Code for Integration Testing: Place Order Flow in Restaurant App

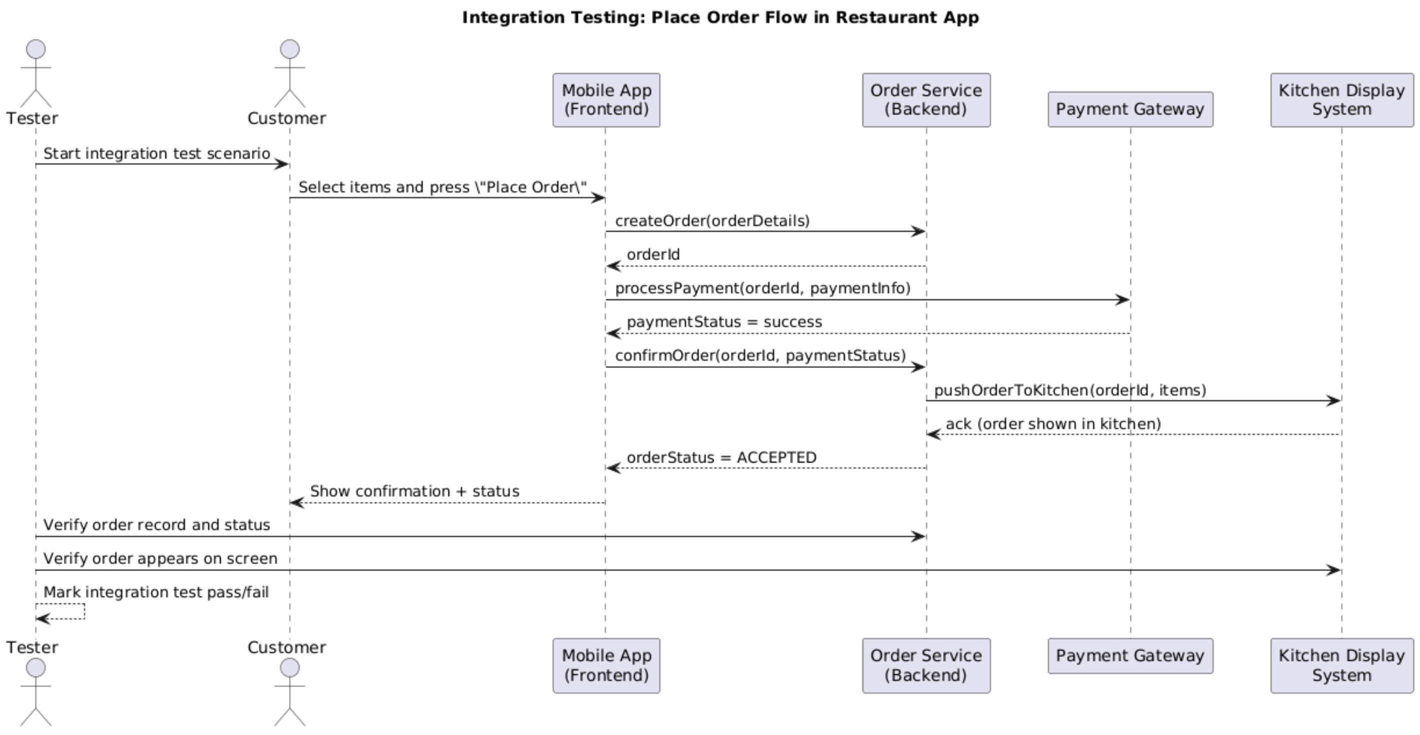
```
@startuml
skinparam defaultFontName Arial
skinparam SequenceParticipantFontSize 14
skinparam SequenceMessageFontSize 14
skinparam ActorFontSize 14
skinparam NoteFontSize 14
skinparam ParticipantPadding 20
```

These lines just format

Note (code start from this step)

```
title Integration Testing: Place Order Flow in Restaurant App
actor Tester
actor "Customer" as Customer
participant "Mobile App\n(Frontend)" as App
participant "Order Service\n(Backend)" as OS
participant "Payment Gateway" as PG
participant "Kitchen Display\nSystem" as KDS

Tester -> Customer: Start integration test scenario
Customer -> App: Select items and press \"Place Order\"
App -> OS: createOrder(orderDetails)
OS --> App: orderId
App -> PG: processPayment(orderId, paymentInfo)
PG --> App: paymentStatus = success
App -> OS: confirmOrder(orderId, paymentStatus)
OS -> KDS: pushOrderToKitchen(orderId, items)
KDS --> OS: ack (order shown in kitchen)
OS --> App: orderStatus = ACCEPTED
App --> Customer: Show confirmation + status
Tester -> OS: Verify order record and status
Tester -> KDS: Verify order appears on screen
Tester --> Tester: Mark integration test pass/fail
@enduml
```

