Structural Design pattern Assignments:

**1️ Adapter Pattern – Payment Gateway Integration**

**Scenario:**  
An e-commerce website supports payments via **PayPal** but wants to integrate **Stripe** and **Google Pay** without changing the existing code.

**Solution:**  
Use an **Adapter Pattern** to create a uniform interface for handling all payment gateways.

🔹 **Implementation:**

* Define an interface IPaymentGateway with a ProcessPayment(amount) method.
* Implement existing PayPalGateway.
* Create StripeAdapter and GooglePayAdapter that **convert their methods** into the expected interface.
* Now, the system can **use any payment gateway seamlessly**.

**2️ Bridge Pattern – Cross-Platform UI Rendering** 🎨

Scenario:

A mobile app must support Android, iOS, and Web with two themes (Light & Dark). The UI logic should be independent of platforms.

Solution:

Use a Bridge Pattern to separate the UI logic from platform-specific implementations.

🔹 Implementation:

Create an IPlatform interface (Android, iOS, Web).

Implement LightTheme and DarkTheme as separate classes.

The AppUI class references both theme and platform, allowing easy scalability.

**3️ Composite Pattern – File System Hierarchy 📂**

**Scenario:**  
A **file manager** must handle files and folders where folders can contain files and other subfolders.

**Solution:**  
Use a **Composite Pattern** to treat individual files and folders **uniformly**.

🔹 **Implementation:**

* Create an interface IFileComponent with DisplayDetails().
* Implement File (leaf) and Folder (composite).
* The Folder class can **contain multiple IFileComponent objects**, enabling **recursive traversal**.

📌 **Example:** Windows File Explorer, Linux File System.

**4️ Decorator Pattern – Custom Coffee Orders ☕**

**Scenario:**  
A coffee shop allows customers to add **extra milk, sugar, whipped cream, and flavors** dynamically.

**Solution:**  
Use a **Decorator Pattern** to wrap base coffee objects with additional features.

🔹 **Implementation:**

* Create ICoffee interface with GetCost() and GetDescription().
* Implement BasicCoffee.
* Decorate it with **MilkDecorator, SugarDecorator, and WhippedCreamDecorator** dynamically.

📌 **Example:** Starbucks, McDonald's custom menu.

**5️ Facade Pattern – Hotel Booking System 🏨**

**Scenario:**  
A travel website integrates **hotel reservations, flight booking, and cab services** into a single interface.

**Solution:**  
Use a **Facade Pattern** to simplify interactions with multiple subsystems.

🔹 **Implementation:**

* Implement subsystems: HotelBooking, FlightBooking, CabBooking.
* Create a TravelFacade class with a BookCompleteTrip() method that **internally handles all bookings**.

📌 **Example:** MakeMyTrip, Expedia, Booking.com.

**6️ Flyweight Pattern – Large-Scale Game Development 🎮**

**Scenario:**  
A **2D game** (e.g., Super Mario) contains thousands of **trees, bricks, and coins**, which consume memory if stored separately.

**Solution:**  
Use the **Flyweight Pattern** to **reuse existing objects** instead of creating new ones.

🔹 **Implementation:**

* Create ITree interface.
* Implement a TreeFactory to manage **a pool of tree objects** and reuse them.
* Instead of creating a new Tree object each time, the game engine **retrieves it from the factory**, saving memory.

📌 **Example:** Unity Game Engine, GTA games.

**7️ Proxy Pattern – Secure Database Access 🔐**

**Scenario:**  
A company has a **centralized database** but needs to restrict access based on **user roles** (Admin vs. Employee).

**Solution:**  
Use a **Proxy Pattern** to **control access** before making direct database calls.

🔹 **Implementation:**

* Create IDatabaseAccess interface with a GetData() method.
* Implement RealDatabaseAccess.
* Create a DatabaseProxy that checks **user permissions** before allowing access.

📌 **Example:** Banking portals, HR management software.