OOAD PROJECT 5 - GARUDA

Project Summary

Title: Garuda – Kanban-Style Project Management System

Team Members:

- Abhinav Venkatesh
- Justin Murillo
- Siddharth Srinivasan

Overview:

- The objective of this project is to make a web-based job board, such that users can manage their tasks across different task boards tailored for a given task state.
- Every user is categorized as either a task reporter or assignee.
    - Task Reporter is responsible for creation and delegation of the epic, stories and tasks at hand.
    - An assignee carries out the tasks in his name, and is responsible for changing task status accordingly.
- Task boards involved in a typical software management lifecycle could be categorized under topics such as Backlog, To-Do, In Progress, Done, Testing and Closed. Every Epic/Story/Task is associated with each of these topics, better known as “Task Status”.
- Once the user visits our website, following is the series of views that he is expected to visit:
    - The login view which demands a valid username/password combination for progressing to the next view
    - The project board view which elaborates on the tasks associated with each status topic
    - The task view which expands on the details and status of a particular task. This section is where edits can be made to an individual task (or story/epic for that matter).

Project Requirements and Responsibilities:

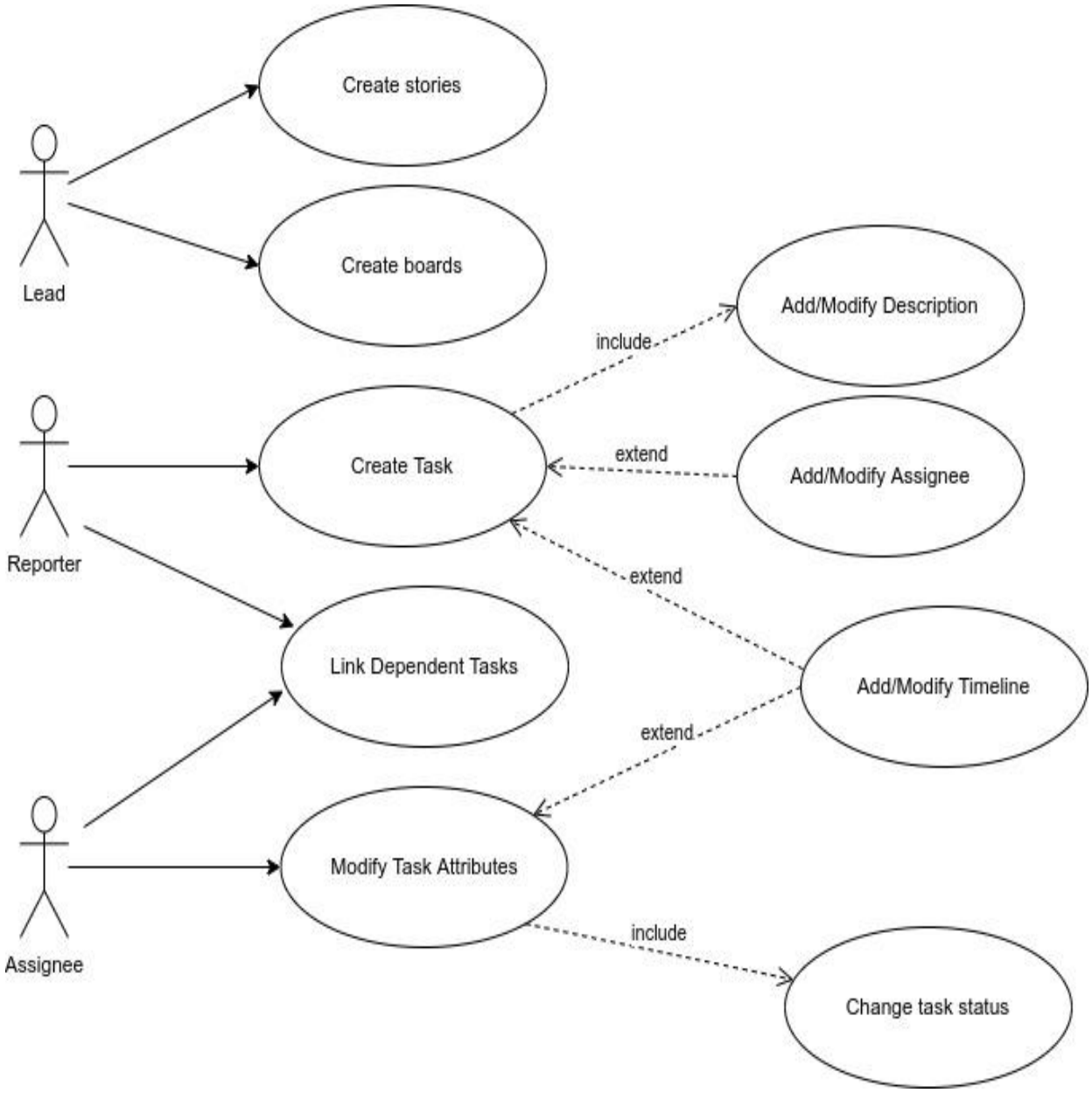
Functional requirements:

- Allow users to sign up to the application
- Provide user login
- Allow user to create tasks, stories and epics
- Allow user to add descriptions to the tasks, stories and epics created
- Associate tasks with users
- Each task is associated with a Reporter and an Assignee
- Each task can be assigned the following states by the user - Backlog, ready, in progress and done
- Each Task is associated with a timeline
- Define dependencies between tasks (optional)
- UI to display the board

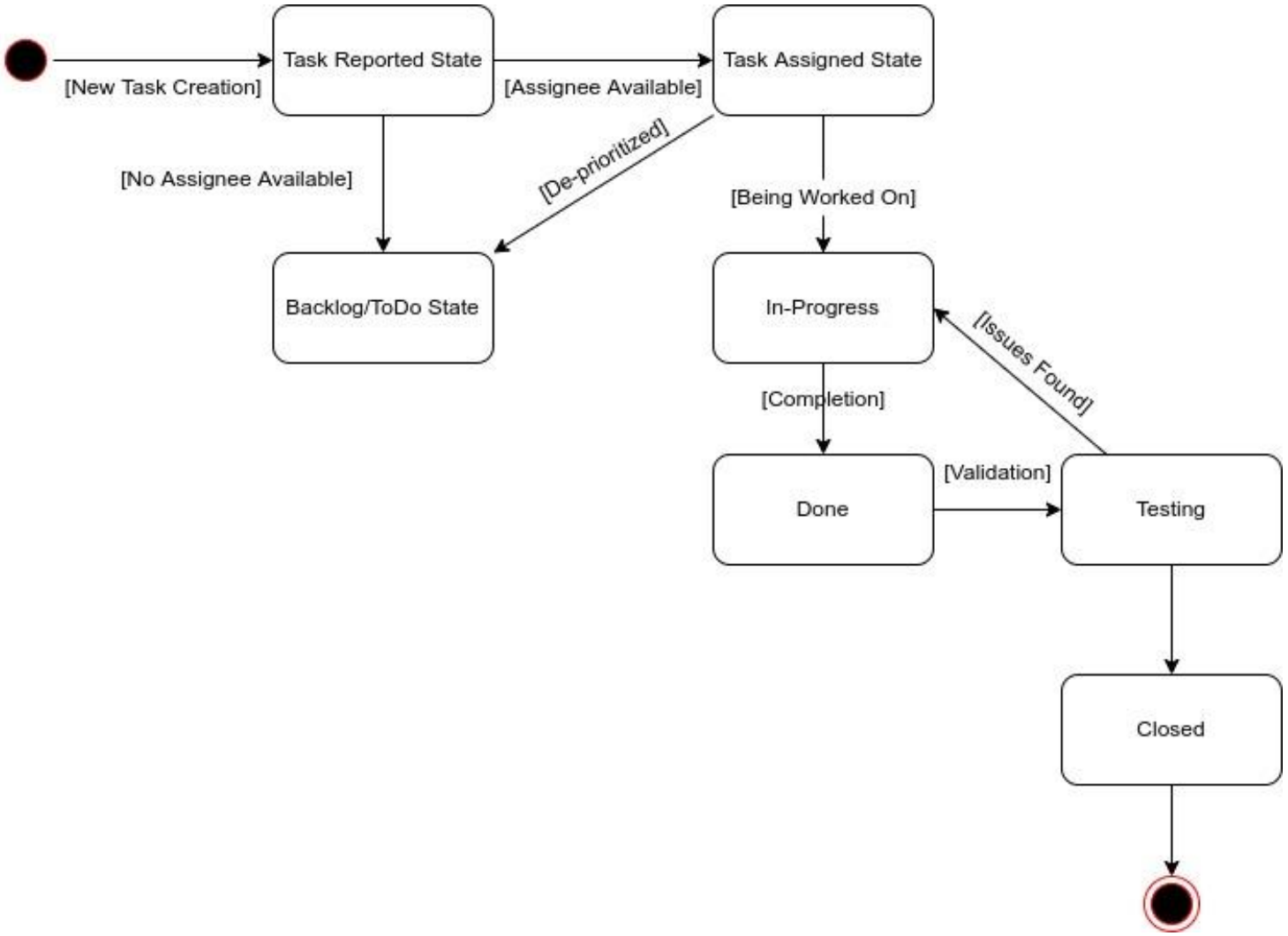
Non-Functional requirements:

