FINAL PROJECT: Software MVP

This is the final project for the course. You will deliver a functioning proof of concept.

The final project requires you to deliver a functioning Minimum Viable Product (MVP) and a comprehensive submission package that documents your process and outcomes.

FINAL Group Submission Due: Nov 27th, 2025

- You must create a release on your primary code repository.
- One team member must submit a link to this final release on Quercus.

FINAL Individual Self-Reflection Due: Dec 6th, 2025:

- Individual uploaded PDF to Quercus.

Deliverables Checklist

1. Product Overview

You must provide a clear outline of:

1. TL;DR (too long; didn't read) (max 65 words) of describing what your product is and what problem it's solving.
2. List of key **jobs to be Done (JTBD)** by your users.
3. List of core **customer user journeys (CUJs)** that your MVP enables.

2. MVP Development Justification

Outline your team's MVP journey, connecting your initial vision to the final implemented features through a process of hypothesis, learning, and pivoting.

- **Initial Hypothesis:** State the core problem, your initial product idea, and key assumptions about users.
- **Key Learnings and Pivots:** Describe critical insights from development and user research (e.g., interviews, testing). Explain how these challenged initial assumptions and led to significant pivots or scope changes in features, design, or target audience.
- **Justification of Final MVP:** Justify the final feature set. Explain why these specific features were prioritized and built, showing how they address validated user needs and represent the most viable solution.

3. Functional and Dynamic MVP

You must deliver a **functional and dynamic MVP** (Minimum Viable Product).

- **Core Functionality:** The MVP must be operational and clearly demonstrate the main functionality.
- **Dynamic vs. Static Content:** Clearly outline which parts of your application are **dynamic** (e.g., pulling live data, user interactions) and which parts are **static** (e.g., placeholder text, pre-seeded data). Your core Customer Use Journeys (CUJs) must be dynamic.

4. Test Coverage

Your codebase must include robust testing.

- **Code Coverage:** Tests should achieve at least **65% code coverage** for your application.
    - Test Coverage
        - Thresholds:
            - 60-65%+ = 10pts
            - 55-59.999% = 8pts
            - 50-54.999% = 5pts
            - < 50% = 0
        - If your application *is not* uploading test coverage and does not provide a link to an example run in the final write, it will be considered to have a coverage of 0%
- **Continuous Integration (CI):** Implement code coverage reporting as part of your CI pipeline.
- **Submission Link:** Provide a link to a **successful CI run** that demonstrates the execution and reporting of your code coverage.

5. Demo Recording and In Class Live Demo

Submit a visual demonstration of your MVP.

- **Audio/Text:** The video must be easy to follow, with either clear audio narration or text overlays.
- **Accompanying Write-up:** Include a brief write-up (bullet points) to guide the teaching staff through testing the app and highlight the most important work.
- **In Class Demo [5 points]:** Live demo in front of the class. Clearly outline which parts are static vs dynamic.

6. Deployment Documentation

Clear instructions for teammates to run your software in production. This should look like a list of steps that a teammate would take to deploy your app. This should include any cloud deployments, databases, app installations, etc.

- **Deployment Documentation:** Document the steps to run/install your application.
  - **Web App:** Provide a public URL and any necessary login credentials.
  - **Desktop/Mobile App:** Provide installers (Mac/Windows/iOS/Android) or a video recording if an iOS certificate is unavailable.

7. Updated Architecture Diagram

The final release must include a current view of your application's structure.

- **Final Architecture Diagram:** Submit a diagram that reflects the architecture of your **final MVP**.
  - Follow the same principle from the roadmap assignment.
  - The teaching team must be able to walk through your code and see if the different architectures fit together.
    - Make this easier for your teaching staff by adding a list of links to the different sections to code which integrate with your different architecture components.
    - Write a single sentence describing the integration.

8. Individual Self-Reflection

Each team member must submit an individual reflection.

Course Reflection Essay (Approx. 300 words)

This reflective essay asks you to consider your experience in the course, focusing on the following areas:

1. **Key Learning Outcomes:**
   - What significant knowledge or skills did you acquire throughout the course?
   - What specific insights did you gain regarding the processes of product and software development?
2. **Lessons for Future Projects:**
   - In hindsight, what specific actions or approaches would you change or implement differently in a future project?
3. **Teamwork and Collaboration:**
   - Describe your experience working with your team.
   - Provide a clear breakdown of your primary contributions to the project, alongside an overview of the work undertaken by your teammates.

Rubric Summary

Section	Description	Worth
Product Overview	The "Product Overview" assessment with a TL;DR, JTBD, and core implemented CUJs.	10.0
MVP Development Justification	Outline your team's MVP journey, connecting your initial vision to the final implemented features through a process of hypothesis, learning, and pivoting.	10.0
Functional and Dynamic MVP	All applications present in the architecture diagram have a repository and an MVP is implemented. The product coherently functions and achieves the intended goal. The majority of the content is functional and dynamic. CUJs outlined in the overview match the MVP Tests should represent the MVP Architecture should match with the MVP	40.0
Code Quality & Test Coverage	All repositories have at least one tool measuring/enforcing code quality, explained in the write-up. CI, Exceptions, Logging, and other required services are set up and functioning. Tests cover at least 65% of your application, demonstrated by a successful CI run.	10.0
Video	A short video (~3 min) and accompanying write-up are provided, indicating implemented flows.	10.0
Deployment Documentation	Detail step-by-step production deployment instructions, including cloud services, databases, application setup, etc.	5.0
Updated Architecture Diagram	Submit the final MVP architecture diagram, following the roadmap assignment's principles. To aid the teaching team's code review, link to the integrating code sections and provide a single sentence describing each integration.	5.0

Individual Self-Reflection (Individual Grade)	Each team member must submit an individual reflection.	10.0
	Total:	**100**

Rating Scale

This scale is used for each line of the rubric above.

Rating	Result
Outstanding, Thoughtful and thorough	100% of pts
Strong, Provides some thought	80% of pts
Acceptable, Simple explanation	60% of pts
Insufficient, Little effort was made to give explanations	40% of pts
Unacceptable, No effort was made or the section was missing	0% of pts

Contact Form

Full Name

Email Address

Type Message Here

Send

Email

contact@dcsil.ca

Social

2nd Floor, Gerstein Science Information Centre, 9 King's College Circle, Toronto, ON M5S 1A5, University of Toronto

---