DegreeFlow

Design Document

Group 04

Supervisor	
Name: **Email:** **Organization:**	Luke Schuurman schuurml@mcmaster.ca McMaster Engineering Society
Team Member 1	
Name: **Email:** **Student Number:**	Anderson Zhou zhoua35@mcmaster.ca 400391994
Team Member 2	
Name: **Email:** **Student Number:**	Aniruddh Arora aroraa33@mcmaster.ca 400318688
Team Member 3	
Name: **Email:** **Student Number:**	Anupam Raj raja5@mcmaster.ca 400351087
Team Member 4	
Name: **Email:** **Student Number:**	Harshit Sharma sharmh17@mcmaster.ca 400349986
Team Member 5	
Name: **Email:** **Student Number:**	Sarah Neelands neelands@mcmaster.ca 400389835
Team Member 6	
Name: **Email:** **Student Number:**	Yanchun Wang wang146@mcmaster.ca 400369124

Table of Contents

1.0 Revision History

Version Number	Authors	Description	Date
0	● Anderson Zhou ● Aniruddh Arora ● Anupam Raj ● Harshit Sharma ● Sarah Neelands ● Yanchun Wang	Started Initial Draft	January 14th, 2024
1	● Anderson Zhou ● Aniruddh Arora ● Anupam Raj ● Harshit Sharma ● Sarah Neelands ● Yanchun Wang	Final Submission	April 3,2025

2.0 Introduction

2.1 Project Purpose

McMaster University students face challenges when planning their academic journey due to scattered information across systems like Mosaic, MyTimetable, and the Undergraduate Academic Calendar. Managing degree requirements, finding relevant course offerings, and organizing a semester-by-semester schedule often requires repetitive manual effort and can lead to mistakes.

DegreeFlow simplifies and streamlines this process by centralizing essential academic planning tools in one platform. Core features include:

- Displays a student's cumulative GPA, total units completed, and the number of units remaining to graduate, helping students monitor academic standing at a glance
- Schedule Generation: Based on a user's selected courses in the What-If planner, DegreeFlow generates a detailed semester-by-semester schedule. This feature uses backend scheduling algorithms and presents an organized front-end view for easy planning.
- Seat Alerts: Real-time seat availability alerts help students secure spots in high-demand courses.

By integrating these tools, DegreeFlow saves students time, reduces planning errors, and improves their overall academic experience through a more intuitive and informed process.

2.2 Document Purpose

This document outlines DegreeFlow's system components, functionalities, technical architecture, and design. It also maps these components to the requirements in the SRS, ensuring a structured approach to enhancing academic planning at McMaster University.

2.3 Project Scope

DegreeFlow is a web-based platform designed to help students efficiently plan and manage their academics by integrating with the Academic Calendar. It offers:

- Degree Progress Tracking: Parsing unofficial transcripts to track course history and degree requirements.

- Real-Time Notifications: Alerts for seat availability in full courses.
- What-If Scenarios: Simulations for academic pathways based on user constraints.
-

Students can upload their unofficial transcripts, monitor progress bars for various degree components, and receive customized course suggestions and alerts via email or push notifications. The tool ensures a centralized, efficient, and user-friendly approach to academic planning.

DegreeFlow is a web-based platform designed to help McMaster University students plan and manage their academics more efficiently by integrating course and program data from the Undergraduate Academic Calendar. The platform provides:

- Degree Progress Tracking: Parses unofficial transcripts to display completed courses, total units earned, remaining units, and cumulative GPA.
- Seat Alerts: Sends real-time notifications when seats become available in full courses.
- What-If Scenarios: Allows students to simulate different academic pathways by selecting hypothetical course plans.

Students can upload their unofficial transcripts, explore potential schedules using What-If plans, and receive seat availability alerts to stay informed during registration periods. DegreeFlow centralizes key academic planning tasks into a streamlined, student-friendly platform, helping reduce manual effort and improve planning confidence.

