critic. Notification to critic is sent via email about the verification and verification result.

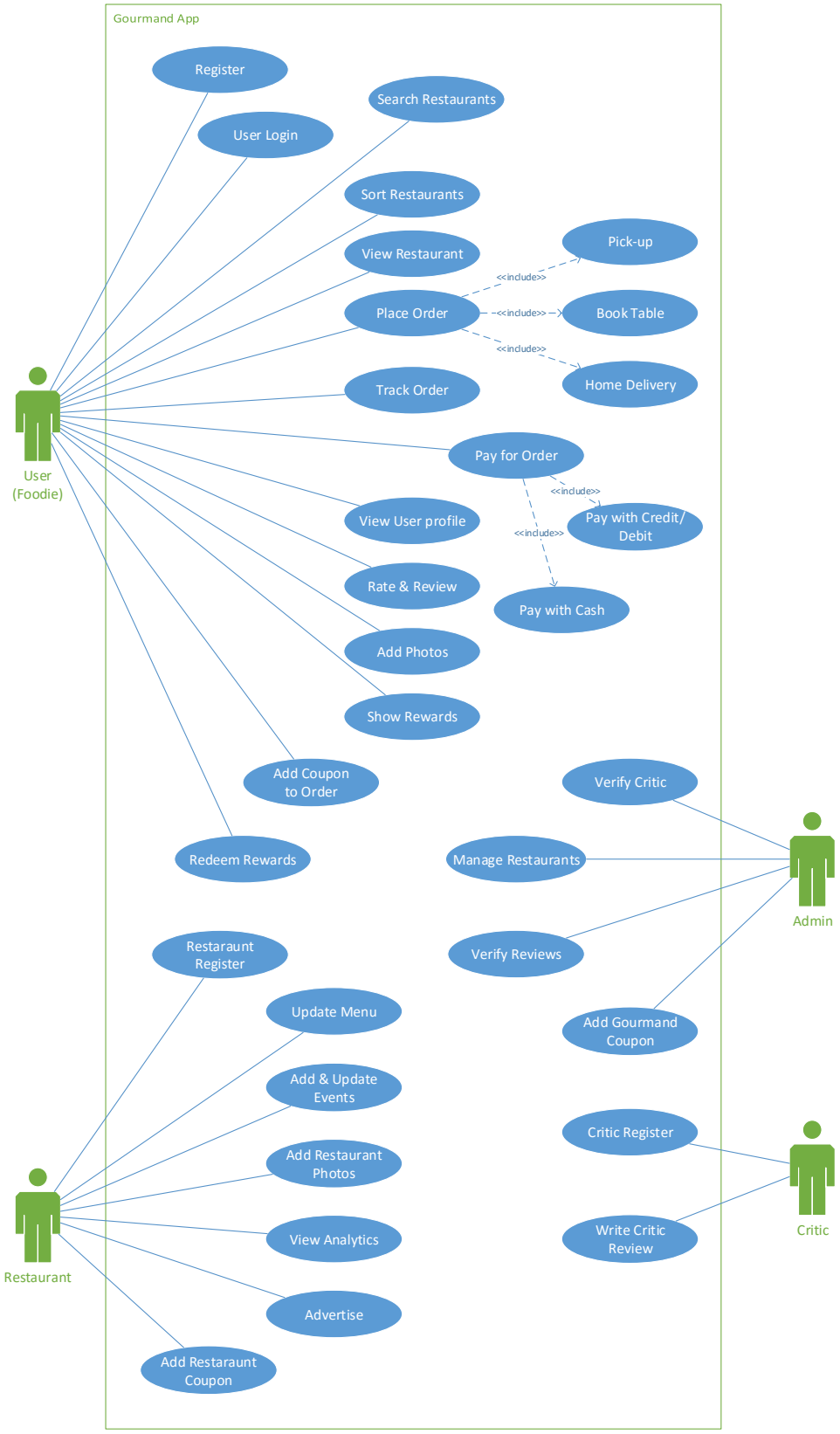
2. Manage Restaurants: Admin selects to add/update/delete a restaurant. System opens the selected page based on the selection of the admin. In case of a new restaurant, a New Restaurant page is opened and admin enters information of the restaurant and submits. System saves the restaurant and updates it to the list of restaurants. In case of

update/delete a restaurant, the system displays Edit Restaurant page to the admin. Admin either edits certain information of the restaurant or deletes the restaurant if the restaurant is closed. System saves and updates the changes to the list of restaurants.

3. Verify Review: System sends a review written by the user to the admin for verification. Admin opens the review and verifies the review. Admin selects to add or flag the review. If the admin selects to add the review, that review is added to the list of reviews and can be seen by the users. If admin selects to flag the review, an email notification is sent to the user indicating inappropriate content in the review and system discards the review from the system.

4. Add Gourmand Coupon: Admin selects to add a gourmand coupon for users. System displays a new coupon page to the admin. Admin enters coupon details in the form and submits. System saves the coupon and is displayed in every user dashboard.

5. Use Case Diagram



6. Use Case Text (Fully Dressed)

6.1 Use Case-1

Use Case Name:	Place Order	
Scenario:	Place a new food order	
Triggering Event:	User opens the mobile app to order food from the catalog of various restaurants.	
Brief Description:	The user selects the order food option on the restaurant profile. The system displays three order food options: Pick-up, Dine-in and Home Delivery. The user selects any one of the options from the three options. System displays the selected page to the user in the app. This use-case is divided into three sub use-cases based on the selection of the order type by the user. The three sub-cases are described below as Pick-up, Book Table and Home Delivery.	
Actors:	User (Primary)	
Related Use Cases:	Search Restaurants, Book Table, Pick-up, Home Delivery	
Stakeholders:	*User*: wants to order food *Restaurant*: wants to prepare and deliver the order/food *Government Tax Agencies*: want to collect tax from every sale (may be multiple agencies, such as national, state or county *Payment Authorization Service*: wants to receive digital authorization requests; wants to accurately account for their payables to the system	
Preconditions:	1. User should be signed-in on the app 2. Home delivery option should be supported in User's area.	
Post conditions:	1. Unique order number must be generated 2. Order must be associated to a valid user 3. Order must be associated and forwarded to selected restaurant 4. Valid discount coupons must be used 5. Valid payment method must be selected	
Flow of Events:	**User (actor)**	**System**
	1. User searches the restaurants through by applying different kinds of filters such as, cuisines, city, ratings etc. 2. User chooses restaurant 3. User views menu and other details 4. User starts order 5. Add item 6. Repeat step 6 until the user wants 7. User ends the order	1.1 Displays search results 3.1 Displays restaurant details 5.1 Create order item. 6.1 Add item to order 7.1 Ends the order

	8. User chooses order type option as Home Delivery 9. User chooses to book a table (if chosen 'dine-in' order type and wish to reserve table) 10. Apply Coupon 11. User selects payment details 12. Do payment	 9.1 Mark and set the delivery type 10.1 Calculate Total 10.2 Display Payment information 12.1 Do order transaction 12.2 Finish Order
Exception Conditions:	1. If the desired restaurant/item in menu isn't available, then User can: 1.1. Choose another restaurant/item 1.2. Choose not to place an order 2. If the discount coupon isn't valid, then the User can: 2.1. Pay the full price 2.2. Choose not to place an order 3. If 'home delivery' service is not present in the User's area, then he/she can: 3.1. Choose either 'take away' or 'dine in' as order type 3.2. Choose not to place an order 4. If User's card authorization fails, then he/she can: 4.1. Choose to pay by 'cash on delivery' (if 'home delivery' is valid) 4.2 Choose not order	

6.2 Use Case-2

Use Case Name:	Book Table
Scenario:	User can reserve a table in a restaurant while pre-ordering the food
Triggering Event:	User selects 'dine in' as order type and opt for reserving a table
Brief Description:	User selects the number of persons to book a table. System displays the table availability to the user. User selects the table and books the table. System displays the confirmation of table being booked and also displays the confirmation number to the user.
Actors:	User (Primary)
Related Use Cases:	Place order, Pay for Order
Stakeholders:	*User*: wants to reserve a table while pre-ordering the food *Restaurant*: A restaurant will get table booked through the app which will increase their business.
Preconditions:	1. User should place an order 2. Restaurant should have registered with the system for booking tables

Post conditions:	Table at the restaurant should be reserved	
Flow of Events:	**User (actor)**	**System**
	1. User places an order 2. User selects 'dine in' as order type 3. User enters the number of persons, date and time for the table booking 4. User selects the table to book.	1.1 Creates order 2.1 Display table details 3.1 System displays the available tables in the time and date selected by the user. 4.1 System displays the table information and book table receipt to the user.
Exception Conditions:	If there are no tables available for booking, then user can: 1.1. Choose to continue to place order as 'dine in' (without table reservation) 1.2. Choose to continue to place order by opting for other order types 1.3 Choose to not proceed with placing the order	

6.2 Use Case-3

Use Case Name:	Rate and review	
Scenario:	User provides rating and reviews for a restaurant	
Triggering Event:	Initiate rating and writing review	
Brief Description:	User selects to Rate & review any restaurant from the restaurant profile of the restaurant. System displays the rate & review screen to the user. User selects the rating out of 10 and writes a review in the review area. User submits the review. System accepts the review and goes to the admin for the verification of review.	
Actors:	User (Primary) Admin (Secondary)	
Related Use Cases:	Verify Review	
Stakeholders:	*User*: Wants to gives the feedback for the restaurant *Restaurant*: Will get the feedback from the user, which will help in marketing of the restaurant. *Gourmand App*: Will get orders from the different users who sees the review and order food from that restaurant.	
Preconditions:	User must be logged in to the system and opened the restaurant for which the user wants to write a review.	
Post conditions:	Ratings and reviews should be submitted into the system for approval of system admin	
Flow of Events:	**User (actor)**	**System**
	1. User opens the Reviews tab from the restaurant profile page	1.1 Displays the list of reviews of the restaurants.

	2. User selects to rate and add a new review for the restaurant 3. User selects a rating from 1 to 10 in the form. 4. User writes a review in the review area inside the review form 5. User submits the review.	2.1 Display the new rate and review form to the user. 3.1 System displays the number selected by the user as a rating. 4.1 System displays the running review currently written by the user. 5.1 System saves the review and checks for any obscene words in the review. If system finds any such words, the review is sent to the admin to verify review and pending review status is shown to the user.
Exception Conditions:	1. User writes some obscene language in the review. Such review will not be immediately posted in the list of reviews for that restaurant. It will be sent to admin for verification. 2. User can rate and review only those restaurants from where he/she has ordered the food.	

6.2 Use Case-4

Use Case Name:	Track order	
Scenario:	User can track the order placed earlier by the user	
Triggering Event:	User selects to track the previously placed order	
Brief Description:	User selects the Track Order feature in the system. System displays the track order screen to the user. User enters the order id in the order id field. System displays the current order status to the user.	
Actors:	User (Primary)	
Related Use Cases:	None	
Stakeholders:	*User*: Wants to track the status of the order placed *Restaurant*: Will let the user know about the current status of the order and thus increase the customer service.	
Preconditions:	User should have placed the order	
Post conditions:	Current status of the order placed should be displayed	
Flow of Events:	**User (actor)**	**System**
	1. User opens the tracking function in the app 2. User enter the order id in the input text.	1.1 Displays the input to enter the order id of the order. 2.1 Displays the current status of the order being tracked
Exception Conditions:	If user provided the invalid order details, then user is prompted to enter the correct/valid details.	

6.2 Use Case-5

Use Case Name:	Search Restaurant	
Scenario:	Search restaurant by applying different filters such as cuisine, city and ratings etc.	
Triggering Event:	User selects to search some restaurants from the app search functionality	
Brief Description:	User selects the filter options from the filter bar and filters the search based on cuisine, rating, price range, distance, and other features like coupons available, order tracking, newly opened, open now, free delivery, express delivery and rapid pick-up. System refines the list and shows a new list of restaurant based on the new filter criteria entered by the user.	
Actors:	User (Primary)	
Related Use Cases:	Sort Restaurant, View Restaurant	
Stakeholders:	*User*: Wants to search various restaurants from the catalog based on some search criteria. *Restaurants*: Will benefit if the user selects to search with their name and increase awareness to the user.	
Preconditions:	User should be signed into the app	
Post conditions:	Should display the search results as per the search criteria	
Flow of Events:	**User (actor)**	**System**
	1. User selects the search functionality from the app 2. User enters the name of the city in the search bar 3. User selects the filter option in the search bar 4. User selects certain filtering options and clicks on the search button again	1.1 Displays the search bar in the app 2.1 Displays the list of restaurants available in the selected city 3.1 Displays the filter options like cuisine, rating, price, range etc 4.1. Displays the filtered list if restaurants matching the filter criteria
Exception Conditions:	1. If the search result is null, then user can choose to search by providing a different search criteria. 2. If the filter selected is not available for any list of restaurants, then the same list as previous will be displayed by the system.	

