CS Capstone Project Management Plan

Date: See SLATE \ Assignments
Type: Individual Submission
Type: **Group Assignment**
Grade: 10%

Summary

Create a project management plan that divides and allocates project work to team members, divides the project into incremental development iterations (i.e. sprints) and defines a risk management plan. The team uses JIRA for project management and creates a site that contains two JIRA projects: Project Management Plan (PMP) and Risk Management Plan (RMP). PMP describes the work-breakdown structure (WBS) of the project as well as the project schedule in the form of a release iteration plan and roadmap while the RMP provides a mechanism for the team to conduct their risk management process.

Submission checklist

1. **JIRA site** link. JIRA site and projects must be shared with the professor such that full access is granted. The JIRA site must contain two JIRA projects that use next-gen templates: PMP (SCRUM Next-Gen) and RMP (Kanban Next-Gen).
2. Link to the **PMP JIRA project** submitted in SLATE as well as added to the team's channel.
3. Link to the **RMP JIRA project** submitted in SLATE as well as added to the team's channel.

NOTE: While this document describes execution and finishing of a sprint please note that the sprint does not need to be completed as part of this assignment. This information is provided to describe how the PMP and RMP are used.

Detailed Requirements

Part I (40%) Work Breakdown Structure (WBS). Create an JIRA site dedicated to your project (https://support.atlassian.com/jira-software-cloud/docs/start-a-new-software-project-for-your-team/). Within the site, create *a SCRUM next-generation* JIRA project that is used as the project management plan (PMP). PMP identifies the tasks to be completed for the project recorded as releases, use-cases, stories, tasks and dev subtasks.

1. Rename the "Epic" issue type to **Use-Case** and add a custom field named "*Use-Case Type*" to specify the type of use-case represented by the issue[1]. The following use-case types shall be configured and used in PMP:
    - *Summary*: identifies a summary use-case

[1] Due to a product limitation, JIRA does not allow multiple "epic" task types which means that use-cases must be used to organize both tasks that implement functionality as well as non-functional tasks

- *User-Goal*: identifies a user-goal use-case which represents a user-requirement
- *Sub-Function*: identifies a low-level reusable use-case that is included in one or more use-cases and does not represent a stand-alone user goal
- *Infrastructure*: identifies a pseudo use-case (PUC) used as collection of tasks that are not use-case specific and that need to be completed to create the technical infrastructure of the project (see 4.d Development Infrastructure Tasks)
- *Support*: identifies a PUC used to plan and manage support activities that need to be completed by the team (see 4.e Project Support Tasks)
- *Research*: identifies a PUC used to plan and manage the research tasks the development team needs to do (see 4.f Research Tasks)

2. Add a custom dropdown field to the story issue type named **Story Type** to identify the following types of story:
    a. *User Story*: a fragment[2] of a use-case that is *user-driven* and can be completed within a single iteration / sprint.[3]
    b. *System Story*: a fragment of a use-case that is *system-driven* and can be completed within a single iteration / sprint.[4]
    c. *Dev Story*: a task that is required for the implementation of a specific use-case, that is *developer-driven* and that can be completed within a single iteration / sprint.[5]

3. Rename the "Subtask" issue type to "**Dev Subtask**" for better alignment with the development process. Subtasks are determined during sprint planning by the developer assigned to a given task.
    a. Add a number field named "Effort Estimate" where the developer can track the number of person-hours they estimate the task to take.
    b. Hide the story point estimate as it will not be used for dev subtasks.

4. Create the project WBS through the **product backlog** by adding use-cases, stories and dev subtasks as follows:
    a. Add the user requirements that have been identified as part of the project inception by adding them as **use-cases** to the backlog.
        i. Describe the use-case in story format in the *Description* field[6]
        ii. Identify the use-case type as "user-goal". Summary and sub-function use-cases will be added when requirements analysis activities are performed.
        iii. Identify the functional area the use-case is part of using a JIRA tag.