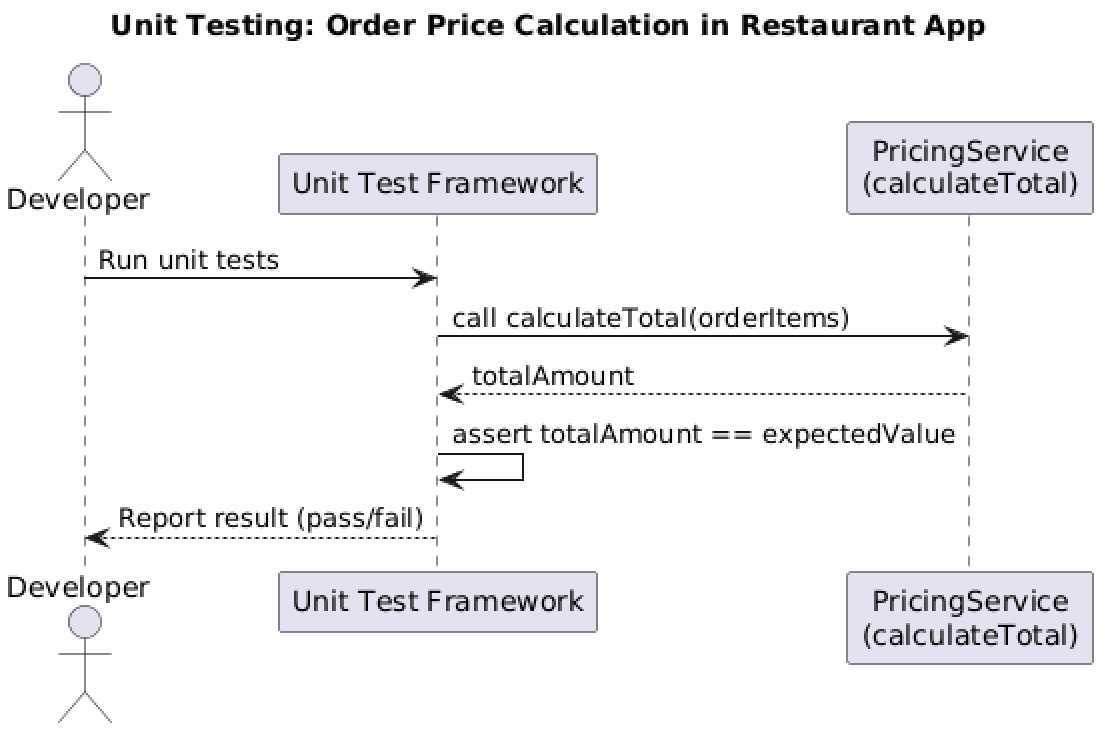
Result Code for Integration Testing: Place Order Flow in Restaurant App


Integration Testing: Place Order Flow in Restaurant App
Tester
Customer
Mobile App (Frontend)
Order Service (Backend)
Payment Gateway
Kitchen Display System
Start integration test scenario
Select items and press \"Place Order\"
createOrder(orderDetails)
orderId
processPayment(orderId, paymentInfo)
paymentStatus = success
confirmOrder(orderId, paymentStatus)
pushOrderToKitchen(orderId, items)
ack (order shown in kitchen)
orderStatus = ACCEPTED
Show confirmation + status
Verify order record and status
Verify order appears on screen
Mark integration test pass/fail

Code for Unit Testing: Order Price Calculation in Restaurant App

```
@startuml

title Unit Testing: Order Price Calculation in Restaurant App

actor Developer

participant "Unit Test Framework" as UTF

participant "PricingService\n(calculateTotal)" as PS

Developer -> UTF: Run unit tests

UTF -> PS: call calculateTotal(orderItems)

PS --> UTF: totalAmount

UTF -> UTF: assert totalAmount == expectedValue

UTF --> Developer: Report result (pass/fail)

@enduml
```