6.2 Use Case-6

Use Case Name:	Add Event
Scenario:	Restaurant can add events on an occasion
Triggering Event:	Restaurant adds event
Brief Description:	Restaurant selects to add a new event. New event screen is displayed by the system. Restaurant enters event name, date, time and location of the

	event and submits the event. System saves the event and updates the events on the dashboard of the restaurant	
Actors:	Restaurant (Primary)	
Related Use Cases:	None	
Stakeholders:	*Restaurant*: Wants to add event provided in the app *User*: Will get notified for the event if the user is already subscribed with the restaurant.	
Preconditions:	Restaurant should be registered with the app/system and have a dashboard	
Post conditions:	System should display the event on its page & profile.	
Flow of Events:	**User (actor)**	**System**
	1. Restaurant sign into the app 2. Restaurant selects to add a new event from the dashboard 3. Restaurant enters details like event name, date and time for the event 4. Restaurant submits the event to the system	1.1 Displays the restaurant's dashboard 2.1 Displays the add new event form to the restaurant. 3.1 Displays the filled form by the restaurant 4.1 Saves the event and displays the event on the restaurant dashboard as well as the restaurant profile which is viewed by the users. Users who are subscribed with the restaurants also get notified about the event.
Exception Conditions:	1. If restaurant's account does not exist, then restaurant can: 1.1. Choose to register and create an account 2. If restaurant forgets username, then it can: 2.1. Choose to request username to be sent on restaurant's registered email-id 3. If restaurant forgets password, then it can: 3.1. Choose to reset the password	

6.2 Use Case-7

Use Case Name:	User Login
Scenario:	Sign-in the user into the app
Triggering Event:	User enters the username and password
Brief Description:	User enters username (email address) and password and clicks on Login button. The system verifies the username and password and if correct, logs the user into the system and displays the dashboard to the user else it shows invalid username/password.
Actors:	User (Primary)

Related Use Cases:	None	
Stakeholders:	*User*: wants to access and manage account; wants to order food	
Preconditions:	User should have an active account	
Post conditions:	Should display user's dashboard after successfully logging into the system.	
Flow of Events:	**User (actor)**	**System**
	1. User enters the username and password in the input fields on the login screen 2. User selects the login option from the login page	1.1 Displays the username and displays the password as asterisks 2.1 System checks the username and password and authenticates the user. The dashboard of the user is displayed to the user
Exception Conditions:	1. If user's account does not exist, then user can: 1.1. Choose to create an account 2. If user forgets username, then user can: 2.1. Choose to request username to be sent on user's registered email-id 3. If user forgets password, then user can: 3.1. Choose to reset the password	

6.2 Use Case-8

Use Case Name:	Add Restaurant Coupon	
Scenario:	Restaurant can add their coupons in the app	
Triggering Event:	Restaurant adds coupon	
Brief Description:	Restaurant select to add a new coupon. System displays the new coupon page. Restaurant enters the coupon fields like coupon name, start date, end date, discount and description and submits the coupon. System saves the coupon and updates the coupon in the system. The restaurant profile will now display this coupon which can be viewed by all the users.	
Actors:	Restaurant (Primary)	
Related Use Cases:	None	
Stakeholders:	*Restaurant*: Wants to add their coupon in the app *User*: Can use the coupon while ordering food for that restaurant to get some discount.	
Preconditions:	Restaurant should be registered with the app/system and have a dashboard	
Post conditions:	System should display the coupon on its page & profile.	
Flow of Events:	**User (actor)**	**System**
	1. Restaurant selects to add a new coupon to their page	1.1 Displays the new coupon page

	2. Restaurant enters the details like coupon name, start date, end date, discount and description of the coupon. 3. Restaurant submits the coupon to the system.	2.1 Displays the filled new coupon form 3.1 Saves the coupon and updates it to the list of coupons available for the restaurant. Restaurant profile is also updated where the users can view the newly added coupon
Exception Conditions:	1. Restaurant is not a registered restaurant. In such case, the restaurant cannot add coupons to their page	

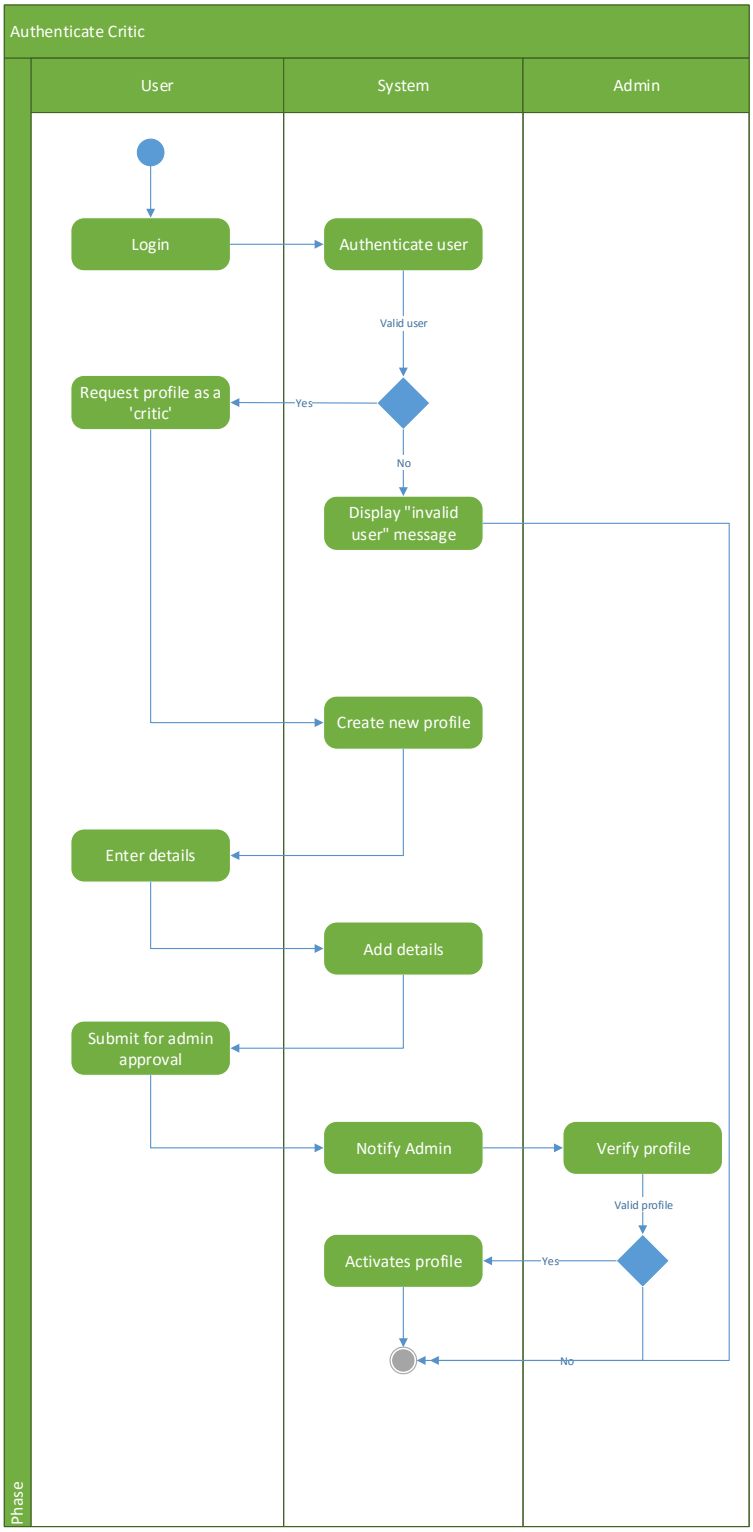
6.2 Use Case-9

Use Case Name:	Manage Restaurants	
Scenario:	Gourmand admin can manage the restaurant in the app	
Triggering Event:	Gourmand admin initiates the restaurant management	
Brief Description:	Admin selects to add/update/delete a restaurant. System opens the selected page based on the selection of the admin. In case of a new restaurant, a New Restaurant page is opened and admin enters information of the restaurant and submits. System saves the restaurant and updates it to the list of restaurants.	
Actors:	Admin (Primary)	
Related Use Cases:	None	
Stakeholders:	*Admin*: wants to manage restaurants in the app	
Preconditions:	1. For adding restaurant, it should not be present in the system 2. For removing restaurant, it should be present in the system 3. For managing restaurant, it should be registered and have a valid dashboard	
Post conditions:	Should update the app with the latest information	
Flow of Events:	**User (actor)**	**System**
	1. Admin opens the app 2. Admin manages the restaurant inventory by performing all or either of the following: 1. Admin adds a new restaurant 2. Admin removes an existing restaurant	1.1 Displays the app content 2.1 Updates the app with latest information

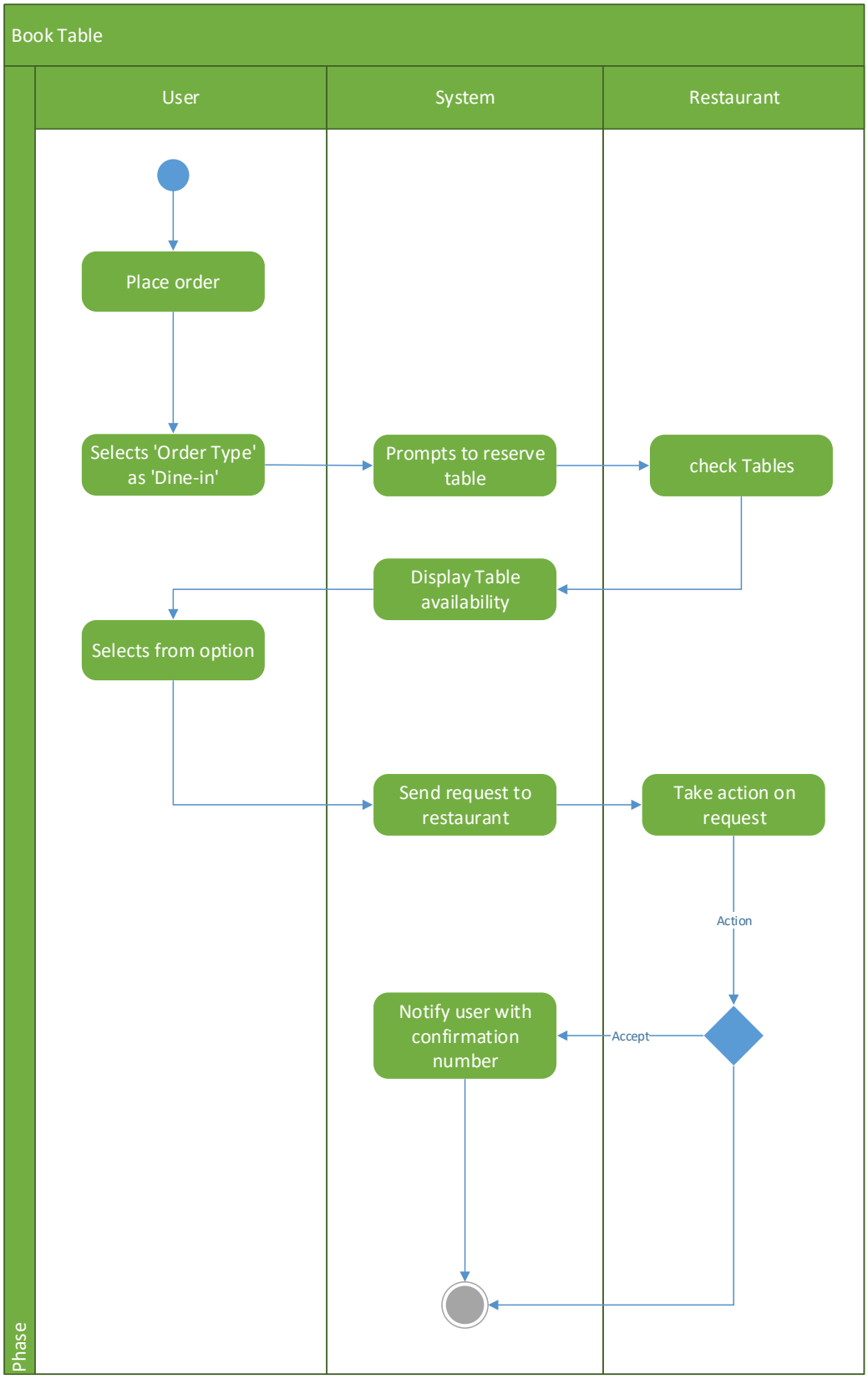
	3. Admin modifies the restaurant dashboard	
Exception Conditions:	1. While adding restaurant, if it is already present in the system then admin can choose to update its details. 2. If a restaurant is not having a valid dashboard, then admin can provide/tag a valid dashboard to the restaurant.	