[2] one or more steps
[3] e.g. "As a student, I want to enter the username and password and request to log in so that I can access the system securely"
[4] e.g. "As a SYSTEM, I want to verify the credentials entered by the user and determine access rights so that users access the system securely"
[5] e.g. "As a DEVELOPER, I want to refactor the password encryption mechanism to use a hash with password salt so that I improve the security of the system"
[6] e.g. "As a doctor, I want to access the system securely so that the privacy and integrity of all medical information is maintained according to regulations and best practices."

b. Select a use-case per team member and **create stories** for functional requirements known at this time for each use-case. Associate the stories with their corresponding use-cases as they are entered in the backlog. [7]
c. Select at least one story per team member to be developed in the first sprint. Each team member shall identify the **dev subtasks** for the story(ies) they have signed up for and estimate the effort required for each task in person-hours.
d. Identify critical **infrastructure high-level tasks** and plan them as *Infrastructure PUCs*[8] by adding them to the backlog. Infrastructure PUCs are not specific to a particular use-case but apply to the system as a whole such as: administration, architecture, system documentation, system testing, integration, database design etc.
    i. Identify some of the *task issues*[9] needed to be carried out to design, create and maintain the infrastructure required to develop the project and associate them with the corresponding infrastructure PUC[10].
e. Identify high-level **project support tasks** and plan them as *Support PUCs* by adding them to the backlog. These represent work required to support the project such as planning, project management, meetings, code reviews, brainstorming sessions etc.
    i. Breakdown each support PUCs into tasks that are added to the backlog to identify the work that is needed[11].
f. Identify **high-level research tasks** as *Research PUCs* such as technologies, domain, existing applications, frameworks, industry and market as well as the creation of throwaway prototypes designed for investigation of research topics.
    i. Breakdown each Research PUC into tasks required for the research to be completed.[12]
g. Create JIRA filters to display important task categories in the project plan as follows:
    i. Identify each functional area of the system, as defined in the requirements model, using a JIRA filter, named using the name of the functional area and showing all use-cases with a given functional area tag[13]
    ii. Identify the list of all use-cases identified for the system ordered by functional area[14]
    iii. Identify all the system infrastructure tasks (PUCs)[15]
    iv. Identify all the research PUCs[16]

[7] In the inception release, each team member is required to analyzed and fully define one use-case.
[8] Use-case issues with the use-case type set to "Infrastructure"
[9] as opposed to story issues
[10] e.g. A system that uses a database need an infrastructure PUC such as "Create Database" with tasks such as "Identify tables", "Create 'Users' table", "Analyze and describe patient data using ERD diagram(s)", "Normalize the database" etc.
[11] e.g. SCRUM methodology requires sprint review and retrospective at the end of each sprint and a sprint planning at the beginning of each sprint.
[12] e.g. A system that uses Augmented Reality would need to research various AR frameworks available on the given platform
[13] JIRA query: *labels = <functional area name>*
[14] JIRA query: *issuetype = "Use Case" AND ("Use-Case Type[Dropdown]" is EMPTY OR "Use-Case Type[Dropdown]" in (Summary, User-Goal, Sub-Function)) ORDER BY labels, summary ASC*
[15] JIRA query: *issuetype = "Use Case" AND "Use-Case Type[Dropdown]" = Infrastructure ORDER BY summary ASC*
[16] JIRA query: *issuetype = "Use Case" AND "Use-Case Type[Dropdown]" = Research ORDER BY summary ASC*

v. Identify all the support PUCs[17]

