OOAD PROJECT 6 - GARUDA

Project Summary

Title: Garuda – Kanban-Style Project Management System

Team Members:

- Abhinav Venkatesh
- Justin Murillo
- Siddharth Srinivasan

Work Done:

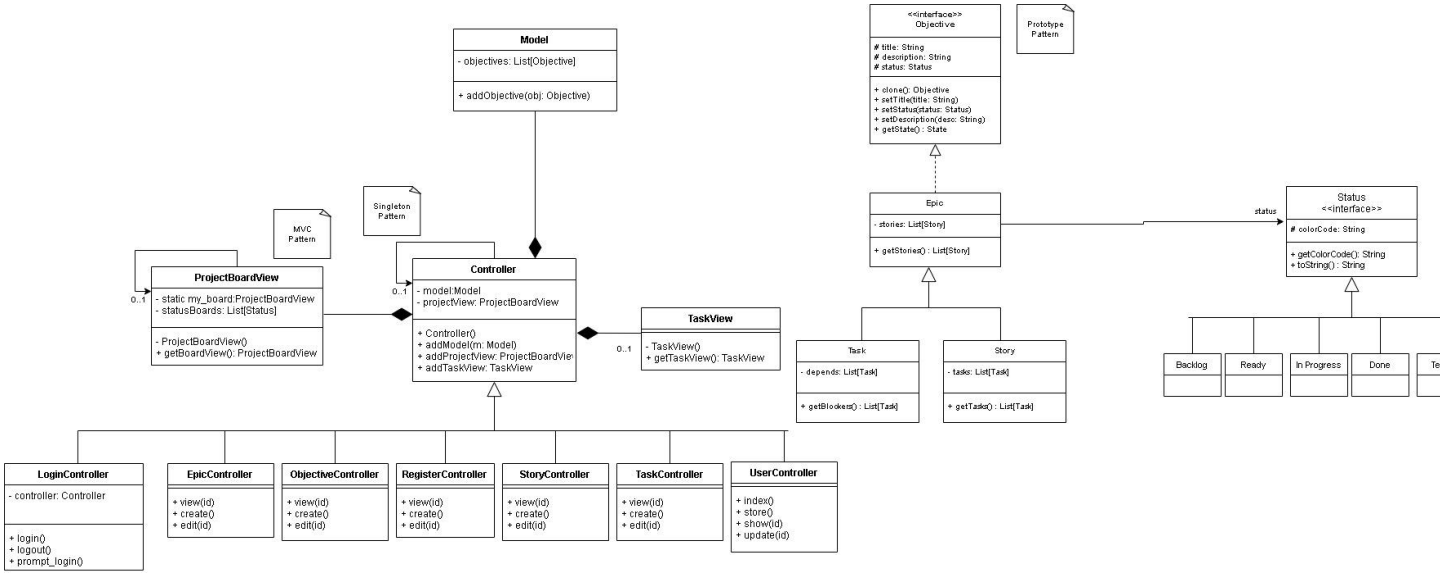
- **Abhinav**: DB Setup & Migration, ORM, View and Controllers for Login, User
- **Siddharth**: Board/Task View Logic, Routes and Controllers for Objective, Epic, Story, Task (in adherence with design patterns)
- **Justin**: UI/UX, Jinja Templating

Changes/Issues Encountered:

- Classes such as User, Login, Epic, Story, Task etc. are now linked with the data model using Object-Relational Mapping.
- We can no longer accommodate Composite Pattern amongst Objective, Epic, Story and Task, since the entity bindings with SQLAlchemy ORM permit only those relationships that are permissible in a Database Model as opposed to an OO hierarchy.
- 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 -> service-> model

(Revised) Class Diagram:



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

Plan for Next Iteration:

- The functionality currently accommodates only Task creation and Status changes at the moment. We will have to extend the same for Epic and Story as well.
- UI/UX must represent the reporter and assignee for each Task/Story/Epic.
- Functionality for modifying and Deleting Epic/Story/Tasks must also be provided.
- To track automatic status updates on epic/story depending on task completion, we plan to implement an observer pattern that publishes state changes up the objective hierarchy once all leaf elements reside in the same status.