Uber
Eats

CONTENT

1. PROJECT IDEA

Uber Eats is an online food delivery platform launched by Uber in 2014. It allows users to order food from local restaurants through a mobile app or website, which is then delivered by drivers using Uber's transportation network. Uber Eats has grown rapidly to become one of the world’s largest food delivery platforms, available in 6,000+ cities across 45+ countries. Uber Eats operates on a three-sided marketplace:

- Consumers
- Restaurants
- Delivery partners

Uber Eats can be accessible via its app or through its website. Uber Eats has become the go-to app for food delivery among consumers due to its convenience, broad selection of restaurants and tech-driven features like real-time order tracking and AI chatbots. While these features offer a seamless experience to individuals, our team aims to optimize the platform to suit the needs of group orders and students. We plan to achieve this by integrating crucial features that make the process of ordering as a group less cumbersome. Some benefits from these features would include reduction of time spent deciding on an order, lessened delivery times, and reduced hassle while paying the bill as a group. On an individual front, we look forward to adding features that encourage students to use the app more, and meal subscription options that would prompt working individuals to subscribe and utilize the app better.

2. PROBLEM STATEMENT

In the current uber eats app, there lacks some crucial features like collaborative planning with effortless splitting, mystery meal, student's discounts, and prepaid meal savings program which if implemented would enhance the user experience, satisfaction, and convenience significantly. Furthermore, a lack of innovative tools on the platform makes it difficult for consumers who are unsure of their options or who are seeking novel experiences to make decisions and explore new restaurants. Also, consumers who frequently order food with friends and family may become frustrated as there are no effective bill splitting options and coordinated decision making. Moreover, there are no discounts, or meal plans, especially for students who are looking for budget friendly options and find it hard to cook food daily due to busy schedules. These added features could help uber eats to fully capitalize on market niche and consumer satisfaction by maximizing convenience, engagement, and loyalty. Addressing this gap could help uber eats to stay competent and ahead of their competitors.

3. PROPOSED SOLUTION

- ***Collaborative Dining with Effortless Splitting***
  Uber Eats can improve the overall dining experience by making ordering easier for groups, reducing the stress of decision-making, offering more cost-effective options for regular users, and providing better support for those focused on their health and nutrition. These improvements will make the platform more convenient, enjoyable, and tailored to individual preferences, creating a smoother and more personalized experience for all users.

- ***Mystery Meal***
  Create a "Mystery Meal" option that selects a restaurant for users based on preferences or cuisine type. Users set their food preferences, and the app takes care of the decision-making. This adds excitement and saves time when groups cannot decide. It helps users discover new restaurants or cuisines. The feature adds an element of surprise and fun to ordering

- ***Prepaid Meal Savings Program***
  Offer a subscription plan where users can pre-pay for a set number of meals each month from select restaurants. This provides a discount on regular orders and ensures convenience for frequent users. Subscribers can easily manage their meals through the app. The plan helps users save money and brings consistent revenue to Uber Eats and partner restaurants.

- ***Student Discounts***
  Uber Eats can attract and retain student customers by offering verified student discounts, such as 10%-20% off orders and free delivery. Campus-focused promotions could involve recruiting student ambassadors to promote the platform through referral programs, hosting pop-up events like food truck days, and distributing flyers with QR codes for discounted offers around campuses. A loyalty program that rewards students with points for every order, allowing them to redeem discounts or free meals, would enhance long-term retention, while flexible payment options, such as "buy now, pay later" or discounted meal bundles, can alleviate financial pressure for students. Together, these strategies provide Uber Eats with a competitive edge in the student market, fostering both engagement and brand loyalty.

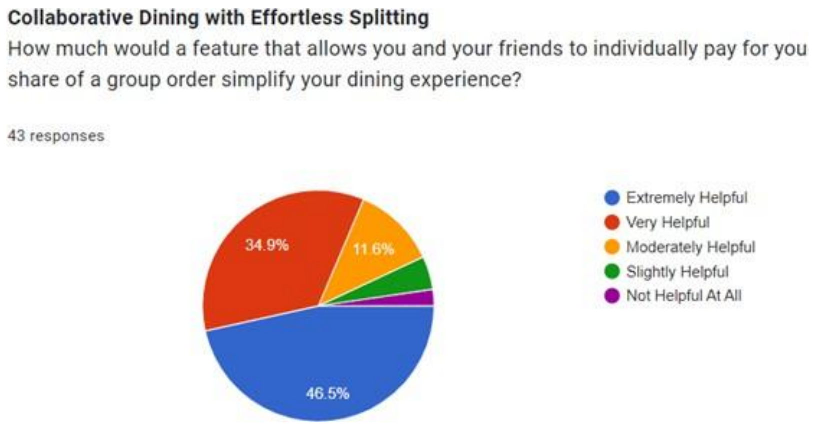
- ***Multi-Restaurant Ordering with Single Pickup***
  Uber Eats can introduce a feature allowing users to order from multiple nearby restaurants in one go, with a single driver picking up all items. This would eliminate the need for separate orders, reducing delivery fees and wait times. Users would receive their entire meal in one delivery, making the experience more convenient, especially for those who want variety or need to accommodate different preferences within a group. This feature would enhance flexibility, save money, and create a smoother, more efficient food delivery experience.

PP

4. SUPPORT (RESEARCH)

- ***Collaborative Dining with Effortless Splitting***

  Simplifying group orders and adding bill-splitting options would be a notable change for users. It eliminates the hassle of manual calculations and coordinating payments, making group dining much easier. Introducing this feature could improve the experience for those who regularly order with friends, leading to higher satisfaction and potentially larger orders.



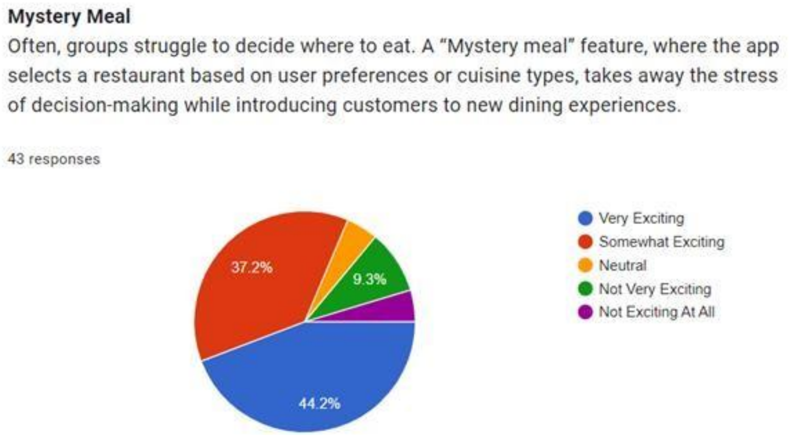
Collaborative Dining with Effortless Splitting

How much would a feature that allows you and your friends to individually pay for you share of a group order simplify your dining experience?

43 responses

- ***Mystery Meal***

  The "Mystery Meal" feature has sparked a lot of excitement, especially among users who enjoy a fun and adventurous dining experience. It is perfect for indecisive customers, helping them explore new restaurants and cuisines with a bit of surprise. By taking the pressure off decision-making, it adds a playful element to ordering while boosting engagement. This feature makes trying new options easy and exciting, making it a smart choice for Uber Eats customers.

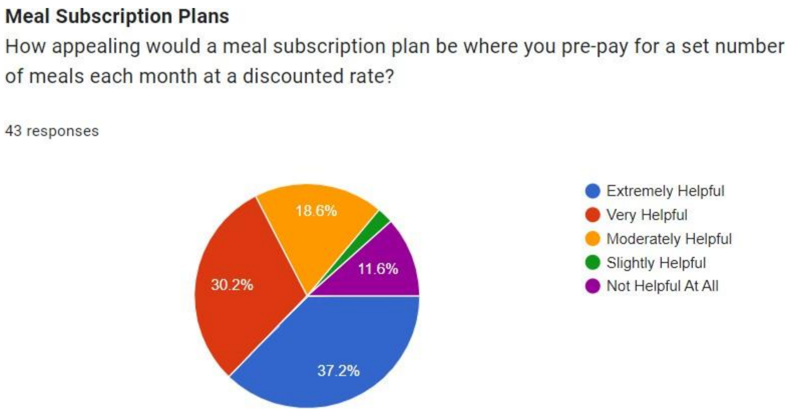


Mystery Meal

Often, groups struggle to decide where to eat. A "Mystery meal" feature, where the app selects a restaurant based on user preferences or cuisine types, takes away the stress of decision-making while introducing customers to new dining experiences.

43 responses

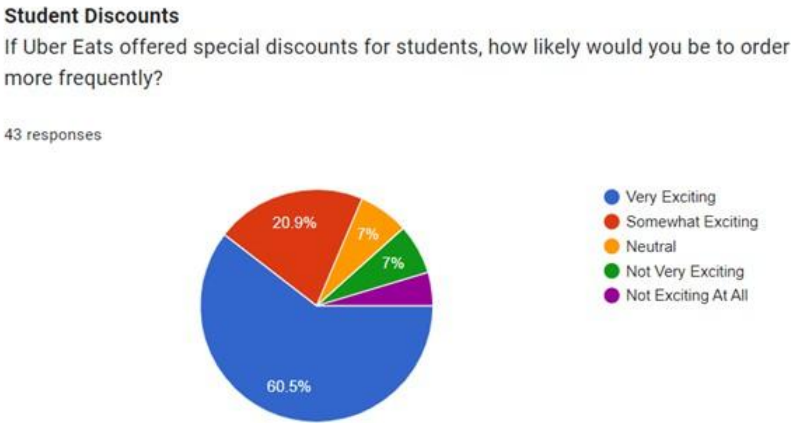
- ***Meal Subscription Plans***
  The concept of a prepaid meal savings plan appeals to many frequent users who want to save money and simplify their ordering routine. Launching a subscription service with discounted meal packages will appeal to regular customers and promote higher order frequency and customer retention.



Meal Subscription Plans
How appealing would a meal subscription plan be where you pre-pay for a set number of meals each month at a discounted rate?

