Finalization of Initial Project Tools and Techniques:

Based on the unique elements of your domain, project, team, and sponsor expectations, your team needs to select appropriate tools to be used throughout the engineering cycle going forward. In plan-driven models, this is part of initial planning; in agile, this is sprint 0 decision making. After your team, sponsor, and coach have agreed to appropriate tools, indicate your selections below. Note that your team (with appropriate approval from sponsor, etc.) is free to change toolsets whenever the need should arise - this is just a snapshot of this moment in the project. Also note that these selections will appear, in some part, in the documentation your team will create within your project plan.

- Project Management Tools - Requirement or User Story Organization, Communication, etc.
    - JIRA
    - Discord for inter-team communication.
    - Google Drive for centralized document management
    - Email
- Technical Development Tools - Versioning, Testing, etc.
    - TBD
- Technical Implementation / Tech Stack - Server, Platform, etc.
    - GitHub Pages for the website.
    - TBD

Development Methodology:

Based upon what you've learned in your academic career and in your co-op experience as well as the individual elements of your project and team, you will need to choose, document, and justify an appropriate development methodology. This goes beyond merely identifying a model, it involves deciding many elements of how your team will work.

The team may be free to select the general methodology that the team believes is most appropriate for the project, or your project sponsor may specify the methodology that you will use, with your team working out the details. The plan should lay out the general approach that you will use throughout the project. For some methodologies, you may be able to identify planned deliverables through all phases of the project. For all methodologies, you must identify when the sponsor can expect to receive any sponsor-required artifacts and deliverables, as specified in the original submission and subsequent discussions.

This element will need to be discussed with and approved by both your coach and sponsor. The resulting plan will need to appear as part of the team's assembled project plan, and a simplified version should be published on the team's website. At the end of your project, this element should be a part of your technical report.

- Planned Activities and Artifacts - How will these proceed? Cyclical?
    - Scrum
    - Bi-weekly sprints ending on Tuesdays.
    - Retro and sprint planning every sprint on Tuesdays.
    - Standups every meeting. Tuesdays first thursdays after meeting.
    - Weekly progress updates and deliverables every other week.
- Roles - How they interface with your methodology
    - Scrum Master who manages the scrum board for us and leads standups
    - Website Manager is responsible for handling metrics
    - Communications Liaison is responsible for taking discussions and sending appropriate information to the sponsor
    - Project Manager keeps the team on track of the methodology.
- Standards and Quality Practices - How will testing, deployment verification, etc. work?
    - Code Reviews for everything
    - Hold UI to W3C standards.
    - CI/CD pipeline through github.
    - Testing of some sort (unit at least).
    - Commenting standards.
    - No magic numbers
    - Standardize variable naming
- Tools - What tools will help with your chosen methodology?
    - Jira
    - GitHub
- Metrics and Measurements - Another element of the project plan that will affect and be affected by this one.

- Velocity
- Man-hours
  - Expected vs Actual
- Mouse clicks per operation
- Response time
- Defect Density
  - Per TLOC/Page
- Cyclomatic Complexity
- Test Metrics:
  - % code coverage, % passing tests, num tests per user story

Our team is planning on using Scrum as our primary development methodology. Our sprints will last two weeks, beginning and ending every other Tuesday.

- At the end of every sprint, we will have a retrospective before beginning sprint planning.
- Every Thursday, we will present a progress report to the sponsor, along with a demonstration of any deliverables completed.
- Every Tuesday and Thursday will have a daily standup.
- If team members need to meet for any other reason, then meetings will be scheduled on-demand.

We have set up a Jira board to keep track of our sprints. We're planning on using requirements instead of user stories, as we feel they are easier to write and work with. Tasks will be created for each deliverable, and split into subtasks, which will be delegated among the members of the team. We have metrics that can be used to determine task delegation, point values, and task complexity.

- We will monitor our velocity in Jira to make sure we are progressing in a timely manner.
- We will chart our expected man-hours per week and compare them to the actual man-hours per week. This allows us to make sure each member is spending adequate time on the project each week, and that the time is spent efficiently.
- We will use the number of expected test cases per task to determine the complexity, and potentially split it into subtasks.
- We will use the expected man-hours per task to determine if it needs to be split into subtasks.

We also have a variety of standards planned to make sure our project holds up to RIT standards. Our primary method of doing this is through peer reviewing everything.

