# K50 Assignment 1 – Individual Submission

# Section 1 – Individual Setup

## Section 1 Part A – Organization and Plan – Mostly Individual

Objective: Understand the goals of the week. Come up with a plan and timetable.

***Individual Work***

Analyze this assignment and plan.

1. Identify the individual deliverables:

The individual deliverables include:

Section 1

Organizing ourselves and planning out the assignment so we can plan accordingly

Section 3

Prioritize the backlog

Section 4

Versioning and release notes

Parts of Section 5

Planning for the future – deliverables, challenges, effort

1. Identify the team deliverables:

The team deliverables include:

Section 2

Research, presentation, demo, refactoring

Parts of Section 5

Teamwork convergence

1. What is your estimate on when the individual portion will be complete? Negotiate with your team to agree upon a time you’ll meet and organize to start working on the team portion.

The idea will be that we will be meeting up at certain points to work as a team as we make progress individually. Ill be trying to finish all the individual stuff as soon as possible to be prepared for a discussion whenever.

https://github.com/SebastianBurke/W2025.git

1. State your estimate and the team rendezvous time as well as any concerns or risks. These must be written down and stated professionally and in a detailed fashion.

Our team has prepared a rendezvous Jan 28th for section 2 and Feb 4th for section 5.

## Section 1 Part B – Updating Work items from Dev I - Individual

Based on your Final comprehensive assignment of 420-K40-HR

1. Add your System Test bugs as “issues” in Azure DevOps.

Your bug report must have the details required to reproduce the bugs. Be sure to classify this as a “bug” work item and not a product backlog item.

1. Add Refactoring Proposals as a backlog item with tag “refactor”.
2. Do not worry if there are duplicates with someone else’s bug or refactor task. You’ll collectively clean that up later.

*Please note for future reference that the* ***cloud-based*** *Microsoft Azure DevOps implementation, Microsoft calls everything generically an “issue” even if it is what we in Computer Science would call a bug. This is due to the “Kanban” nature of tasks in Azure DevOps.*

**Update the tables below with your updates:**

**Bug List**

|  |  |
| --- | --- |
| **Azure DevOps Issue URL** | **Bug Title** |
| [Bug 848: Ability to add more than one app admin to an app - Boards](https://csazure.cegep-heritage.qc.ca:8080/F2024/Team%201%20-%20HIA3%20Redux/_workitems/edit/848/) | Ability to add more than one app admin to an app - Boards |
| [Bug 849: Display details about the app(s) that an app admin manages - Boards](https://csazure.cegep-heritage.qc.ca:8080/F2024/Team%201%20-%20HIA3%20Redux/_workitems/edit/849/) | Display details about the app(s) that an app admin manages - Boards |
|  |  |

**Refactor Task List**

|  |  |
| --- | --- |
| **Azure DevOps Issue URL** | **Backlog Item** |
| [Bug 853: Local Authentication Password Hashing - Boards](https://csazure.cegep-heritage.qc.ca:8080/F2024/Team%201%20-%20HIA3%20Redux/_workitems/edit/853/) | Local Authentication Password Hashing - Boards |
| [Bug 852: Add confirmation popup for deletion of app admin - Boards](https://csazure.cegep-heritage.qc.ca:8080/F2024/Team%201%20-%20HIA3%20Redux/_workitems/edit/852/) | Add confirmation popup for deletion of app admin - Boards |
|  |  |

## Section 1 Part C – Review Process and Standards - Individual

1. Review the team’s coding standards document and reconcile against samples of your code. We will be holding code inspections later on. Ideally, your team’s code meets this standard.
2. Find at least 3 defects and submit them to Azure DevOps.

|  |  |
| --- | --- |
| **Defect parameter** | **Possible values** |
| Origin | Requirements, Design, Implementation, Testing |
| Type | Missing, Wrong, Extra, Usability, Performance, Style, Clarity |
| Severity | Major, Minor |

**Code inspection defect report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Filename/DevOps version/line#** | **DevOps Issue #** | **Origin** | **Type** | **Severity** | **Description/Comments** |
| **AppAdminsController.cs (L25)** | **#850** | **Implementation** | **Missing** | **Major** | **Details action returns an empty view. Needs logic to fetch display details.s** |
| **AppController.cs (L78)** | **#1115** | **Design** | **Extra** | **Minor** | **ViewBag is used multiple times instead of a strongly typed ViewModel.** |
| |  | | --- | | **AppController.cs (L49)** |  |  | | --- | |  | | **#1114** | |  | | --- | | **Usability** |  |  | | --- | |  | | **Clarity** | **Minor** | **Index method catches exceptions but does not log errors. Consider logging for debugging.** |

1. Review the team’s coding standards document. If the document is wrong or needs to be updated, now is your chance to make some proposed changes.