43 responses

- ***Student Discounts***
  Many students are highly enthusiastic about discounts, and offering student-specific promotions could significantly boost engagement. With students often being cost-conscious, these targeted deals can make a significant difference. By introducing exclusive student discounts, not only would it increase order frequency, but it could also strengthen loyalty, turning occasional users into regular customers.

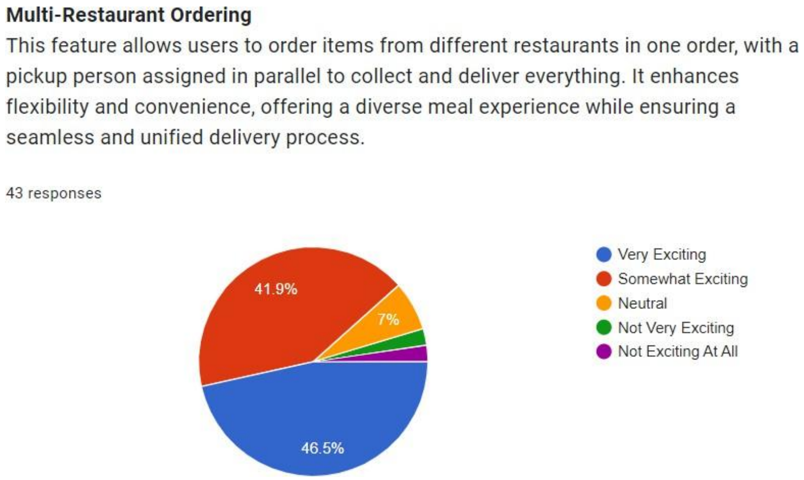


Student Discounts
If Uber Eats offered special discounts for students, how likely would you be to order more frequently?

43 responses

- ***Multi-Restaurant Ordering***

  Customers are excited about the convenience of ordering from multiple restaurants in one go, and this feature could really boost engagement. People love flexibility, and being able to enjoy a variety of meals in one seamless delivery is a game changer. Offering multi-restaurant ordering not only makes the experience more enjoyable, but it could also lead to more frequent orders and stronger customer loyalty, turning occasional users into regulars.



Multi-Restaurant Ordering

This feature allows users to order items from different restaurants in one order, with a pickup person assigned in parallel to collect and deliver everything. It enhances flexibility and convenience, offering a diverse meal experience while ensuring a seamless and unified delivery process.

43 responses

5. USE CASE DIAGRAM

Actors

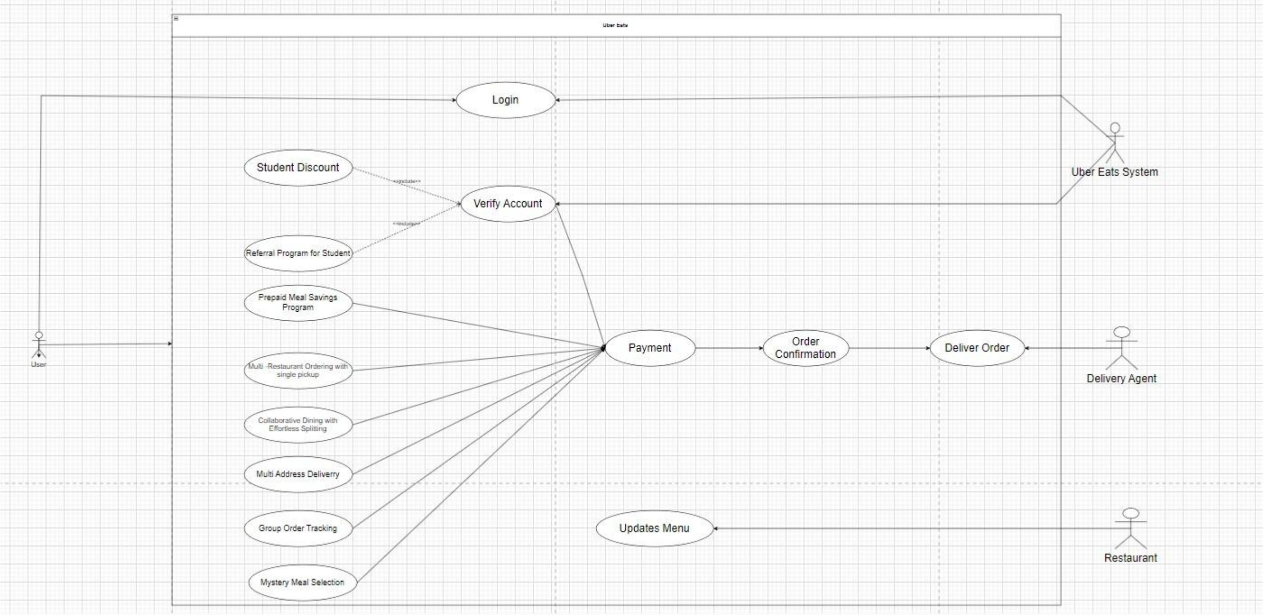
- Consumer (User) – People using Uber Eats for individual or group orders.
- Student – A specific user type, eligible for student discounts and features like prepaid meals.
- Delivery Partner – Drivers who pick up and deliver orders.
- Restaurant – Local restaurants registered on Uber Eats.
- Uber Eats System – The platform that processes orders, payments, and deliveries.
- External System – Platform which verifies student ID or another platform which processes payments.

Use Cases

- Login
- Place an Individual Order
- Collaborative Dining with Effortless Splitting
- Mystery Meal Selection
- Prepaid Meal Savings Program
- Student Discounts
- Multi-Restaurant Ordering with Single Pickup
- Process Payment (Split Payment)

- Delivery Order
- Restaurant Updates Menu
- Multi Address Delivery
- Group Order Delivery Tracking
- Referral Program for Students

Use Case Diagram:



6. USE CASE DESCRIPTION

Use Case Description 1:

Use case name: Login
Primary Actor: User
Stakeholder: Users, System Admin
Brief Description: Users log into the Uber Eats app to access their account and place orders.
Trigger: User opens the app and selects the login option
Normal Flow of Events: 1. User opens the app and selects "Login." 2. User enters credentials (email/password or social login).

3. System verifies the credentials. 4. If successful, the user is granted access to their account.
Exception: Invalid credentials; user is prompted to try again or reset their password.

Use Case Description 2:

Use case name: Place an Individual Order
Primary Actor: User
Stakeholder: Uber Eats, Restaurants, Delivery Drivers
Brief Description: User places an individual order from a restaurant using the Uber Eats app.
Trigger: User selects a restaurant and adds items to the cart.
Normal Flow of Events: 1. User selects restaurant and browses the menu. 2. User adds items to the cart. 3. User proceeds to checkout and selects delivery location. 4. Payment is processed, and the order is sent to the restaurant. 5. The restaurant prepares the order and assigns a delivery driver. 6. User receives order confirmation and estimated delivery time.
Exception: Payment failure, restaurant out of stock, or delivery delay.

Use Case Description 3:

Use case name: Collaborative Dining with Effortless Splitting
Primary Actor: Group of Users
Stakeholder: Uber Eats, Restaurants, Payment Gateways
Brief Description: A group of users places an order and splits the bill based on individual selections.
Trigger: Group initiates a shared cart for collaborative ordering.
Normal Flow of Events: 1. User creates a group order and invites participants. 2. Each group member adds items to the shared cart. 3. The app calculates individual costs and displays split payment options. 4. Each user approves their selection and splits the payment.

5. Orders are placed and processed by the restaurant. 6. Delivery is initiated once the order is ready.
Exception: One or more users fail to make payment; items unavailable from the restaurant.

Use Case Description 4:

Use case name: Mystery Meal Selection
Primary Actor: User
Stakeholder: Uber Eats, Restaurants
Brief Description: Users opt for a "mystery meal," allowing Uber Eats to choose a restaurant and meal based on preferences.
Trigger: User selects the "Mystery Meal" option.
Normal Flow of Events: 1. User selects “Mystery Meal” and inputs food preferences. 2. App chooses a restaurant and selects a meal based on preferences. 3. User is informed of the restaurant and meal. 4. Payment is processed, and the order is placed. 5. Delivery is initiated once the order is ready.
Exception: No restaurants match user preferences or dietary restrictions.

Use Case Description 5:

Use case name: Prepaid Meal Savings Program
Primary Actor: Frequent Users
Stakeholder: Uber Eats, Restaurants, Payment Gateways
Brief Description: Users subscribe to a prepaid meal plan, receiving discounts on monthly orders.
Trigger: User subscribes to a meal savings plan.
Normal Flow of Events: 1. User navigates to the prepaid meal plan section. 2. Users select a plan and the number of meals they want per month. 3. Payment is processed for the subscription. 4. Meals are credited to the user’s account, and discounts are applied to orders.

5. Users order meals as needed and use prepaid credits.

Exception: Payment failure, user exceeds monthly meal limit.

Use Case Description 6:

Use case name: Student Discounts

Primary Actor: Students

Stakeholder: Uber Eats, Universities, Restaurants

Brief Description: Students receive discounts and promotions by verifying their student status.

Trigger: User inputs student verification information.

Normal Flow of Events:

1. User navigates to the student discount section.
2. Users enter their student ID or connect through university email.
3. Verification is processed.
4. Discounts are applied to the user’s account.
5. Users place an order, and discounts are reflected at checkout.

Exception: Student verification fails; discount not applied.

Use Case Description 7:

Use case name: Multi-Restaurant Ordering with Single Pickup

Primary Actor: User

Stakeholder: Uber Eats, Multiple Restaurants, Delivery Drivers

Brief Description: Users place orders from multiple restaurants in one order, with a single driver delivering all items.

Trigger: User adds items from more than one restaurant to the cart.

Normal Flow of Events:

1. User selects items from multiple restaurants and adds them to the cart.
2. User proceeds to checkout and payment is processed.
3. Each restaurant prepares the order.
4. One driver picks up all the items and delivers them in a single trip.

Exception: One or more restaurants are delayed or unable to fulfill the order.

