CSCI 3100 Software Engineering

Course Project Specification (Spring 2026)

1. Project Overview

The goal of this project is to apply Agile methodologies and Cloud-native architecture to build a **SaaS application** using **Ruby on Rails**. You will work in a team of **4-5 students**.

Unlike traditional projects, we emphasize **Working Software** over lengthy documentation. However, since we follow Agile principles, your development history (Git) will be audited to ensure iterative progress, even without intermediate deadlines.

1.1 The "Non-Negotiables"

- **Framework:** Ruby on Rails 7+ ~~(Mandatory)~~.(Optional but strongly suggested). Because our TAs will follow the grading scheme in Section 6B to grade your project based on whether you use TDD and BDD for development. If the language/framework your team uses can support TDD and BDD, you can also use it, but you need to clarify how you use TDD and BDD in your project, as our TAs are not experts in all languages or frameworks. **Another risk of not using Ruby on Rails is that you may get fewer practices with Rails, which will be part of our final exam.**
- **Deployment:** Public Cloud (Heroku, AWS, etc.). The app must be publicly accessible.
- **Version Control:** GitHub (If private repo, invite TAs as collaborators. Contact our Head TA for this matter).
- **Database:** select one from SQLite, PostgreSQL, or MySQL.
- **Testing:** RSpec (Unit) & Cucumber (BDD), they must be integrated in **GitHub Actions**.

---

2. Project Topics (Choose One)

Option A: CUHK Venue & Equipment Booking SaaS

A multi-tenant system for Student Societies to book rooms/equipment from different Colleges/Departments.

- **Core Feature:** Conflict detection (prevent double booking), Approval workflow.
- **SaaS Angle:** Different Departments (Tenants) manage their own resource pools independently.

Option B: CUHK Second-hand Marketplace SaaS

A centralized trading platform for students to buy/sell textbooks, furniture, etc.

- **Core Feature:** Search & Filtering, Item Status Management (Available -> Reserved -> Sold).
- **SaaS Angle:** Different Hostels/Colleges acts as "Communities" with different listing rules.

Option C: CUHK Mock-Fund League SaaS

A virtual investment competition platform. Organizations can host trading contests with custom rules.

- **Core Feature:** Virtual Trading execution (Buy/Sell), Portfolio Valuation, Leaderboard.
- **SaaS Angle:** Each competition (League) has its own participants and rules (e.g., starting capital).
- **Compliance:** Must clearly state "Virtual Trading Only".

Option D: Open Topic

Propose your own SaaS idea.

- **Constraint:** Must be a SaaS (Multi-tenant or complex Roles). Must also follow 1.1 “Non-Negotiables”.
- **Guardrails:** No anonymous social apps, no gambling, no academic dishonesty facilitation.

---

3. The "N-1" Complexity Rule

To ensure adequate engineering depth, a team of N members must implement **at least N-1 Advanced Features** (beyond basic CRUD operations).

Advanced Feature Examples:

- **External APIs:** Google Maps, SendGrid (Email), Stripe (Mock payment).
- **Real-time:** ActionCable (WebSockets) for chat/notifications.
- **Background Jobs:** Sidekiq/Redis for scheduled tasks (e.g., daily reports).
- **Analytics:** Interactive Dashboards (Chart.js).
- **Search:** Fuzzy search / Auto-complete.

---

4. Schedule & Deliverables

Phase	Timeline	Weight	Deliverables
Phase 1	**Proposal** (**DDL: Feb 17)**	**10%**	1-Page PDF Plan + GitHub Repository URL + **Deployed "Hello World" Link**
Phase 2	**Final Submission** (**DDL: April 14**)	**90%**	GitHub Repository + Deployed SaaS URL + Video

---

5. Detailed Requirements

Phase 1: Proposal Submission (10%)

- **Goal:** Confirm team, topic, and technical stack setup.
- **Submission:**
    1. **Proposal PDF (1 page, single-spacing, Times New Roman, 12 points):** Team members (ID+Name+GitHub Account), Selected Topic, Pain Points, Solution Overview, and list of planned "$N-1$" features.
    2. **"Walking Skeleton" Link:** A public URL (e.g., on Heroku) showing a basic "Hello World" Rails page.
    3. **GitHub Repo:** Created with all members added.

Phase 2: Final Submission (90%)

This is the only major code submission. It will be graded comprehensively.

Submission Package:

1. **GitHub Repository Link.**
2. **SaaS Deployment URL.**
3. **5-Minute Demo Video:** Narrated walkthrough of the user journey.
4. **README:** Setup guide (instructions to run and test your product), list of implemented features (and who), and a screenshot of the SimpleCov report.

---

6. Final Submission Grading Scheme (Total: 90%)

A. Product Functionality (30%)

- **Completeness:** For example: Does the app work? Are all Core Features implemented?
- **Complexity ($N-1$):** For example: Are the advanced features functional?
- **UX/UI:** For example: Is the interface intuitive?
- **SaaS Architecture:** For example: Is data properly isolated between Tenants/Users?

B. Engineering Quality (30%)

- **Testing (Critical):**
    - **RSpec:** High coverage for Models/Services.
    - **Cucumber:** Acceptance tests covering key User Stories.
    - *Expectation:* >80% Test Coverage.
- **Code Quality:** Follows Rails conventions (DRY, Skinny Controller), clean code style.
- **Deployment:** The live app must not crash during TAs' testing.

C. Process Audit (20%)

- **TAs will audit your Git History.**
- **Agile Evidence:** We expect to see regular commits throughout weeks
- **Anti-Pattern:** Projects with 80% of code committed in the last week will be penalized heavily in this section.
- **Collaboration:** Members with a relatively small contribution will be penalized heavily in this section.

D. Demo Video (10%)

- **Demo Video:** Clarity of presentation and "selling" the product.

F. Feature Ownership

- In the GitHub Readme, there should be a table showing the ownership of each implemented feature (example shown below). We will evaluate the overall contribution of each member. Students with significantly low contributions will be penalized by a factor of range 0.0 to 1.0.
  Suppose the overall group grade is 90 and we identify the factor for a student (XX) as 0.8, then the student XX's grade will be 90*0.8=72. The scores of other group members will not be affected.
- **We will not try to penalize a group member unless necessary.**

Feature Name	Primary Developer (Name)	Secondary Developer	Notes
User Auth & Roles	**Alice**	**Bob**	**Uses Devise + Pundit**
Payment Integration	**Bob**	**x**	**Integrated Stripe API**
Search Engine	**Charlie**	**x**	**Uses Elasticsearch**

---

7. Policies

- **Free-Rider Policy:** We use GitHub Insights. If a member has negligible commits, they get **0 marks** for the project, regardless of the group's grade.
- **AI Policy:** You may use AI (ChatGPT/Copilot), but you must be able to explain every line of code if asked.

- **Plagiarism:** Copying code from other groups or past years will result in immediate failure of the course.