Brian Alfano
Sung Bae
Eric Lindemann
Artur Meletyan

Project Part 4: Final Report
Project Brian Team 08

Question 1.) Our first feature implemented was MySQL connector for python. We stored data in tables about users, boards, and tasks. The MySQL connector allowed Project Brian to extract the data from the databases' tables. Another key feature for Project Brian were *terminaltables*. The layout of a terminal can be dull and boring, but with *terminaltables* we are able to split boards, tasks, and task states in a neat and fashionable way. Project Brian used a State Design Pattern. This design pattern was extremely useful for object oriented programming.

Final system's class diagram:

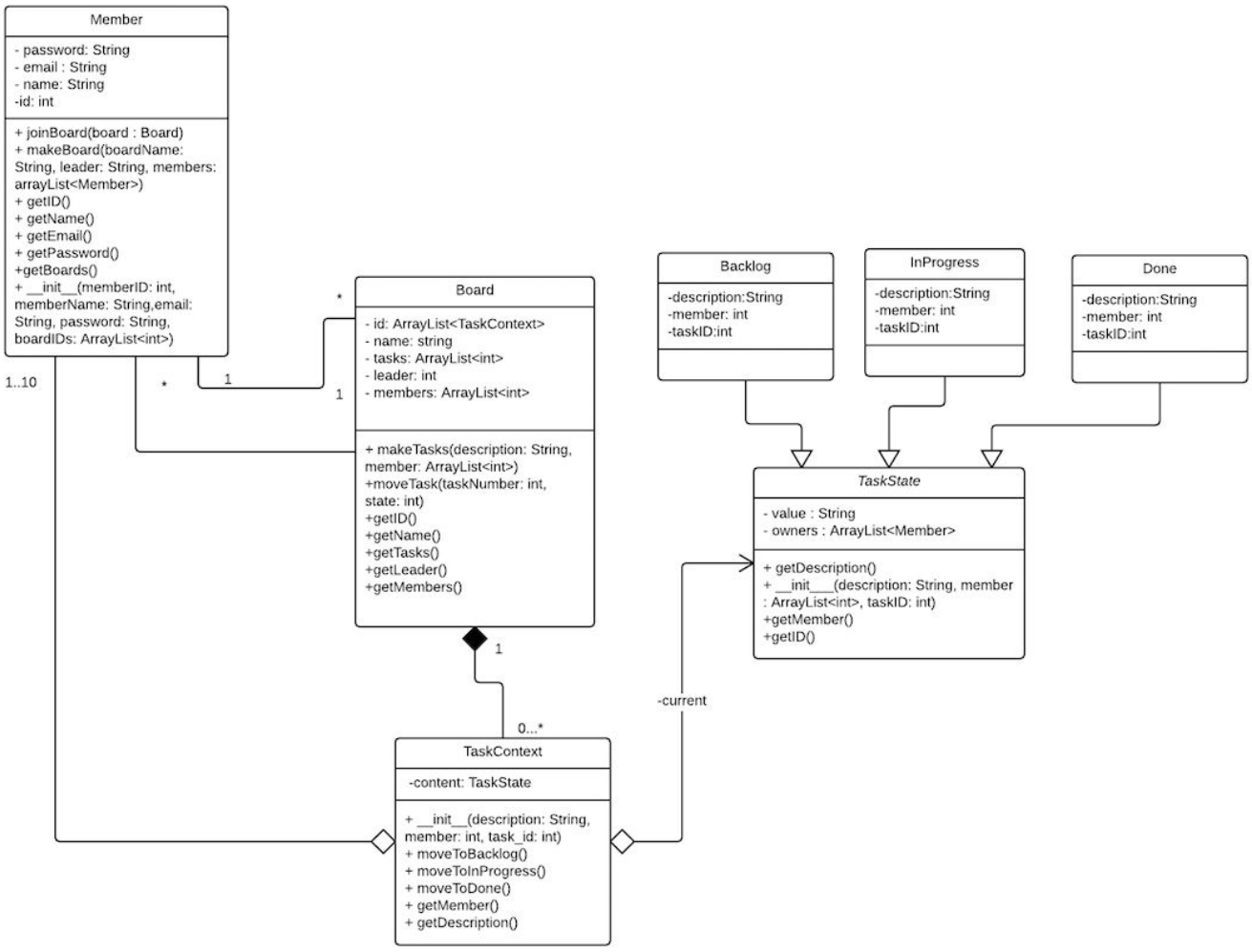

Figure 1: For Project Part 2 we turned a class diagram to help guide our vision for the final system. This was incredibly helpful and lead to this class diagram for our final system.

Some additional attributes and behaviors are the main changes to our final system's class diagram. We also removed the leader class entirely. The additional instance variables and functions are due to unanticipated actions that our system required. We believe the design and analysis process is crucial to understand most of the systems functionality, and that is exactly what our diagrams did for us. It is very difficult to know exactly what you need before programming but we did a great job in anticipating the functionality of our system.