Use Case Description 8:

Use case name: Process Payment (Split Payment)
Primary Actor: Group of Users
Stakeholder: Uber Eats, Payment Gateways, Banks
Brief Description: A group of users splits the payment for an order based on individual selections or equally.
Trigger: Group members confirm their items and proceed to payment.
Normal Flow of Events: 1. The app calculates the total order amount. 2. Users confirm their individual selections and payment method. 3. The app processes each payment individually. 4. Once all payments are successful, the order is confirmed.
Exception: One or more payments fail, preventing order completion.

Use Case Description 9:

Use case name: Delivery Order
Primary Actor: Delivery Driver
Stakeholder: Uber Eats, Users, Restaurants
Brief Description: Delivery driver picks up the order from the restaurant and delivers it to the user
Trigger: Order is prepared by the restaurant and hands over to driver.
Normal Flow of Events: 1. Restaurant prepares the order. 2. The delivery driver is assigned and notified. 3. The driver picks up the order from the restaurant. 4. The driver delivers the order to the user. 5. User confirms receipt of the order in the app.
Exception: Delivery delay, incorrect delivery address, or traffic issues.

Use Case Description 10:

Use case name: Restaurant Updates Menu
Primary Actor: Restaurant
Stakeholder: Uber Eats, Users
Brief Description: Restaurants update their menu on Uber Eats to reflect added items or availability.
Trigger: Restaurant needs to make menu updates.
Normal Flow of Events: 1. Restaurant logs into their Uber Eats account. 2. Restaurant updates menu items (add/remove/change). 3. Menu changes are saved and published on the app. 4. Users can see the updated menu.
Exception: Menu update fails, or added items are not displayed properly.

Use Case Description 11:

Use case name: Multi Address Delivery
Primary Actor: User
Stakeholder: Uber Eats, Delivery Drivers, Restaurants
Brief Description: User places an order for multiple addresses in one checkout (e.g., for friends or family members at various locations).
Trigger: User selects the option to deliver to multiple addresses.
Normal Flow of Events: **1.** The user selects the items and enters multiple delivery addresses. **2.** The app calculates delivery fees for each address. **3.** User confirms and processes the payment. **4.** Restaurants prepare the order. **5.** Multiple delivery drivers are assigned to different addresses. **6.** Users receive their deliveries at the specified addresses.
Exception: Incorrect address, delivery failure for one or more locations.

Use Case Description 12:

Use case name: Group Order Delivery Tracking

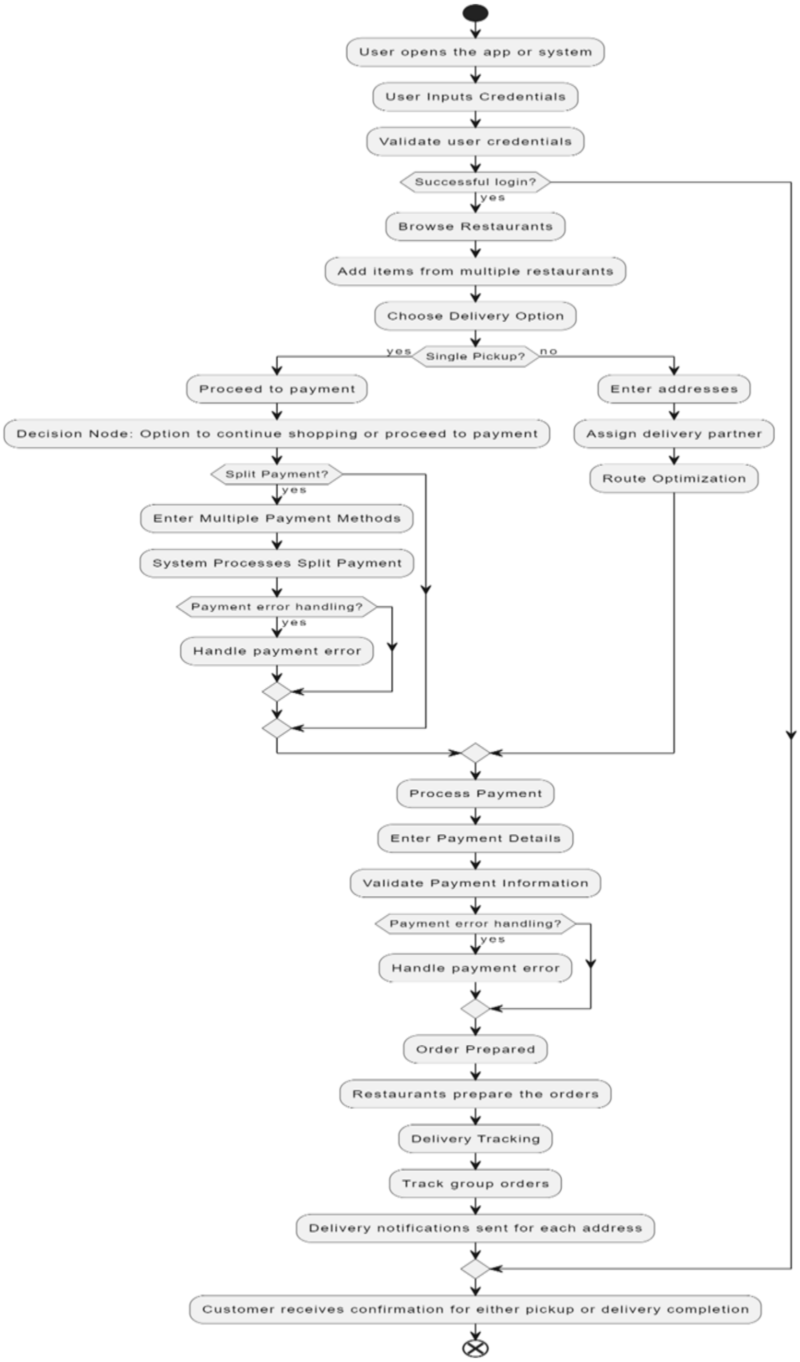
Primary Actor: Group of Users
Stakeholder: Uber Eats, Delivery Drivers, Restaurants
Brief Description: Group of users can track the status and location of their shared order in real time.
Trigger: Group places a collaborative order.
Normal Flow of Events: 1. The group collectively places a collaborative order through the app. 2. The restaurant begins preparing the shared order and updates are sent to the group. 3. The app tracks the shared order's progress (preparation, pickup, and delivery), with all group members having access to real-time updates. 4. The driver delivers the order, and the group is notified upon successful delivery.
Exception: Delivery tracking is unavailable or delayed.

Use Case Description 13:

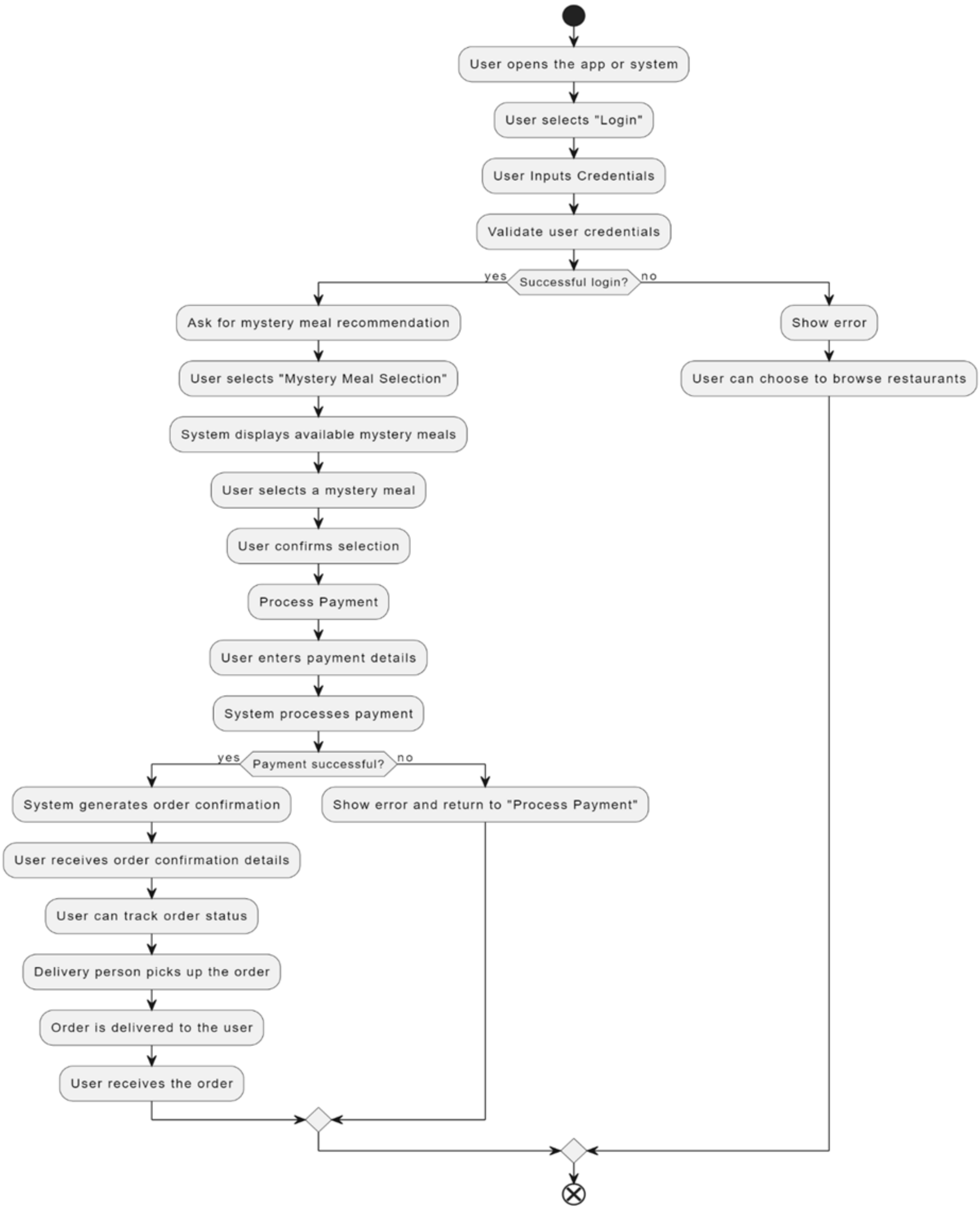
Use case name: Referral Program for Students
Primary Actor: Students
Stakeholder: Uber Eats, Universities
Brief Description: Students refer friends to Uber Eats and receive rewards for each successful referral.
Trigger: Student shares referral code with a friend.
Normal Flow of Events: 1. Students access their referral code in the app. 2. A student shares the referral code with a friend. 3. Friend signs up and places an order using the referral code. 4. The referring student and the new user both receive rewards.
Exception: Referral code fails to apply, or rewards are not issued.