Result Code for Integration Testing: Place Order Flow in Restaurant App


Unit Testing: Order Price Calculation in Restaurant App
Developer
Unit Test Framework
PricingService (calculateTotal)
Run unit tests
call calculateTotal(orderItems)
totalAmount
assert totalAmount == expectedValue
Report result (pass/fail)

Code for Validation Testing: Requirement "Online Order and Payment"

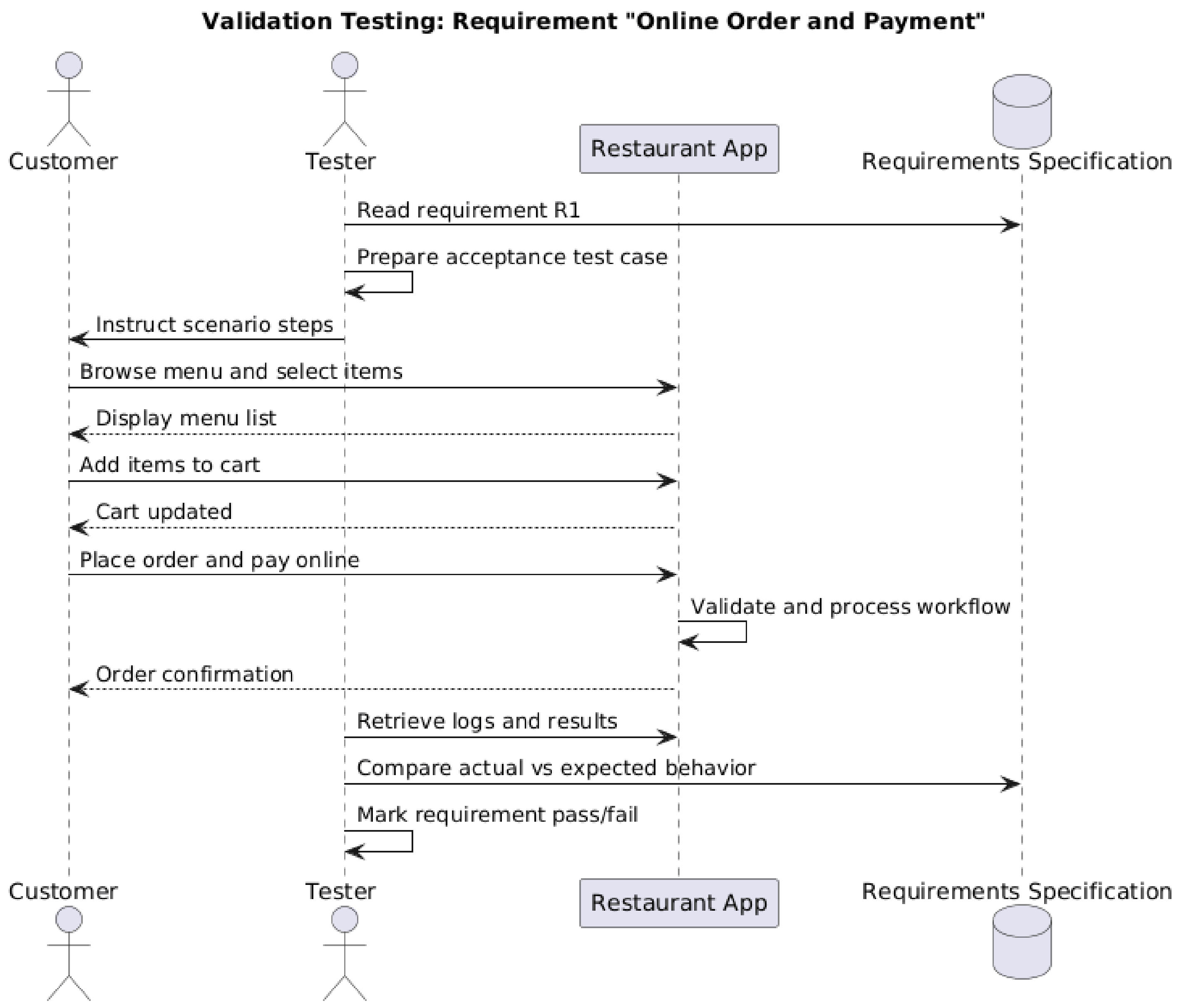
```
@startuml
skinparam defaultFontName Arial
skinparam SequenceParticipantFontSize 14
skinparam SequenceMessageFontSize 14
skinparam ActorFontSize 14
skinparam NoteFontSize 14
skinparam ParticipantPadding 20
```