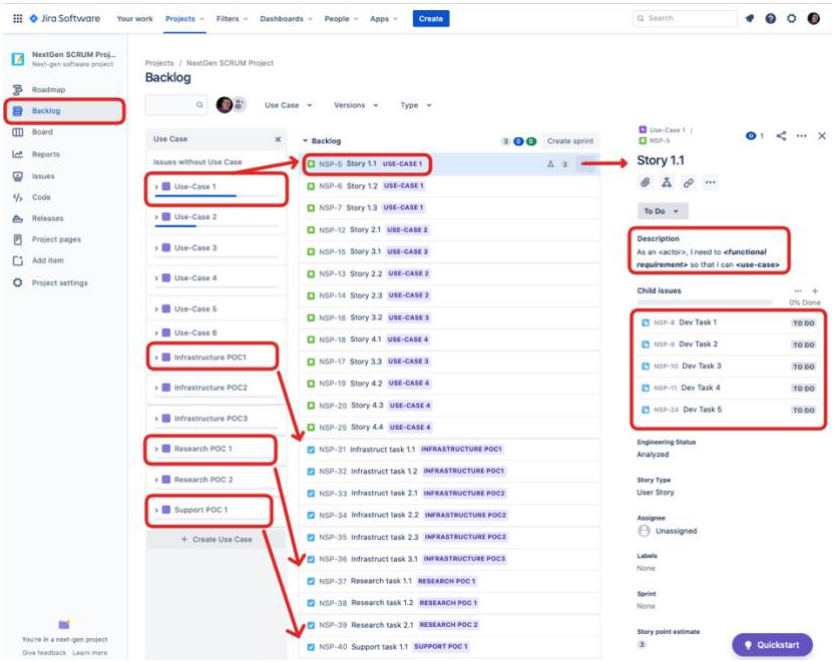

Figure 1: WBS Views with JIRA Filters

Figure 2: Work Breakdown Structure (WBS) with JIRA

Part II (20%) Release Plan (RP). In the Backlog of the same JIRA PMP project, create a release plan by creating a set of iterations (sprints) and releases. The release plan shall contain all the iterations you envision will be needed to fully implement the project not just the iterations until the end of the term. It is understood that the project, by design, would need to extend past the end of the term given the problem's complexity.

1. Iterations that take places this term shall be 1 week in length while iterations that are planned past the end of the term shall be 2 weeks in length
2. Each iteration shall have a clear goal written from a user perspective that describe the goal for the potentially releasable product increment.
3. Each sprint name shall follow the naming convention *Iteration <n>*[18].

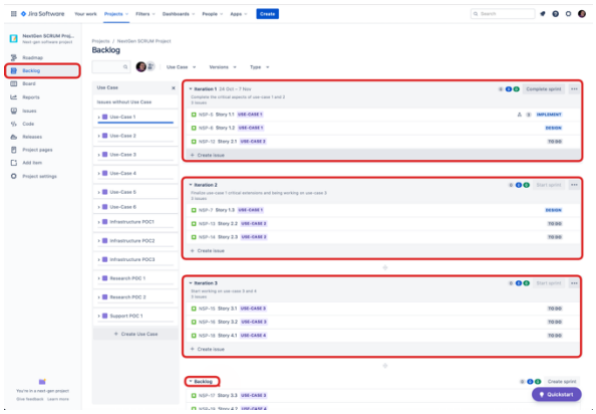

Figure 3: Release Plan

[17] JIRA query: *issuetype = "Use Case" AND "Use-Case Type[Dropdown]" = Support ORDER BY summary ASC*
[18] Once you create your first sprint, JIRA will continue to use the same naming convention.

4. In Releases, create five releases: Inception Release, Elaboration Release (the two releases this term), Alpha Release, Beta Release and Final Release (happening sometime in the future).

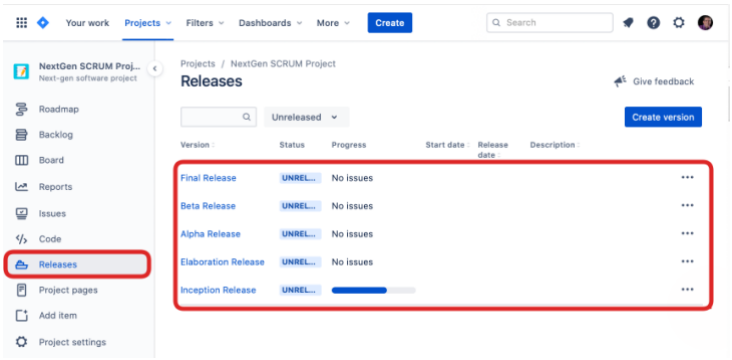

Figure 4: Project Releases in JIRA

5. Plan and start the first sprint

Sprint Planning

1. As a team, select the product backlog items (PBIs)[19] to be completed in the sprint. Evaluate PBIs for readiness, priority and completeness. New PBIs can be added as determined by the product owner or removed. If PBIs are removed, they shall be added back to the backlog.
2. Collectively as a team, discuss the relative complexity of the stories selected and assign story points according to the following strategy:
    a. 1 story-point: small tasks that can be done in one short sitting[20]
    b. 2 or 3 story-points: small-to-medium tasks that can be done in a longer sitting
    c. 5 story-points: medium items that require multiple sittings[21]
    d. 8 story-points: large item that will take several concentrated periods of time
    e. 13+ story-points: feature that has to be broken into smaller stories before it can be allocated to a sprint
3. Each team member must select and assign themselves at least one story

