## 

Horse Racing Website Documentation

12.2022

****

Jiacheng Shen

Jiani Yu

Junyan Feng

# Table of Contents

**Table of Contents** [**2**](https://docs.google.com/document/d/1_MmcLJokzLVBkVGPH4qisiuUDepsDlId/edit#heading=h.fgm6himu3ebe)

1. Description [**3**](https://docs.google.com/document/d/1_MmcLJokzLVBkVGPH4qisiuUDepsDlId/edit#heading=h.ymr915ohwo3o)

2. Process **3**

3[.](https://docs.google.com/document/d/1_MmcLJokzLVBkVGPH4qisiuUDepsDlId/edit#heading=h.1fob9te) Requirements & Specifications [**4**](https://docs.google.com/document/d/1_MmcLJokzLVBkVGPH4qisiuUDepsDlId/edit#heading=h.1fob9te)

4[.](https://docs.google.com/document/d/1_MmcLJokzLVBkVGPH4qisiuUDepsDlId/edit#heading=h.tyjcwt) Architecture & Design [**5**](https://docs.google.com/document/d/1_MmcLJokzLVBkVGPH4qisiuUDepsDlId/edit#heading=h.tyjcwt)

5[.](https://docs.google.com/document/d/1_MmcLJokzLVBkVGPH4qisiuUDepsDlId/edit#heading=h.k0ea6al3733u) Reflections & Lessons Learned [**7**](https://docs.google.com/document/d/1_MmcLJokzLVBkVGPH4qisiuUDepsDlId/edit#heading=h.k0ea6al3733u)

# Description

This is a horse racing information web application. The stakeholders of our project are people who love horses and watching horse races. They will have the access to know many horses, learn bits of knowledge about horses, join forums to participate in discussions of horses, and get exposed to actual horse events. Our website consists of a home page, news page, forum, horse information page, and calendar page. At the same time, we added several user interaction functionalities, including login, like/dislike, comments, and searching, to make our web more user-friendly. For the administrator side, we enable the administrator to manage accounts, publish news, and post announcements, besides the basic user functionalities.

# Process

During our developing process, we choose to follow the Scrum process and one week for each Sprint. We first decide to use Flask and nextjs as the frameworks of the backend and frontend. And then, we start designing UI drafts, user case diagrams, and apis. After the initialization, we then officially start to implement concrete functionality in each Sprint.

In each Sprint plan after designing, we assign tasks based on certain functionality. For example, if one of us is going to implement forum functionality, he will finish the viewer and controller together. If one of the tasks doesn’t finish before the next Sprint, the unfinished task will distribute to other team members whose task list is clean.

After all the features are implemented, we start using Postman to perform Api test and check if all the functionalities run well.

At the very end, we use eslint to check the code format by following the Standard javascript style and manually commenting on the functions and components.

However, during our iterative development, we encountered some conflicting circumstances. For instance, if one of us is assigned to finish the login functionality, the menu component needs to change as well. However, since the menu component is assigned to another member, he/she may not understand what the login function is doing and have no idea how to modify the corresponding items in the menu. So this may result in incompleteness in some Sprint.