These lines just format

Note (code start from this step)

```
title Validation Testing: Requirement "Online Order and Payment"
actor Customer
actor Tester
participant "Restaurant App" as RA
database "Requirements Specification" as Req
Tester -> Req: Read requirement R1
Tester -> Tester: Prepare acceptance test case
Tester -> Customer: Instruct scenario steps
Customer -> RA: Browse menu and select items
RA --> Customer: Display menu list
Customer -> RA: Add items to cart
RA --> Customer: Cart updated
Customer -> RA: Place order and pay online
RA -> RA: Validate and process workflow
RA --> Customer: Order confirmation
Tester -> RA: Retrieve logs and results
Tester -> Req: Compare actual vs expected behavior
Tester -> Tester: Mark requirement pass/fail
@enduml
```

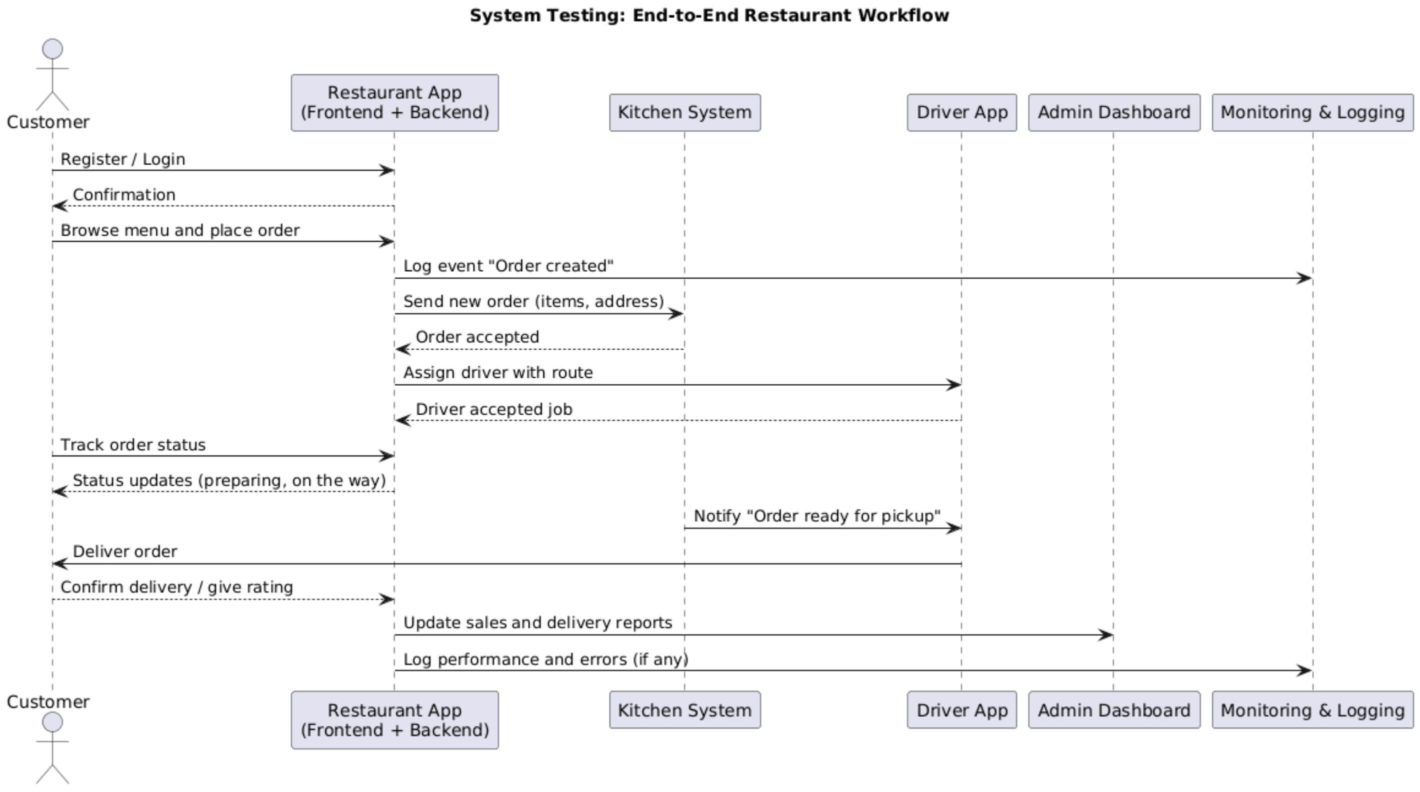
Result Code for Validation Testing: Requirement "Online Order and Payment"