Sprint Execution[22]

1. As team members work on PBIs they move them through the following workflow as shown in **Error! Reference source not found.**:
    a. *Analyze*: PBI is divided into dev-subtasks and effort is estimated for each subtask. Any requirements details are analyzed in the requirements model.
    b. *Design*: PBI implementation plan is prepared by designing the necessary interaction in the Interaction Model and adding necessary elements to the Design Model

[19] Could be stories or tasks. The first iteration shall consist at least of some user stories that provide some functionality that is visible to the user
[20] One of the strategies used by teams is to think of the smallest story that has a user outcome and give that 1 point. The rest of the point measurements are relative to this unit.
[21] Another strategy used by teams is to think of a story of a medium complexity, in the middle of the complexity spectrum and assign it 5 points. All other measurements are then related to this, either lower for simpler stories or higher for more complex ones.
[22] While sprint execution is presented here for completeness, the assignment does not require that you complete the sprint.

c. *Implement*: PBI is implemented and debugged
d. *Test*: PBI is tested through unit-tests or manual testing as necessary
e. Done: PBI code is integrated into the master branch

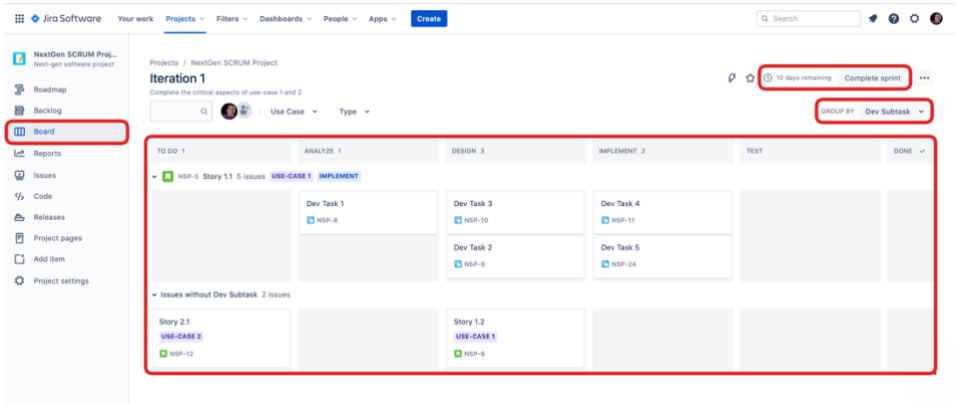

Figure 5: Sprint Board is used to manage the execution of the sprint

2. Add comments to PBIs as work is done to identify any roadblocks, technical debt[23], dependencies.
3. The SCRUM master monitors progress, determines any bottlenecks and ensures the team communicates efficiently.

Sprint Finish

1. *Sprint Review*. Conduct a sprint review meeting and review the status of all the PBIs in the sprint as follows:
    a. Evaluate the story point estimation to identify miss-estimations. Compare estimation with actual effort
    b. Move any PBIs that were not started in the backlog to be considered in future iterations
    c. PBIs that were partially completed within the defined iteration time shall be revised to identify the work completed and record it as such. Revise the name, description and dev tasks of the partially completed PBI as necessary to reflect the work completed. Create a new PBI to record the work remaining and the associated dev tasks and place it in the backlog. Comments shall be placed to describe the breakdown and the reason the PBI could not be completed. The two PBIs shall be linked together for traceability.
    d. At the end of the sprint review activity, all the PBIs shall be marked as completed and the sprint itself is completed.
    e. Review the risks recorded and managed in the risk management plan.

2. *Sprint Retrospective*. Conduct a sprint retrospective to review the process used, the communication between team members, how dependencies and risks were managed and identify any improvements that are necessary in the next sprint.

[23] shortcuts taken

Part III (20%) Risk Management Plan. Create Kanban Next-Generation JIRA project that will be used to manage the risks throughout the project.