# Requirement & Specification

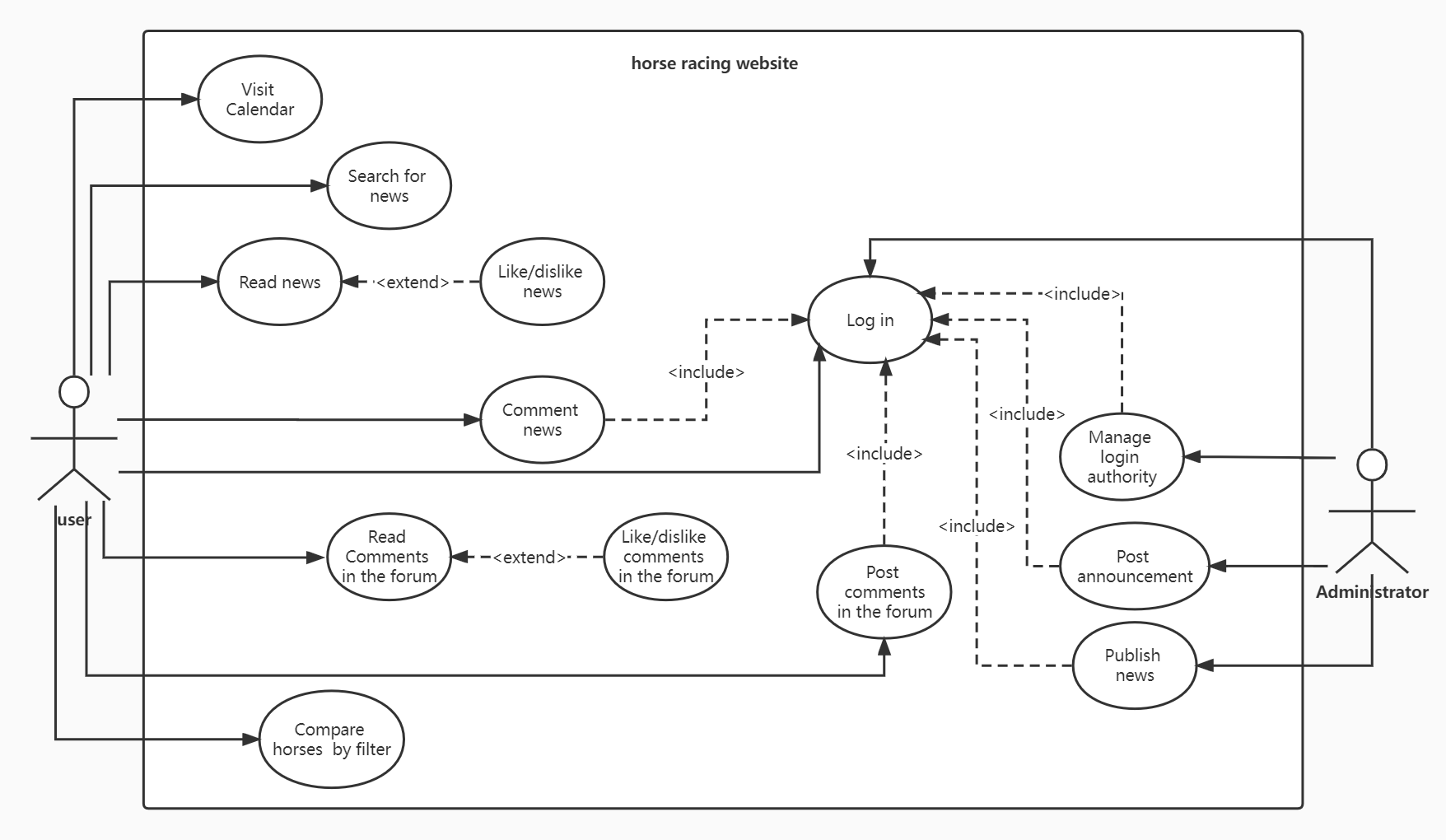


Figure1 - Use Case Diagram

We determined two types of participants, the user, and the administrator. We implemented the web app following the above use case diagram. All the required functions are implemented. The relationship between use cases is also well reflected in the final app. In addition, in our final product, administrators can also access users’ actions, which is not indicated in this diagram. The following is the user story for posting messages in the forum

1. UC: “Post messages in the forum”

1.1 Preconditions:

Log in

1.2 Main Flow:

After logging in, users can type in the text box and post messages in the forum[E1].

1.3 Subflows:

None

1.4 Alternative Flows:

[E1] User’s comment cannot be an empty string. If a user tries to post an empty string, the system will warn the user to retype his comment.