- Availability (System should be available whenever user wishes to view the task)
- Consistency/Reliability (Tasks shown in the board must contend with the details entered during task creation)
- Performant (The system must be able to handle as many tasks as was keyed in by the task creator)

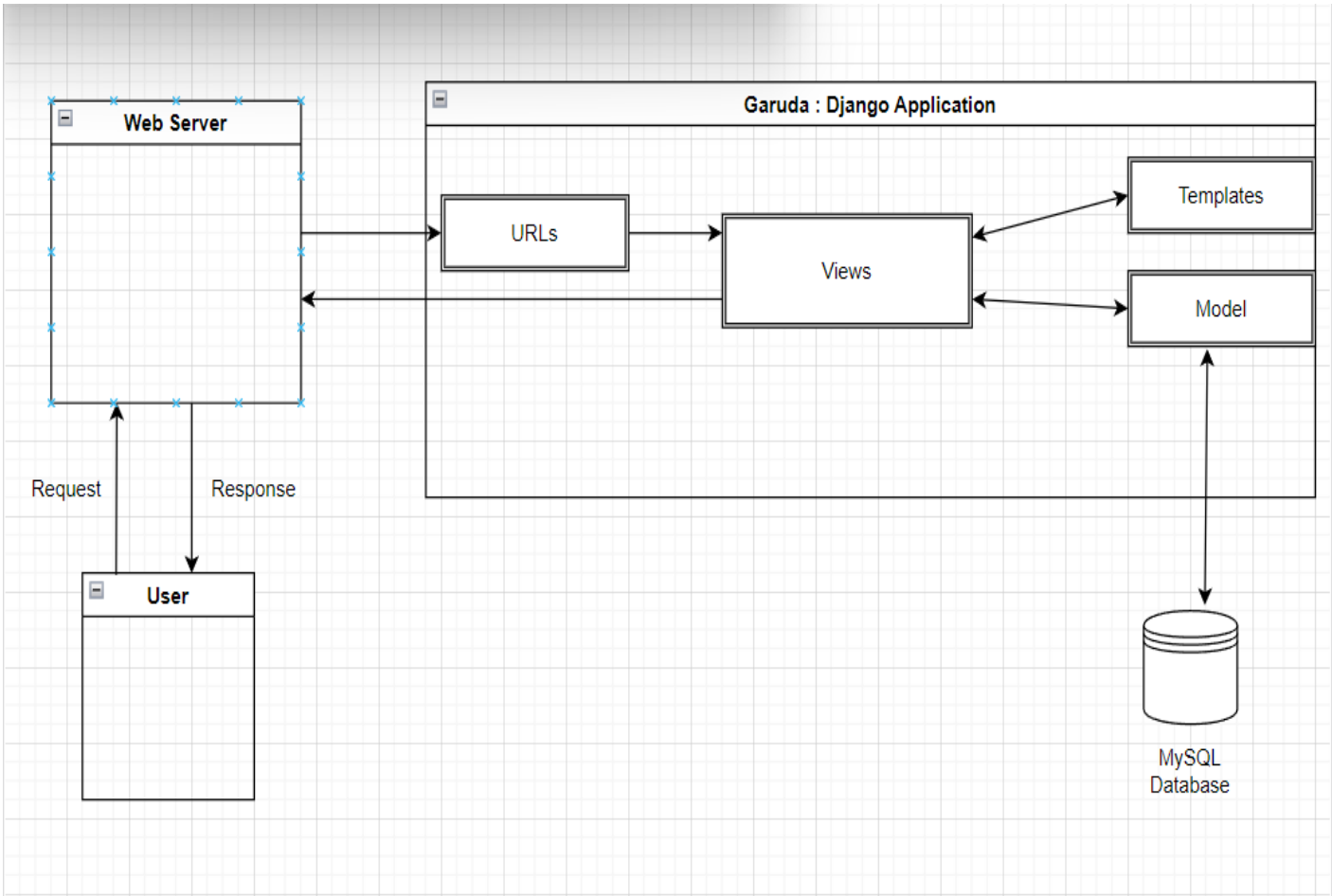