7. Activity Diagrams

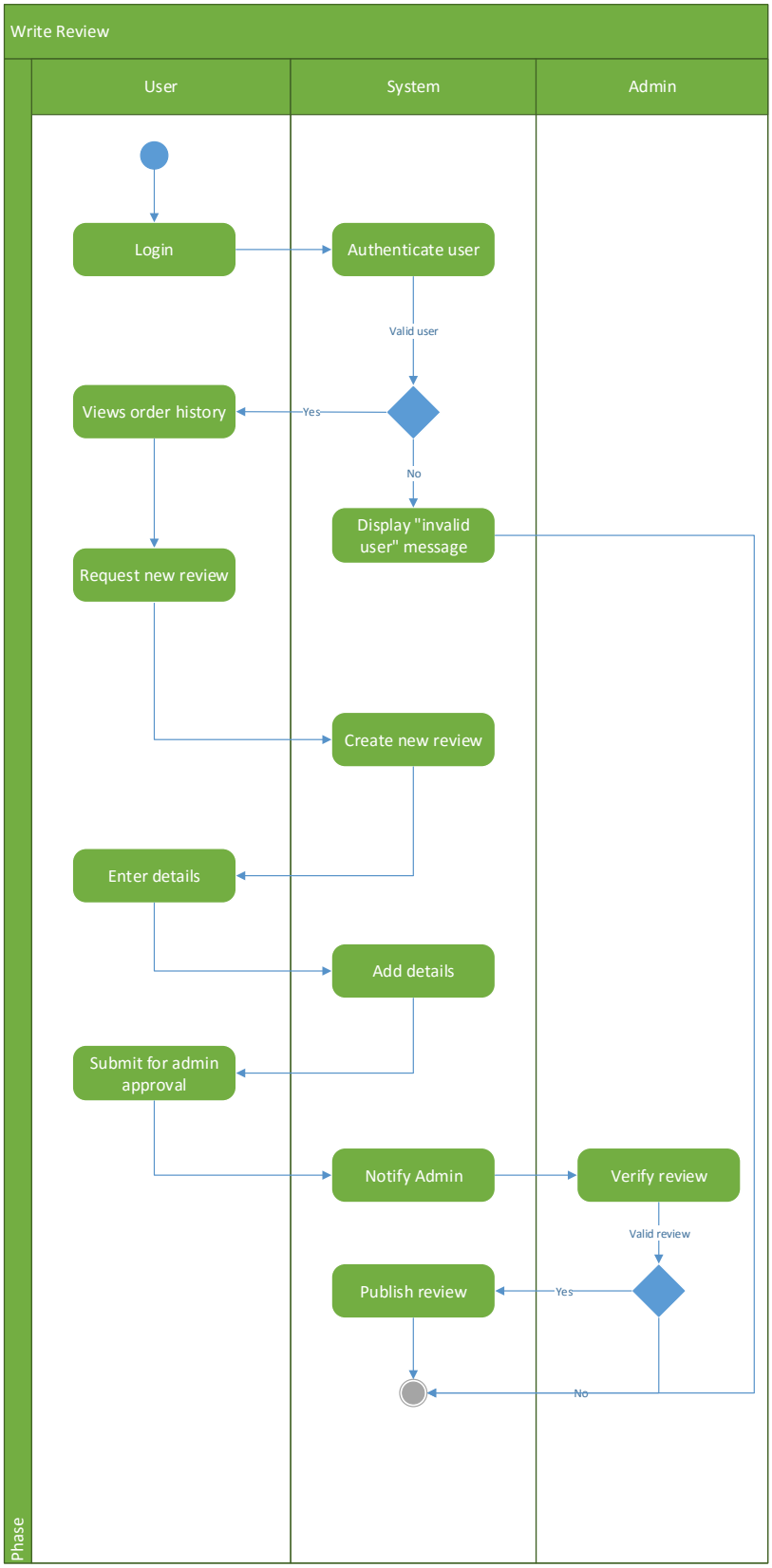
1. Authenticate Critic



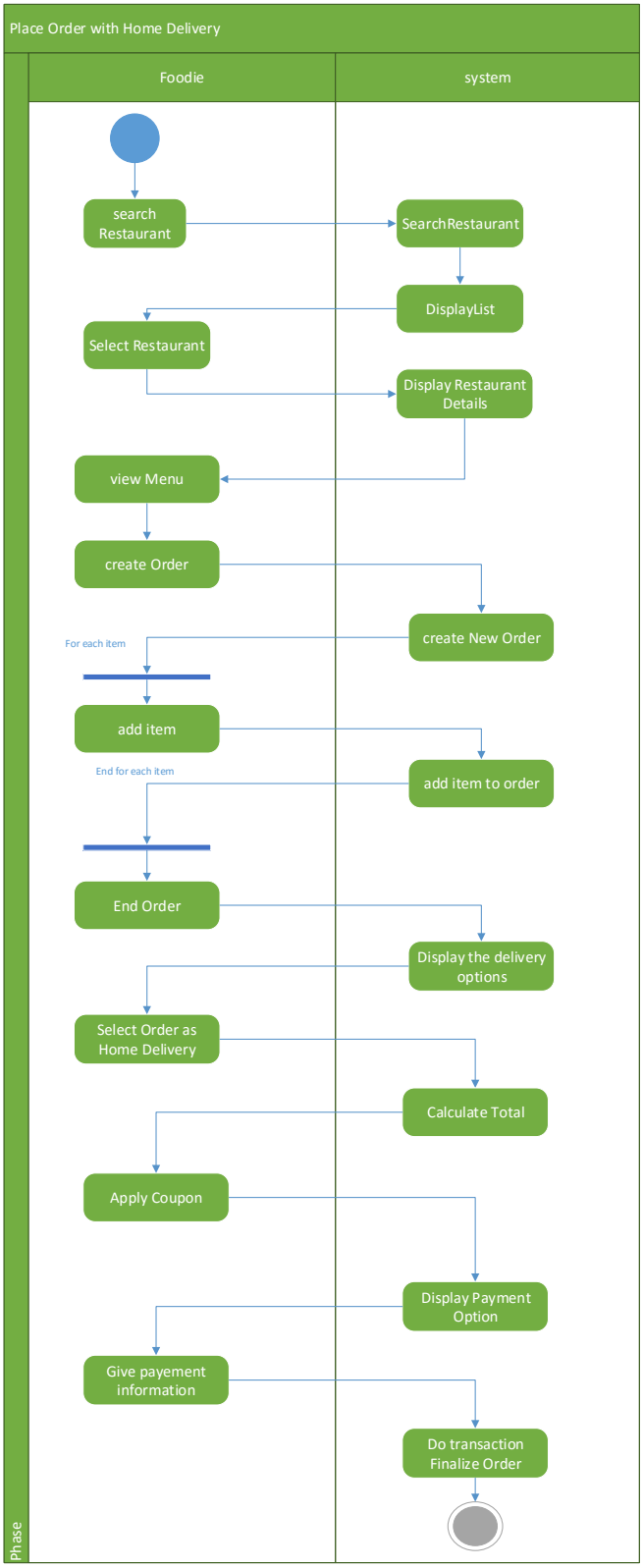
2. Book Table


Book Table
User
System
Restaurant
Place order
Selects 'Order Type' as 'Dine-in'
Prompts to reserve table
check Tables
Display Table availability
Selects from option
Send request to restaurant
Take action on request
Action
Notify user with confirmation number
Accept
Phase

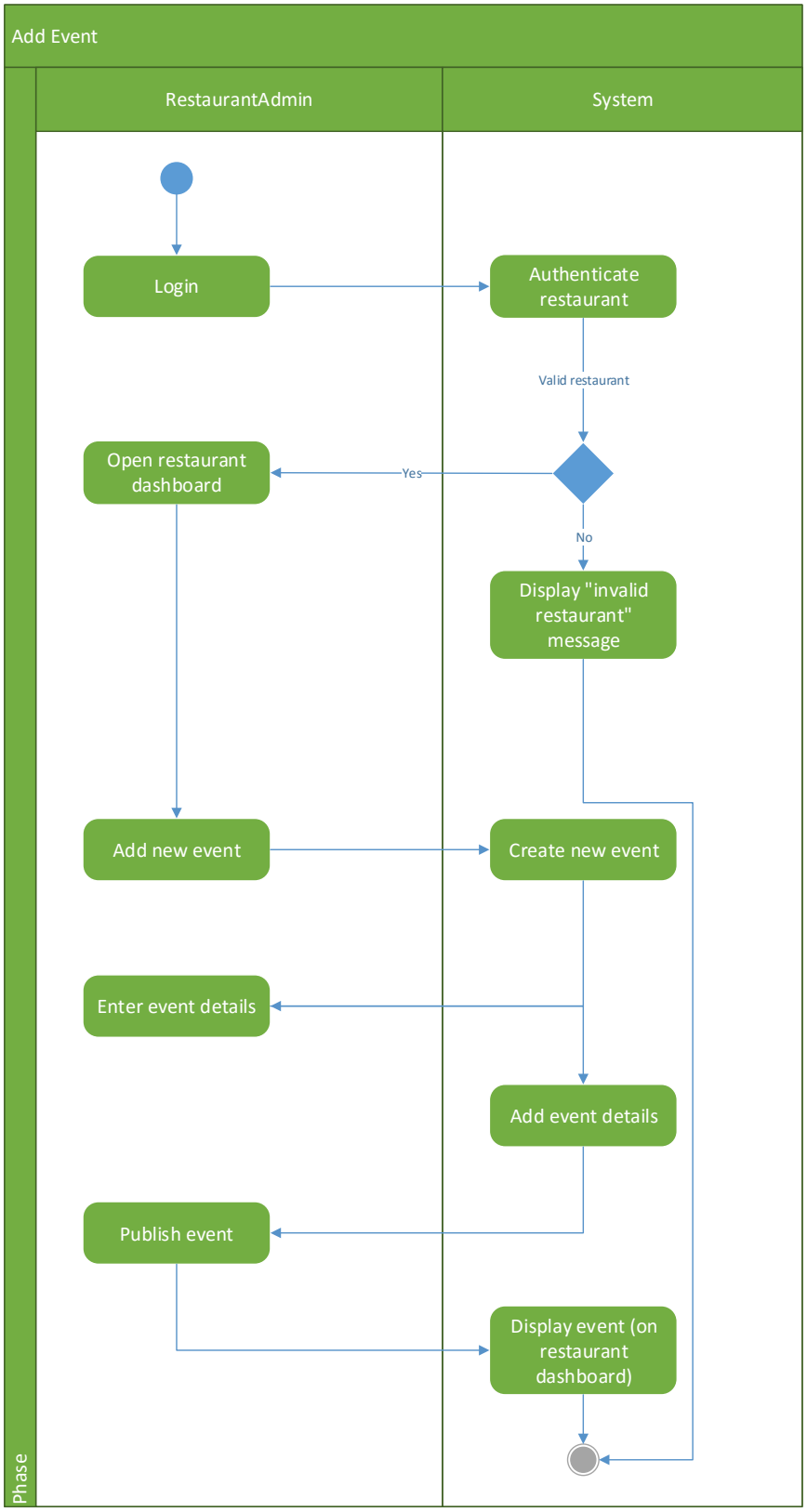
3. Write Review



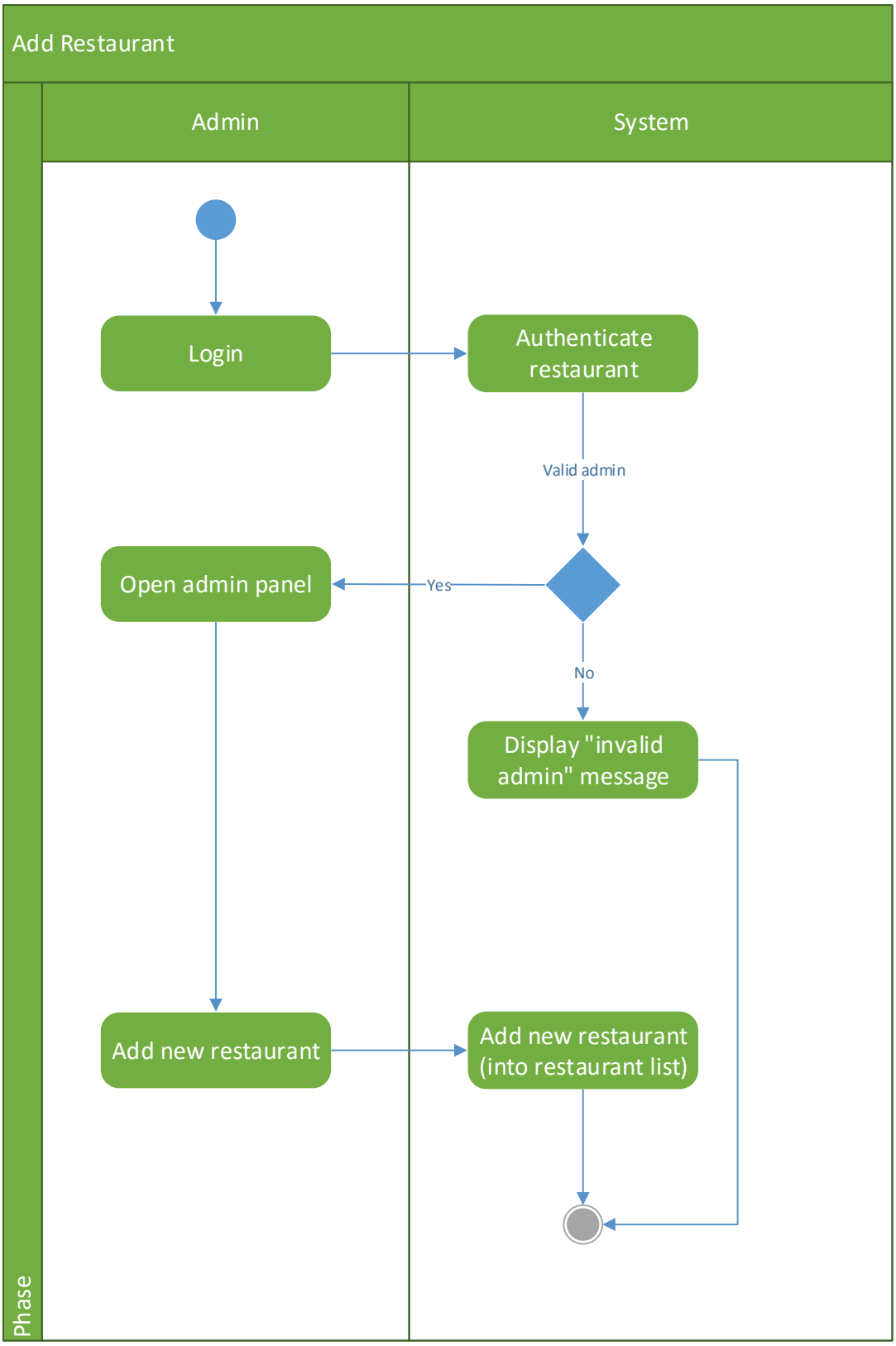
4. Place Order



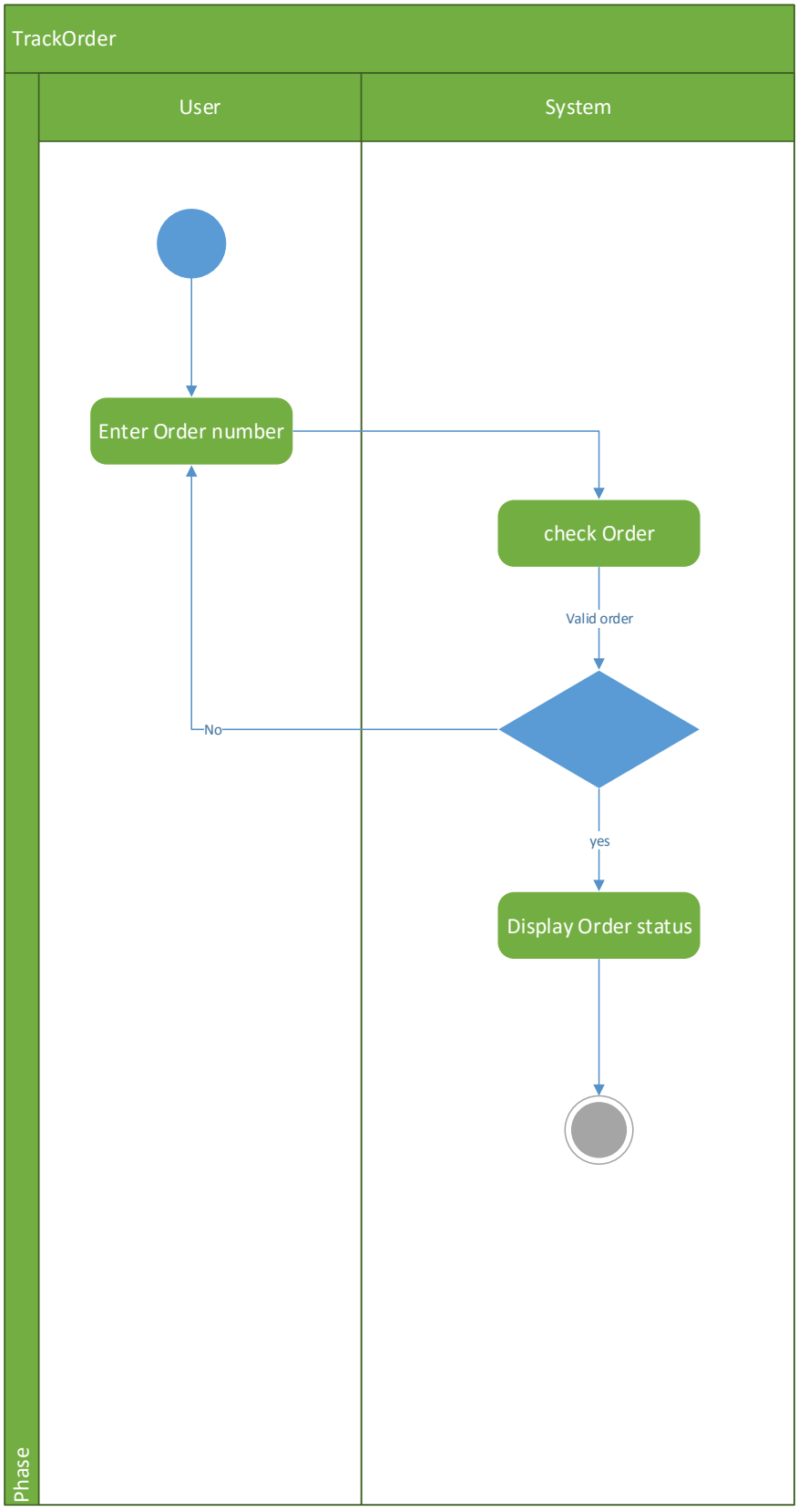
5. Add Event


Add Event
RestaurantAdmin
System
Login
Authenticate restaurant
Valid restaurant
Open restaurant dashboard
Yes
No
Display "invalid restaurant" message
Add new event
Create new event
Enter event details
Add event details
Publish event
Display event (on restaurant dashboard)
Phase