# Architecture & Design

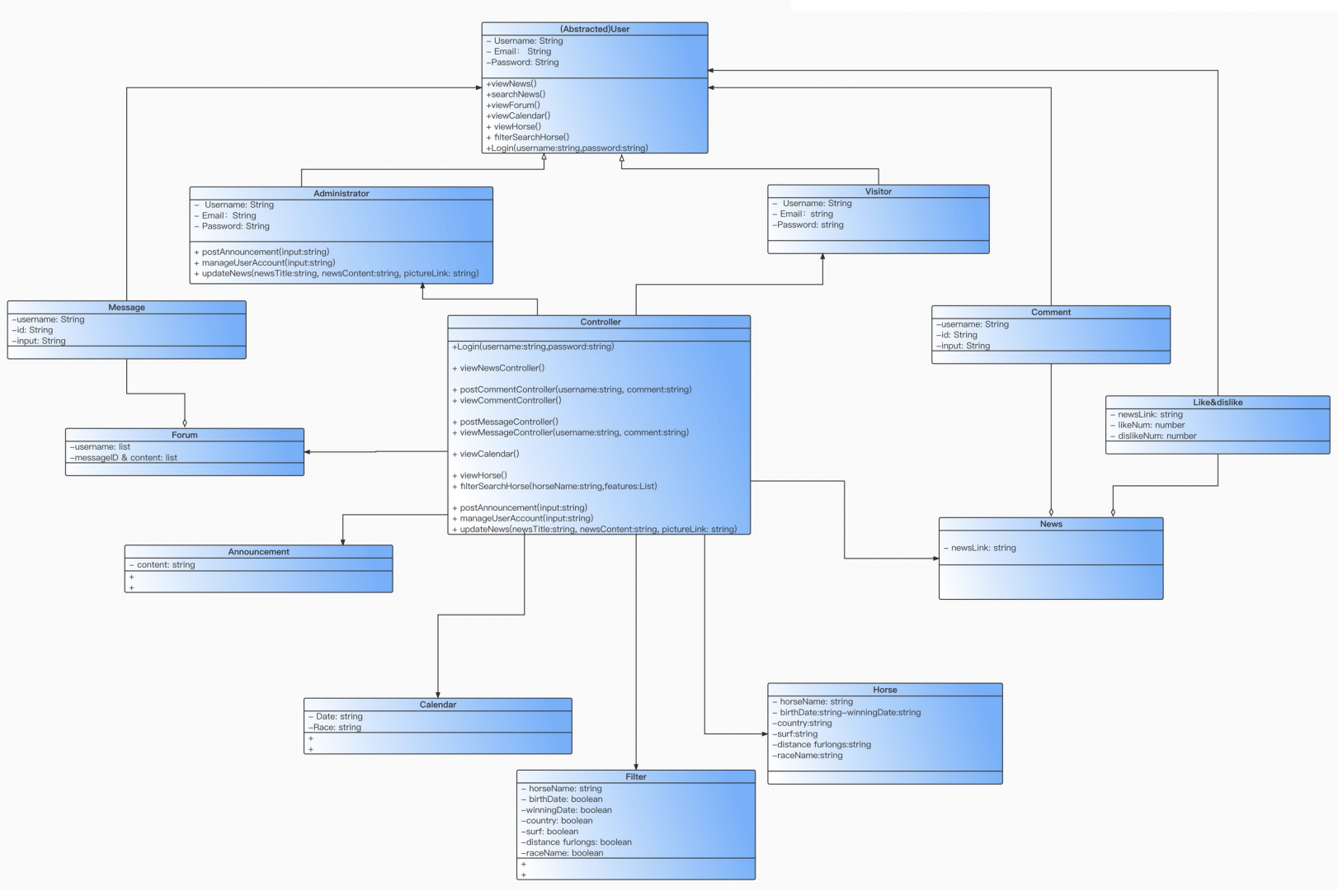


Figure 2 - Class Diagram

The core entity of our software is “User”. In our design, we have two types of users: “Administrator” and “Visitor”. Only administrators have permission to visit the “admin” page, which includes three components (functions): post announcements, manage user accounts, and update news. Only administrators and visitors can post comments.

Everyone (even without login) can use the horse filter/sorter, calendar, press the like/dislike buttons and read news/announcement/forum messages, which are all components (entities). We make a navigation menu to help users find the function they want to use.

