OOAD PROJECT 7 - GARUDA

Project Summary

Title: Garuda – Kanban-Style Project Management System

Team Members:

- Abhinav Venkatesh
- Justin Murillo
- Siddharth Srinivasan

Work Done:

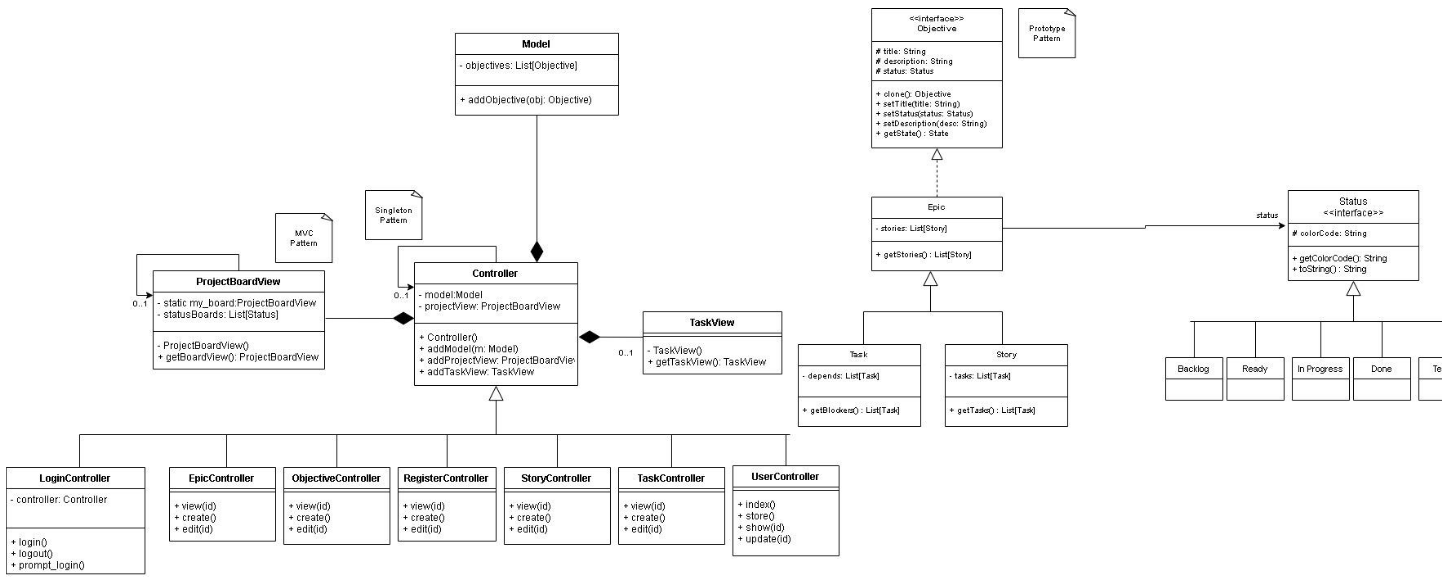
- **Abhinav**: DB Setup & Migration, ORM, View and Controllers for Login, User
- **Siddharth**: Board/Task View Logic, Routes and Controllers for Objective, Epic, Story, Task (in adherence with design patterns)
- **Justin**: UI/UX, Jinja Templating

Final System State:

- Classes such as User, Login, Epic, Story, Task etc. are linked with the data model using Object-Relational Mapping.
- Features implemented:
  - Addition of objectives to the Kanban board
  - Kanban Board View, Task View
  - User Registration/Login
- Features not implemented: Implementation
  - Modify/Delete views
- Patterns implemented:
  - **ORM:** As described above
  - **Prototype Pattern:** Leveraged the deepcopy submodule of Python within the __*copy*__ dunder so that duplicate attributes of an Objective can be reused while cloning an Epic or Story.

