// App Architecture Assignment

i. Creational Design Patterns

1. Factory Method: is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.
2. Abstract Factory: is a creational design pattern that lets you produce families of related objects without specifying their concrete classes.
3. Builder: is a creational design pattern that lets you construct complex objects step by step. The pattern allows you to produce different types and representations of an object using the same construction code.
4. Prototype: is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.
5. Singleton: is a creational design pattern that lets you ensure that a class has only one instance, while providing a global access point to this instance.

ii. Structural Design Patterns

1. Adapter: Allows objects with compatible interfaces to collaborate.
2. Bridge: lets you split a large class or a set of closely related classes into two separate hierarchies – abstraction and implementation – which can be developed independently of each other.
3. Composite: lets you compose objects into tree structures and then work with these structures as if they were individual objects.
4. Decorator: Lets you attach new behaviors to objects by placing these objects that contain the behaviors.
5. Façade: Provides a simplified interface to a library, a framework, or any other complex set of classes.

6. Flyweight: Lets you fir more objects into the available amount of RAM by sharing common parts of state between multiple objects instead of keeping all of the data in each object.
7. Proxy: Lets you provide a substitute or placeholder for another object. A proxy controls access to the original object. Allowing you to perform something either before or after the request gets through to the original object.

iii. Behavioral Design Patterns

1. Chain of Responsibility: lets you pass request along a chain of handlers. Upon receiving a request, each handler decides either to process the request or to pass it to the next handler in the chain.
2. Observer: Lets you define a subscription mechanism to notify multiple objects about any events that happen to the object they are observing.
3. State: Lets an object alter its behavior when its internal state changes. It appears as if the object changed its class.