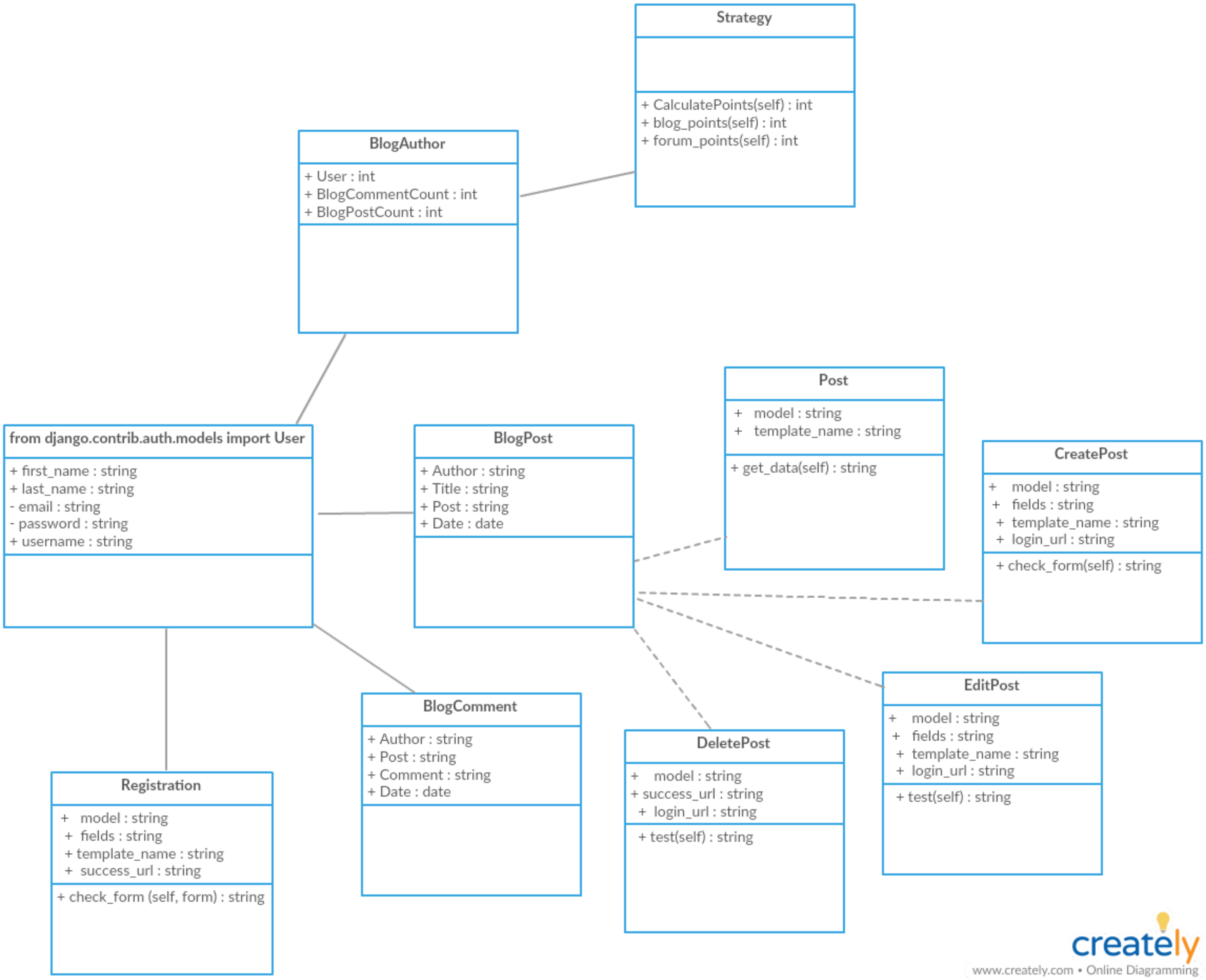
1. **Name**
   Richard Li
2. **Project description.**
   Blogging website where users may submit, comment and rate forum and blog posts. Post and comment functionality includes the creation of, editing, rating and deletion of them.
3. **List the features that were implemented (table with ID and title).**

| Req ID | Requirement Description |
|---|---|
| UR-1A | Blog users may sign up for an account |
| UR-1B | Users may log in to the system |
| UR-2 | Users may view their account activity amongst all public/private forums or blogs |
| UR-5 | Users may contribute to (submit text comments) any forum or blog they have access to |
| UR-6 | Users may star forum and blog posts as a voting system to promote more popular content |
| UR-7 | Users may search for other users |

4. **List the features were not implemented (table with ID and title).**

| Req ID | Requirement Description |
|---|---|
| UR-3A | Users may subscribe to/follow any public forum or blogger |
| UR-3B | Users may unsubscribe from/unfollow any public forum or blogger |
| UR-4A | Public and private forums are globally searchable for keywords |
| UR-4B | Private forums and blogs will only appear in searches if users have read permissions for those particular forums or blogs |
| UR-8 | Users with the most starred content may be identified in a global leaderboard of the most prominent users |

**5. Show your final class diagram.**
**What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.**



The design changed significantly as the project was originally designed to be implemented in Java, but was later changed to Python using a Django web framework. The way in which Django in particular is set up encourages a different methodology in design, with its MVC/MTV(Model template view) framework and the diagram does the best to portray that. All classes stemming from BlogPost are a part of views.py, which essentially acts as a controller in a typical MVC design pattern. Ultimately, due to the change in languages and framework choice, the project design had to change as well.

**6. For each design pattern implemented,**

- **Show the classes from your class diagram that implement each design pattern.**

  BlogAuthor, Strategy

- **Show the class diagram for the design pattern.**

  See above, specifically for classes BlogAuthor and Strategy

- **Explain how you implemented the design pattern, why you selected that design pattern.**

Strategy makes sense for altering algorithms that use similar formulas behind the scenes and changing them at run time. Implementation was not in the traditional Java sense, using an interface and having subclasses implement it due to Pythons limitations. As Python does not have interfaces, I used a single class called Strategy and implemented an abstract method that added points to a users score. This method was then overridden by other methods that were designed for differing applications. This is ultimately the closest you can get to a Strategy implementation in Python.

7. **What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?**

   Design is incredibly important in the implementation of systems and a good design can save hours in implementation time. The process of analysis can also help to streamline a project and show us where poor design can inhibit the successfulness of a system. Regarding the process itself, I’ve learned that the design of a system is equally important as the end implementation simply due to the usage and implementation of systems being decided by good designs.