Validation Testing: Requirement "Online Order and Payment"
Customer
Tester
Restaurant App
Requirements Specification
Read requirement R1
Prepare acceptance test case
Instruct scenario steps
Browse menu and select items
Display menu list
Add items to cart
Cart updated
Place order and pay online
Validate and process workflow
Order confirmation
Retrieve logs and results
Compare actual vs expected behavior
Mark requirement pass/fail

Code for System Testing: End-to-End Restaurant Workflow

```
@startuml

title System Testing: End-to-End Restaurant Workflow


actor "Customer" as C

participant "Restaurant App\n(Frontend + Backend)" as RA

participant "Kitchen System" as KS

participant "Driver App" as DA

participant "Admin Dashboard" as AD

participant "Monitoring & Logging" as MON


C -> RA: Register / Login

RA --> C: Confirmation


C -> RA: Browse menu and place order

RA -> MON: Log event "Order created"

RA -> KS: Send new order (items, address)

KS --> RA: Order accepted


RA -> DA: Assign driver with route

DA --> RA: Driver accepted job


C -> RA: Track order status

RA --> C: Status updates (preparing, on the way)


KS -> DA: Notify "Order ready for pickup"

DA -> C: Deliver order

C --> RA: Confirm delivery / give rating

RA -> AD: Update sales and delivery reports

RA -> MON: Log performance and errors (if any)


@enduml
```

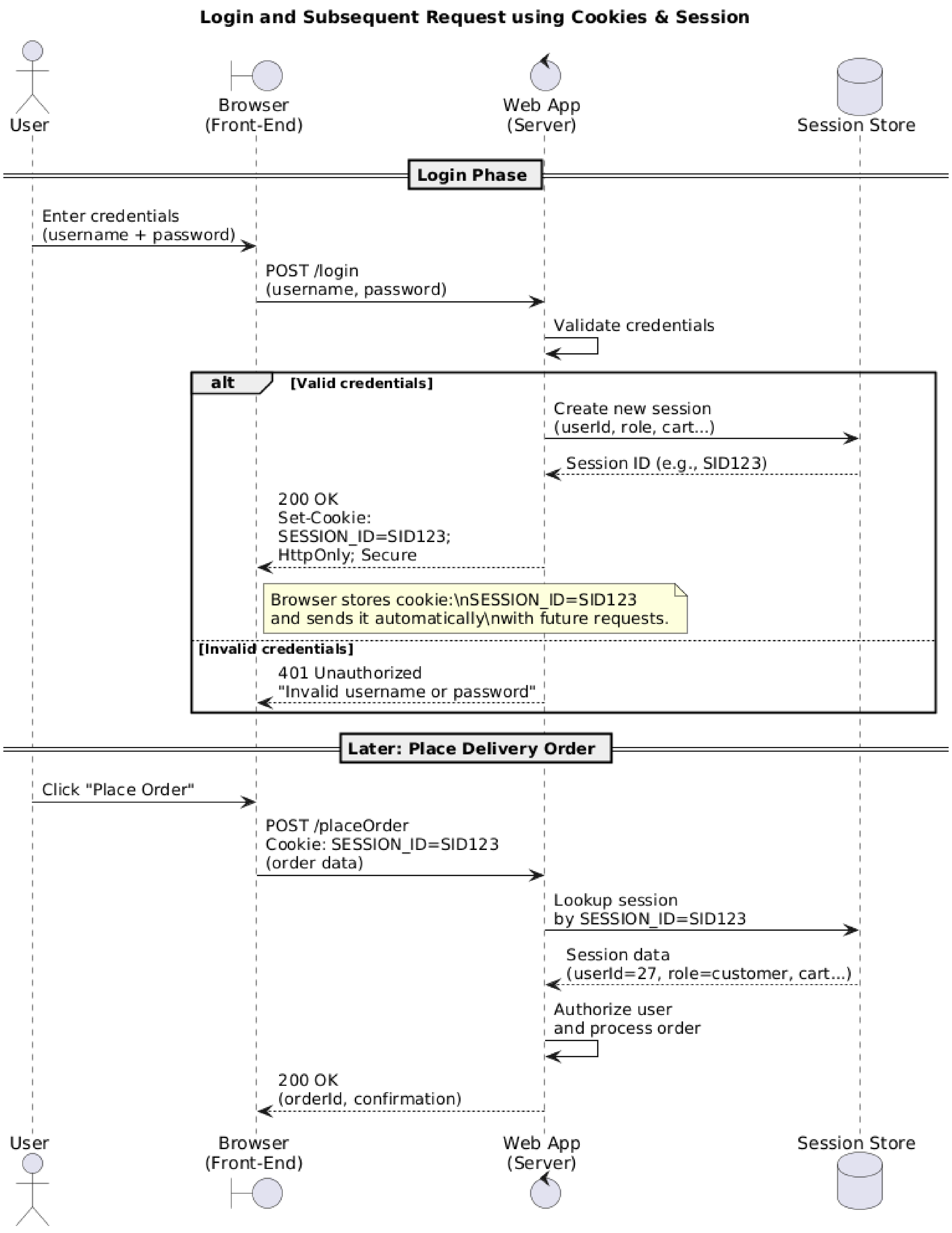
Result Code for System Testing: End-to-End Restaurant Workflow