7. ACTIVITY DIAGRAM

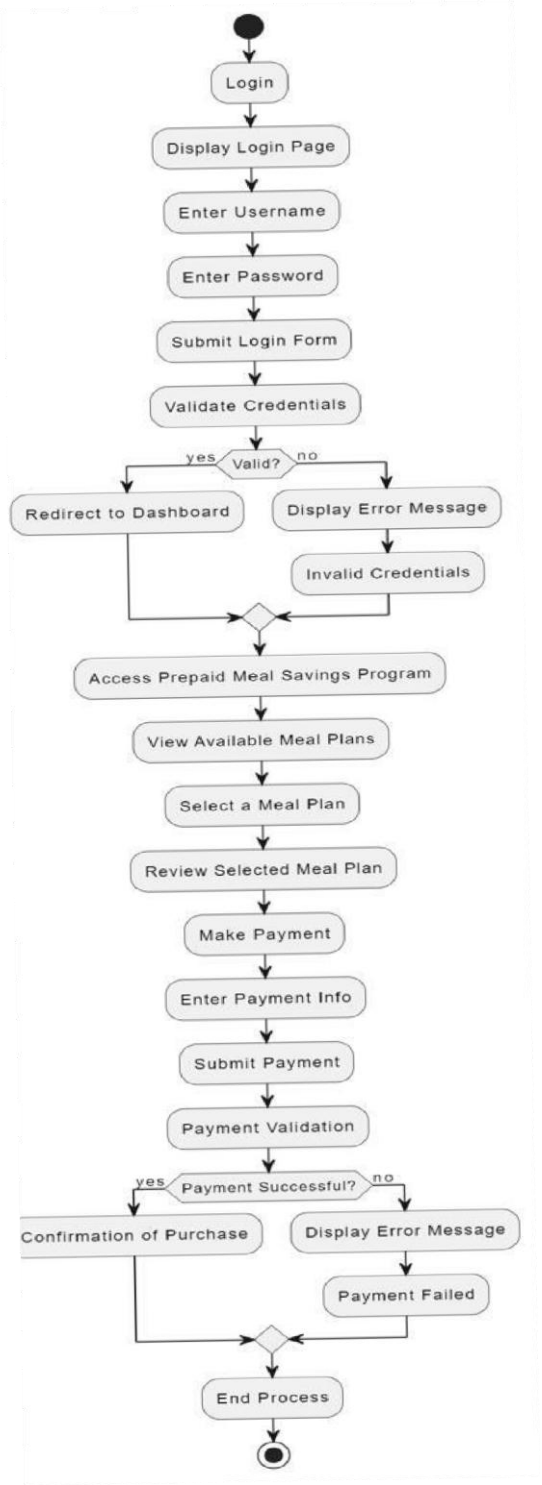
Group Orders:


User opens the app or system
User Inputs Credentials
Validate user credentials
Successful login?
yes
Browse Restaurants
Add items from multiple restaurants
Choose Delivery Option
yes
Single Pickup?
no
Proceed to payment
Enter addresses
Decision Node: Option to continue shopping or proceed to payment
Assign delivery partner
Split Payment?
Route Optimization
yes
Enter Multiple Payment Methods
System Processes Split Payment
Payment error handling?
yes
Handle payment error
Process Payment
Enter Payment Details
Validate Payment Information
Payment error handling?
yes
Handle payment error
Order Prepared
Restaurants prepare the orders
Delivery Tracking
Track group orders
Delivery notifications sent for each address
Customer receives confirmation for either pickup or delivery completion

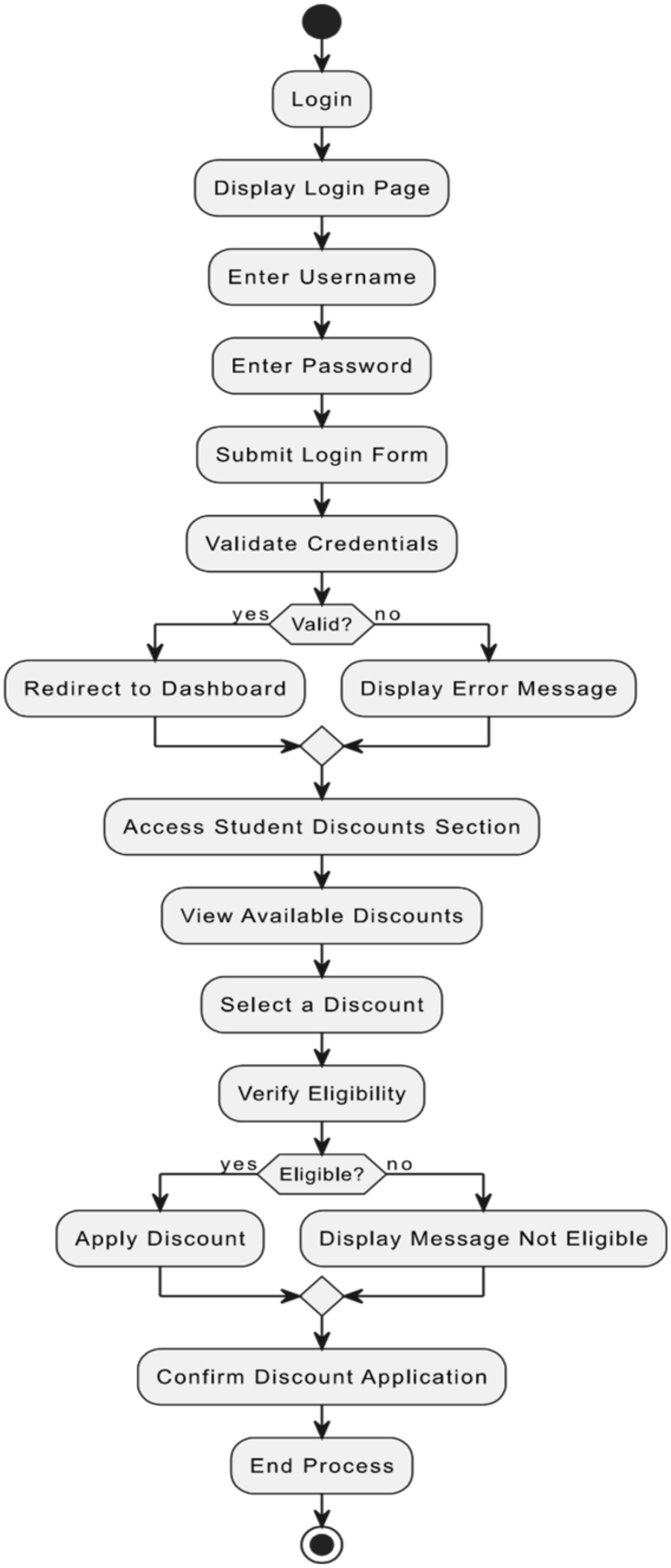
Mystery Meal Recommendation:


User opens the app or system
User selects "Login"
User Inputs Credentials
Validate user credentials
yes
Successful login?
no
Ask for mystery meal recommendation
User selects "Mystery Meal Selection"
System displays available mystery meals
User selects a mystery meal
User confirms selection
Process Payment
User enters payment details
System processes payment
yes
Payment successful?
no
System generates order confirmation
User receives order confirmation details
User can track order status
Delivery person picks up the order
Order is delivered to the user
User receives the order
Show error and return to "Process Payment"
Show error
User can choose to browse restaurants

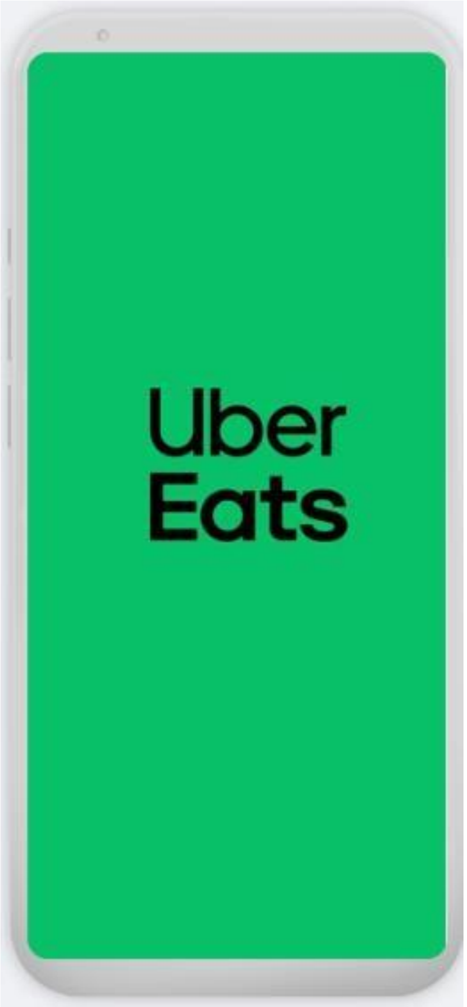
Prepaid meals plan program:

