An architecture diagram for an Ola-like ride-hailing software would involve the following components:

1. User App:

- User registration and login

- Ride booking and scheduling

- Payment processing

- Rating and feedback system

2. Driver App:

- Driver registration and login

- Ride acceptance and navigation

- Earnings tracking and payment processing

3. Admin Panel:

- User and driver management

- Ride management and monitoring

- Payment and revenue tracking

- Analytics and reporting

4. Database:

- User and driver information storage

- Ride history and transaction storage

- Payment and revenue data storage

5. API Gateway:

- Handles requests from User and Driver apps

- Routes requests to microservices

6. Microservices:

- Authentication Service: handles user and driver authentication

- Ride Service: handles ride booking, scheduling, and management

- Payment Service: handles payment processing and revenue tracking

- Notification Service: handles push notifications and alerts

7. Third-Party Integrations:

- Payment gateways (e.g. Stripe, PayPal)

- Mapping and navigation services (e.g. Google Maps, Waze)

- Cloud services (e.g. AWS, Google Cloud)

Here's a high-level architecture diagram:

+---------------+

| User App |

+---------------+

|

|

v

+---------------+

| API Gateway |

+---------------+

|

|

v

+---------------+

| Microservices |

| (Authentication, |

| Ride, Payment, |

| Notification) |

+---------------+

|

|

v

+---------------+

| Database |

+---------------+

|

|

v

+---------------+

| Third-Party |

| Integrations |

+---------------+

|

|

v

+---------------+

| Admin Panel |

+---------------+

Note that this is a simplified diagram, and a real-world architecture would involve additional components and complexity.

**Title:** IRCTC Web Application Architecture using the MVC Pattern

**Diagram Structure:**

1. **IRCTC Browser (Top Element):**
   * Positioned at the top center.
   * Acts as the client interface where users search for trains, check availability, and book tickets.
2. **Controller (Left Box):**
   * Positioned below the IRCTC browser and to the left.
   * **Content:**
     + Processing HTTP requests from the IRCTC browser
     + Application-specific logic, such as handling search queries, login authentication, and session management.
     + Data validation to ensure the correctness of user input, such as dates, train numbers, and passenger details.
   * **Arrow directions:**
     + **From IRCTC Browser to Controller:** Represents an HTTP request sent from the browser, such as a ticket search or booking request.
     + **From Controller to Model:** Represents the update request for data processing, such as querying the database for train availability or confirming a booking.
     + **From Controller to View:** Represents form generation and handling user events, like displaying search results or booking confirmations.
3. **Model (Bottom Box):**
   * Positioned directly below the IRCTC browser.
   * **Content:**
     + Business logic, such as rules for seat allocation, fare calculations, and cancellation policies.
     + Database access for retrieving train schedules, seat availability, user accounts, booking history, etc.
   * **Arrow directions:**
     + **From Model to Controller:** Represents change notification after data is processed, like updating the booking status.
     + **From Model to View:** Represents refresh request, such as sending updated data (like available seats) to the view.
4. **View (Right Box):**
   * Positioned below the IRCTC browser and to the right.
   * **Content:**
     + Dynamic page generation for displaying search results, booking details, and payment confirmations.
     + Forms management, such as handling input forms for passenger details, payment methods, and ticket confirmation.
   * **Arrow directions:**
     + **From View to IRCTC Browser:** Represents the response with dynamically generated pages, such as the ticket booking confirmation page or search results.
     + **From View to Model:** Represents notification for any data refresh requirements, like updating seat availability after a booking