## 1. SCRUM Roles & Events:

- Briefly describe how you would contribute to a SCRUM-based team as a student intern.
- What SCRUM ceremonies would you participate in, and what would be your input?

I, as a student intern, would be part of the development team, where I can learn the most about Touchstone's goals as I'm still familiarizing myself.

**Throughout each SCRUM ceremony…**

**Sprint planning**: Planning specific, measurable, attainable goals for the next couple weeks.

- The Product Owner discusses what they wish to accomplish, adding each task to a backlog.
- The Scrum Leader divvies up tasks, ensuring each person has a reasonable workload. They also provide support, such as scheduling, training, etc.
- I would state what I'm able to accomplish and what support I need.
- I would also contribute ideas about how to improve the product for consumers (ex: UI/UX, features, small improvements, etc.).

**Sprint**: Development time (about 2 weeks).

- The Product owner will evaluate their vision as development and new challenges unfold.
- The Scrum Leader makes sure people know what they're doing and keeps team morale high.
- I would be working on the development of my tasks and requesting help whenever needed.

**Daily scrum / stand-up**: Short, daily checkups.

- The Product Owner will make sure the team is on track with their vision.
- The Scrum leader makes sure that people have the support they need.
- I would state what I finished the previous day, any obstacles, and what I plan on completing.

**Sprint review**: Reviewing and showcasing the work.

- Everyone celebrates :)

**Sprint retrospective**: Discussing what did and didn't work.

- The Product Owner gathers information about how to improve the next, or current product.
- The Scrum Leader gathers feedback on how to improve the team next time.
- The development team and I will give our thoughts on the development.

## 2. SDLC Planning:

- Describe how you would approach redesigning the automatic flagging system following the SDLC.
- Specify the phases (Requirements, Design, Implementation, Testing, Deployment, Maintenance) and what you would do at each stage.

**Requirements:**

Frontend:

1) Applicants:
    a) Input form with input validated (email, nothing blank, etc.)
2) Reviewers portal:
    a) Be able to navigate between multiple applications.
    b) When clicked into a specific application, be able to easily see flags without having to manually mark everything down.

Backend:

1) Applicants: Can submit an application and have it be stored.
2) Reviewers: Backend service that automatically flags issues and categorizes them by severity.

Additionally, ask what stakeholders need, analyse their requests, and update requirements from there.

**Design:**
Wireframe: Only has bare essentials (requirements) and describes how buttons navigate between pages.
API design:

1) Decide how to store data (JSON? XML?).
2) Decide how data will be structured (how to group data for easier access, ex: personalInfo: {firstName:"", lastName:"", email:"", birthday:""}).
3) Decide what tasks are handled on frontend vs backend (ex: flag data in frontend then send to backend? Or have the backend do all the flagging?).
4) Plan ways to make code readable and maintainable (ex: Single Responsibility Principle to keep code clean and reusable).

UI/UX Design: Make wireframe prettier (animations, colors, etc.) using, for example, Figma while also ensuring the design is intuitive.

**Implementation:**

1) Frontend (view + controller): Create each page and validate functionalities as you implement them (ex: form correctly validates that required fields are all filled).
2) Backend (controller): Ensure backend APIs are correctly hooked up the frontend & are performing functions as expected.
3) Data (model): Ensure data is being updated and is correctly formatted.
4) Throughout it all, ensure proper documentation for future maintainability.

**Testing:**
Software testing:

1) List out all the possible inputs & anything else that can go wrong (ex: email input is not a valid email).
2) Write unit tests with, for example, Mocha, for each test case.
3) Run, test, fix bugs, repeat.

User Experience testing:

1) Find people to test the software (best to find people within and without the company for fresh perspectives).
2) Depending on sample size, ask each individual what they thought. Or make a survey asking about ease of use, how intuitive the software is, any other thoughts people had, etc.
3) Find the most common criticisms and tackle those first.

**Deployment:**

1) Push all code you wish to deploy to the dev/test branch.
2) Host backend with a provider (ex: AWS, Azure, etc.)
3) Host the frontend with a hosting provider (ex: Vercel, Hostinger, etc.)
4) Do some smoke testing (small tasks to ensure functionalities are working) to see if code can move to the next stage of testing.
5) Continue until code is pushed to production.

**Maintenance:**

1) Ensure that there is a feedback form and a way for the site to send any bugs and data about the bugs back to the development team (ex: with LogRocket).
2) Regularly check in on the program to do bug fixes, small improvements, or even to add new features.