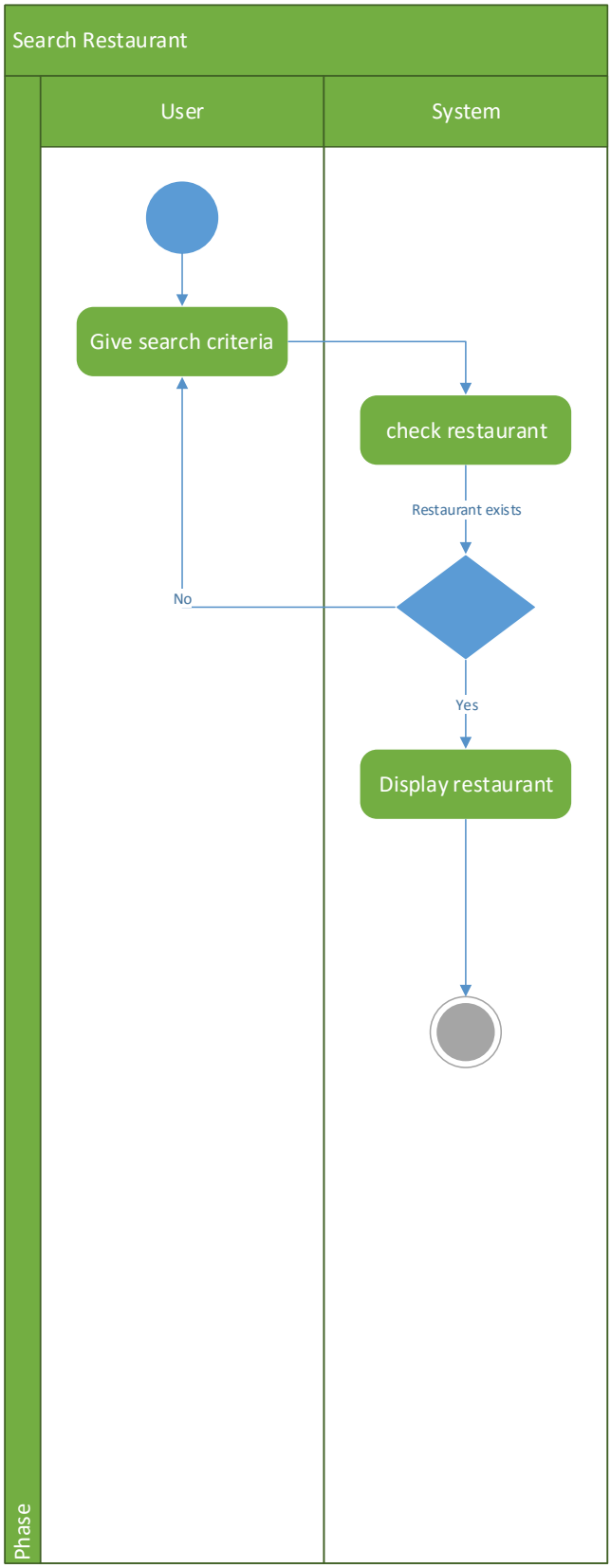
6. Add Restaurant



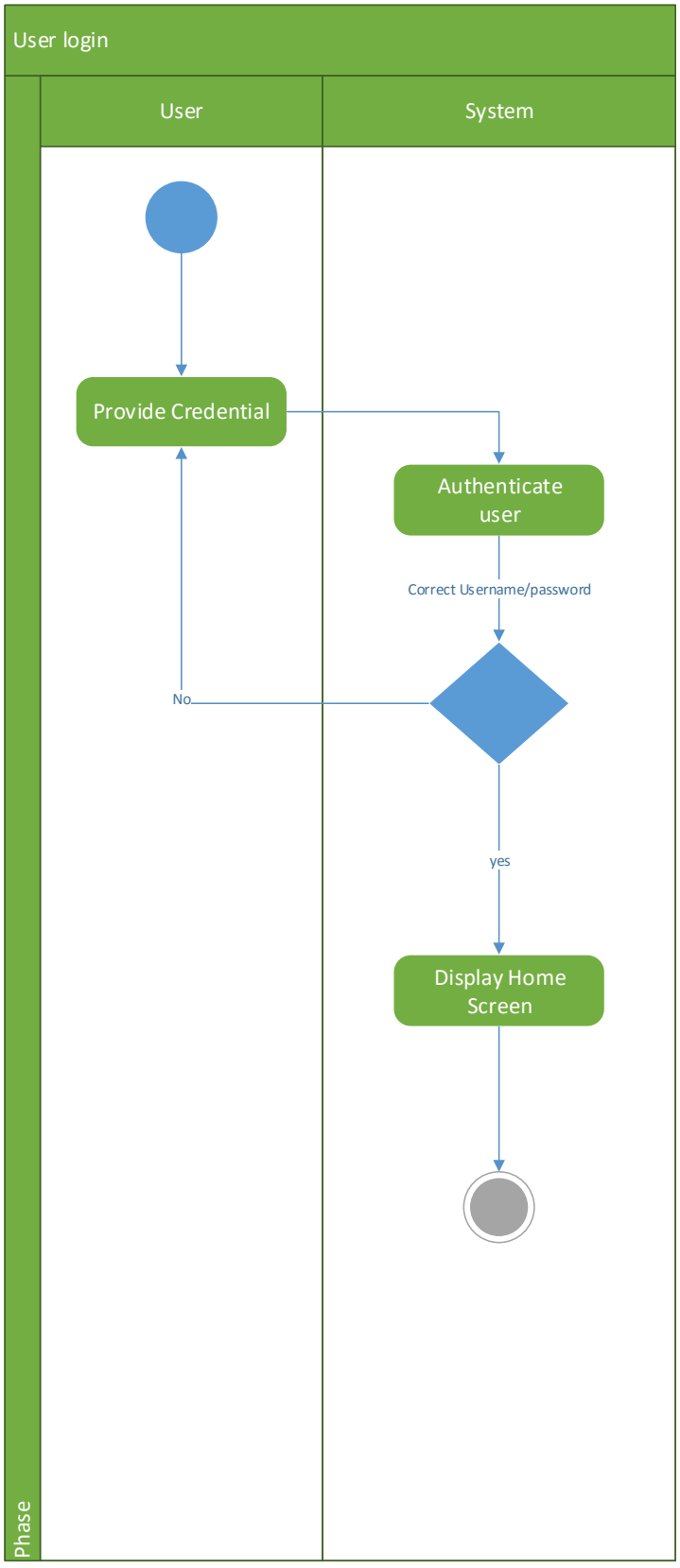
7. Track Order



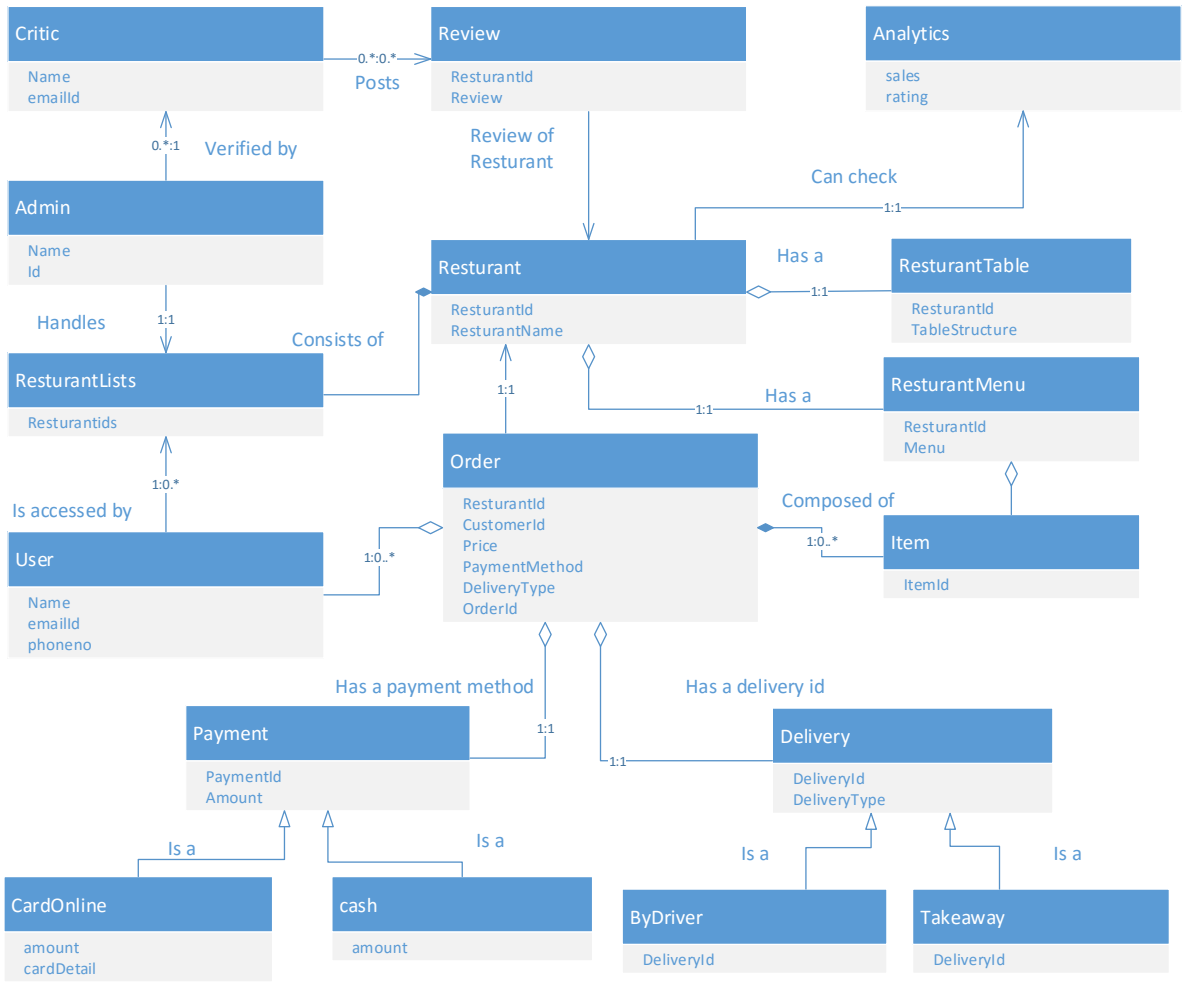
8. Search Restaurant



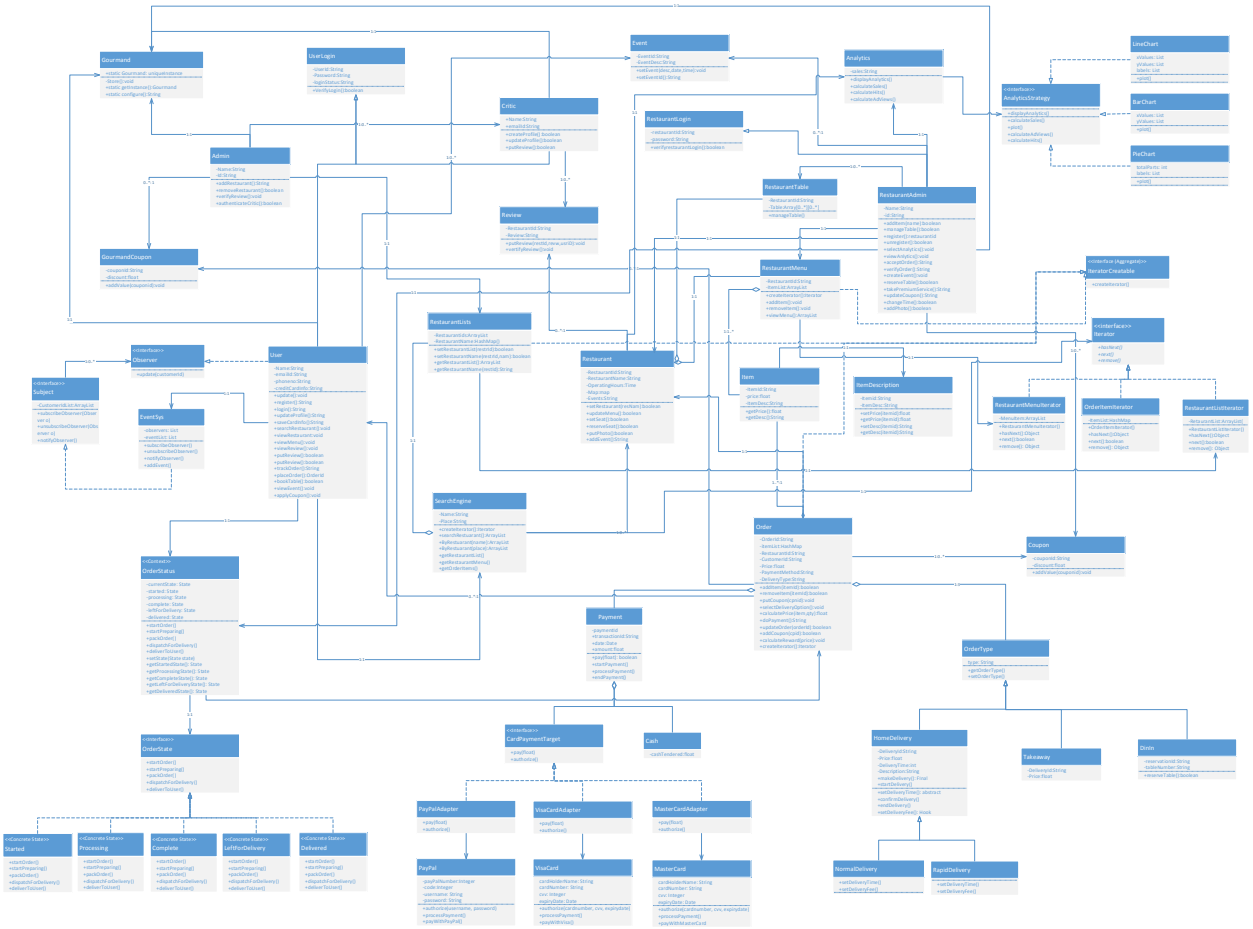
9. Login


User login
User
System
Provide Credential
Authenticate user
Correct Username/password
No
yes
Display Home Screen
Phase