1. Add a new issue type named "Risk" with the following custom fields
    a. *Mitigation Strategy*: paragraph field that identifies mitigation strategies
    b. *Probability*: dropdown field that identifies the probability of the risk to be one of Very High, High, Moderate (default), Low
    c. *Impact*: drown-down field that identifies the impact on the system to be one of Catastrophic, Serious (default), Tolerable, Insignificant
    d. *Risk Area*: drop-down field that identifies the area of risk to be Project (default), Product or Business
2. Delete the Epic and Task issue types as they are not used in this project
3. Configure the Kanban board to have columns for each phase of the risk management process:
    a. *Identification*: risk is being identified with a summary and an optional description
    b. *Analysis*: risk is being analyzed to determine probability and impact
    c. *Planning*: mitigation strategies are being determined
    d. *Monitoring*: risk is being monitored and updates are being recorded as comments
    e. *Active*: risk is actively impacting the project. Use subtasks to identify the actions that are being taken to mitigate the risk and who is responsible for them
    f. *Resolved*: the risk has been eliminated or is no longer relevant

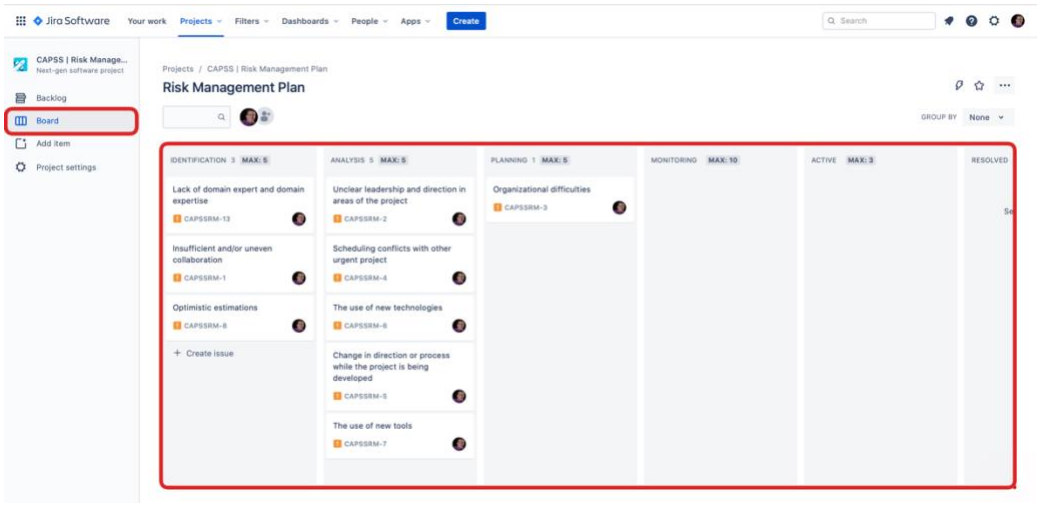

Figure 6: Kanban board used for risk management

4. Identify, analyze and plan risks that are relevant to the project. As you perform this work move each risk through the risk management process using the JIRA Kanban board.
    a. All the risks that are being managed must be present on the Kanban board

b. Risks that are in the backlog are risks that have been considered but have been deemed as not relevant to the project.

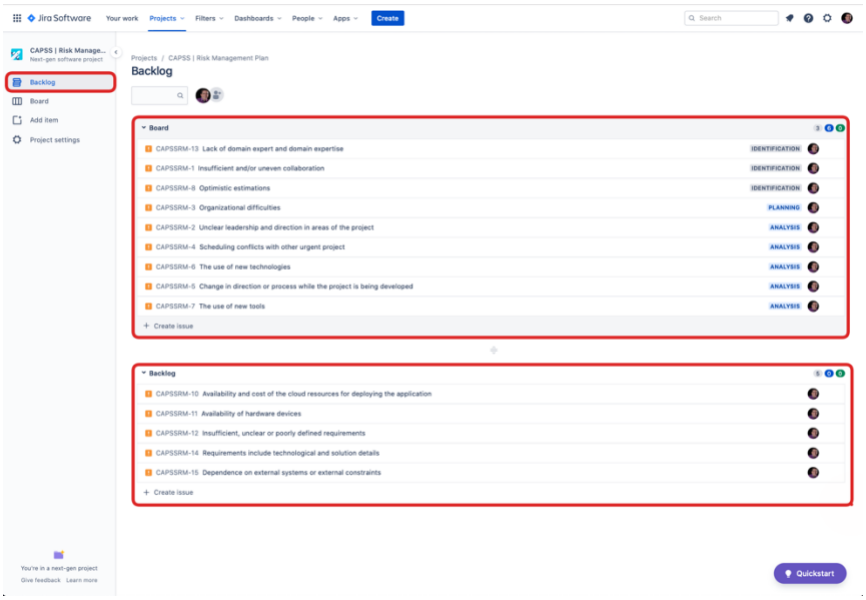

Figure 7: Risk Identification with Kanban JIRA

5. Continuously monitor risks throughout the project and post updates as comments at the end of each sprint as part of the sprint review activity.
6. Identify any active risks and determine the subtasks needed to mitigate or resolve the risk. Ensure the tasks are assigned and they are completed promptly.
7. Create JIRA filters to identify risks by risk area: project, product and business.[24]