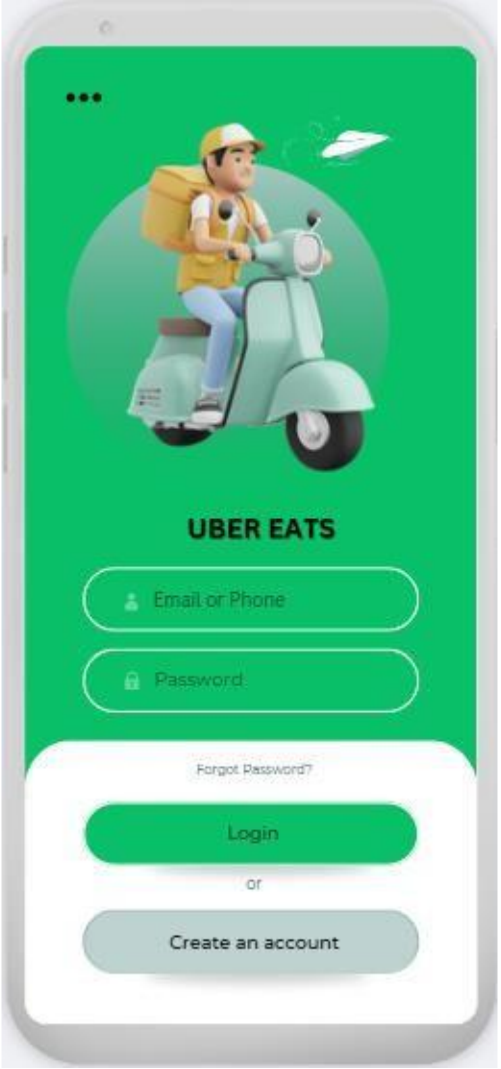


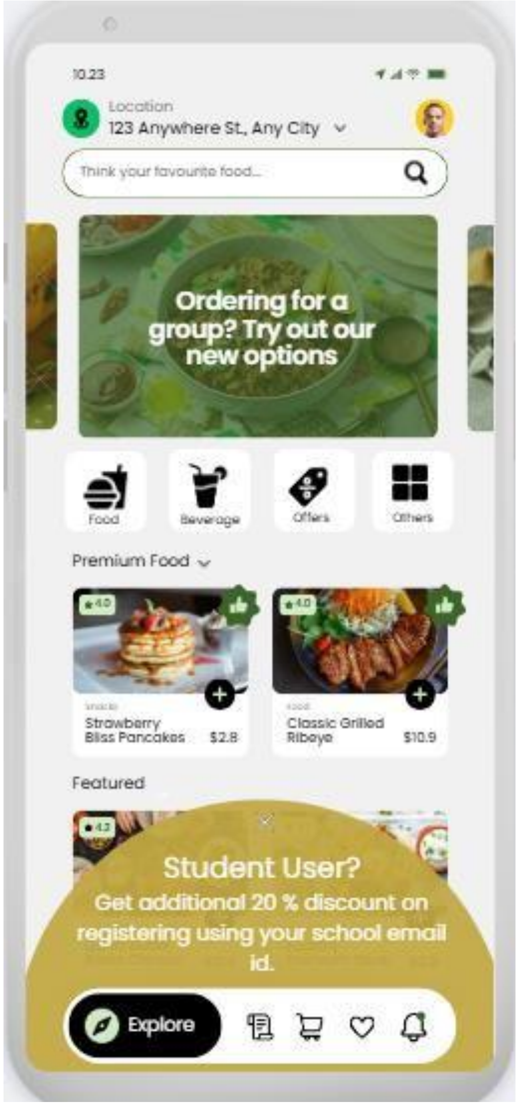
Student discount:


Login
Display Login Page
Enter Username
Enter Password
Submit Login Form
Validate Credentials
yes
Valid?
no
Redirect to Dashboard
Display Error Message
Access Student Discounts Section
View Available Discounts
Select a Discount
Verify Eligibility
yes
Eligible?
no
Apply Discount
Display Message Not Eligible
Confirm Discount Application
End Process

8. USER INTERFACE

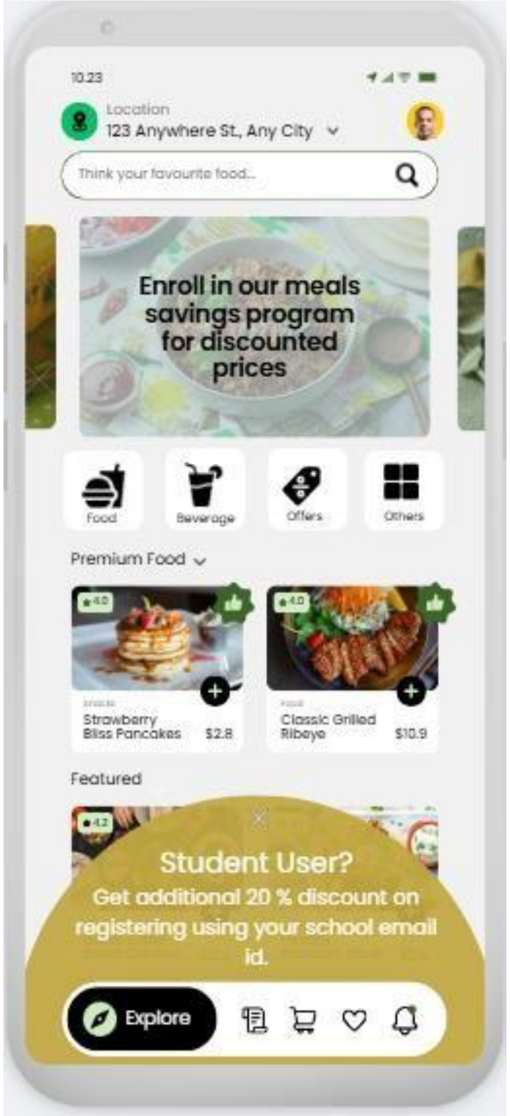






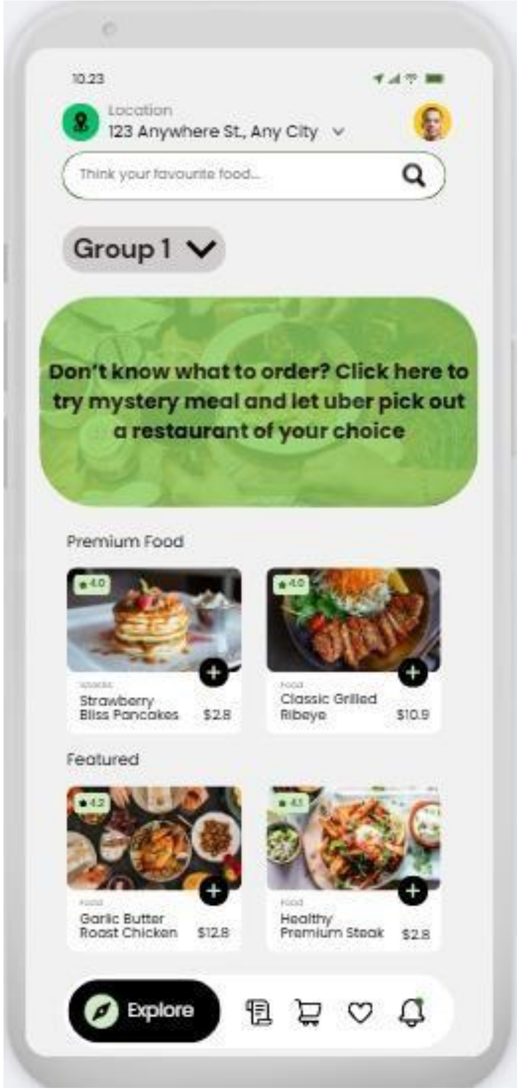
Ordering for a group? Try out our new options

Student User?
Get additional 20 % discount on registering using your school email id.

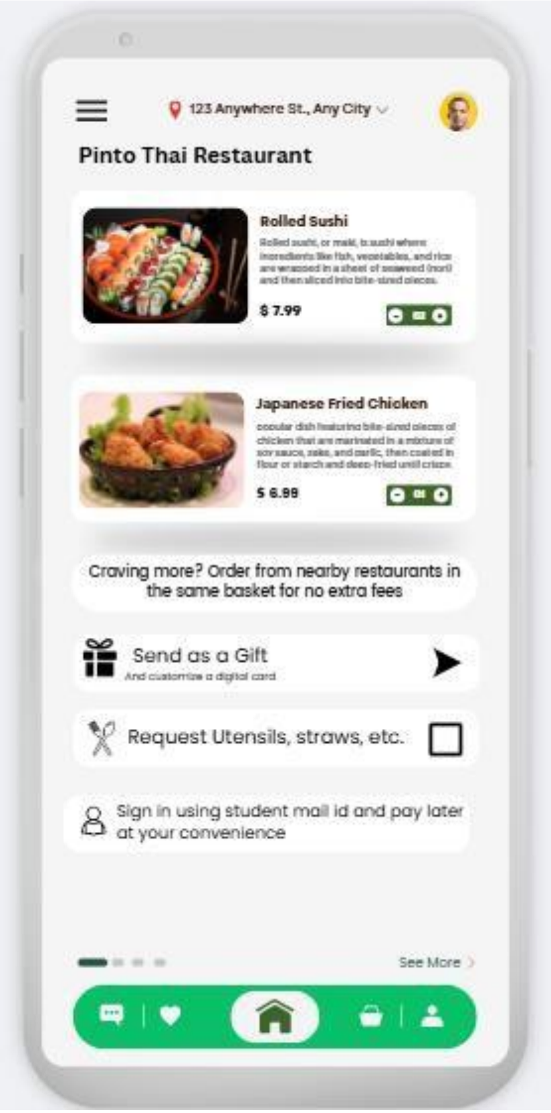


Enroll in our meals savings program for discounted prices

Student User?
Get additional 20 % discount on registering using your school email id.