8. Domain Model

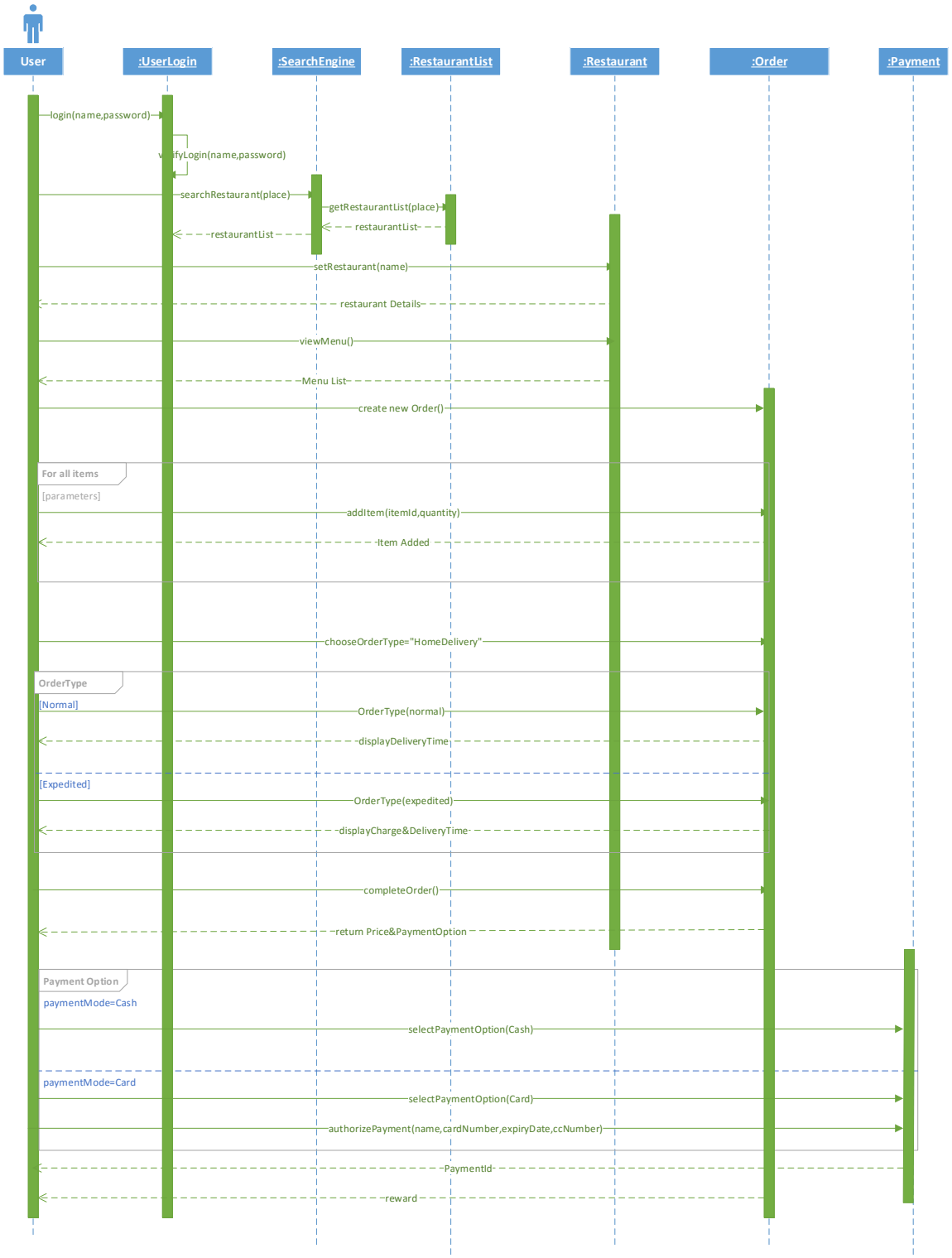


9. Design Model

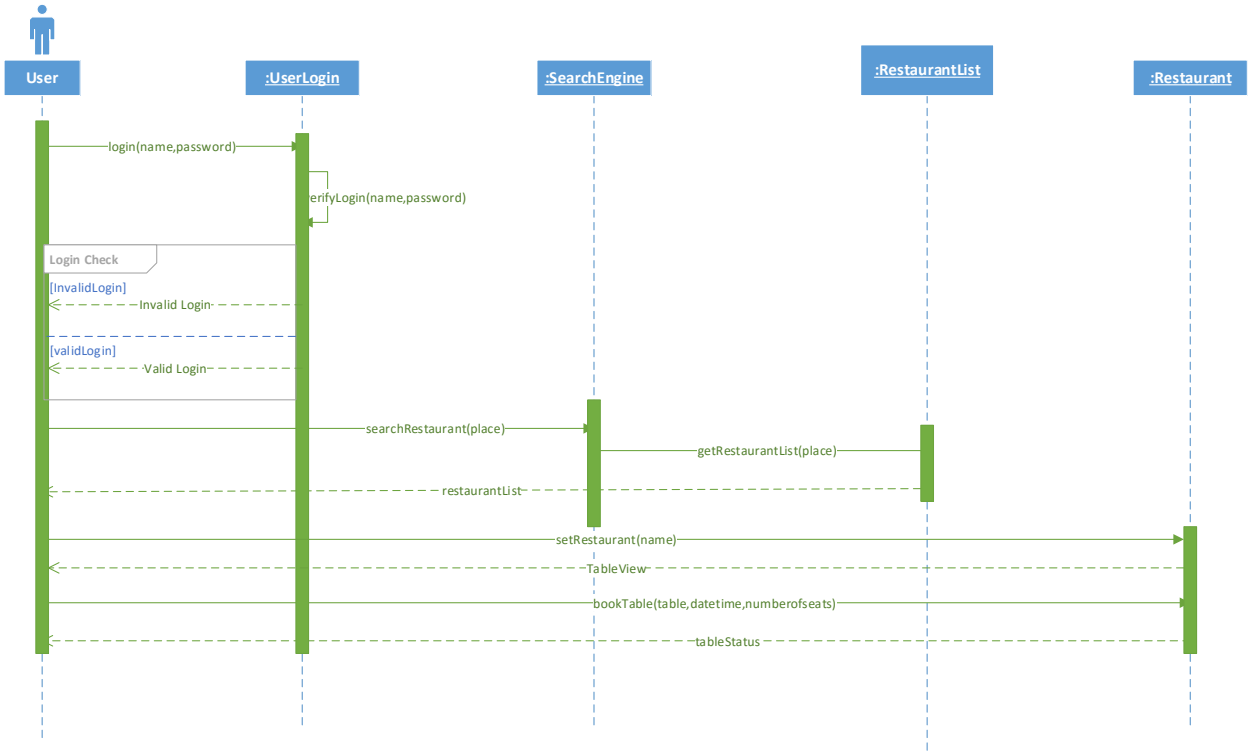


10. Sequence Diagrams

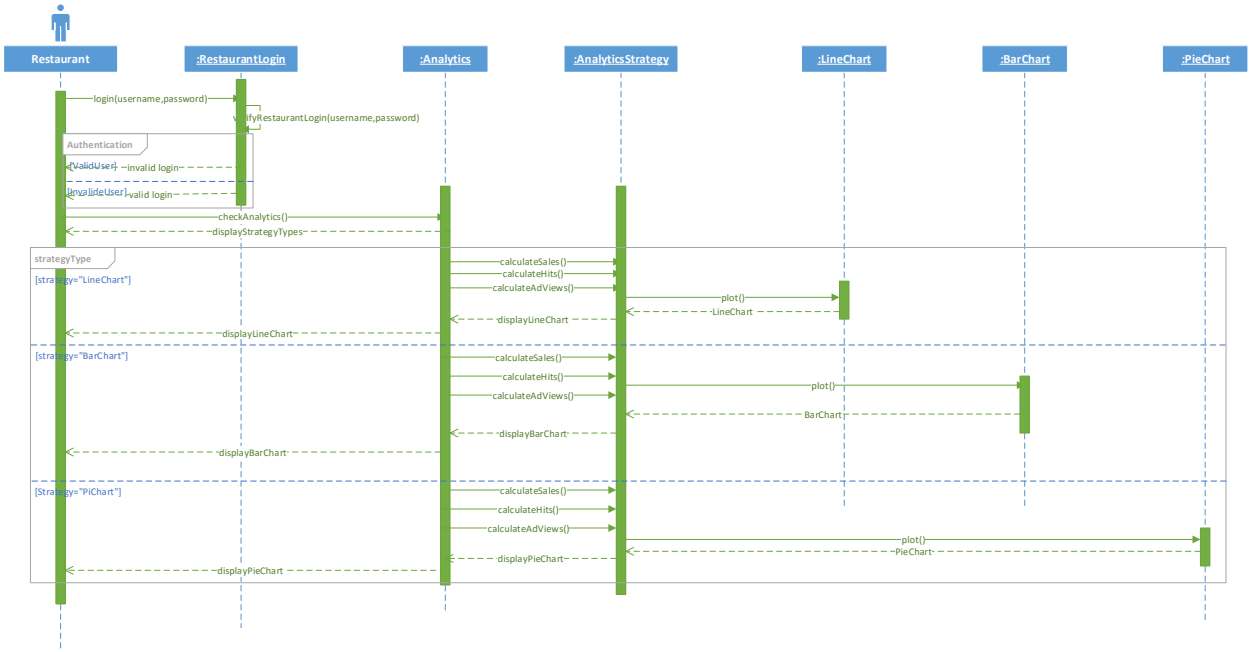
10.1 Place an Order (with Order Type Home Delivery)