System Testing: End-to-End Restaurant Workflow
Customer
Restaurant App (Frontend + Backend)
Kitchen System
Driver App
Admin Dashboard
Monitoring & Logging
Register / Login
Confirmation
Browse menu and place order
Log event "Order created"
Send new order (items, address)
Order accepted
Assign driver with route
Driver accepted job
Track order status
Status updates (preparing, on the way)
Notify "Order ready for pickup"
Deliver order
Confirm delivery / give rating
Update sales and delivery reports
Log performance and errors (if any)

Code for Login and Subsequent Request using Cookies & Session

```
@startuml
title Login and Subsequent Request using Cookies & Session
actor "User" as U
boundary "Browser\n(Front-End)" as B
control "Web App\n(Server)" as S
database "Session Store" as SS
== Login Phase ==
U -> B: Enter credentials\n(username + password)
B -> S: POST /login\n(username, password)
S -> S: Validate credentials
alt Valid credentials
  S -> SS: Create new session\n(userId, role, cart...)
  SS --> S: Session ID (e.g., SID123)
  S --> B: 200 OK\nSet-Cookie:\nSESSION_ID=SID123;\nHttpOnly; Secure
  note right of B
    Browser stores cookie:\nSESSION_ID=SID123
    and sends it automatically\nwith future requests.
  end note
else Invalid credentials
  S --> B: 401 Unauthorized\n"Invalid username or password"
end
== Later: Place Delivery Order ==
U -> B: Click "Place Order"
B -> S: POST /placeOrder\nCookie: SESSION_ID=SID123\n(order data)
S -> SS: Lookup session\nby SESSION_ID=SID123
SS --> S: Session data\n(userId=27, role=customer, cart...)
S -> S: Authorize user\nand process order
S --> B: 200 OK\n(orderId, confirmation)
@enduml
```

Result Code for Login and Subsequent Request using Cookies & Session


Login and Subsequent Request using Cookies & Session
User
Browser (Front-End)
Web App (Server)
Session Store
Login Phase
Enter credentials (username + password)
POST /login (username, password)
Validate credentials
alt [Valid credentials]
Create new session (userId, role, cart...)
Session ID (e.g., SID123)
200 OK Set-Cookie: SESSION_ID=SID123; HttpOnly; Secure
Browser stores cookie:\nSESSION_ID=SID123 and sends it automatically\nwith future requests.
[Invalid credentials]
401 Unauthorized "Invalid username or password"
Later: Place Delivery Order
Click "Place Order"
POST /placeOrder Cookie: SESSION_ID=SID123 (order data)
Lookup session by SESSION_ID=SID123
Session data (userId=27, role=customer, cart...)
Authorize user and process order
200 OK (orderId, confirmation)
User
Browser (Front-End)
Web App (Server)
Session Store