Question 2.) For Project Brian we implemented a State Design Pattern in order for our Task Object to alter its behavior when its internal state changed. Tasks we delegated into three subclasses: class Backlog, class InProgress, and class Done. When a task in Backlog would change to InProgress its internal state would change. Using a design pattern helped code faster because we knew exactly how the subclasses should interact with the super class *TaskState*.

Question 3.) In Project Part 2 we envisioned our system to be hosted on a website and for users to see a GUI. With the lack of knowledge in coding a sufficient GUI the main focus turned to using a terminal for our final system. This was by far our biggest change to our project. It is less user friendly, but gets the exact same job done for the user. Another reason for disregarding a GUI was to narrow our focus on only object oriented programming. By focusing on mostly back end code we were able to learn more, create a better back end system, and finish the project on time with minimal difference. Almost all of our requirements from business, user, functionality, and non-functionality remained the same. Our activity diagram for creating a board remained the same except when a user doesn't exist in the database we do not send the user an email. Instead, we make the board leader retype the user's he wants on the board until

all of the users match the users already in our database. Our initial idea for a database in Project Part 2 was MySQL, and we stuck with it. Project Part 2 was extremely useful to the final system's design because doing it correctly helped us program everything.

Project Part 2 class diagram:

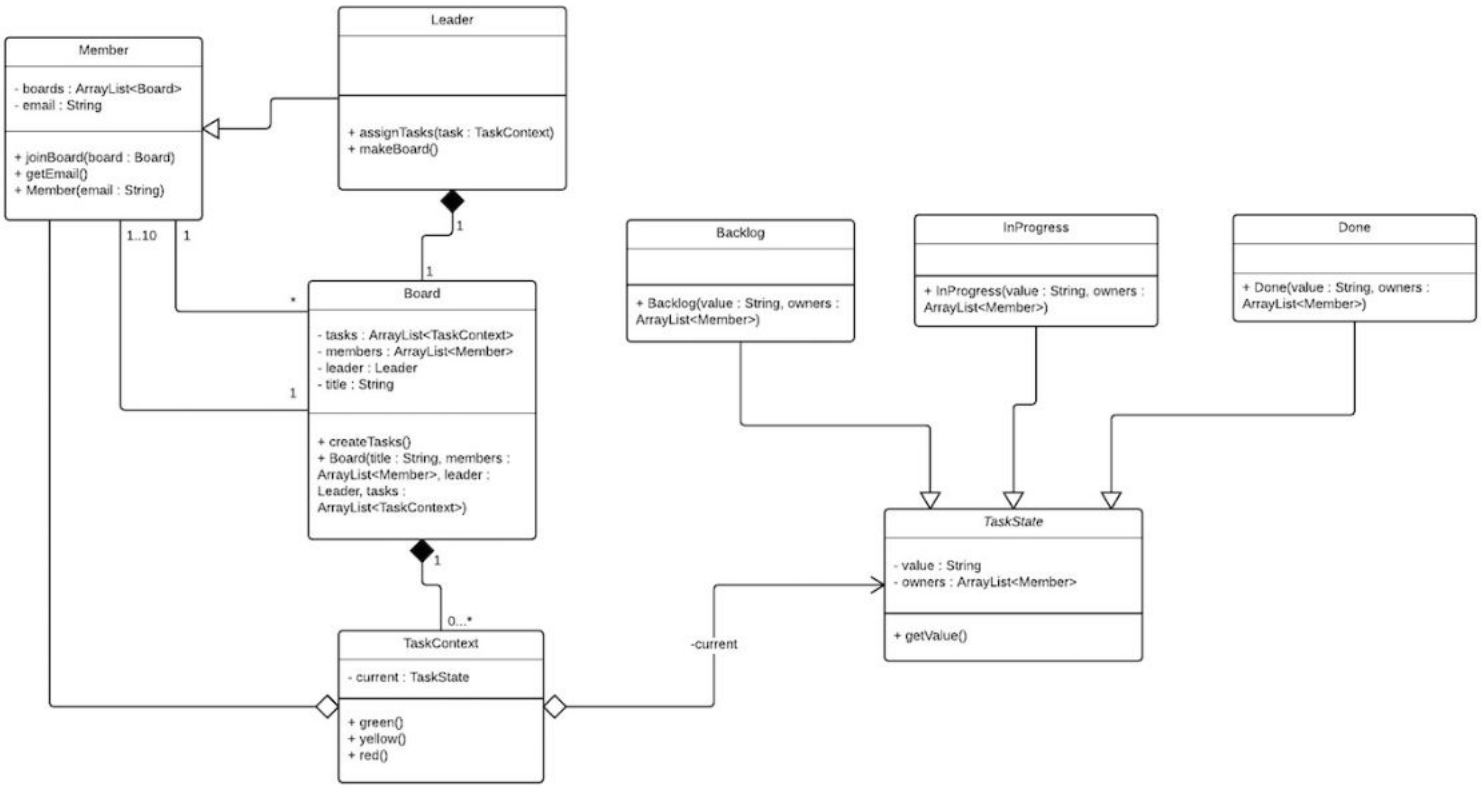

Figure 2: Project Part 2 class diagram when we anticpated most of the attributes and behaviors but not all of them.