Fill in the table below with your proposal. No more than 5 items.

**Proposed Changes to Coding Standards**

|  |  |  |
| --- | --- | --- |
| **Proposal #** | **Category** | **Proposal/Description** |
| **1** | Error Handling | Require logging of exceptions in a central logging service instead of only using ViewBag.ErrorMessage. |
| **2** | |  | | --- | |  |  |  | | --- | | **Controller Actions** | | Ensure Details actions fetch and return appropriate data before rendering the view. |
| **3** | Code Readability | Replace ViewBag with strongly typed ViewModel to ensure type safety and improve readability. |

1. Reflect on last semester’s sprints to generate process change proposals

Identify if new lessons learnt should be added. Define TWO process changes. Modify the below table as necessary.

What behaviours, approaches or processes would you change to improve the sprint process this semester to improve in the following categories:

* 1. Ability to deliver and complete User stories
  2. Ability to deliver quality (cover code standards, white and black box testing)
  3. Team and individual evaluation and scoring
  4. Meeting overhead (scrums, sprint reviews/planning, retrospectives).

**Proposed Changes to Process**

|  |  |
| --- | --- |
| **Category** | **Proposal/Description** |
| **Ability to complete User Stories** | Sit down and break down tasks for each user story to confirm that it is able to be complete. |
| **Ability to Deliver quality** | Reduce scope every day and prioritize quality of code and speed of development. |
| **Team and Individual Scoring** | Keep communication ongoing with teammates to know where everyone is at |
| **Meeting Overhead** | Make time periods that can be consistently used for meeting overhead |

## Section 1 Part D – Budgeting - Individual

1. Review the timeframes for each of the following and provide the available person-hours to be applied to each.

Hint: count the number of your teammates. Look at the course outline for the expected hours per week in this course, then do the math. The course outline also contains the dates for each release. Assume “end of week” deadlines apply to assignments and sprints *( i.e. midnight on Fridays )*.

This is to understand how many hours were available in past timeframes, so that you can map it onto what is achievable moving forward.

|  |  |  |
| --- | --- | --- |
| **Timeframe** | **Start date** | **Available person hours** |
| Release 0.1 – Sprint 1  *(Fall 2024)* |  | Note: ponderation and team size |
| Release 0.2 – Refactor Sprint 1  *(Winter 2025)*  *(2nd dev sprint overall)* | Jan 27th | 120 |
| Release 0.3 – K50 Dev Sprint 1  *(Winter 2025)*  *(K50 Dev Sprint 1, 4th dev sprint overall)* | February 3, 2025 | 120 |
| Release 0.4 – K50 Dev Sprint 2  *(Winter 2025)*  *(K50 Dev Sprint 1, 4th dev sprint overall)* | February 17, 2025 | 120 |
| Release 0.5 – K50 Testing Sprint 1  (Winter 2025) | March 17, 2025 | 120 |
| Release 0.6 – K50 Testing Sprint 1  *(Winter 2025)* | April 14, 2025 | 120 |
| Beta Release 1 – K50 Testing Sprint 2  *(Winter 2025)* | April 14, 2025 | 120 |

# Section 3 - Execution

## Section 3 Part A – Work on backlog prioritized issue list

*Individual Work:*

Take a task (bug or refactoring) from your prioritized list for the Refactor iteration and get working!

This is of course all utilizing the CSAZURE DevOps taskboard, as for every other Dev Proj sprint.

Do NOT work on any task that has not been committed to without agreement with your team and your project manager.

Follow the process. All changes must be reviewed.

There will be a team assessment done for this next week. You will scrum daily on this and treat it like a regular iteration. We’ll have a quick review on the iteration results later in the week.

**To submit**

When you have completed the assignment, the following deliverables are required:

**Individual:**

upload the **YourUserName\_K50\_A01\_SetupReassessRefactor.docx** document to Moodle

**Team:**

Meeting Minutes from Best Practices Presentation (email)

Refactor commit (email)

Memo from Group discussions (Moodle, ref 2.C)

Team self-assessment of commitments. Write a few statements indicating how you performed here, as a team.