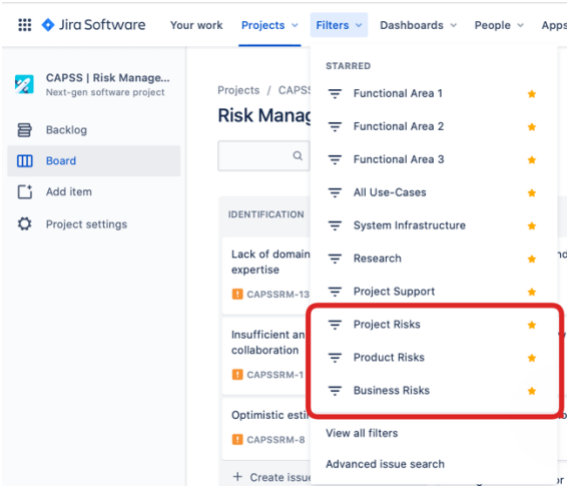

Figure 8: JIRA Filters for Risk Management

[24] JIRA query: *issuetype = Risk AND "Risk Area[Dropdown]" = <risk area>*

Part IV (20%) Project Management Report in the PID. Continue the development of the PID by filling in the section on Project Management which must provide a clear overview of the project planning process and outcome for the project:

1. Describe the project management process that is being used by the team.
2. Using a responsibility matrix as shown in Table 1, describe the allocation of responsibilities in the team:
    a. Identify the allocation project management responsibilities
    b. Identify the allocation requirements engineering responsibilities
    c. Identify the allocation of software architecture responsibilities
    d. Identify the allocation of construction responsibilities
    e. Identify the allocation of testing responsibilities
    f. Identify the allocation of support responsibilities
3. Describe the release plan succinctly by identifying the list of iterations and their goals.
4. Identify the Top-5 risks identified by the risk management plan, their analysis and planning details.

Table 1: Project Responsibility Matrix

Project Responsibility	Team Member 1	Team Member 2	Team Member 3
Project Management			
Project Owner			
SCRUM Master			
Risk Analyst			
Requirements Engineering			
Requirements / Business Analyst			
Stakeholder Champion (by Stakeholder)			
Functional Area Champion (by functional area)			
User Experience Design Lead			
Software Architecture			
Software Architect			
Requirements Model Lead			
Domain Model Lead			

Project Responsibility	Team Member 1	Team Member 2	Team Member 3
Design Model Lead			
Deployment Model Lead			
Interaction Model Lead			
Construction			
Full-Stack Developer (UI, code and unit-tests)			
Integration / DevOps Lead			
Testing			
QA Lead			
Verification & Validation Champion (by Functional Area)			
Test Model Lead			
Support			
Tool and Devices Support			
Communication Support			

Notes:

1. The **professionalism of your submission**, clarity of written communication is extremely important. The ability to communicate your knowledge is as important as the knowledge itself. Up to 40% of the mark for any written work can be deducted due to poor presentation / communication: document organization (10%), layout (10%) spelling (10%), title page (10%)

2. **All assignment shall be submitted by the deadline.** Late submissions will be penalized with 10% per day for up to 3 calendar days after which the assignment cannot be submitted anymore. **An email must be sent** should you choose to submit a late assignment. If no such emails are received the solution will be posted. **Assignments are not accepted after the solutions have been posted.**

3. In this group assignment, **team members must be responsible for contributing an equal share to *EACH PART* of the assignment**. Each individual member must clearly be able to demonstrate

their individual contribution to each of the requirements outlined in this assignment. See the Academic Honesty at Sheridan.

4. Submission is done in electronic format **using SLATE DropBox. DO NOT email your submission.**