Regina Engineering Competition 2025
Programming
Director: Amr Azouz
Introduction
The aim of this competition is to design a Minimum Viable Product (MVP). A Minimum Viable Product is a version of a product that permits maximized features with the least effort. The resulting product should be fully functional and ready for user testing.

**Competition Details:**

- Date: Nov 8th, 2025
- University of Regina Campus
- Breakfast & Social Activities: 8:00 AM - 9:00 AM
- Competition Time: 9:00 AM - 12:00 PM
- Lunch Break & Career Fair: 12:00PM - 1:00 PM
- Competition Time: 1:00 PM - 4:30 PM
- Presentations and Judge Deliberation: 4:30 - 5:30 PM
- Teams are 1- 4 people in size.

**Safety Guidelines:**

- Designate specific areas for eating and drinking to prevent spills on equipment.
- Keep the workspace clean and organized.
- Avoid downloading and using unverified applications

**Problem Statement:**

- University students often struggle to stay organized and manage their study time effectively across multiple classes, assignments, labs, midterms, appointments, etc. This can lead to missed deadlines, poor academic performance, and burnout.
- Your task is to design a minimum viable product (MVP) that helps students efficiently manage their academic life. The tool should be intuitive. There should be a **calendar** and a **dashboard**. Users should be **able to input tasks at specific dates and times.**

**Bonus Ideas (Optional):**

1. Since some students study late at night, a dark/light mode toggle helps reduce eye strain and improves comfort.
2. You can also add priority labels (e.g. High, Medium, Low) so that students would be able to sort tasks accordingly

**Equipment:**

- Computers will be provided at the Computer Lab.
- Workspace for 4 people will be provided as well.
- Competitors may bring in personal computers as well as any physical writing material (notebooks, papers, etc.)

**Rules and Regulations:**

- The use of AI is permissible; however, you need to specify where it was used. -
- The use of external code such as Stack overflow is permissible; however, proper citation is mandatory.
- All team members must contribute to the final product. Judges may ask questions to verify contributions.
- Each team's presentation must not exceed 15 minutes. Time penalties may apply if exceeded.
- All coding must be completed during the official competition hours. No work may be done outside of allocated time.
- All code must be submitted via GitHub with proper documentation and commit history before the deadline.

**Deliverables:**

- A usable program (make sure to also include the code on GitHub)
- A user guide on how to operate the program
- A presentation (PowerPoint) that is ten minutes long which may include
  1. Introduction
  2. Methodology
  3. Features
  4. Issues encountered
- * Please note that these are the suggested requirements for the presentation. Additional information is beneficial to ensure a better score.

**Judging Criteria:**

- Program/GUI - /35
- Presentation - /30
- Problem-Solving - /35
- Total - /100
- Penalty if rules are compromised -20

**Scoring and Evaluation:**

| **Programming /GUI (35 TOTAL)** | **Presentation (30 TOTAL)** | **Problem Solving (35 TOTAL)** | **Penalties** | **Comments:** |
|---|---|---|---|---|
| | | | | TOTAL: /100 |

Here are three creative brainstorming angles for your MVP, designed to be flashy for a hackathon while still being achievable in 6.5 hours.

The problem isn't just *missing* features; it's that standard calendars are "dumb" (they just store info) and to-do lists are "overwhelming" (they just show a giant list). A *creative* solution makes the tool *smart* and *reduces* student burnout.

---

## 💡 Angle 1: The "Syllabus Parser"

- The Problem: The single worst part of any planner is data entry. Students get 5 syllabi at the start of term with 50+ dates. Nobody wants to type those in.
- The Creative Solution: Your app's main feature is a syllabus parser.
- How it Works (MVP):
    - Don't mess with PDF uploads. Just have a giant text box: "Paste your syllabus course outline here."
    - The user copies-and-pastes the text from their syllabus PDF into the box.
    - A JavaScript function uses Regex (Regular Expressions) to find keywords and dates.
    - You'll need a few patterns, but it's very doable:
        - Find `(Assignment|Lab|Quiz) \d: .* (Due|due on) (Jan|Feb|...|Dec) \d+`
        - Find `(Midterm|Final Exam): (Jan|Feb|...|Dec) \d+ at \d+:\d+ (AM|PM)`
    - The app presents a list: "We found these 12 deadlines. Do you want to add them to your calendar?"
- Why it's a Winning Idea:
    - MVP Definition: This is the *perfect* "maximized features with the least effort."
    - Problem-Solving Score: You're not just building a list; you're solving the *real* problem of data entry.
    - Presentation: This is a *fantastic* "wow" moment in your 10-minute demo.

