Objectives

To use GitHub and the GitHub flow model to collaboratively engineer, deliver, and evaluate a software product. Along with using GitHub features like the issue tracker and reviewing pull requests and using ZenHub board to manage the project, in this assignment you will use Markdown to complete technical writing tasks and the Python programming language and many Python packages (e.g., Pytest for automated testing) to implement a production quality software. As a side effect of working in a team, you will also experience challenges, such as the creation of merge conflicts in a version control repository, that force you to develop practical solutions. You will also gain experience in talking with team members and leaders, technical leaders, and the course instructor. Students will work together in a development team while mastering the technical and professional skills in the field of software engineering, completing a weekly assessment of their progress. Students will also adhere to a code of conduct that governs how all team members will interact during the completion of software products. Students will regularly receive both percentage and mastery grades on this project.

Reading Assignment

If you have not done so already, please read all of the relevant “GitHub Guides”, available at https://guides.github.com/, that explain how to use many of the features that GitHub provides. In particular, please make sure that you have read guides such as “Mastering Markdown” and “Documenting Your Projects on GitHub”; each of them will help you to understand how to use both GitHub and GitHub Classroom. To do well on this assignment you should also read all of the assigned chapters in the following textbooks: Cooperative Software Design, Head First Software Development, Engineering Software as a Service, Think Python. You are also expected to find and read all of the online resources that you need to complete this software project. Please see the course instructor if you have questions on these reading assignments.

In addition to these reading assignments, you are to read and learn about the structure of ZenHub. Specifically, please watch two and a half minute videos on ZenHub at https://www.youtube.com/playlist?list=PLFIGvQyXSp3CGS8X300Aj-hUZ-VLiHbK. You can learn more about ZenHub by reading one of their ebooks at https://www.zenhub.com/resources#ebooks.

Developing and Releasing Text Mining Tool

Unlike the practical assignments, this laboratory assignment asks you to work in a entire-class team to create and release a software product. For this laboratory assignment, your task is to collaborate with the members of your class to design, implement, test, deploy, and maintain an enhanced text
mining tool that measures technical responsibility of developers. You can learn more about this tool, called GatorMiner, by visiting its repository: https://github.com/Allegheny-Ethical-CS/GatorMiner. GatorMiner was developed by Dr. Bonham-Carter and our TL, Enpu You, as a part of the project for the Mozilla Foundation to incorporate ethics into computer science. There are many software engineering tasks to complete before this tool is ready for production use by faculty at Allegheny College and other academic and industrial institutions. You should review GatorMiner’s issue tracker in GitHub to learn about some of the features that you must implement in order to successfully complete this software project. Since this assignment’s goal is for all students to work together to finish a production quality version of GatorMiner, the tasks for this project are not limited to those that are currently in the issue tracker. In fact, you are required to go through the process of requirement gathering, develop user stories, and convert them to specific tasks. In summary, you are required to implement additional features in GatorMiner to ensure it is production ready.

Software Engineering Collaboration

To encourage expedited implementation of the features and to keep team-work manageable, our class will adopt agile scrum framework (with some adaptations for the classroom environment). As we discussed in class, there are three main roles in scrum framework: development team, scrum lead, and product owner. Their descriptions, taken from the text book are below. In addition to these roles, there are stakeholders involved, in the form of customers/clients.

- **Development team**: Individuals responsible for delivering the software.

- **Scrum Lead**: A team member who acts as buffer between the development team and external distractions, keeps the team focused on the task at hand, enforces team rules, and removes impediments that prevent the team from making progress.

- **Product Owner**: someone who represents the voice of the customer and prioritizes user stories.

In our adaptation of these roles to a classroom environment, your instructor will serve in a role of a customer, while technical leaders will act as product owners and will help each team identify and prioritize tasks. Each team will consist of 4-5 individuals, including a scrum lead and a development team. Teams will be self-selecting, taking into consideration a range of skills identified by each member. To ensure cohesive implementation of a full-featured tool, we will also select a “Chief Scrum Lead”, a person who will oversee the technical development of all teams' work and ensure collaboration among teams when necessary.

The project will utilize ZenHub project board to manage tasks and to ensure timely completion. If ZenHub tab is not visible for you on GitHub, you may need to add a browser extension (https://www.zenhub.com/extension). The iterations will be completed in sprints of one week each. After each team selects a feature to implement, they are to organize their sprints. Each sprint will consist of multiple related tasks that a team can complete in a week. Each developer can be assigned a specific task in a sprint or multiple developers can work on a single task together. After a sprint ends and team conducts sprint review and retrospective, a new sprint is planned for a subsequent
week and the board is updated accordingly. Ideally, each sprint should produce incremental working implementation of a task. Incomplete tasks after a sprint must be moved back to the backlog and be dealt with during the next sprint planning. Sprint review, retrospective and planning will occur during each lab session.