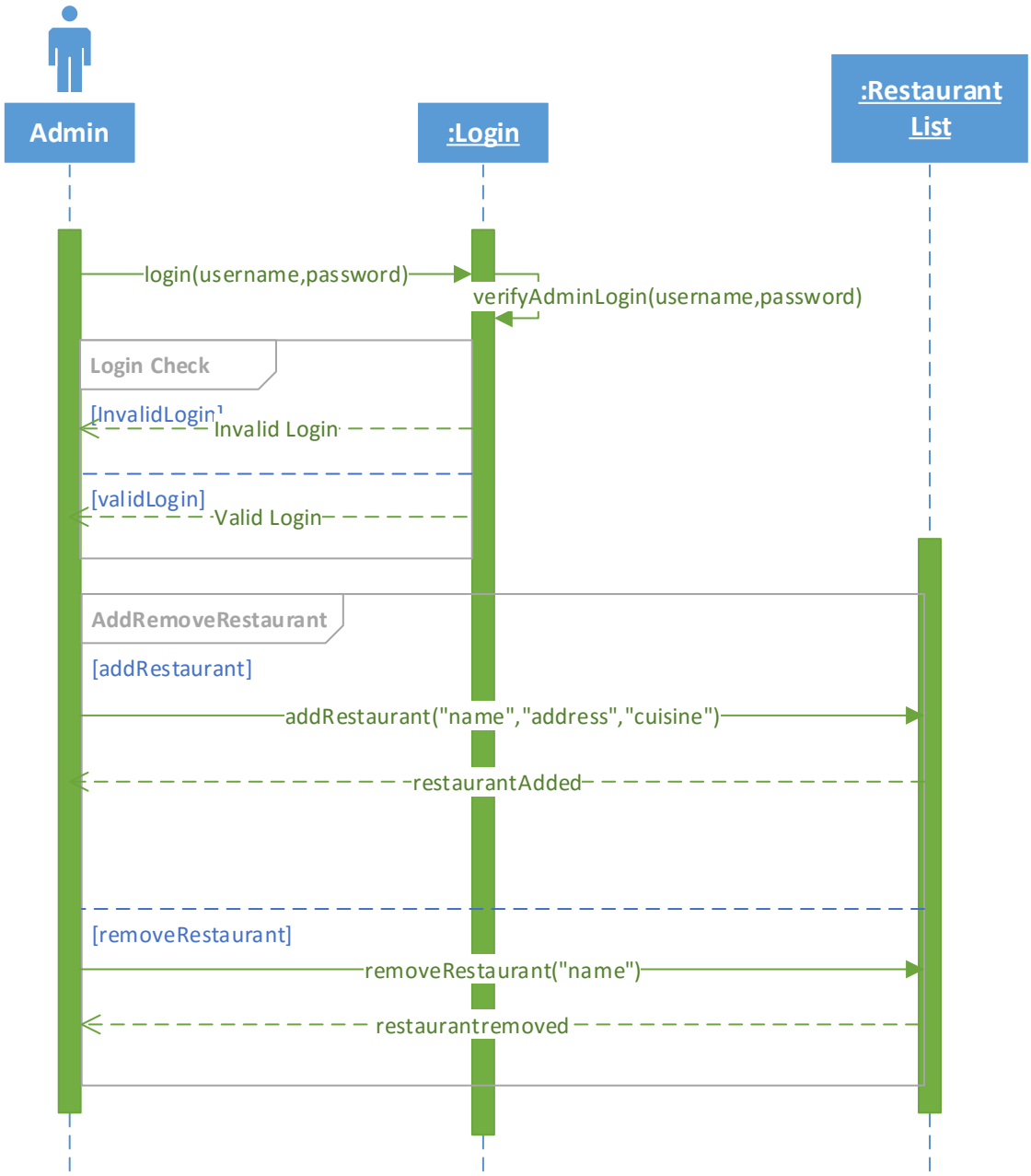
10.2 Book Table



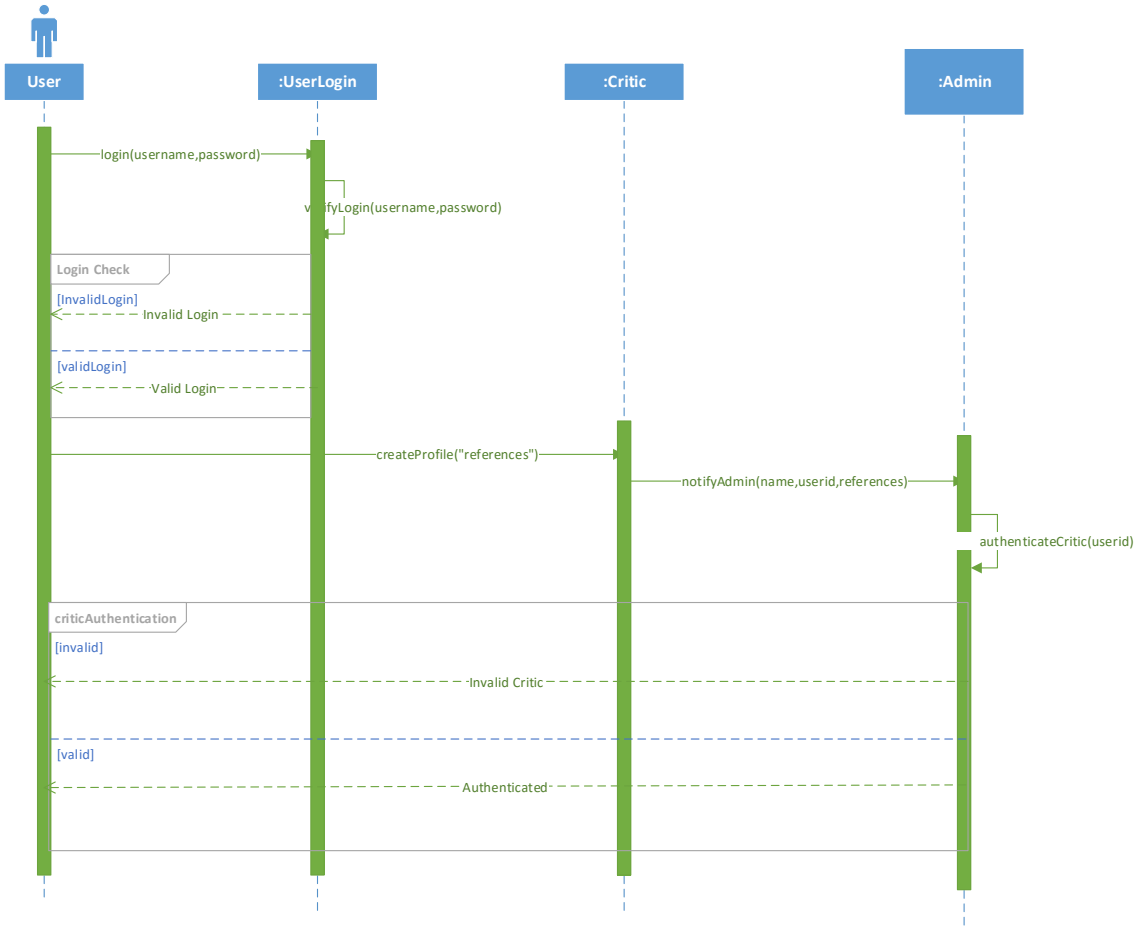
10.3 View Analytics



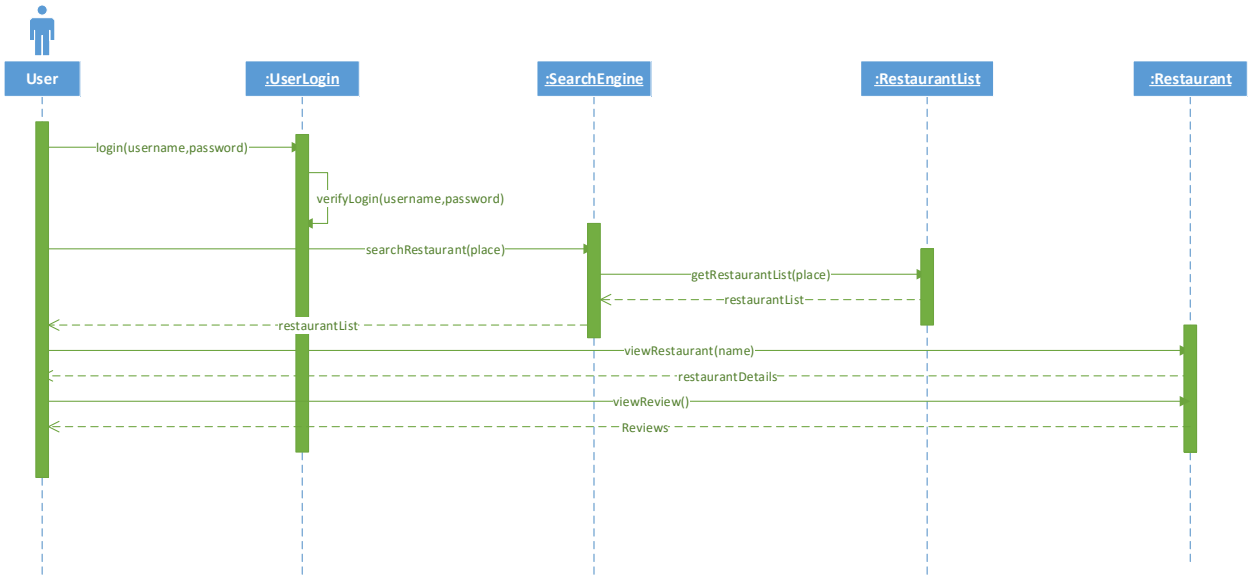
10.4 Manage Restaurants



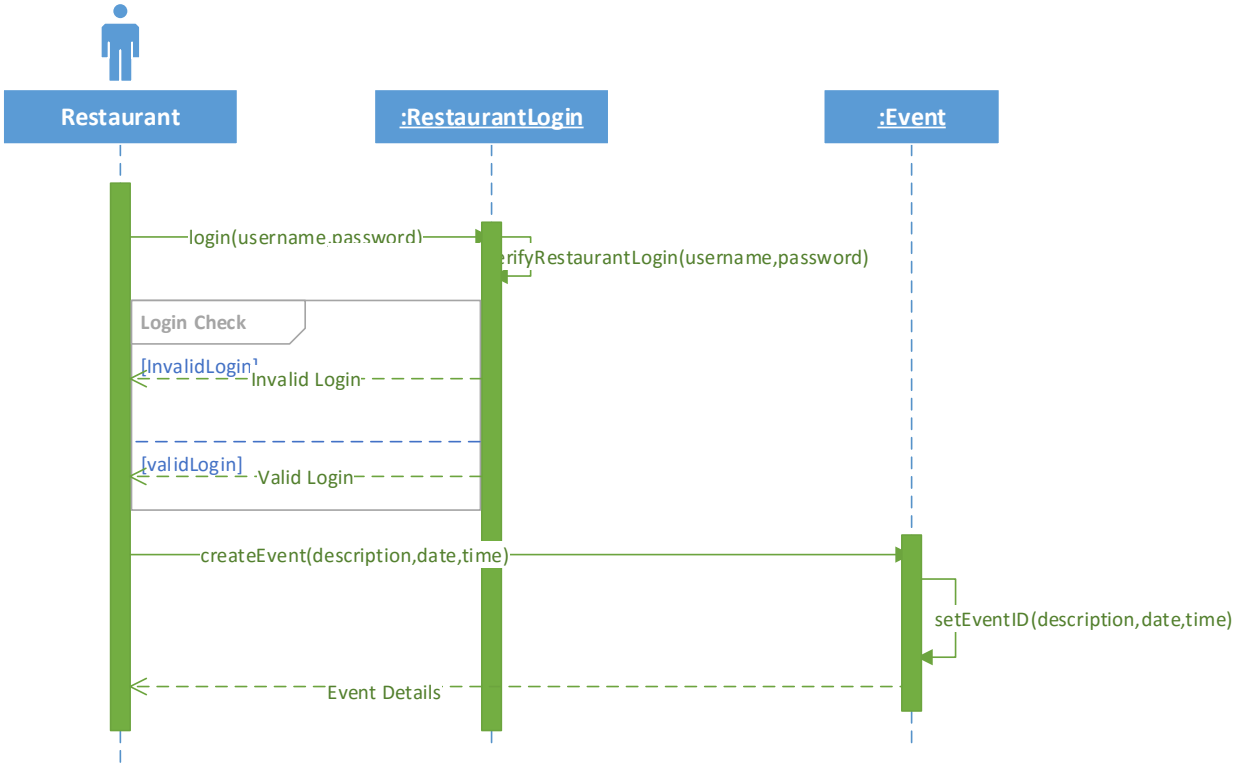
10.5 Authenticate Critic



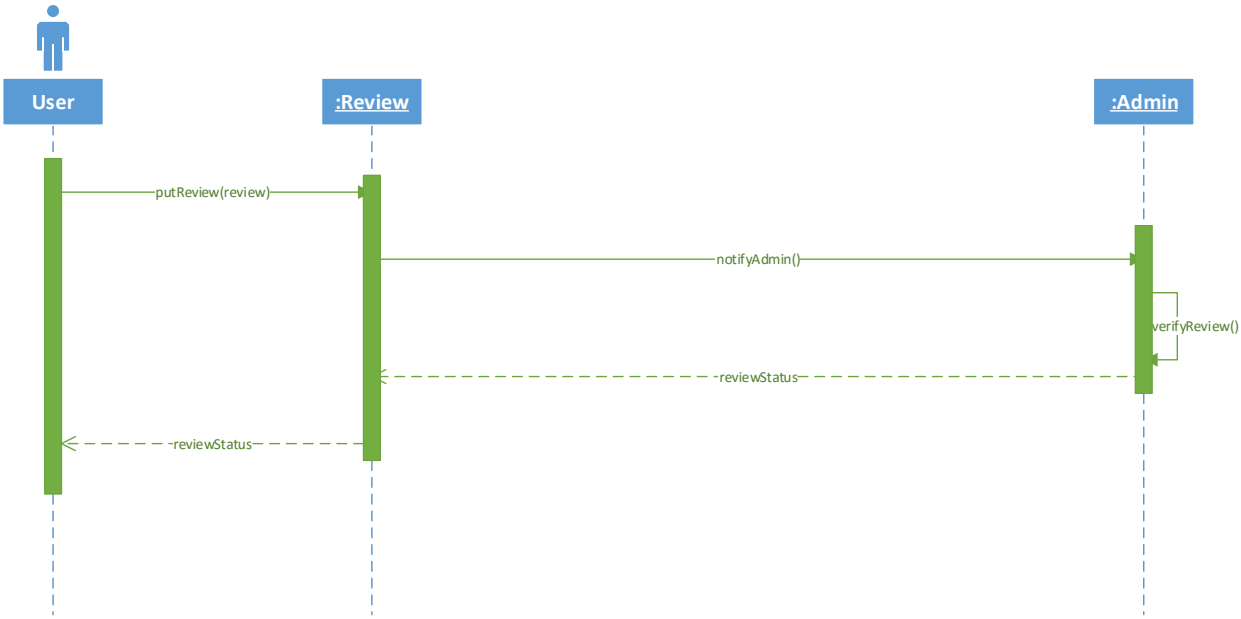
10.6 Read Review



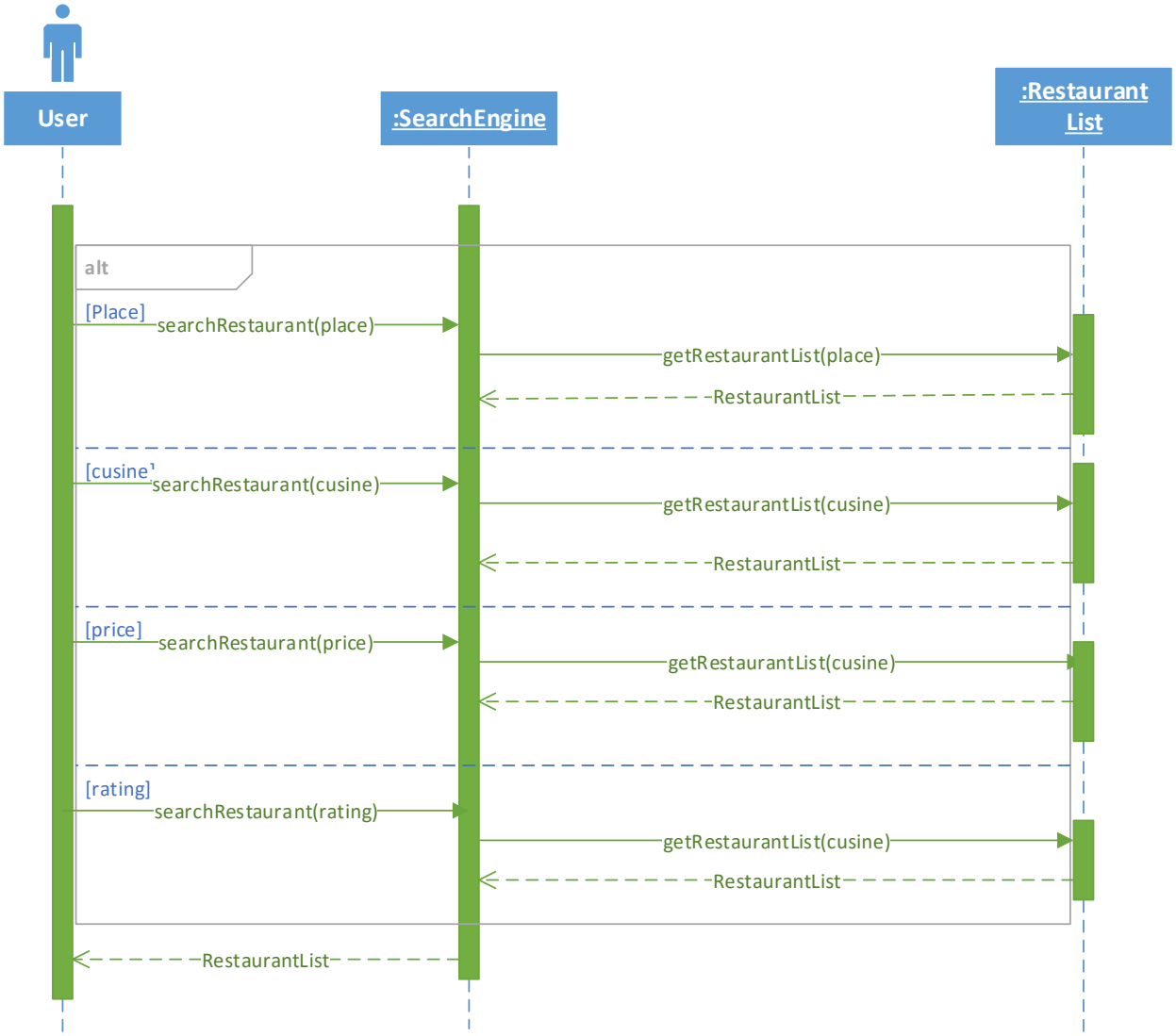
10.7 Add Event



10.8 Write Review



10.9 Search Restaurant



11. Design Patterns Documentation

11.1 The Template Design Pattern

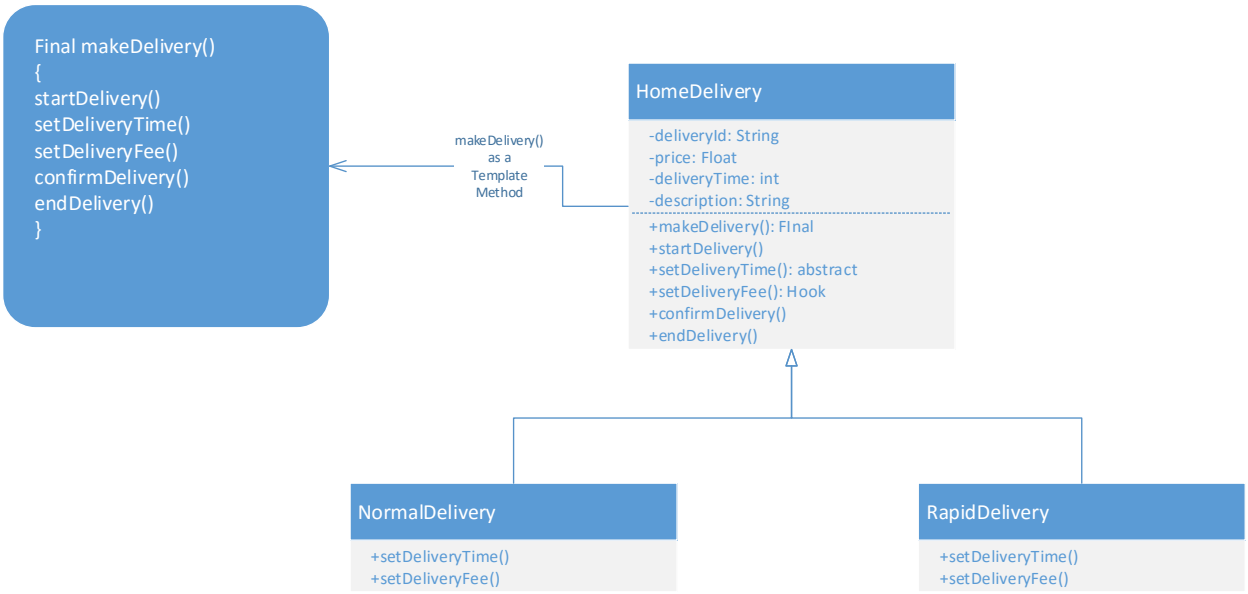
Object Oriented Goal: Code Re-use. Designing to interface and not to implementation.