---

## 💡 Angle 2: The "OS Scheduler" Dashboard

- The Problem: A student looks at 10 tasks due this week and gets "analysis paralysis." They don't know where to start, get overwhelmed, and end up on TikTok.
- The Creative Solution: Treat tasks like processes in an operating system. Your app isn't a "dashboard"; it's a "Task Scheduler." (This is a great tie-in to computer engineering).
- How it Works (MVP):
    - When a user adds a task, they add the (Bonus) Priority

(High/Medium/Low).
    - The Dashboard has a single, prominent button: "What should I do next?"
    - When clicked, your app runs a simple "scheduling algorithm" to suggest *one* task.
    - The Algorithm: You can invent one. For example, it could weigh Priority and Deadline. It finds the task with the highest "Urgency Score."
        - `Urgency = (Priority_Weight) / (Days_Until_Due)`
        - `Priority_Weight`: High=10, Medium=5, Low=1
        - *This is a fun, "engineery" way to solve the bonus "priority" feature.*
- Why it's a Winning Idea:
    - Problem-Solving Score: You are directly tackling "burnout" and "ineffective time management" by making the *decision* for the student.
    - Presentation: You get to put a "scheduling algorithm" on your PowerPoint slide. This looks highly technical and impressive.
    - Relatability: The judges (likely engineers) will love the OS analogy.

---

## 💡 Angle 3: The "Energy-Based" Dashboard

- The Problem: It's 10 PM. You have a "High Priority" task to "Study for ENPE Midterm," but you're burnt out. You also have a "Low Priority" task to "Watch recorded ENPE lecture." A normal to-do list says "DO THE HIGH PRIORITY THING," but you can't. So you do nothing.
- The Creative Solution: A dashboard that filters tasks by required energy level.
- How it Works (MVP):
    - When adding a task, the user includes one more field: Energy Level: (Brain Dead, Low Focus, Deep Work).
    - The Dashboard is simple. It has three buttons at the top: `[Brain Dead 🧠]`, `[Low Focus ☕]`, `[Deep Work 🚀]`.
    - When the user clicks a button, the app filters the "Due Today/This Week" list to show *only* tasks matching that energy level.
- Why it's a Winning Idea:
    - Intuitive: This is extremely user-centric. It acknowledges that "study time" isn't all the same.
    - Problem-Solving Score: It directly addresses the "burnout" part of the problem statement, which is a key insight. It keeps students productive even when they aren't at 100%.
    - GUI Score: This is very simple to implement but makes the GUI feel dynamic and responsive to the user's *state*, not just their data.

---

## My Recommendation

**Go with Angle 1 (Syllabus Parser). It's flashy, achievable, and perfectly matches the "MVP" definition of "maximized features with least effort." You can still add the bonus priority/dark mode features to it.**

**Would you like some help thinking about the Regex pattern for the parser?**

## Most important components

- Calendar to see important deadlines
- Main dashboard
    - Task list
        - Approximation of effort
        - Due date
        - Weightage
    - Task insertion
    - Syllabus insertion
- Tab system
    - Opens up on dashboard
    - Calendar Tab

## Additional Features

- Accessibility?
- Bonus: Schedule filter – change between school and work schedule etc.

## How we want the user to feel

- Already overwhelmed, so we want the page to be welcoming and easy to use
- Should not take much energy to learn how to navigate the application
- Minimalistic design to lower stress

## Technical requirements

## Tech Stack