Don't know what to order? Click here to try mystery meal and let uber pick out a restaurant of your choice

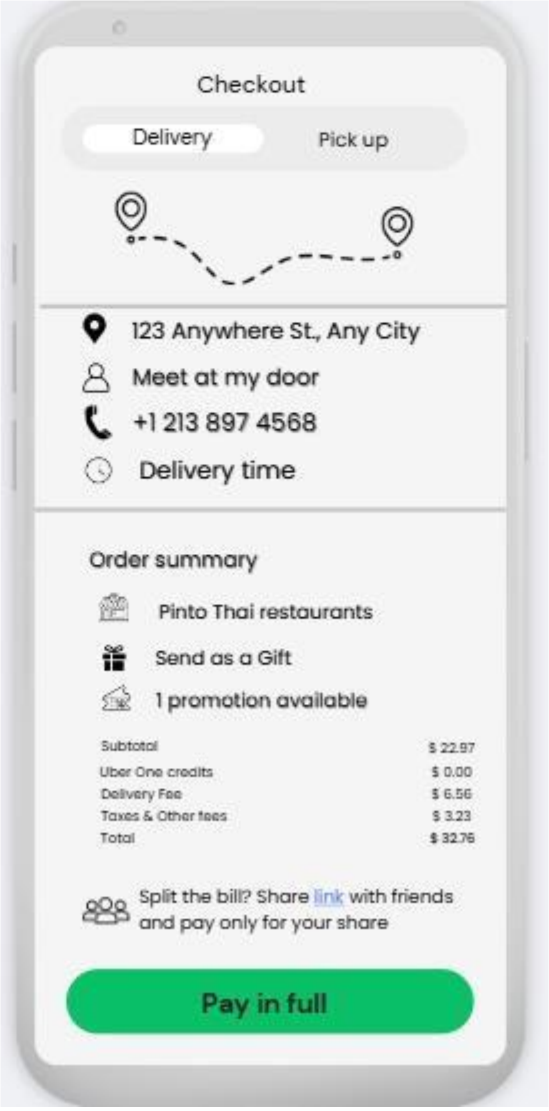


Rolled sushi, or maki, is sushi where ingredients like fish, vegetables, and rice are wrapped in a sheet of seaweed (nori) and then sliced into bite-sized pieces.

popular dish featuring bite-sized pieces of chicken that are marinated in a mixture of soy sauce, sake, and garlic, then coated in flour or starch and deep-fried until crispy.

Craving more? Order from nearby restaurants in the same basket for no extra fees

Sign in using student mail id and pay later at your convenience



Checkout

123 Anywhere St., Any City

Meet at my door

+1 213 897 4568

Delivery time

Order summary

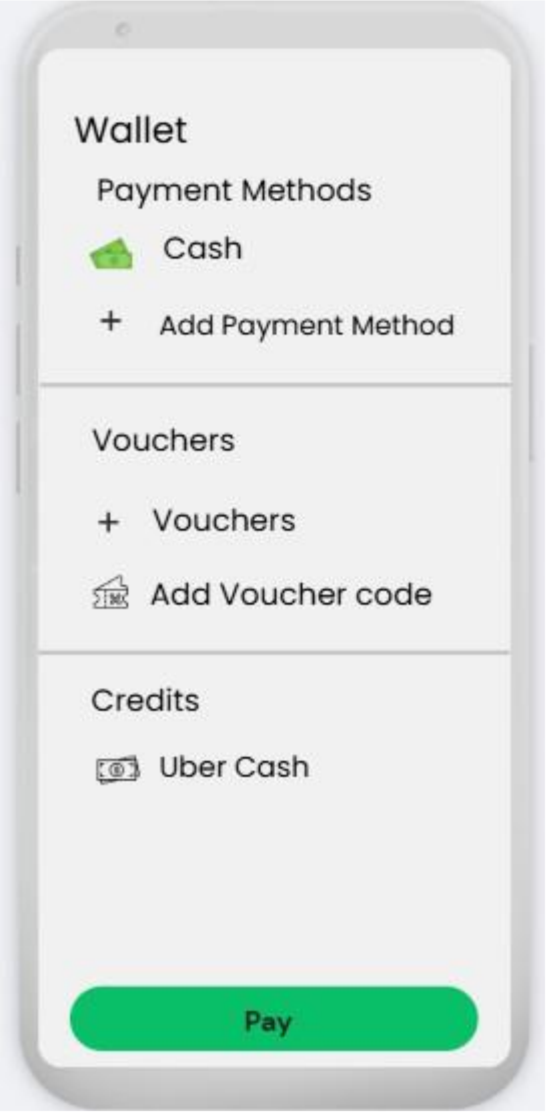
Pinto Thai restaurants

Send as a Gift

1 promotion available

Subtotal	$ 22.97
Uber One credits	$ 0.00
Delivery Fee	$ 6.56
Taxes & Other fees	$ 3.23
Total	$ 32.76

Split the bill? Share link with friends and pay only for your share



Wallet

Payment Methods

Cash

+ Add Payment Method

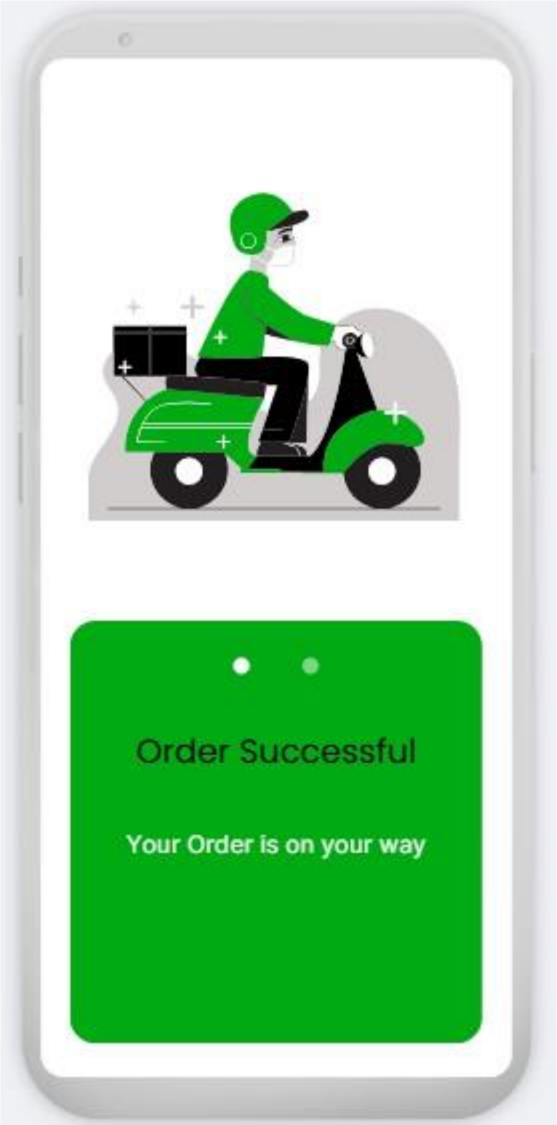
Vouchers

+ Vouchers

Add Voucher code

Credits

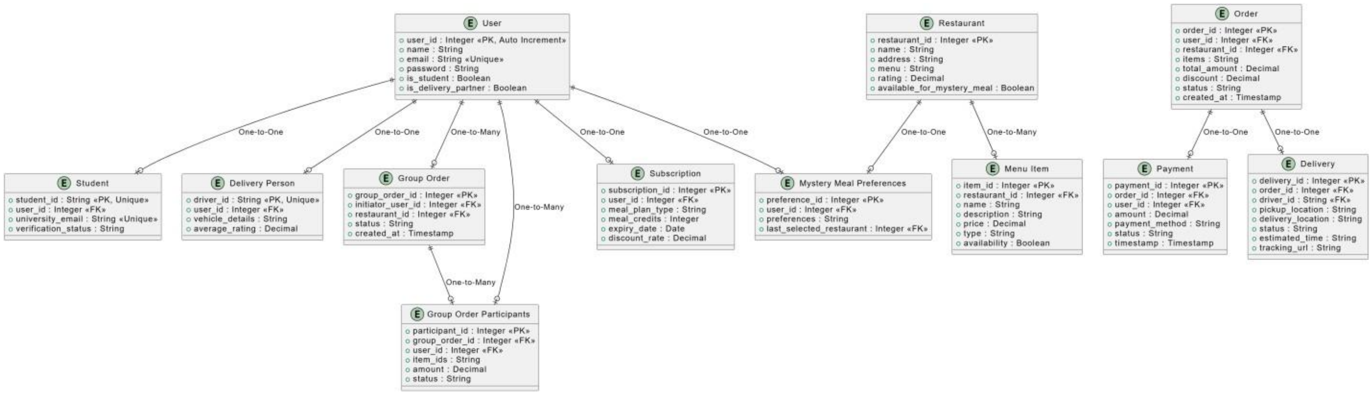
Uber Cash



Order Successful

Your Order is on your way

9. DATABASE DIAGRAM


E User
o user_id : Integer «PK, Auto Increment»
o name : String
o email : String «Unique»
o password : String
o is_student : Boolean
o is_delivery_partner : Boolean
E Restaurant
o restaurant_id : Integer «PK»
o name : String
o address : String
o menu : String
o rating : Decimal
o available_for_mystery_meal : Boolean
E Order
o order_id : Integer «PK»
o user_id : Integer «FK»
o restaurant_id : Integer «FK»
o items : String
o total_amount : Decimal
o discount : Decimal
o status : String
o created_at : Timestamp
One-to-One
One-to-One
One-to-Many
One-to-Many
One-to-One
One-to-One
One-to-One
One-to-Many
One-to-One
One-to-One
E Student
o student_id : String «PK, Unique»
o user_id : Integer «FK»
o university_email : String «Unique»
o verification_status : String
E Delivery Person
o driver_id : String «PK, Unique»
o user_id : Integer «FK»
o vehicle_details : String
o average_rating : Decimal
E Group Order
o group_order_id : Integer «PK»
o initiator_user_id : Integer «FK»
o restaurant_id : Integer «FK»
o status : String
o created_at : Timestamp
E Subscription
o subscription_id : Integer «PK»
o user_id : Integer «FK»
o meal_plan_type : String
o meal_credits : Integer
o expiry_date : Date
o discount_rate : Decimal
E Mystery Meal Preferences
o preference_id : Integer «PK»
o user_id : Integer «FK»
o preferences : String
o last_selected_restaurant : Integer «FK»
E Menu Item
o item_id : Integer «PK»
o restaurant_id : Integer «FK»
o name : String
o description : String
o price : Decimal
o type : String
o availability : Boolean
E Payment
o payment_id : Integer «PK»
o order_id : Integer «FK»
o user_id : Integer «FK»
o amount : Decimal
o payment_method : String
o status : String
o timestamp : Timestamp
E Delivery
o delivery_id : Integer «PK»
o order_id : Integer «FK»
o driver_id : String «FK»
o pickup_location : String
o delivery_location : String
o status : String
o estimated_time : String
o tracking_url : String
One-to-Many
E Group Order Participants
o participant_id : Integer «PK»
o group_order_id : Integer «FK»
o user_id : Integer «FK»
o item_ids : String
o amount : Decimal
o status : String