Use-Case Diagram



State Diagram

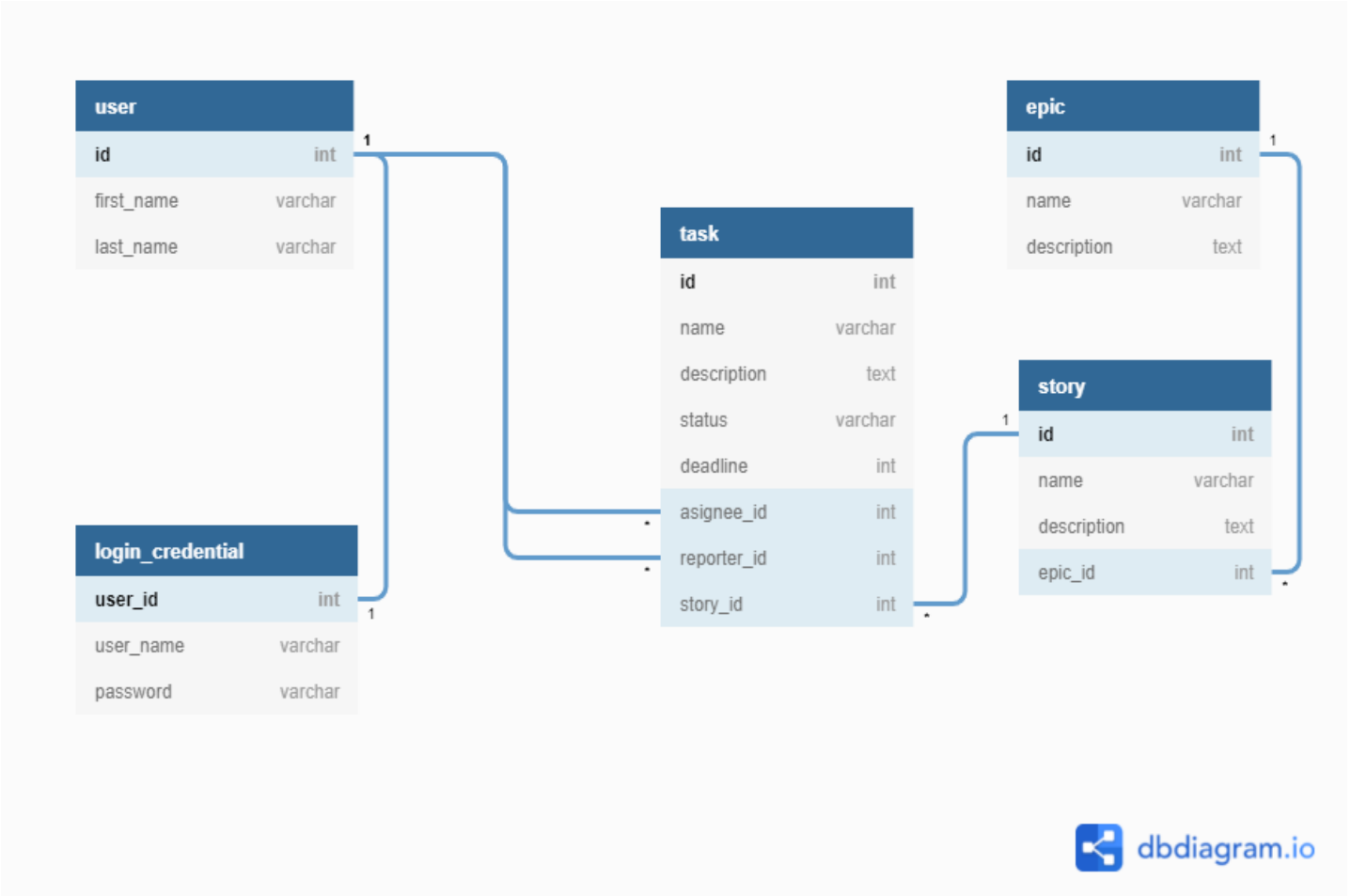

Task Reported State
[New Task Creation]
[Assignee Available]
Task Assigned State
[No Assignee Available]
[De-prioritized]
[Being Worked On]
Backlog/ToDo State
In-Progress
[Issues Found]
[Completion]
Done
[Validation]
Testing
Closed

Architecture Diagram



Data Storage