Response to commit email at the end of the Refactor evaluation.

(email, end of refactor sprint).

*(“K50 Team #” must be in the email subject line or marks will be forfeited.)*

# Section 4: Versioning, Individual work

For the individual part of the assignment, create a new Word document named **YourUserName\_Teamxx\_K50\_A02a.docs.** Include your (short) username and team number.

Effort estimate for this: < 60 minutes (for a focused individual)

## Part A – Software Versioning

1. Propose a numbering plan on how your product should be versioned. Explain the rules/conditions for when each number is updated and how.

Our product should be versioned by changes. Minor changes will increase the third number by 1. Major changes will that change or add functionality increase the second number by 1. Changes that encompass a large rework of the application will increase the first number by 1.

Release number visualization: x.x.x

1. Propose a mockup of where the software version will be displayed on your product. It should be visible on every page.

Our software version will be displayed in the footer of our application, and maybe near the logo for the application, if we deem this as important.

## Part B – Release Notes

1. Identify the purpose of Release Notes and the intended audience.

Release notes contain information about new enhancements and known issues of the build. They are usually for clients, and can include updates and feature launches, regular improvements, and bug fixes.

1. Identify 3 articles on Release Notes Best Practices. Provide the links and a summary for each. List 5-8 best practices that you’d like to adopt described that you’d recommend.

<https://theproductmanager.com/topics/software-release-notes/>

Be clear

Keep it customer-centric

Highlight bug fixes

Be mindful of the tone

Sprinkle in content

<https://www.launchnotes.com/blog/how-to-write-great-product-release-notes-the-ultimate-guide>

Focus on the value to the customer

Keep them punchy

Use plain language

Include images, videos, gifs, and links

Inject your brand’s tone of voice

Group them logically

Include specific dates

Segment them by impacted user

<https://userguiding.com/blog/release-notes>

Don’t have a product first approach

Users needs and expectations are top priority

UserGuiding, AnnounceKit, and Releasenotes.io to automate

Distribute notes across multiple channels to increase the visibility

I would probably adopt the following practices:

Focus on the user’s needs first

Use tools to make the process easier

Group them logically

Include dates

Using images, videos, gifs, and links

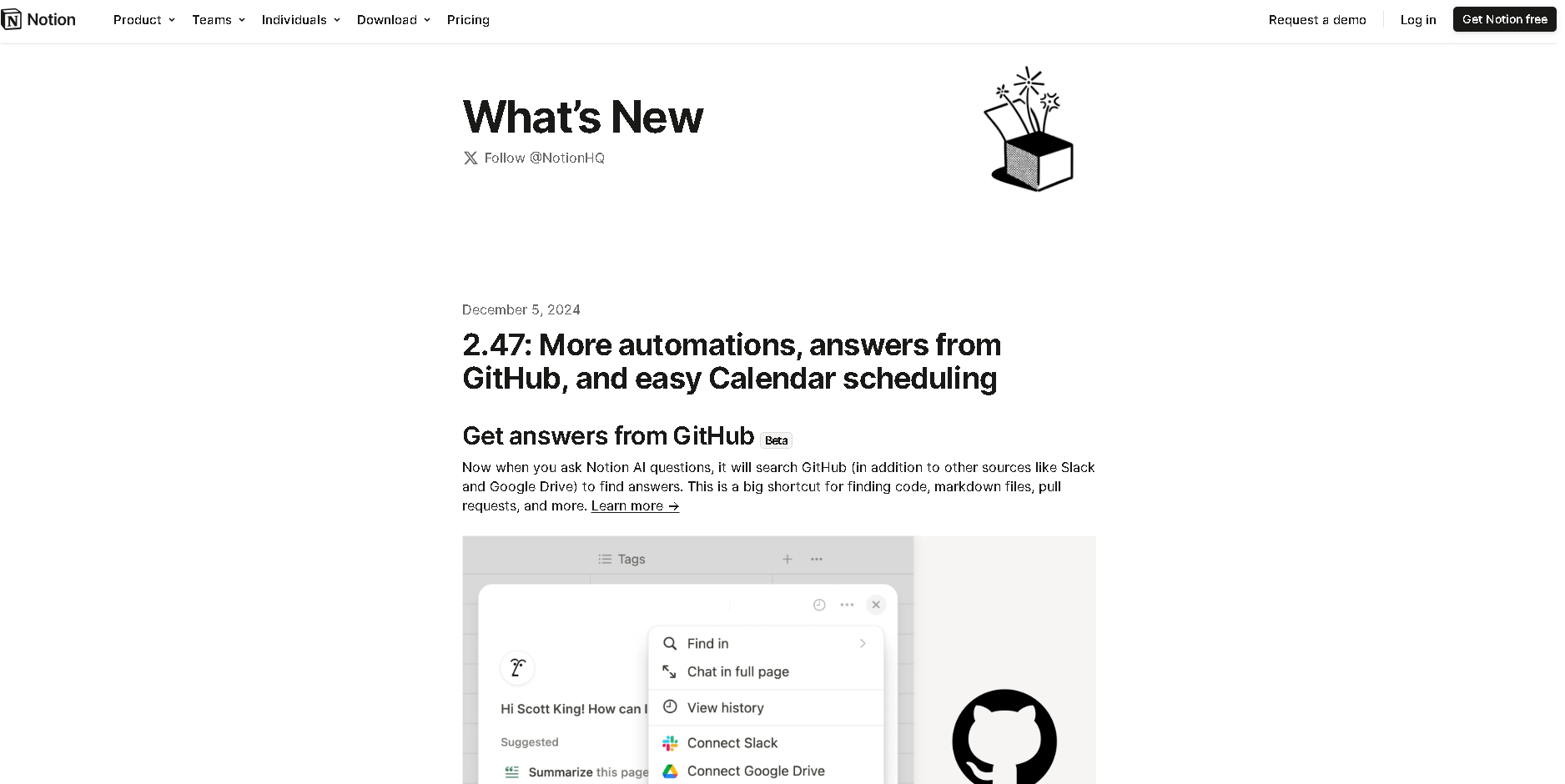
Use brand’s tone of voice

1. Design your Project Team’s Release Notes. Provide a template for Release Notes that you would propose to your team. Your proposal should have enough detail so that a teammate would understand how to fill it out.

Capture the above in your document. The goal is to convince your teammates that you’ve got a well thought-out and achievable approach. Include answers to the items 1, 2 above, and a sample of your proposal would look like for the first sandbox release you provided to the user.

<https://www.notion.com/releases>

I think we should follow Notion’s standards for release notes. They are clear, engaging, and not overwhelming. They also have video guides on how to use the new features they have added, avoiding confusion.



Using bold fonts and contrasting colors is also going to be important. They look more like articles than release notes, and I think this is a good thing, as users often skip over release notes because of the information bomb. I could take care of the videos as I have experience in video-making. These notes from Notion check off all the boxes for what makes good release notes.

# Section 5 – Individual Work

During this part of the assignment, you and the team will need to:

1. Plan ahead to determine what needs to be done to the end of the project
2. Negotiate a migration plan for the project with the operational owner. How would they start using your product given existing workflows?

Read and analyze the K50 Production Criteria for Development Projects.

Also review the Course outline to understand the future deliverables.

Note all the work in this section is captured in a single spreadsheet document.

## Section 5 Part A – Identify Requirements and Deliverables and current status

Put together a spreadsheet that captures:

1. Requirements and Deliverables. Organize and number these based on the numbering in the “K50 Production Criteria for Development Projects” document for traceability. (i.e. 3.2 refers to Operational documentation).
2. Current status of each deliverable: (New, In Progress, Done)
3. Comment/concerns/risks – add a text column to capture any comments, concerns or risks

## Section 5 Part B – Identify outstanding work items, loose planning

For the items that are not **Done** from Section 5 Part A, add columns in your spreadsheet to capture

1. Deliver by – Options are {Week 6,7,8,9}. **Propose a “deliver by” deadline.**
2. Effort estimate – in person hours, how much would this be to get to done?

(Exceptions: Do NOT estimate any of the system test and bug fix items)

## Section 5 Part C – Roll up the results and report

Either manually or using your spreadsheet prowess, report on

1. Number of Requirements and Deliverables
2. % done (vs required)
3. Number of outstanding hours to complete

As computer science near-graduates,it’s expected that this chart be comprehensive, detailed, have a proper narrative and correct in its content. An example will be provided.