10. DATABASE DESCRIPTION

Entities

User

- **user_id**: Integer (Primary Key, Auto Increment)
- **name**: String
- **email**: String (Unique)
- **password**: String
- **is_student**: Boolean (Indicates if the user is a student)
- **is_delivery_partner**: Boolean (Indicates if the user is a delivery partner)

Student

- **student_id**: String (Primary Key, Unique)
- **user_id**: Integer (Foreign Key referencing User.user_id)
- **university_email**: String (Unique)
- **verification_status**: String (e.g., verified/pending)

Delivery Person

- **driver_id**: String (Primary Key, Unique)
- **user_id**: Integer (Foreign Key referencing User.user_id)
- **vehicle_details**: String
- **average_rating**: Decimal

Group Order

- **group_order_id**: Integer (Primary Key)
- **initiator_user_id**: Integer (Foreign Key referencing User.user_id)
- **restaurant_id**: Integer (Foreign Key referencing Restaurant.restaurant_id)
- **status**: String (e.g., active/completed)
- **created_at**: Timestamp

Group Order Participants

- **participant_id**: Integer (Primary Key)
- **group_order_id**: Integer (Foreign Key referencing Group Order.group_order_id)
- **user_id**: Integer (Foreign Key referencing User.user_id)
- **item_ids**: String (Comma-separated list of menu item IDs)
- **amount**: Decimal
- **status**: String

Subscription

- **subscription_id**: Integer (Primary Key)
- **user_id**: Integer (Foreign Key referencing User.user_id)
- **meal_plan_type**: String (e.g., basic, premium)
- **meal_credits**: Integer
- **expiry_date**: Date
- **discount_rate**: Decimal

Mystery Meal Preferences

- **preference_id**: Integer (Primary Key)
- **user_id**: Integer (Foreign Key referencing User.user_id)
- **preferences**: String (e.g., cuisine preferences, dietary restrictions)
- **last_selected_restaurant**: Integer (Foreign Key referencing Restaurant.restaurant_id)

Restaurant

- **restaurant_id**: Integer (Primary Key)
- **name**: String
- **address**: String
- **menu**: String (General description or categories)
- **rating**: Decimal
- **available_for_mystery_meal**: Boolean

Menu Item

- **item_id**: Integer (Primary Key)
- **restaurant_id**: Integer (Foreign Key referencing Restaurant.restaurant_id)
- **name**: String
- **description**: String
- **price**: Decimal
- **type**: String (e.g., appetizer, main course)
- **availability**: Boolean

Order

- **order_id**: Integer (Primary Key)
- **user_id**: Integer (Foreign Key referencing User.user_id)
- **restaurant_id**: Integer (Foreign Key referencing Restaurant.restaurant_id)
- **items**: String (Comma-separated list of menu item IDs)
- **total_amount**: Decimal
- **discount**: Decimal

- **status**: String (e.g., pending/completed)
- **created_at**: Timestamp

Payment

- **payment_id**: Integer (Primary Key)
- **order_id**: Integer (Foreign Key referencing Order.order_id)
- **user_id**: Integer (Foreign Key referencing User.user_id)
- **amount**: Decimal
- **payment_method**: String (e.g., credit card, wallet)
- **status**: String (e.g., success/failure)
- **timestamp**: Timestamp

Delivery

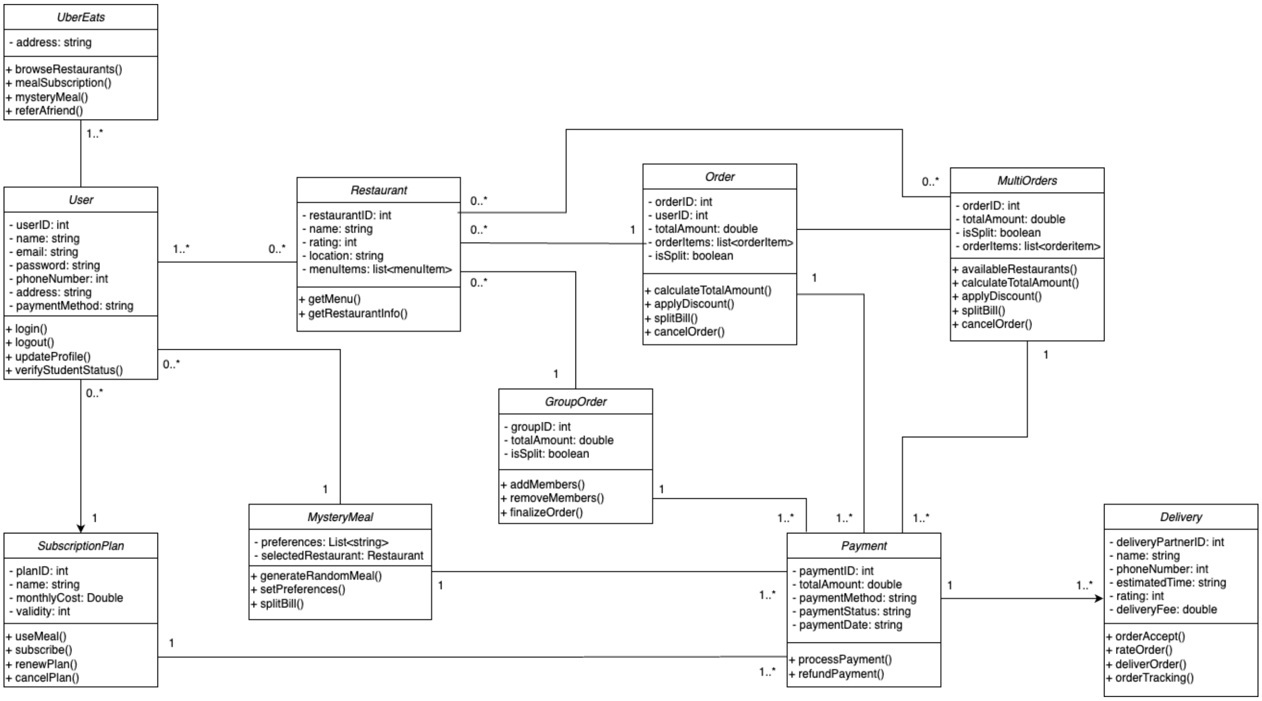
- **delivery_id**: Integer (Primary Key)
- **order_id**: Integer (Foreign Key referencing Order.order_id)
- **driver_id**: String (Foreign Key referencing Delivery Person.driver_id)
- **pickup_location**: String
- **delivery_location**: String
- **status**: String (e.g., in transit/delivered)
- **estimated_time**: String
- **tracking_url**: String

Relationships

1. **User to Student**: One-to-One (A user can optionally be a student).
2. **User to Delivery Person**: One-to-One (A user can optionally be a delivery partner).
3. **User to Group Order**: One-to-Many (A user can initiate multiple group orders).
4. **Group Order to Group Order Participants**: One-to-Many (A group order can have multiple participants).
5. **User to Subscription**: One-to-One (A user can have one subscription).
6. **User to Mystery Meal Preferences**: One-to-One (A user can define their meal preferences).
7. **User to Order**: One-to-Many (A user can place multiple orders).

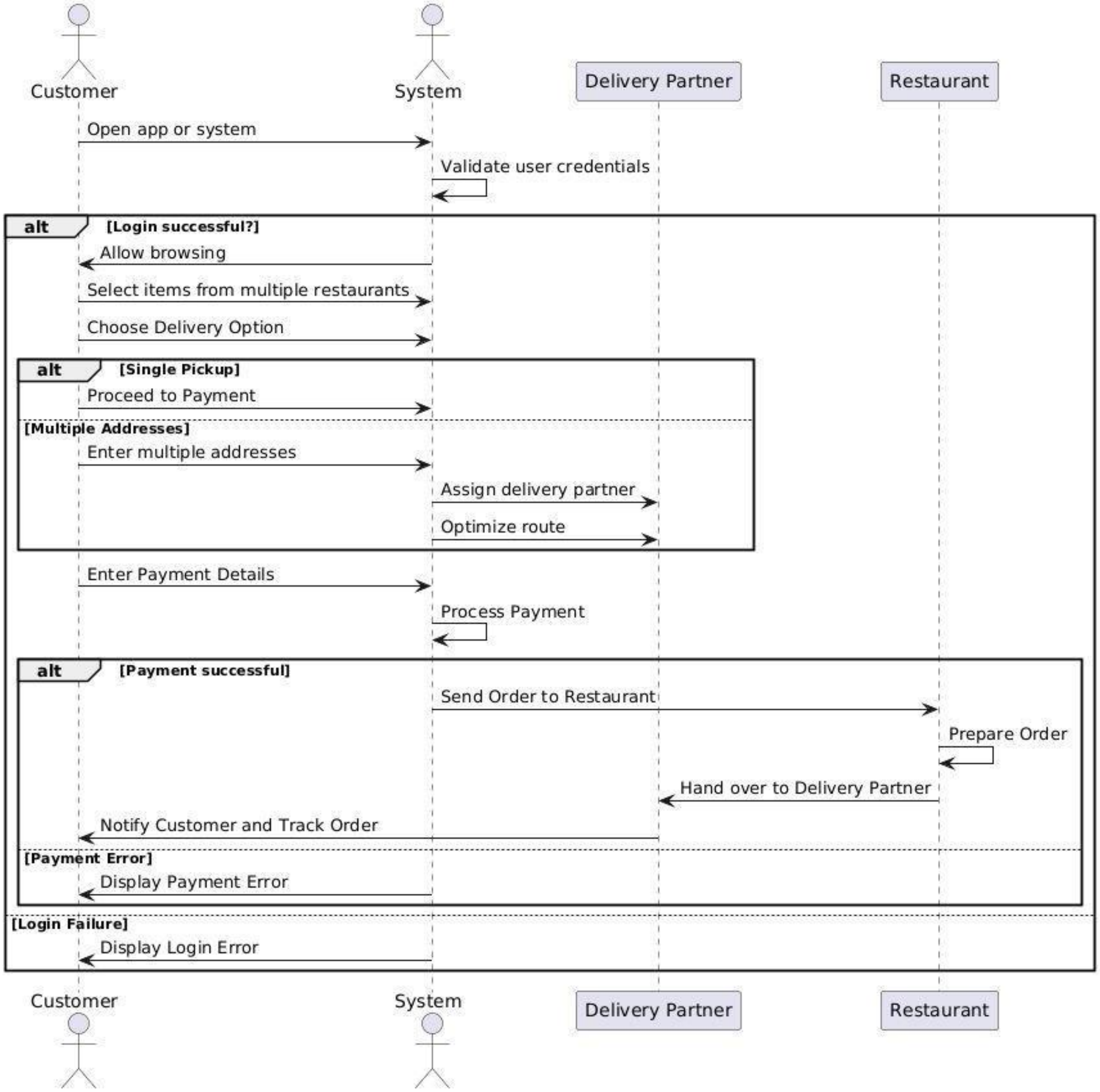
8. **Order to Payment**: One-to-One (Each order has a corresponding payment record).
9. **Order to Delivery**: One-to-One (Each order has a corresponding delivery record).
10. **Restaurant to Menu Item**: One-to-Many (A restaurant offers multiple menu items).
11. **Restaurant to Order**: One-to-Many (A restaurant can fulfill multiple orders).