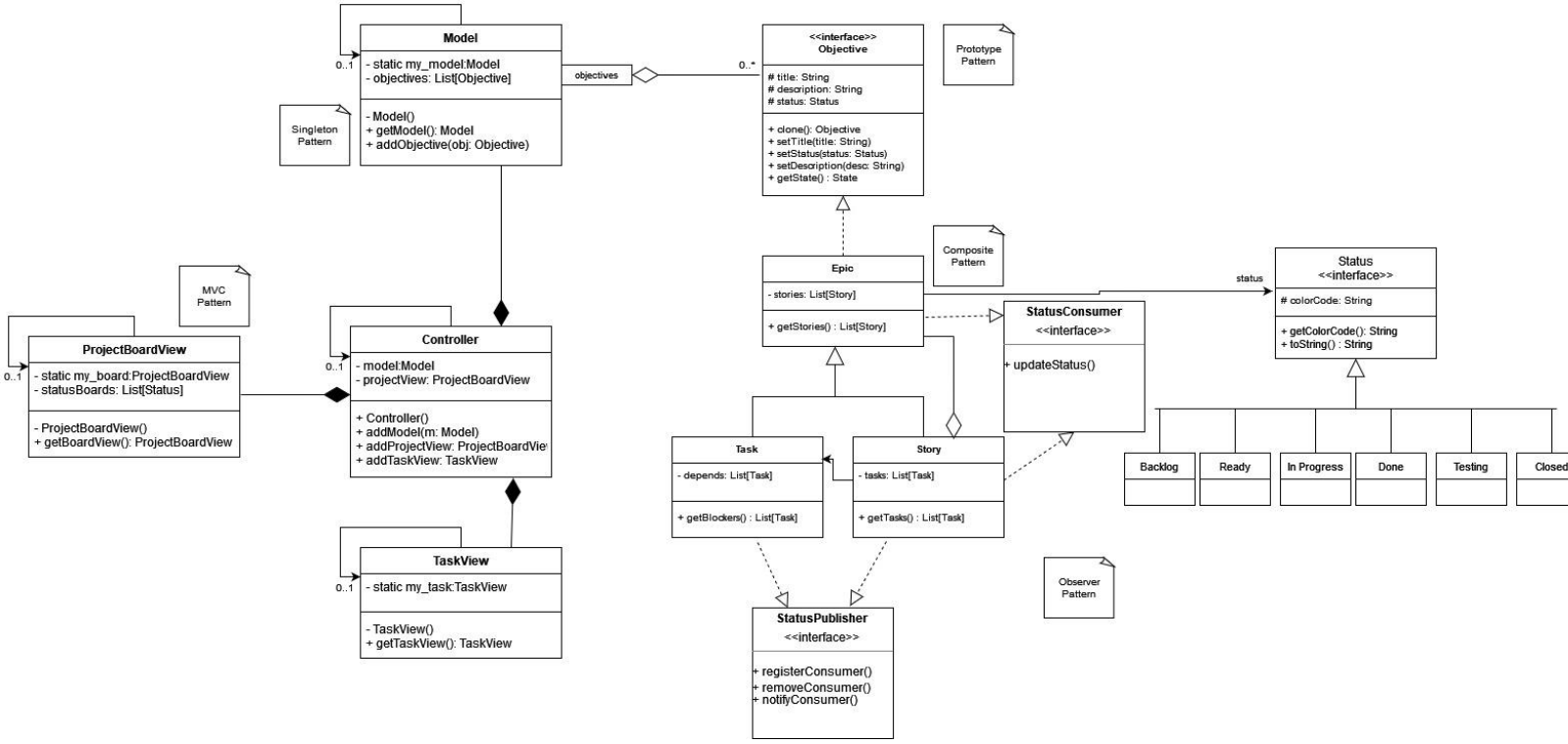
The application would be using a relational database (Ex. MySQL) for storing and retrieving the data. The different tables include user, login credential, task, epic and story. The ER diagram for the tables is shown below. The application would also leverage ORM for communicating with the database.



