PM3 Design and Prototype - Bagel Devs

Zachary Lahlou
Maddie Oehler
Conor Edwards
Arik Scofield
Maceo Cardinale Kwik
Esther Kim
Annie Tran

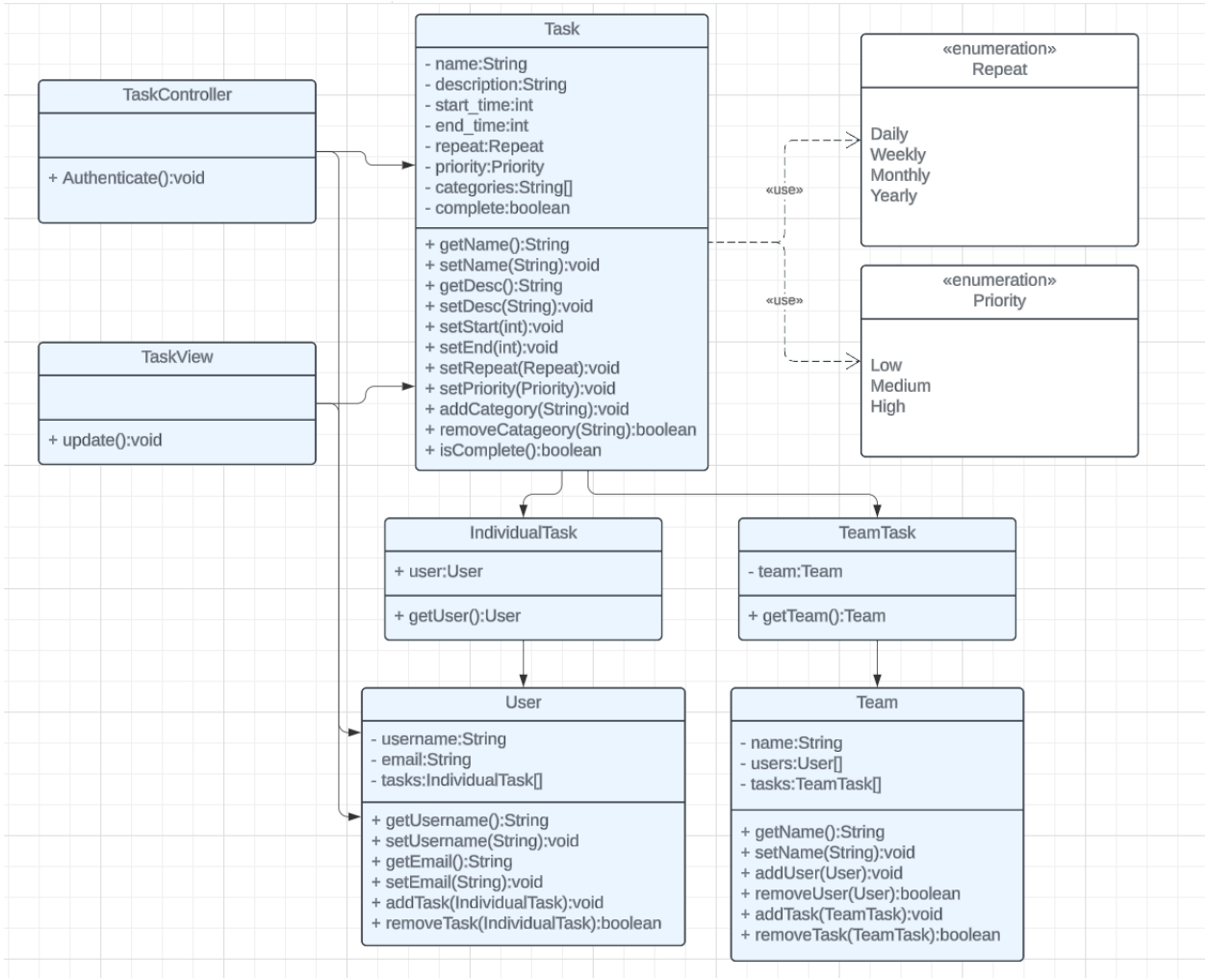
High-level Design

Bagel Dev's To-List application will use the Model-View-Controller architecture. The MVC architecture supports increasing levels of abstraction during design. The model would include classes and methods responsible for tasks, categories, recurrence settings, and system interactions. The views aspect focuses on the visual appearance of the interface that is different from the underlying data management or application flow. The controller acts as an intermediary which handles user application and flow. Bagel-To-Do has features that act as the intermediary such as the task completion marking, setting recurring tasks, filters for tasks, etc. This is the bridge that affects data and UI design. The different levels of abstraction are structured by the model, view, and controller.

Low-level Design

The Creational design pattern family would likely be the most helpful in designing this project, as the TODO app mainly focuses on the creation and storage of tasks. These tasks can be of different types as well, such as a team task vs an individual task.