Your team should use GitHub and its features (e.g., issue tracker, pull requests, commit log, and code review request) to complete all of the tasks referenced in the previous section. Aiming to manage risk and estimate the effort required for individual team members to complete this project, you should assign people to specific roles and tasks. While it is acceptable for you to have in-person discussions with your team members or to talk about the project through Slack, please remember that all important discussions and decisions must be documented through GitHub and ZenHub board. Finally, as you are working with your team, you should carefully document your experiences and contributions so that you can share them through writing stored in the repository created by GitHub Classroom.

Since multiple approaches may support the effective completion of the required software, this assignment does not dictate team organization or communication strategies. The students in the course should instead work with each other, the technical leads, and the course instructor to identify team roles and strategies for effective organization and communication. With that said, you should plan to use either forks or branches of a GitHub repository to organize your work. For example, each feature implementation could live in a separate branch (called feature branch). Once a specific branch/fork contains the finished version of its associated feature, a team member should create a pull request for discussion and review. If the team leaders, the technical leaders, and the course instructor judge that the pull request has all of the expected characteristics, then it should be merged into the “main” branch of the GatorMiner repository. If the pull request is not accepted, then team member(s) should improve it until it meets every reviewer’s expectations. Your team should continue to use this model, called “GitHub flow”, to support the completion of all deliverables.

Ceremonies

At the minimum, three scrum meetings a week are required, two of which will occur at the beginning of each class and practical session. During a scrum meeting, each team member should take turns answering the following questions, you will also be required to document these answers in your reflection document for each lab.

• What did you since last meeting?

• What will you do before the next meeting?

• Are there any obstacles in your way?

Sprint review, sprint retrospective and next sprint planning will all occur during each week’s lab session. The sprint review is an informal meeting between the development team, the scrum lead, the product owner and the stakeholders. The team gives a demonstration and will discuss which planned tasks are finished and which are not.
After the sprint review, each team (development team, scrum lead, and possibly product owner) will conduct a **sprint retrospective**, which is a meeting time used to figure out which things the team is doing well, what should be continued, and what else can be done to improve the next sprint.

Finally, after the sprint retrospective, each team should engage in the next **sprint planning**, so the next iteration could begin. The goal of the spring planning is to determine the sprint plan and set a sprint goal, specifically which tasks in the backlog to tackle in the sprint.

**Skill Assessment and Code of Conduct**

Each week, you will receive a separate GitHub repository that will contain links to the assessment sheet and code of conduct that we jointly created during lab 1. Please copy those files into your weekly repository and complete them fully. For example, each week you will be asked to sign the Code of Conduct to indicate that you adhered to it throughout the completion of the software project. Additionally, each class member will be required to assess their efforts during each sprint and use the assessment document to record their progression of technical and professional skill development. You should thoughtfully reflect on your current areas of expertise and opportunities for improvement. After each sprint, using the weekly repository, you should also document the work that you completed for this project, basically reporting on scrum meeting discussions.

**Suggested Schedule for the Software Project**

The course instructor invites the students in this class to work together to devise a schedule by which they can complete the software product by the stated deadline. Overall, you will work on this assignment for seven weeks. Here is a suggestion for a schedule to complete GatorMiner.

- **Sprint 0** (Week One: March 17-24): After organizing your team and then learning how to build, test, and use the current version of GatorMiner, review the existing issues and organize them into tasks using appropriate information and labels, create new user stories and create new tasks/issues in ZenHub. Once you have finished these tasks explore the Python language features and packages you will need to implement some of the features.

- **Sprint 1** (Week Two: March 24-31): Identify a feature or features and associated tasks that your team will implement during the first sprint. Each task should have a title, acceptance criteria, estimate, priority, appropriate label identifying the team. Assign tasks to the team members. Implement the tasks associated with your selected features.

- **Sprint 2** (Week Three: March 31-April 7): Plan to finish implementing a feature(s) from sprint 1 ensuring it is bug-free and fully tested. If the feature has been fully implemented and tested, make a pull request to incorporate it into the main branch and then identify another feature or features to implement during sprint 2.

- **Sprint 3** (Week Four: April 7-April 14): Plan to finish implementing a feature(s) from sprint 2 ensuring it is bug-free and fully tested. If the feature has been fully implemented and tested, make a pull request to incorporate it into the main branch and then identify another feature or features to implement during sprint 3.
- **Sprint 4** (Week Five: April 14-April 21): Finish implementing all key features for the tool and complete all major fixes, ultimately leading to a release of a production quality tool suitable for use by others.