***Final system's class diagram*:**

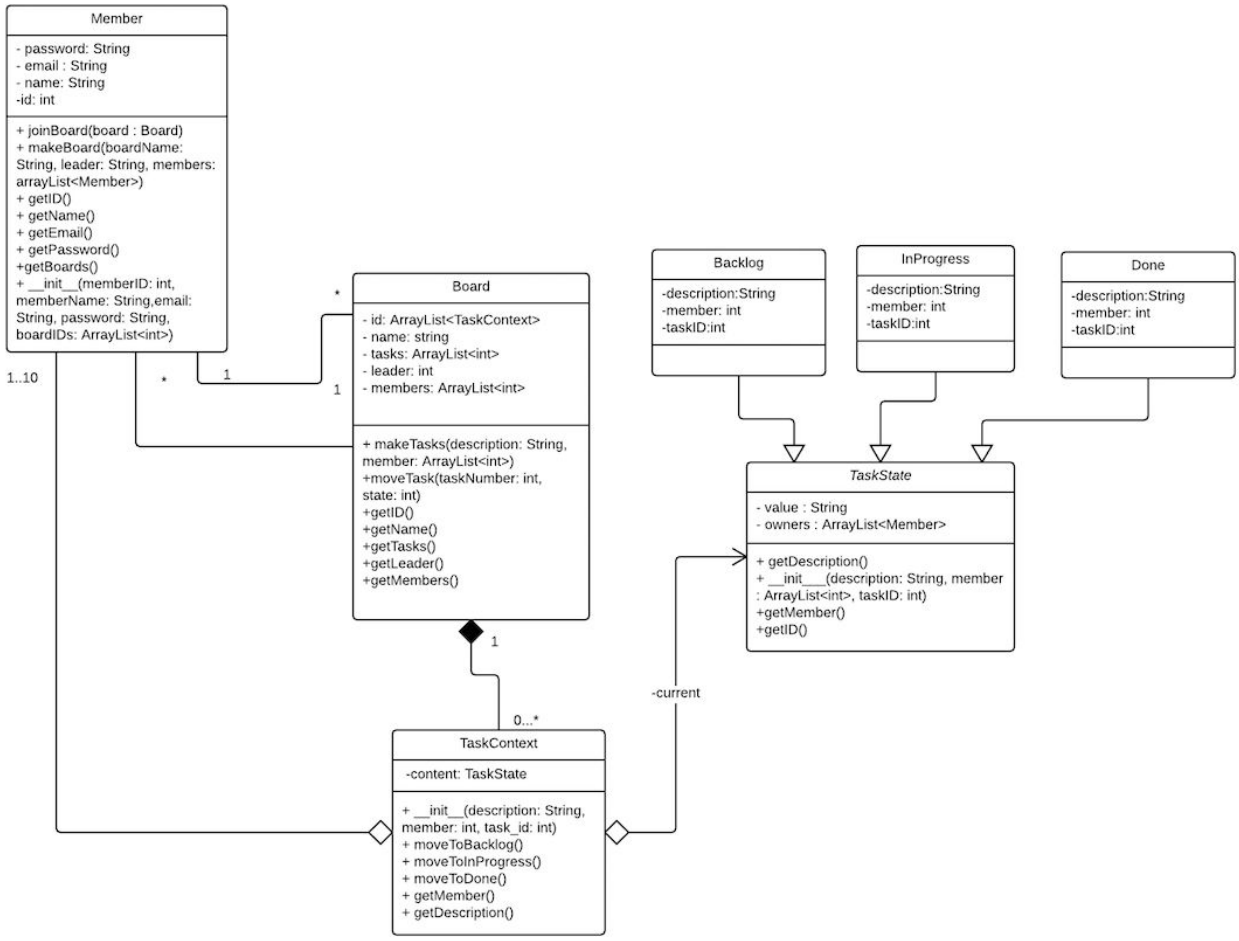

Figure 3: Final System's class diagram where we removed the leader class and added more attributes and behaviors. All the class relations remained the same.

In question one we described what changes occurred in Project Brian's class diagram and why the changes happened. Now you can visually see the differences between Project Part 2's class diagram and our Final System's class diagram. We found that our leader's behaviors were exactly the same as our member class therefore we removed it. "Get" methods were added to all of the class diagrams because we obviously needed them. On Project Part 2 we got a 59/60 because our methods "red", "yellow", and "green" were terribly named. You can see in the TaskContext class that we changed all these bland names to "Backlog", "InProgress", and

“Done”. Thanks to this recommendation we were able to make the code more readable.

Question 4.) Project Brian learned through the design and analysis process the capabilities of a project. Without design and analysis each day of coding would have been done blindly and making the coding process take longer. With a set of diagrams and analysis we were able to code more fluently because the diagrams helped us plan for the future. There is no way by the end of the semester we would remember every minor detail we wanted to implement in our final project. Through design and analysis we were able to look back in our records and see exactly what to implement. A critical part of design and analysis for our group was constantly comparing our code to our diagrams. When we recognized a section of a diagram was not yet coded it would remind us that it needs to be done. Without the design and analysis part of this project we would have left a lot of critical details out of our final system.