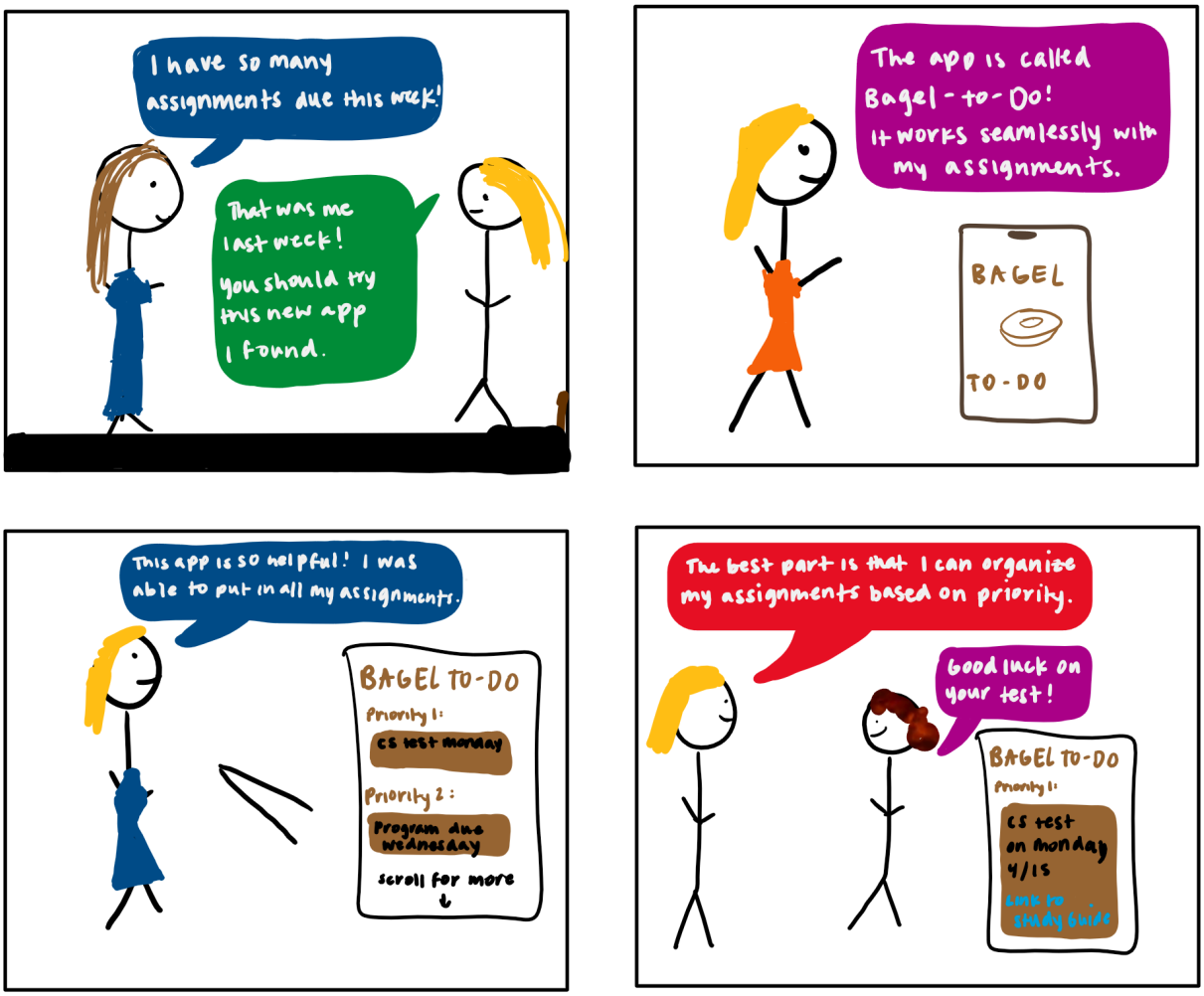
Class Diagram:



Pseudocode:

```
enum Repeat:
    Daily
    Weekly
    Monthly
    Yearly

enum Priority:
    Low
    Medium
    High


interface Task:


class IndividualTask implements Task
    String name;
    String description;
    int startTime;
    int endTime;
    Repeat repeat;
    Priority priority;
    String[] categories;
    bool complete;

    setName(String n):
        this.name = n;

    setDesc(String d):
        this.description = d;

    setStart(int s):
        this.startTime = s;

    setEnd(int e):
        this.endTime = e;

    setRepeat(Repeat r):
        this.repeat = r;

    setPriority(Priority p):
        this.priority = p;

    bool addCategory(String cat):
        this.categories.add(cat);

    bool isComplete():
        return complete

    setComplete(bool c):
        this.complete = c;

class TeamTask inherits IndividualTask:
    private Team team;
    getTeam():
        return this.team;

    setTeam(Team t):
        this.team = t;

class User:
    String username;
    String email;
    IndividualTask[] tasks;

    String getUsername():
        return this.username;

    setUsername(String u):
        this.username = u;

    String getEmail():
        return this.email;

    setEmail(String e):
        this.email = e;

    addTask(IndividualTask t):
        this.tasks.add(t);

    bool removeTask(IndividualTask t):
        return this.tasks.remove(t);

class Team:
    String name;
    User[] users;
    TeamTask[] tasks;

    addUser(User u):
        this.users.add(u);

    bool removeUser(User u):
        return this.users.remove(u);

    addTask(TeamTask t):
        this.tasks.add(t);

    bool removeTask(TeamTask t);
        return this.tasks.remove(t);

class TaskController:
    authenticate(User):

class TaskView:
    updateView():
```

Design Sketch



One design choice that we chose to illustrate in the design sketch was our choice to order our to-do list by priority. This is specifically helpful in school - situations as shown in the drawing above. By organizing the list of tasks by priority level, users can keep track of their most important assignments coming up in the week.

Process Deliverable

Our team has been developing the prototype on Figma. View our work with this link. The labels on each frame describe the flow when presenting the prototype.

To present the working prototype, use this link.