11. COMPLETE CLASS DIAGRAM

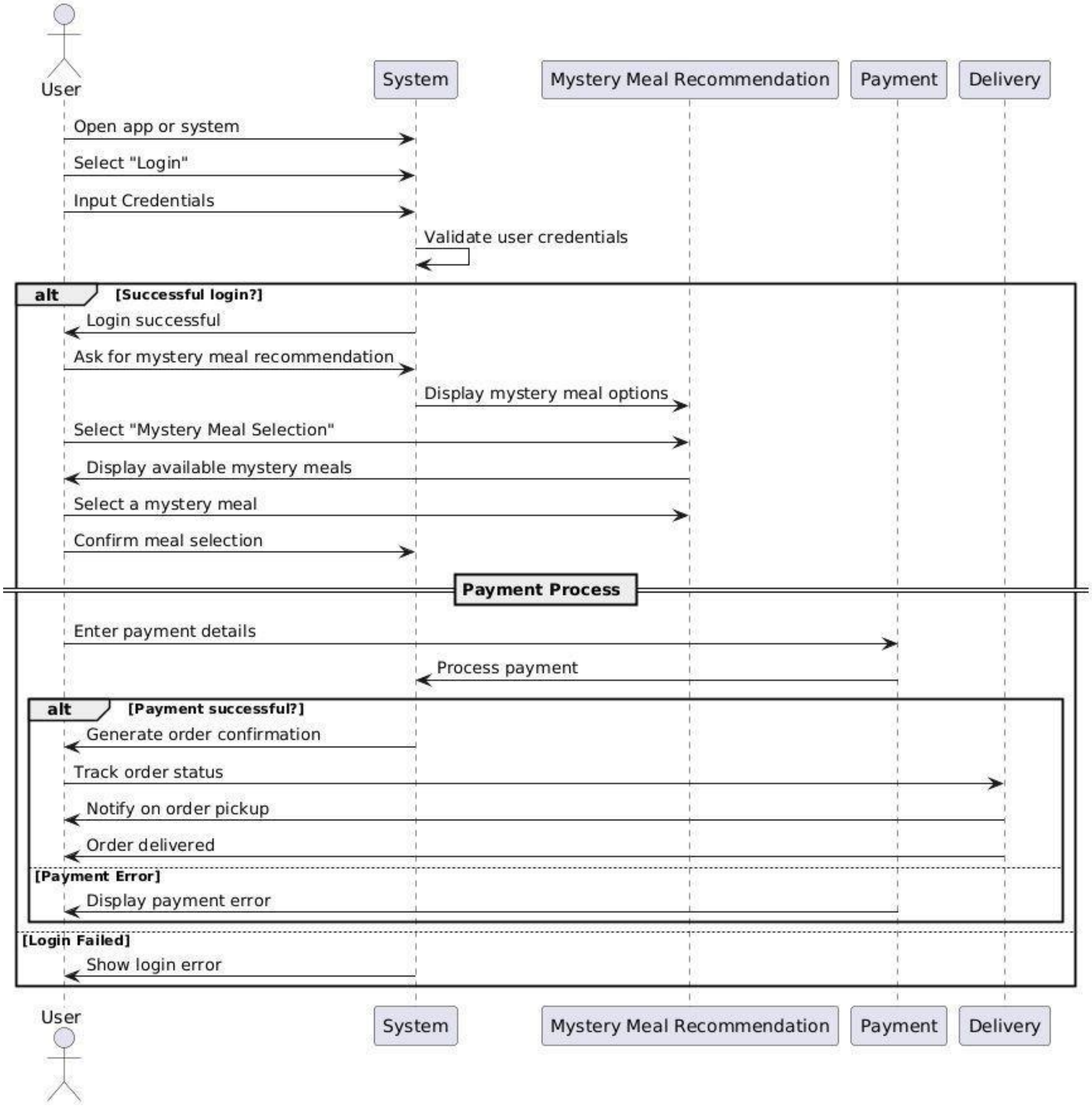


12. SEQUENCE DIAGRAMS

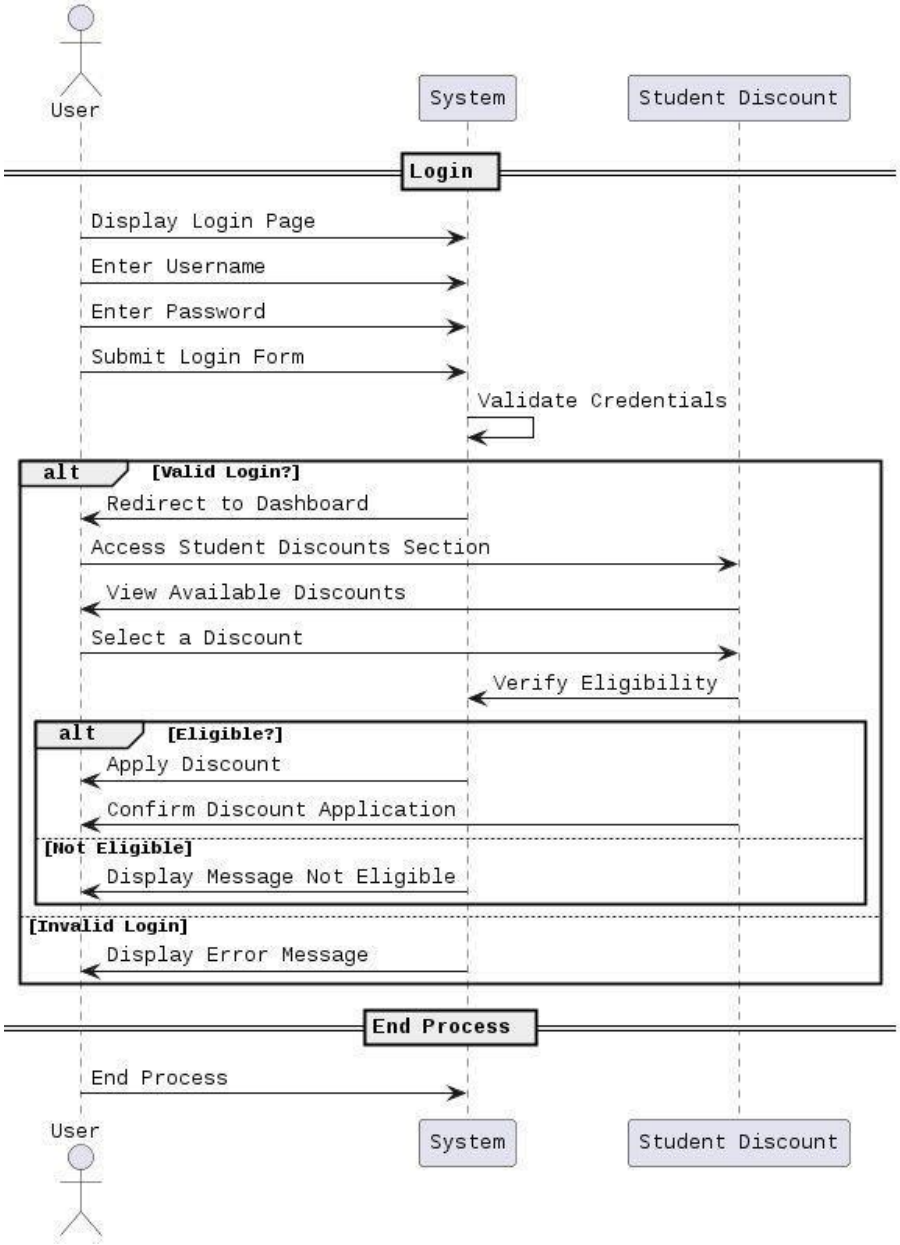
1. Group Order Delivery



2. Mystery meal recommendation



3. Student Discounts


User
System
Student Discount
Login
Display Login Page
Enter Username
Enter Password
Submit Login Form
Validate Credentials
alt
[Valid Login?]
Redirect to Dashboard
Access Student Discounts Section
View Available Discounts
Select a Discount
Verify Eligibility
alt
[Eligible?]
Apply Discount
Confirm Discount Application
[Not Eligible]
Display Message Not Eligible
[Invalid Login]
Display Error Message
End Process
End Process
User
System
Student Discount