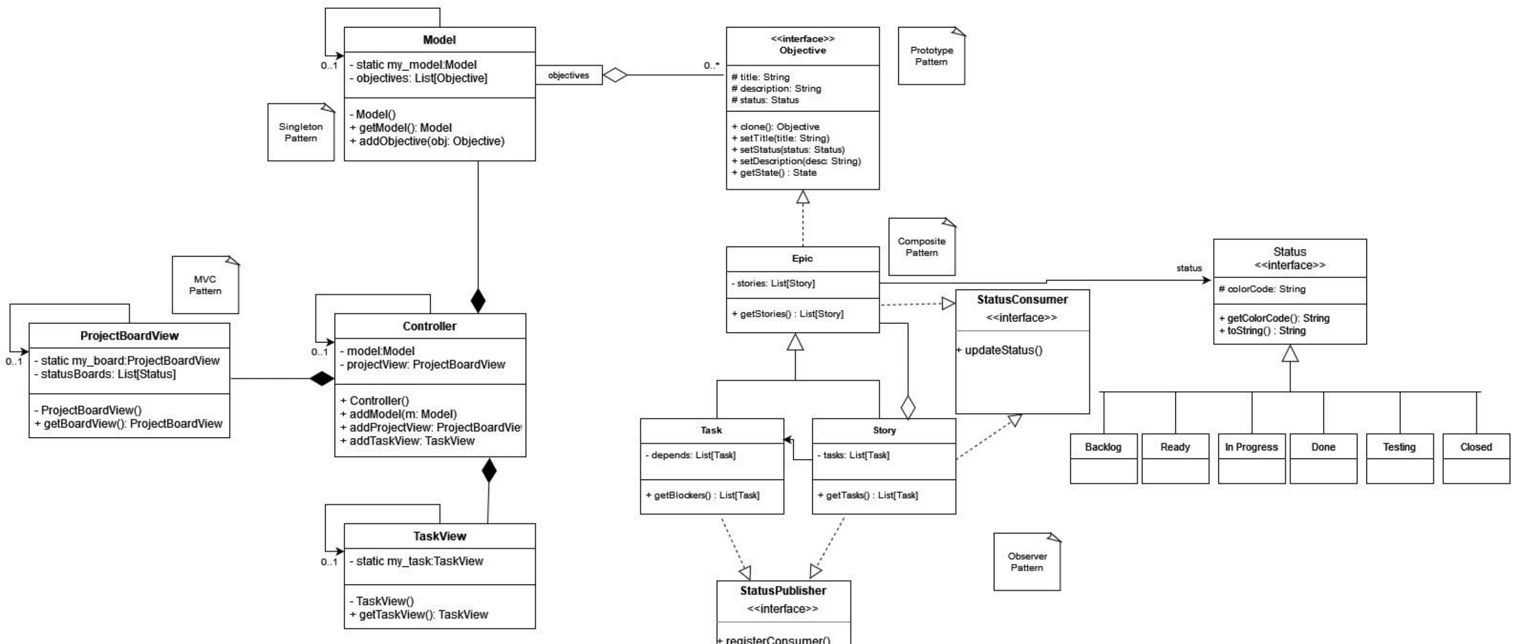
- **Singleton Pattern:** Every Controller must be instantiated only once for reuse with corresponding routes and views. We leverage the __*new*__ dunder to implement the pattern during object creation.
- **MVC Pattern:** In an extension to the typical design of the MVC pattern, we ensure the control flow includes routes for the mapping between the API endpoints and method handlers, and service logic which accommodates any specialized compute/processing apart from business logic handled on the controller side. The new sequence of control would now be:
    - route -> controller -> model -> view

(Final) Class Diagram:



https://i.imgur.com/TC663va.jpg

(Initial) Project 5 Class Diagram:



Key differences between Project 5 and Project 7 deliverables are:

- Introduction of ORM pattern
- Database migration primitives
- Absence of Composite Pattern
- Absence of Observer Pattern
- Refactoring of MVC pattern
    - Several controllers bearing common functionality
    - Each View and Controller class can be effectively treated as a Singleton

Third-Party vs Original Code Statement:

The Project Codebase is entirely original, excluding functionality provided by external tools and third-party frameworks:

- Flask Micro Web Framework: https://flask.palletsprojects.com/en/2.1.x/
- ORM: https://flask-sqlalchemy.palletsprojects.com/en/2.x/
- DB Driver: https://pypi.org/project/PyMySQL/
- DB Migrations: https://flask-migrate.readthedocs.io/en/latest/

OOAD Process Elements:

- Incorporating ORM pattern to interact with the data model more effectively
- Major difficulty faced retaining Composite Pattern after incorporating ORM, so reverted the same to a more traditional inheritance hierarchy between Epic, Story and Task classes
- Observer Pattern has been discarded since we estimate more effort will be required to notify users and Objective class-hierarchies of state changes at this point in time.
- As discussed before, we separately maintain the routes away from their core functionality dictated by the controllers, and we bind both the routes and controller methods using a ‘Blueprint Mapping’.

Code:

Codebase: https://github.com/Garuda-PMS/garuda_dev
Documentation: https://github.com/Garuda-PMS/garuda_docs