- Any task listed on our Jira which requires code to be written must have that code reviewed by at least one other team member.
- Any task that changes the UI must have that change reviewed by at least one other team member and approved by the sponsor.
- Any task which requires large-scale architectural or UI design must be reviewed by the entire team, and approved by the sponsor.
- Any task which is a documentation deliverable must be reviewed by the team, and approved by the coach and sponsor.

During these reviews, we will be looking for a few specific things.

- Any code written must be commented where necessary.
- Code should avoid magic numbers wherever possible.
- Variable names should be standardized (no current standard selected).
- Tests should be written for the code, if possible and reasonable.
- UI changes must be held to W3C standards.

We are planning on using GitHub as our primary repository for the project. Branches will be created whenever new functionality is worked on, and branches will not be merged into main

unless they pass a code review. We will not be performing code reviews for every commit, only for task completion and pull requests.

In addition to code reviews, we also have a variety of metrics that we plan to keep track of to make sure code is being held to standard.

- Mouse Clicks per Operation, which will be used to make sure the website UI is effective.
- Response Time, which will be used to make sure the UI and database aren't unnecessarily laggy.
- Defect Density, which will be tracked on a page, method, and TLOC basis.
    - Defects per page will be used to inform UI design decisions and determine if a redesign is necessary.
    - Defects per method will be used to inform architectural design decisions and determine if a refactor is necessary.
    - Defects per TLOC will be used to get an overall idea of the code quality for the project.
- Code Coverage Percentage, which will be used to determine how effective our unit tests are.

In order to make sure all of this runs smoothly, each of our team roles has designated responsibilities.

- The Scrum Master manages the Jira board. They create and assign tasks, split tasks into subtasks, and make sure requirements are well defined. They also keep the team updated to changes in velocity.
- The Communications Liaison is responsible for keeping the coach and sponsor in the loop. They make sure both are informed of any information pertinent to their role in the project.
- The Website Manager is responsible for keeping track of metrics and displaying them on the team website. Because all metrics need to be displayed on the website anyways, delegating metrics management to them is the easiest.
- The Project Manager is responsible for leading project meetings and making sure everything runs smoothly.

We believe that this is the most effective methodology for our team. It allows us to keep up consistent progress and communication with the sponsor, while giving us enough flexibility to make sure we are able to adapt in the face of changing requirements. It has enough structure to make sure we're spending our time efficiently, without requiring us to spend more time on our process than needed.

Domain Model:

Early in the first semester, you will create a domain model for your project. This will provide a common understanding between the team and project sponsor of the domain and scope for the project. The domain model is not a high-level implementation design. It uses only the vocabulary of the sponsor and domain. Unless software engineering is the domain of the project, the domain model should be devoid of software terminology. There are many videos and examples available online, possibly even related to the particular domain of your project. A slide deck covering the basics is available here.

This element will need to be discussed with and approved by both your coach and sponsor. The resulting output will need to appear as part of the team's assembled project plan, and a simplified version should be published on the team's website. At the end of your project, this element should be a part of your technical report.

- Forums Page
- Private Messaging Page
- Listings Page
- Database
- Upload Page
    - Listing
    - Video
    - Diagrams
- Login Page
- Main Page
- Simple Secure Storage
- Payment Page?

Process and Progress Metrics:

In addition to tracking individual and total time/effort worked on the project, you are to select at least one product and at least one process metric that the team will track. The team should select metrics that are appropriate for the project and process methodology chosen. It may be later in the project when the product and process metrics will have meaning, but at the least you should start tracking each metric as soon as possible. At any point in the project, if the team feels that one of the chosen metrics is not paying benefits, the team can choose to start tracking a different metric - as long as the team continues to track time/effort, product, and process in some manner.

A List of Useful Metrics

The metrics below can be used for traditional waterfall/iterative development as well as agile approaches, and can also be applied to projects other than pure development work.