Definition: The Template Method Pattern defines the skeleton of an algorithm in a method, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

Implemented on classes:

1) HomeDelivery
   a) NormalDelivery
   b) RapidDelivery

Design Snapshot:



Design Explanation:

We want our subclasses to implement methods in a certain fashion just like a steps of an algorithm. Moreover, we want to enforce that some methods be implemented by the subclasses itself as a subclass specific implementation. We need a template so that we can make a steps of methods which must be followed by the subclasses

In the delivery options, we have two different methods of delivery but the steps of processing both the types is same. So we define a template method which indicates steps of the algorithm which needs to be implemented by the subclasses.

- The HomeDelivery class has a makeDelivery() method which acts as a template for the subclasses (NormalDelivery and RapidDelivery). The makeDelivery() is declared as final so that the subclasses cannot change the method. Subclass needs to implement the setDeliveryTime() method and the setDeliveryFee() method because every type of HomeDelivery has different delivery time and fees for the delivery.
- So, the subclass will use the makeDelivery() method. All the other methods are already implemented in the abstract class HomeDelivery. Only setDeliveryTime() and setDeliveryFee() needs to be implemented in the subclasses.
- There is setDeliveryFee() hook method which is implemented in the abstract class. The hook method may or may not be implemented by the subclasses. It depends on the subclasses if they want to implement it or not. So this method is kept as void. Subclasses which implement this method will add behavior to this method.

11.2 The Observer Design Pattern

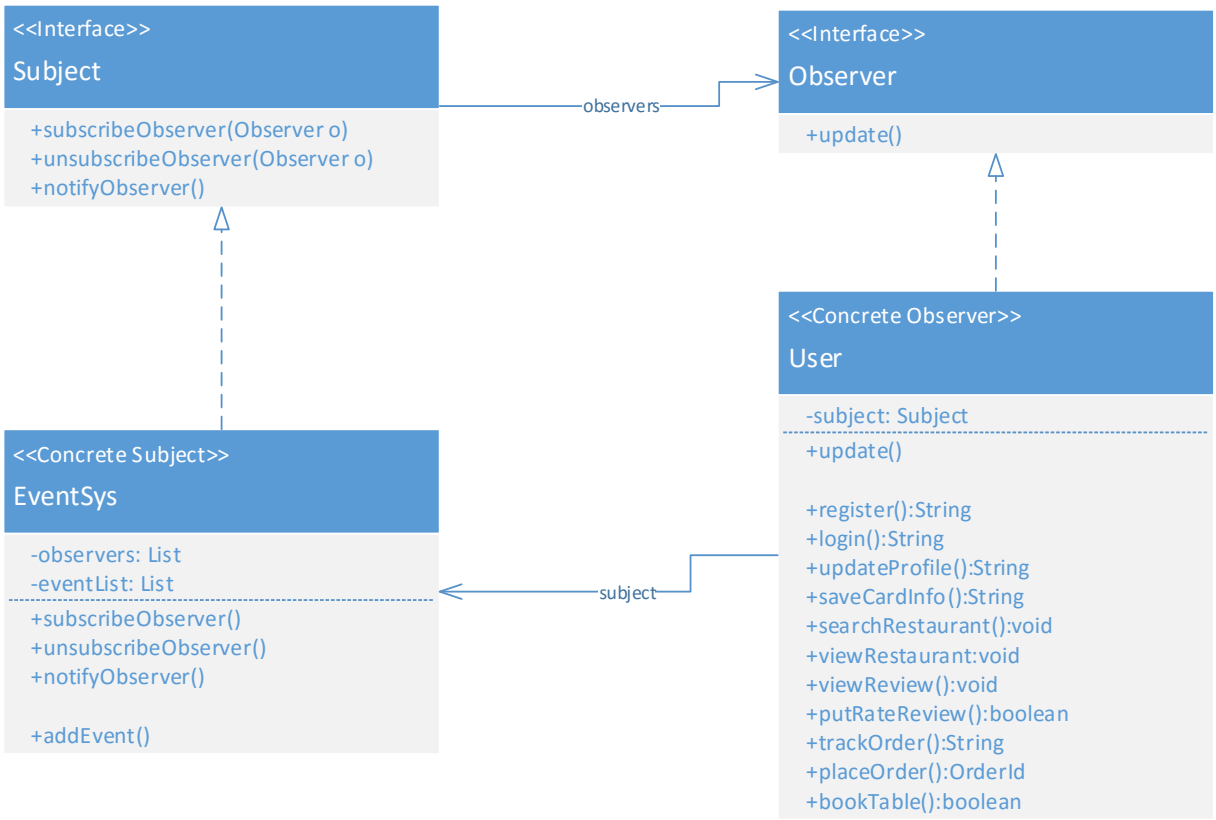
Object Oriented Goal: Favoring Composition over Inheritance, Programming to interface and not implementation, maintaining loosely coupled designs between the objects that interact.

Definition: The Observer Pattern defines a One-to-Many dependency between objects so that when one object changes state, all of its dependents are notified and updated automatically.

Implemented on classes:

1) Subject (Interface)
   a) EventSys (Concrete Subject)
2) Observer (Interface)
   a) User (Concrete Observer)

Design Snapshot:



Design Explanation:

Whenever a restaurant creates a new event, all the users subscribed with the restaurant must be notified of the events by email or text message. This requirement can be designed using the Observer Design Pattern. Following are the key points of the Observer pattern implemented in our design model:

- The Subject interface acts as a supertype of the concrete subject classes. Users use this interface to subscribe as observers and also to remove themselves from being observers.
- The EventSys class is the concrete subject which implements the Subject interface. Thus, this class has subscribe and unsubscribe methods which are used by observers to subscribe with the particular event notifications. This class also has a notifyObserver() method which is used to notify the Users if any event is added by the restaurant. In addition to the methods implemented from the Subject interface, this class also has its own methods like addEvent(), getEvent(), setEvent() and others which are specific to the class.
- The Observer interface is an interface which needs to be implemented by the Users who want to receive notifications from any event. It has one method update(), which has to be implemented by all the concrete observers, which in our case are the Users.
- The User class implements the Observer interface and thus becomes a concrete observer. The User class needs to implement the update() method. The update method will be called whenever any new event is available to send the notification to the User.
- Every User object subscribes with the concrete subject (EventSys) by calling the subscribe method through the Subject interface. The User can also call unsubscribe if the User no more wants any notifications from the restaurant.

11.3 The Strategy Design Pattern

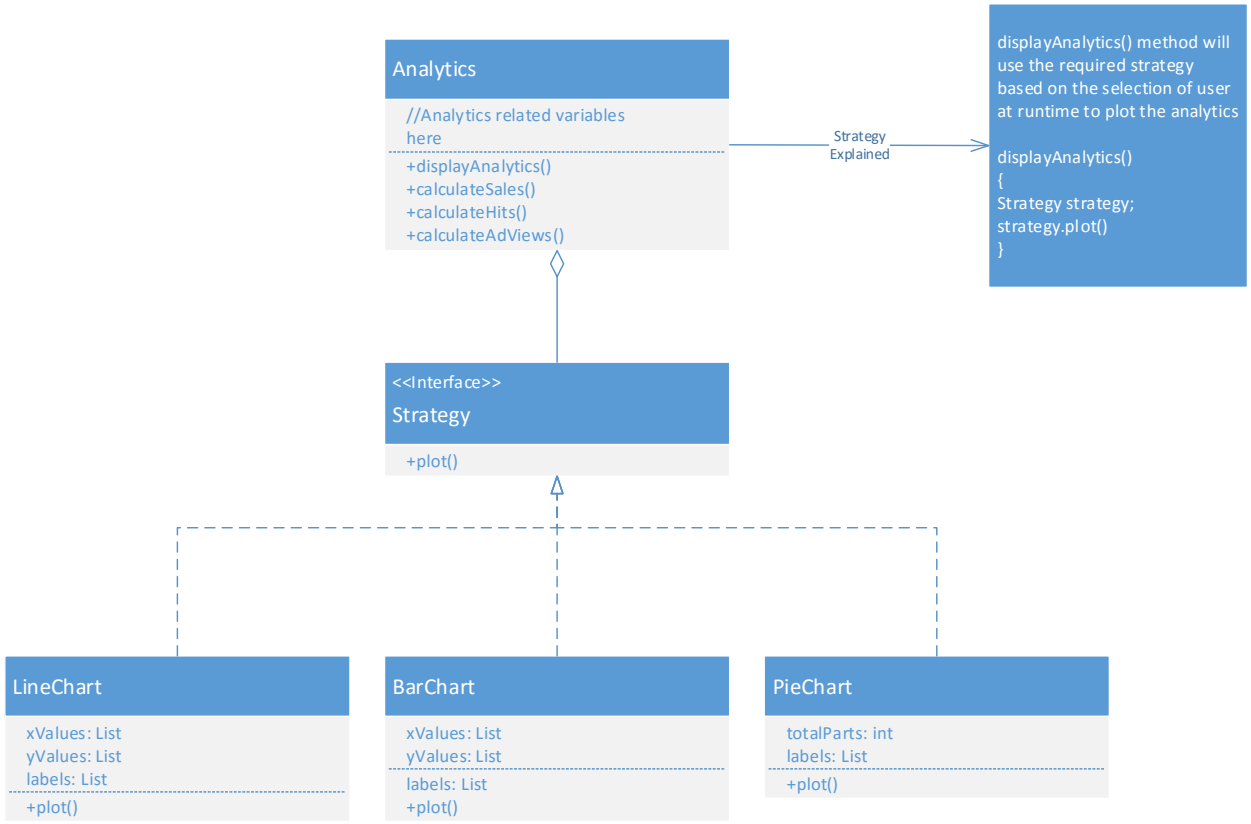
Object Oriented Goal: Keeping the implementation details as flexible as possible. Abstracting as much as possible.

Definition: Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

Implemented on classes:

1. Analytics
2. Strategy
3. LineChart
4. BarChart
5. PieChart

Design Snapshot:



Design Explanation:

There are three types of Analytics options and all the three types of analytics have different implementation details. It must be decided on runtime that which type should be used to process the Analytics. When the client selects the type of Analytics, the proper processing of Analytics needs to be done on runtime.

We can apply the strategy pattern to use the three types of Analytics processing options interchangeably. Whenever the client selects the type of Analytics, the proper processing strategy is created and processing of the Analytics is done based on the selection of the client. Below are some steps which explain the implementation of strategy pattern in our design.

- The Analytics class is the context class which will have a method which will require the processing of Analytics. In our case that method is the displayAnalytics() method.
- The Strategy is an interface which provides a function/method which has to be implemented by the classes which implement this interface. In this case, the plot() method is the method which will be implemented by the classes implementing Strategy interface.
- LineChart, BarChart and PieChart are the classes which have different implementation of the plot() method and are called on by the context whenever a strategy is desired to be performed.
- The plot() method contains different implementation in all the three classes and thus gives the client ability to select a strategy at runtime.

11.4 The Iterator Design Pattern

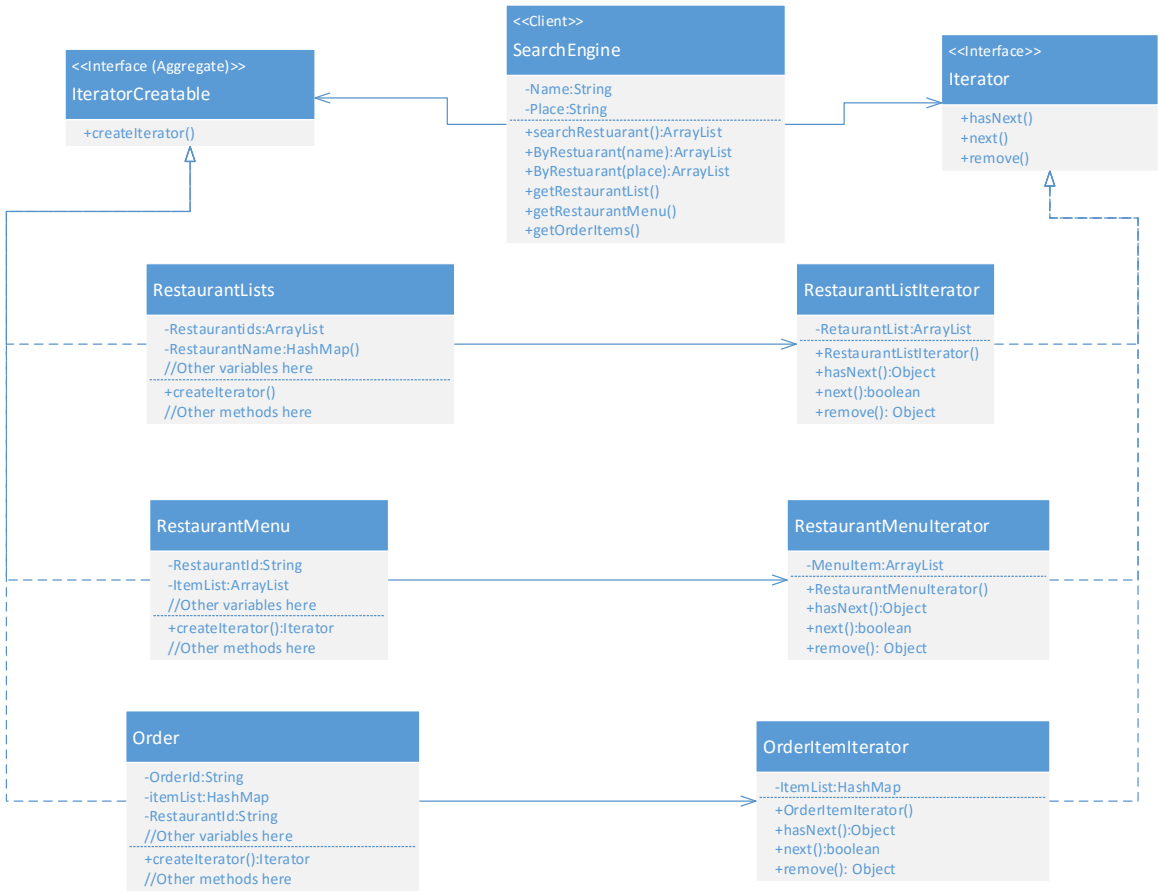
Object Oriented Goal: Traversal of elements of an aggregate without exposing the underlying implementation.

