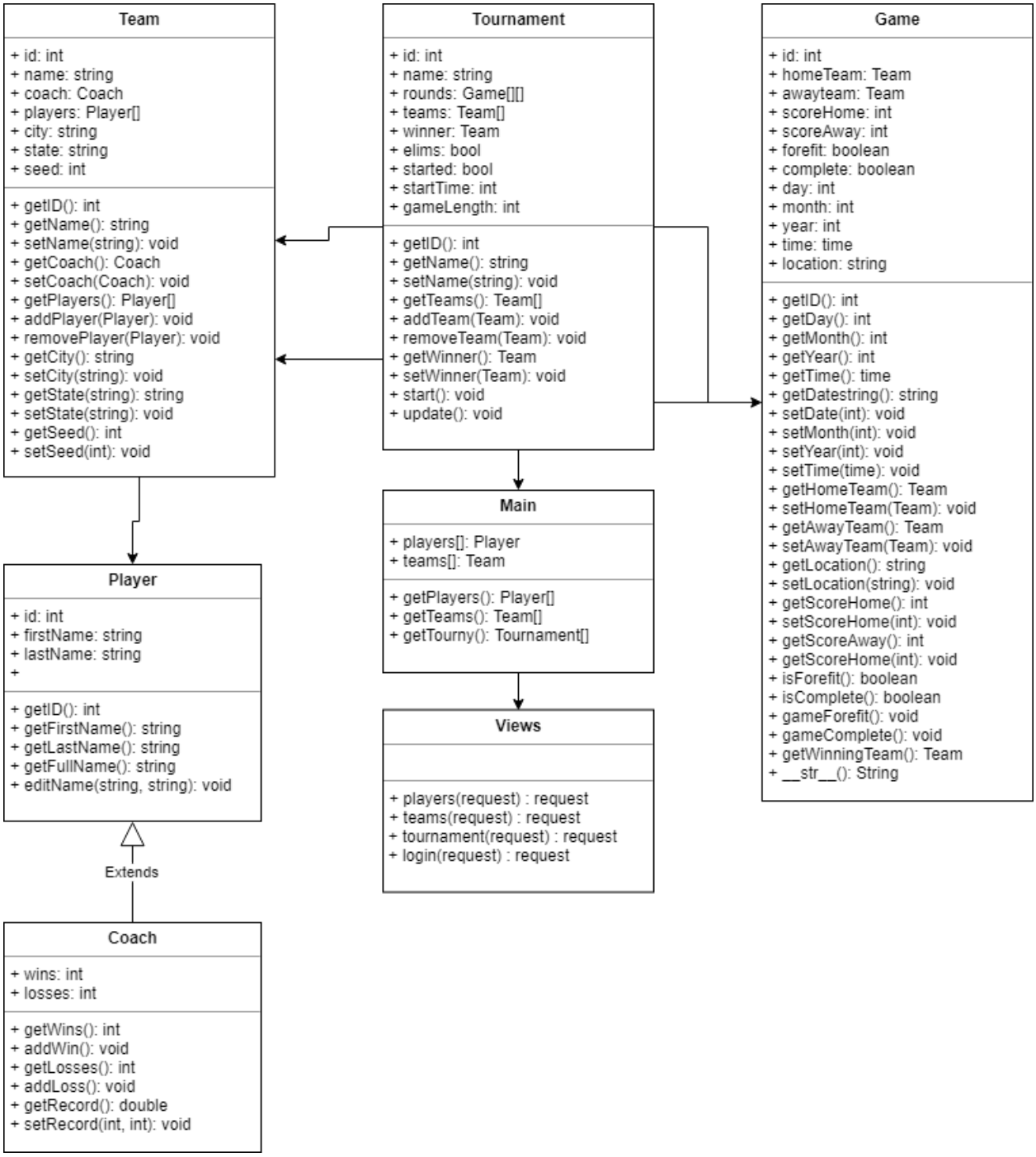
Name: Andrew Casner
Title: Online Brack Manager
Project Summary: This online bracket manager will allow users to create and manage tournaments for any different type of competition. It will allow users to define a number of competitors, the type of competition, and then be able to visually manage the tournament.
Compleated Project Requirements:

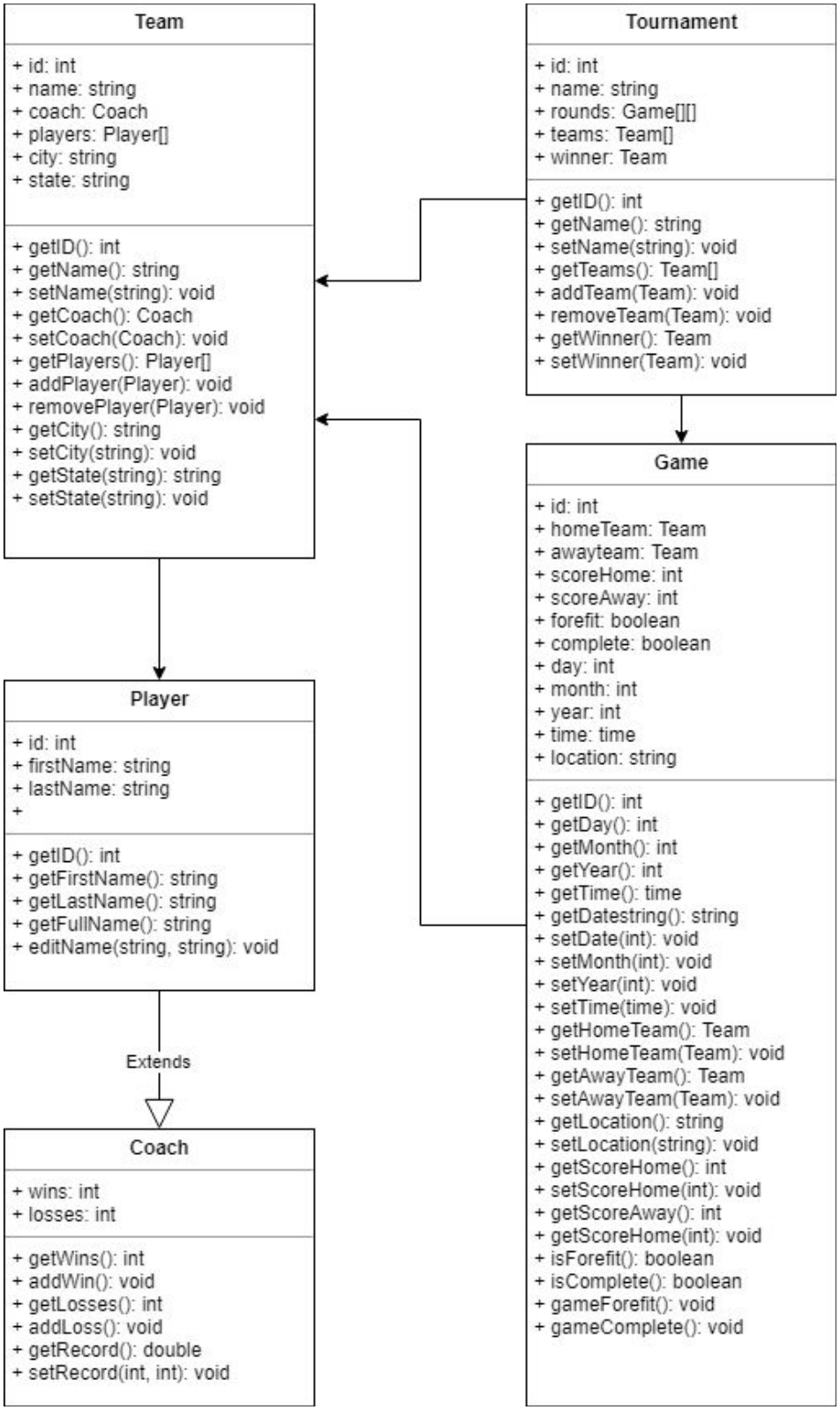
UR-001	Users can sign in
UR-002	Users can specify the number of competitors in each tournament and they type of tournament it is.
UR-003	Users can view their tournament visually.
UR-004	The software will automatically generate a tournament based on the input variables.
UR-005	Users can update the status of the tournament.
UR-006	Users can view the status of a tournament.

Uncompleted Project Requirements:

Updated Class Diagram:

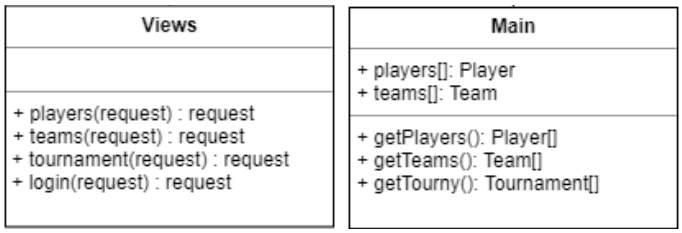


Original Class Diagram:



Changes: The majority of the changes came with the addition of the main class to run and control the web app as well as the views class to update the UI shown to the user. I had not taken this class into account at the beginning of the semester. Planning out the classes and their variables and functions were very helpful and made me really think through the design before I started programming. This kept my code clean, clear, and easily readable.

Design Pattern Implemented: The design pattern I choose to implement was the model view controller pattern. I choose this pattern as it seemed very relevant to designing a web app. I implemented this framework within the Django framework. The models are the Team, Player, Coach, Game, and Tournament classes as these contain all the “business logic”. The Main class is responsible for the controller. This class pieces all the other classes together and is responsible for passing information to the view. The View class renders and returns templates with the data it is given by the Main class.



Key Takeaways: I learned a lot during this project, but the most valuable thing I believe I learned was that planning and setting requirements before you start can lead to much cleaner, easily maintainable code, then just jumping into a project and coding straight away. The class diagram really forced me to think through all use cases of my app and figure out the best way to implement it. This resulted in clean, maintainable code that makes sense and is organized properly. If new features are to be added to the project, it is built in a way where you can quickly and easily implement new functionality, due to its solid, planned out base.