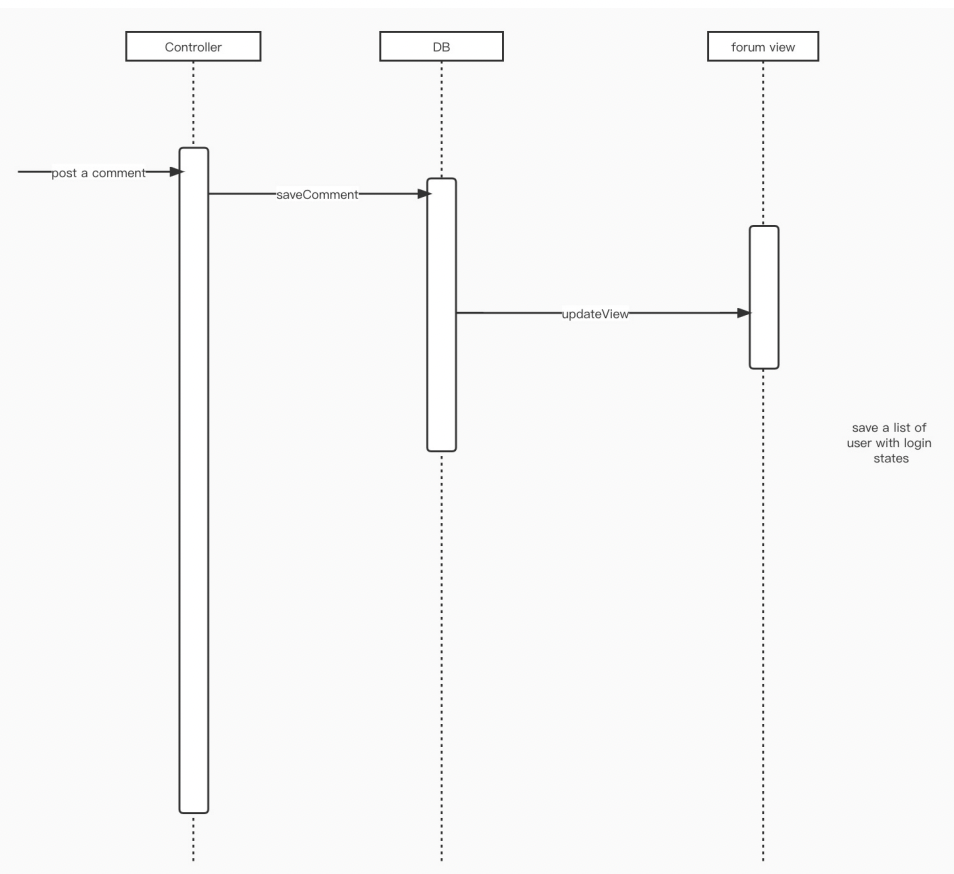


Figure 3 - Sequence Diagram

To describe our design for a concrete entity, let’s take the forum as an example. When a user interacts with the forum (press the “post” button), the “PostComment” method is triggered. This method will send a post request to the backend. The login authentication decorator at the corresponding backend API will check the login status (the permission). If permission is granted, the comment content will be stored in the database and the forum view will be refreshed.

We choose Nextjs (a popular framework of React) as our frontend framework. The “React hook” enables us to use state and other React features without writing a class as well as automatically apply state and observer design patterns to our frontend. Another critical feature of React is always breaking giant components into smaller pieces of components. Thus, when we design our system and write the codes, we keep considering the componentization of entities as well as the reusability of the components.

We choose Flask as our backend framework. Flask (and python itself) has a helpful feature called function decorator. This feature automatically applies the decorator design pattern to the backend. Besides the necessary “router” decorator, the jwt related decorators make the login authentication code neat and simple. Since we use Nextjs as our frontend, there is no need for us to create html files at backend.

# Reflections

Jiacheng Shen:

In the developing process, I learned that the basic knowledge of backend and frontend is very important. I spent nearly one and half months learning React and javascript. Besides that, I also learned that knowing the manners of doing the project is important as well. At the very beginning, I push my code as long as I stop, so it has a lot of errors in my commit, which may confuse other team members. In fact, I should only commit the code I finished instead of the unfinished version. Additionally, I forget to update the package JSON when I install a new package. This also results in some components can’t run on other teammates’ side after they pull the newest version from GitHub.

Jiani Yu:

I enrolled in this course with little knowledge about the implementation of frontend and completely no idea about frameworks, backend, databases, and software design and development. I struggled to understand the software engineering concepts covered in classes. This project gave me the opportunity to connect abstract concepts with concrete coding practice. In the early stage of implementing the web app, I found it hard to build connections between the project with the knowledge taught in class. I was fully focusing on learning the frameworks and writing correct codes. But in the final weeks, when I was thinking about refactoring the code, I started to see how design patterns can be used in our app. Some design patterns were used without intention. Although I didn’t really implement them (because it takes time for me to figure out how to do that with correct syntax that follows the framework ), I came up with some ideas to improve our design using the single responsibility principle. I also gained a basic understanding of network protocol, communication between frontend and backend, and version control, mostly from my failures and the processing of debugging.

I was eager to understand what API is and how it works because I used APIs for a few of my IMA projects and I never know about it. I put a lot of effort into learning it and I’m very glad about what I obtained. Building APIs for our application from scratch also greatly assists this process. In particular, I learned API testing and postman for the technique presentation and completed the API testing for our web application. I learned from the official document of postman and online tutorials, which indeed improved my self-learning ability.

Junyan Feng:

When I enrolled in “software engineering”, I didn’t expect that this course could be that helpful. This course gave me a concrete and deep understanding of software process, design pattern, testing, API, etc., which are the concepts I may have gotten in touch with before but never really understood. Especially, through this course I understand how important and useful design patterns are as well as how the design patterns are applied, intentionally or unintentionally, in the software.

The experience of “software creation” is of vital importance to me. When I was doing my internship, I had few chances to participate in software design. In this course, however, I can participate in the whole software development process, from planning, designing, analyzing to developing, testing and documentation writing (but no maintenance and customer support, that’s a pity). This experience gives me a comprehensive understanding of the software development life cycle. Also, it’s my first time to try combining Next Js with Flask, which is a precious experience.