Definition: Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.

Implemented on classes:

1. SearchEngine
2. IteratorCreatable
    a. RestaurantLists
    b. RestaurantMenu
    c. Order
3. Iterator
    a. RestaurantListsIterator
    b. RestaurantMenuIterator
    c. OrderItemIterator

Design Snapshot:



Design Explanation:

We need a way to iterate over different types of data structures present in different classes. We need a common interface which defines a way to iterate over a collection of elements. In this design, the Pending Orders and Order Items need a common interface through which the Order handler can easily traverse the elements from both the classes.

We can implement the Iterator pattern in this case. The below points explain the implementation of Iterator Pattern in our design.

- The SearchEngine is the client class which requires to traverse over the restaurant menu, restaurant list and order list collection.
- The IteratorCreatable is the interface which has an abstract method createIterator() which will be implemented by the classes which needs to create an iterator for themselves.
- The RestaurantLists, Order and RestaurantMenu are the classes which have the different types of data structures to store the lists of items respectively. These classes implement the IteratorCreatable interface and so also implement the method createIterator() which creates particular concrete iterator for the class to traverse the list of elements.
- The Iterator interface defines a set of methods which needs to be implemented by the concrete iterators so that traversal methods can be implemented.
- The RestaurantListsIterator, OrderItemIterator and RestaurantMenuIterator are the concrete Iterator classes which implements the methods hasNext(), next() and remove() for the traversal and removal of elements for the respected collections of the classes.
- Now, the Order Handler client only knows about the IteratorCreatable interface & the Iterator interface. Whenever the SearchEngine class wants to iterate or traverse over the collections from RestaurantLists, Order and RestaurantMenu class, it deals with the interfaces mentioned above and uses the methods from the concrete iterators to iterate over the elements.

11.5 The Adapter Design Pattern

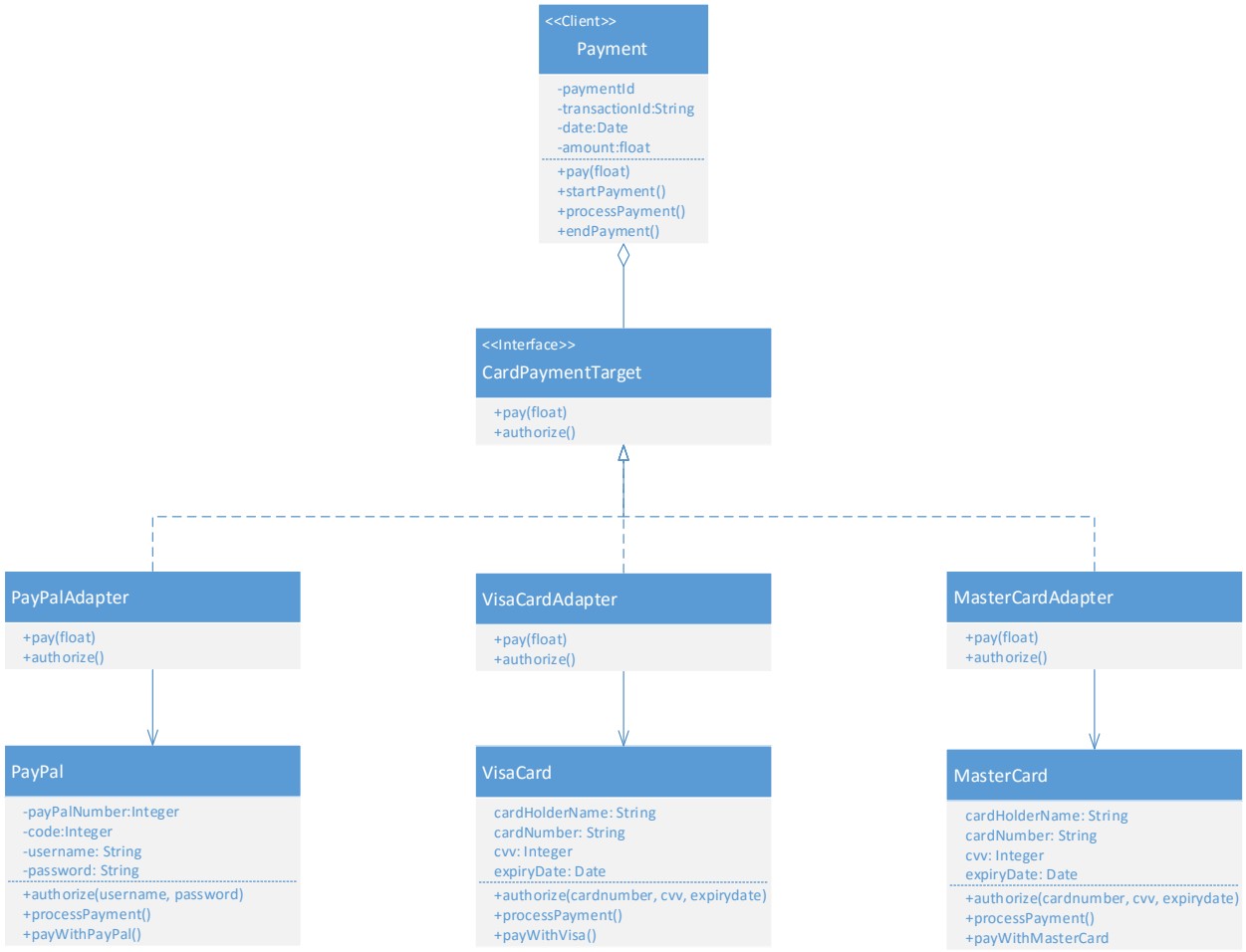
Object Oriented Goal: Favoring object composition over inheritance. Bind the client to an interface and to implementation.

Definition: The Adapter Pattern converts the interface of a class into another interface the clients expect. Adapter lest classes work together that couldn't otherwise because of incompatible interfaces.

Implemented on classes:

1. Payment
2. CardPaymentTarget (Target Interface)
3. PayPalAdapter
    a. PayPal
4. VisaCardAdapter
    a. VisaCard
5. MasterCardAdapter
    a. MasterCard

Design Snapshot:



Design Explanation:

In our application, there are multiple ways for the user to pay for the order like Credit Card, PayPal and Cash on Delivery and we also accept Master Card and Visa Card both. Thus the classes for Master Card, Visa Card and PayPal differ in method implementations and also have different interfaces. We need a way for our card payment option to adapt with different types like Master Card, Visa and PayPal.

In this situation, we can add the adapter pattern for the clients to easily use the adapters for payment rather than going for the implemented classes to perform functions. The Adapter Pattern implemented in our design is explained as follows:

- The Payment class is our client class which will only know about the CardPaymentTarget interface and will only implement the pay and authorize methods using the Target interface as the type of different objects like MasterCard, Visa and PayPal.
- The CardPaymentTarget is the target interface which has methods like pay and authorize which needs to be implemented by the respective adapters.
- PayPal, MasterCard and VisaCard are the Adaptee classes which will get the calls that Payment will make on Target Interface.
- PayPalAdapter, VisaCardAdapter and MasterCardAdapter are the class which compose the PayPal, VisaCard and MasterCard classes respectively. The adapter classes implements all the methods from the target interface and in those methods are the method calls to the particular classes like PayPal, VisaCard and MasterCard.
- The client Payment invokes method on the target (CardPaymentTarget) interface. The PayPalAdapter, VisaCardAdapter and MasterCardAdapter gets the method call and it delegates the calls to their respective classes.

11.6 The State Design Pattern

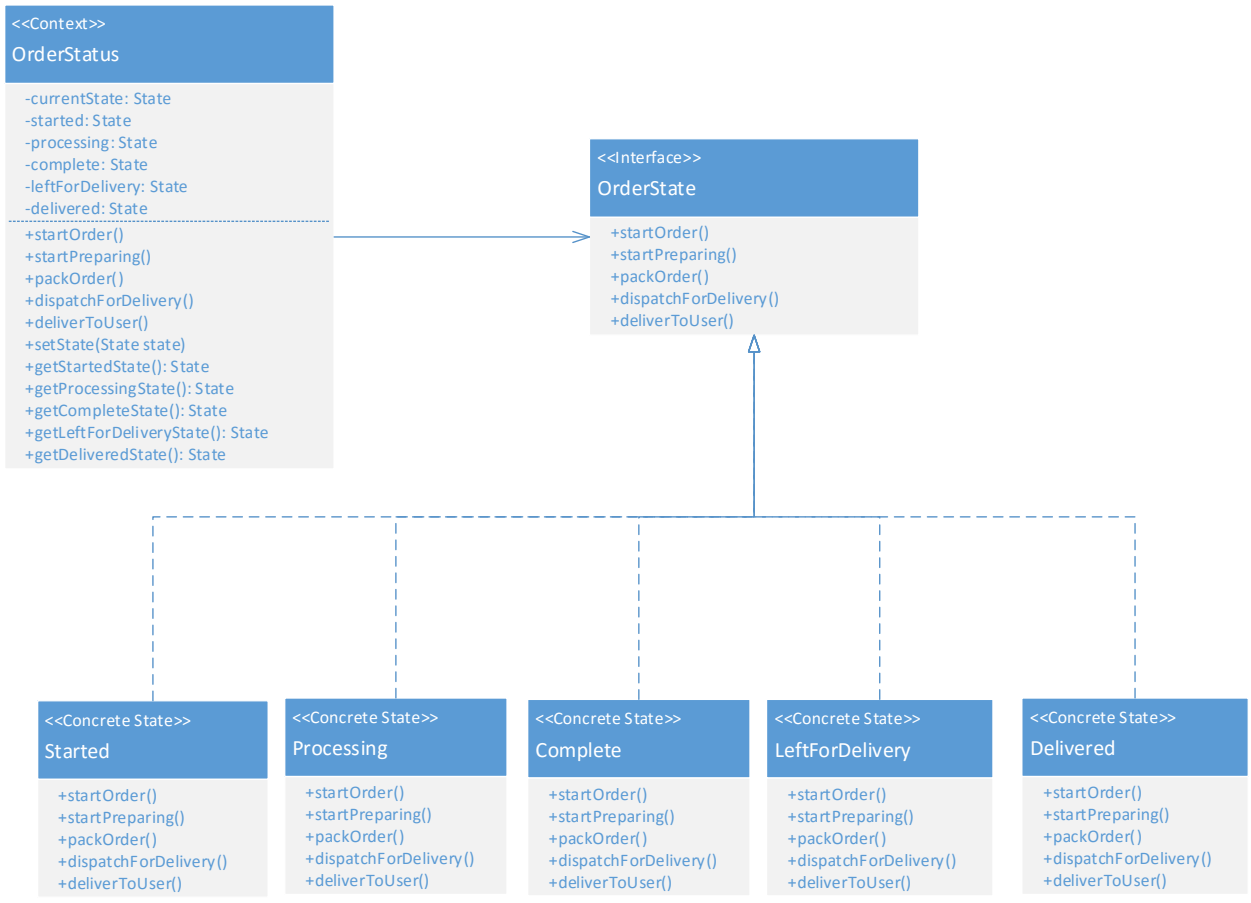
Object Oriented Goal: Encapsulate what varies. Favor composition over inheritance. Strive for loosely coupled designs. Depend on abstractions. Do not depend on concrete classes.

Definition: The state pattern allows an object to alter its behavior when its internal state changes. The object will appear to change its class.

Implemented on classes:

1. OrderStatus (Context)
2. OrderState (Interface)
3. Started
4. Processing
5. Complete
6. LeftForDelivery
7. Delivered

Design Snapshot:



Design Explanation:

There is a functionality in our design in which the user can track the order status in the app and the app updates the order status whenever the order makes a transition from one state to another. We have 5 different states for our order.

To implement this functionality, we have used the state design pattern and its implementation in our design is as follows:

- The OrderStatus is our context class and it can have a number of internal states. This is the class where all the changes of the events will be done. Every time a change of event occurs, the current state is changed and a new state is delegated to the current state object. This class has reference to all the concrete state objects and thus it makes easier to delegate the objects to the current state.
- The OrderStatus class also has setState method which will change the state to the required state as soon as an event occur. The different events have been implemented as methods which will call the particular State object to perform its functionality.
- The OrderState is the State Interface which has some methods which needs to be implemented by the concrete state classes. This interface defines a template for the concrete states. This methods are basically the events which can occur on the order status and thus will change the state of the order.
- The concrete state classes are Started, Processing, Complete, LeftForDelivery and Delivered. These classes implement all the methods from the OrderState interface and implements the method according to their own state.
- The concrete states handle requests from the context. Each concrete state provides its own implementation. Thus, when the context changes state, its behavior will also change.

11.7 The Singleton Design Pattern

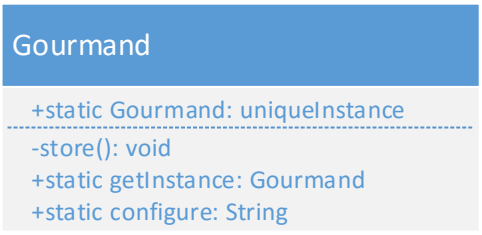
Object Oriented Goal: Ensuring only one instance of the singleton object available to all the classes.

Definition: The Singleton Pattern ensures a class has only one instance, and provides a global point of access to it.

Implemented on classes:

1. Gourmand

Design Snapshot:



Design Explanation:

The Gourmand class is instantiated only once because it is the class used to initiate the app. Thus, after initializing the variable for the first time, that same variable is used as an instance whenever it is required by any other object.

The Singleton Pattern makes a class singleton by providing only one static instance variable to all the objects who wants to access that object. Thus ensuring only one global object. It also defines one static public method which is used to access the unique instance. The implementation of Singleton Pattern in my design is explained as follows:

- The Gourmand class is the singleton class. The variable uniqueInstance is marked as a static variable and thus there is only one copy of that variable. That will be the class variable and anyone who wants to access the instance of Gourmand will get this global instance from the getInstance() method
- The Gourmand can be implemented as follows so as to make the class a singleton class:

```
Public class Gourmand
{
        //declaring the unique instance as a private and static method so that there
        remains only one instance of the class and no one can access the instance directly
        Private static Gourmand uniqueInstance;

        //making the constructor of the class as private, so no one can instantiate the
        class using the constructor
        Private Gourmand() {}
```

```
        //providing the single unique instance of the class by checking if the instance
        already exists or not
        Public static Gourmand getInstance()
        {
                If(uniqueInstance == null)
                        uniqueInstance = new Gourmand();

                //this will return the unique instance to whoever called this method
                return uniqueInstance;
        }
}
```