UI Mockups/Sketches

Link:
https://www.figma.com/file/pqzv4NkCkkeEpgCBCZrcZE/Garuda?node-id=0%3A1

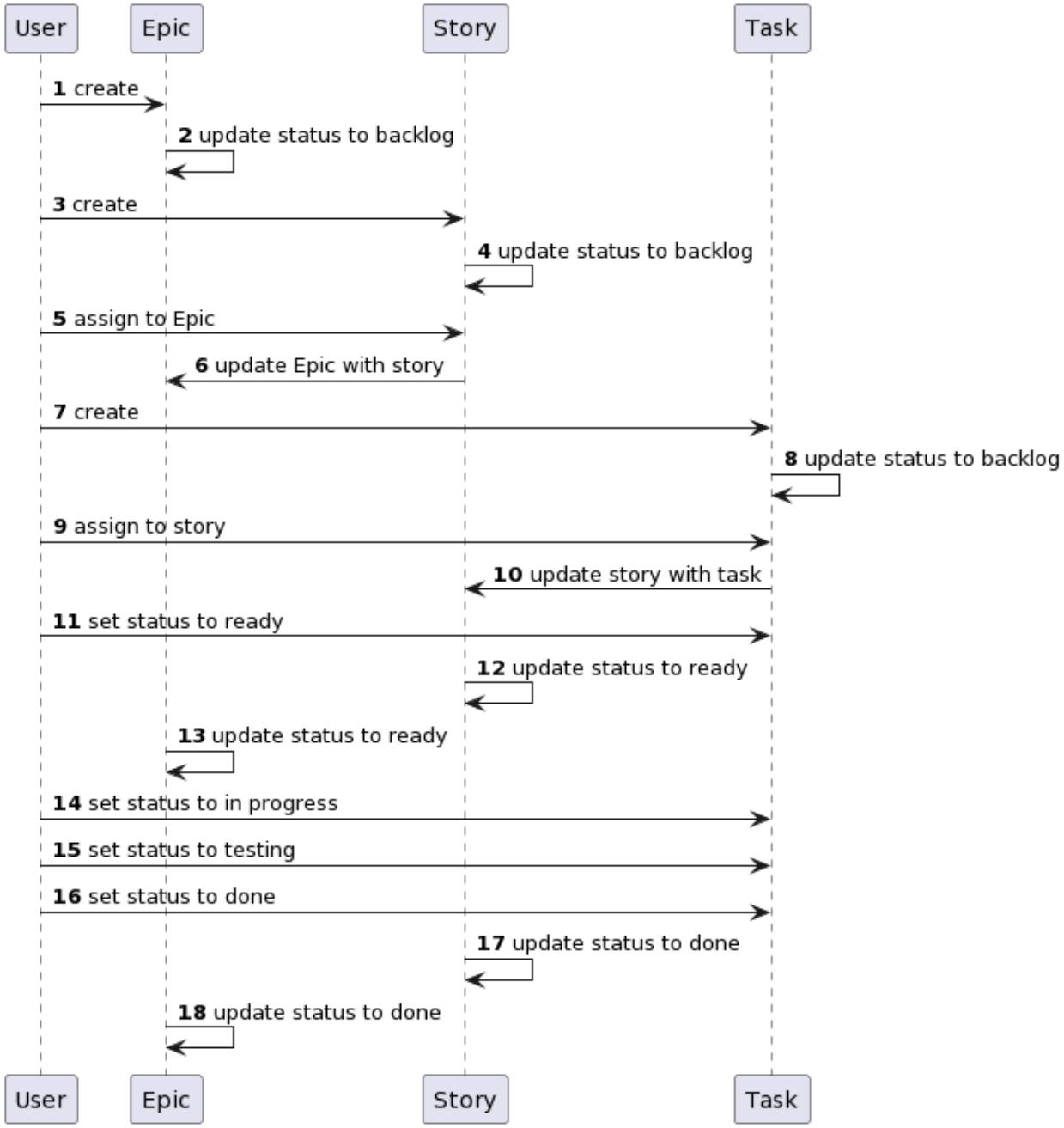
Class Diagram & Patterns



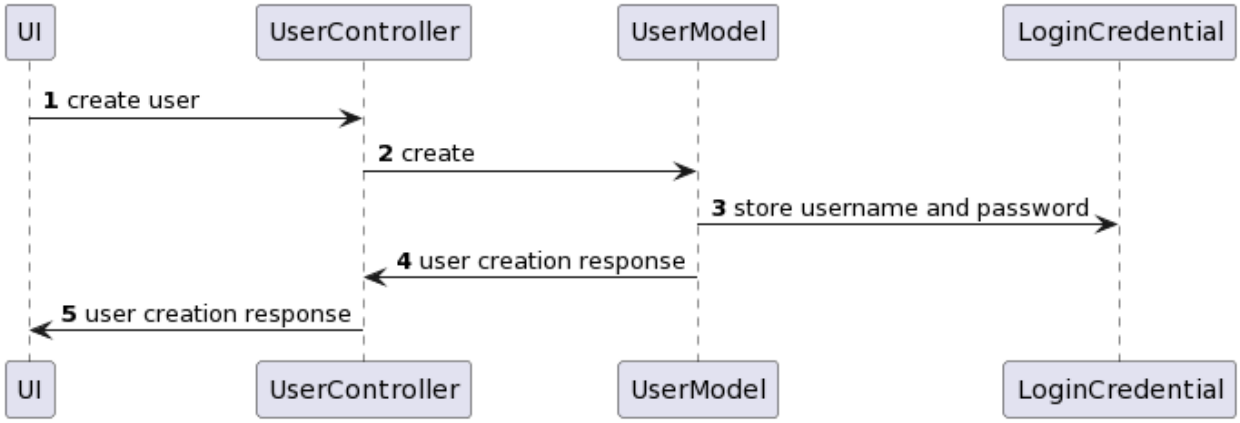
https://i.imgur.com/y8JczlM.jpg

UML Sequence Diagram

1. Task life cycle
   This sequence diagram captures the steps involved in working with tasks. The user has to initially create an epic. Each epic could contain one or more stories. The user therefore needs to create stories and link them with an epic. Similarly, each story can have one or more tasks. The user can thus create tasks and link them with stories. The user can update the status of the tasks. If all the tasks associated with a story are completed, then the story is auto-updated to completed as well. Epics also follow a similar pattern with respect to stories.


User
Epic
Story
Task
1 create
2 update status to backlog
3 create
4 update status to backlog
5 assign to Epic
6 update Epic with story
7 create
8 update status to backlog
9 assign to story
10 update story with task
11 set status to ready
12 update status to ready
13 update status to ready
14 set status to in progress
15 set status to testing
16 set status to done
17 update status to done
18 update status to done
User
Epic
Story
Task

2. User creation
   Every user is required to sign up with the application before proceeding to use the application. This sequence diagram details the different components involved in the process. Since the application follows an MVC pattern, the sequence diagram captures this interaction. The user data is stored in the user model and the user login credentials are stored separately. The password is encrypted before being stored.



3. User Login
   The sequence diagram captures the sequence of steps that take place when a user logs in. The user data is stored in the user model as well as with the login credentials. This can easily be accessed in order to validate the user login.