1. **Progress metrics:** These show whether project execution is on schedule
    - Slippage chart: Shows number of days you are ahead/behind schedule, based on planned dates for reaching milestones. To use this, you would plan your project as a series of milestones, identify target dates for reaching each, track actual dates of reaching each milestone, and plot the difference
    - Earned value chart: Identify the activities for your project, and associate an earned value (# of points) with each. You earn the value (only) when you complete the activity - no points for partially complete activities. Plot total points earned against time
2. **Defect metrics:** Keep track of errors in your artifacts: requirements, design, code, test cases, process definition etc. The best practice is to use a defect tracker and enter problems as they are found, and also track the status of the defect (found, assigned, fixed, deferred). From this information, you can derive several metrics
    - **Defect density: number of errors per KLOC, page, use case**
    - Pie chart of defects by type (include raw number and density)
    - Bar chart of defects by module
    - If you are conducting code inspections on all deliverables, you can create a Defect Removal Effectiveness chart, which is a grid showing the phase in which each defect originated, and phase it was found. Very useful to measure effectiveness of inspection activities and also phase activities, but probably too much work for the size of projects you are doing
3. **Effort metrics:** Keeping track of hours spent
    - **Estimation accuracy: Planned vs actual effort for each activity**
    - Effort by type of activity: Classify effort based on activity type: requirements, design, implementation, coordination, documentation, customer interaction etc. Useful to identify where your time is going, so that you can try to improve efficiency in that area
4. **Activity metrics:**

    - Requirements metrics: Requirements volatility (% of requirements that were changed during each week)
    - Design metrics: methods per class, depth of inheritance hierarchy, fan out (# of methods called per class or per method)
    - **Test metrics: Coverage, # of test cases per requirement (use case, user story etc), % of tests passed successfully**
    - **Code metrics: cyclomatic complexity**, LOC per method
    - **Documentation metrics**
5. **Other metrics:**
    - Productivity: Volume of deliverable produced per unit effort (e.g. pages per hour, LOC per hour)
    - **Product quality metrics: mouse clicks per operation (interface usability), average response time (performance)**
    - Average time to fix bugs / respond to customer requests
    - Other metrics suggested by the sponsor or preferred by the team

This element will need to be discussed with and approved by both your coach and sponsor. The resulting documentation will need to appear as part of the team's assembled project plan, and most metrics should be published and tracked on the team's website. At the end of your project, the final results of this element should be a part of your technical report.

Progress Metrics:

- The primary progress metric we will be using is velocity.
- We will be giving every task and subtask a point value, and using that to determine how many points we are doing per sprint. The more points we have in a given sprint, the more progress we make.
- We can track this over time, and display it in a graph or chart on our team’s website.

Defect Metrics:

- We will use a defect tracker in our Jira or GitHub to log them and use that data to inform our analysis.
- The number of defects in a task can be used to determine if the task is ready to be marked as finished, or if it needs more work.
- The number of defects on a page can be used to determine if the page is well designed or not.
- The number of defects per thousand lines of code can be used to determine how generally effective our coding practices are.

Effort metrics:

- We plan to keep track of how many hours of work we expect to do per week, vs how many hours of work we actually do. This can be used to make sure everyone is putting in the necessary amount of effort to get things done.
- This is easily tracked on our team website, using a bar graph or a table.

Activity Metrics:

- We plan on having a variety of test metrics.

    - The code coverage percent will be used to make sure all of our code is being properly tested.
    - The percentage of tests passed will be used to help inform defect metrics.
    - The number of tests per task can be used to determine how tasks should be split into subtasks.
- We also plan to hold ourselves to a documentation standard, and we can keep track of how many of our functions are properly commented.

Performance Metrics:

- We intend to keep track of the response time for the UI and database to make sure performance is up to par. This will likely not be shown on the website, but will still be tracked for the sponsor's sake once we finish.
- We want to keep track of mouse clicks per operation to make sure that the most important things are easy to do. Given that accessibility is important for this application, this is a high priority.

Additional Elements:

In addition to the methodology, domain model, and metrics (in other actions), your team needs to consider the rest of the plan, appropriately considering and drafting elements as needed by your particular domain, project, team, and sponsor. These elements should include, but are not limited to:

- Project goals and scope statement - this may be in part obtained from the initial sponsor submission, available in the "Projects" tab
- Planned milestones and major deliverables
- Initial requirements, user stories, and/or work breakdown structure
- Communication and stakeholder management plan
- Risk management plan - this may be needed if the weekly 4-Ups are insufficient to cover the project's anticipated risks and planned mitigations
- Other elements as needed

These elements as well as the completed project plan will need to be discussed with and approved by both your coach and sponsor. If appropriate, they should be published in some form on the team's website. At the end of your project, these elements should be a part of your technical report.