Participating team members: Rohan Kansal, Ishaan Reddy, Adinay Khatri

SOLID
The diagram implements **Liskov Substitution Principle** as the specialized task classes RecurringTask and HighPriorityTask may substitute the Task class they inherit within the system without compromising functionality or creating an error. We also implement **Interface Segregation Principle** with the team members, as we use a general TeamMember class and then have 2 classes, BackendDev and FrontendDev that inherit that and then implement a special interface that each have their own specific method relevant to each subclass, the Backend interface providing apis() functionality to the BackendDev and the divs() functionality provided to the FrontendDev to segregate responsibilities for different people by different interfaces. **Dependency Inversion Principle** is also followed as the task is structured to depend on the generalized TaskStatus interface rather than on specific implementations like CompletedStatus and PendingStatus, thereby enhancing both flexibility and extensibility.

GRASP
There are multiple instances of **inheritance**, one example including the classes RecurringTask and HighPriorityTask that extend the Task class, thereby facilitating polymorphism. Low Coupling is accomplished as the design maintains loose dependencies between Task and TaskStatus, ensuring that modifications in status management do not adversely affect the task system. Interfaces for Flexible Status Handling is implemented through the adoption of the TaskStatus interface which enables a flexible and extensible approach to task status management, while decoupling the Task class from specific implementations.