2.4 Assumptions and Invariants

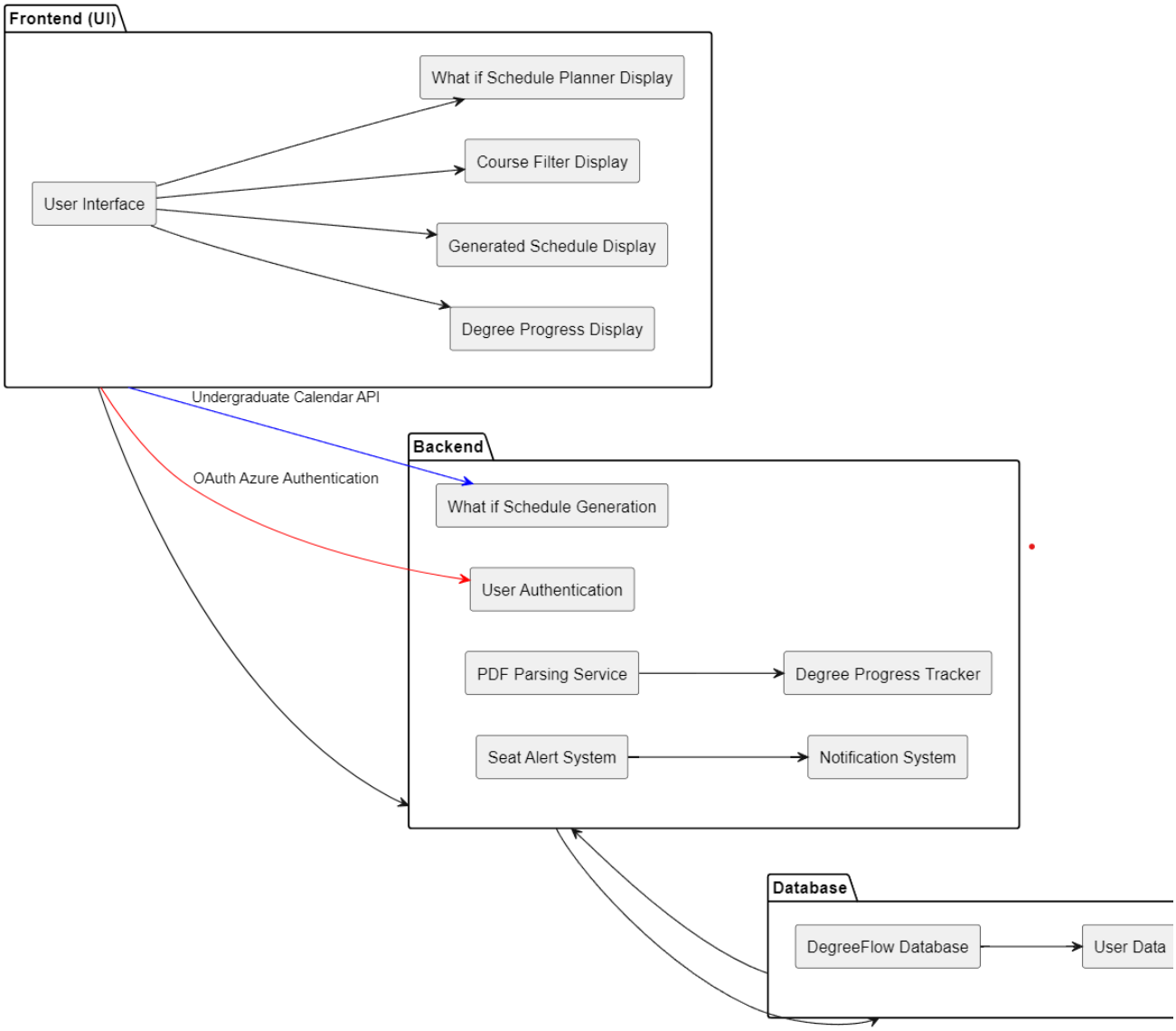
2.4.1 Assumptions

- Students have access to unofficial transcripts in PDF format from Mosaic.
- DegreeFlow retrieves course information and program structures by integrating with McMaster's Undergraduate Academic Calendar.
- Students must verify the accuracy of their transcript data and ensure it is up to date.
- Seat availability notifications rely on real-time queries or periodic updates from Mosaic.
- The system assumes students follow standard academic pathways (e.g., prerequisites, corequisites; enforcement is not handled by the platform.).
- DegreeFlow is a supplementary tool, not a replacement for academic advisors or official degree audits.
- A stable internet connection is required for real-time updates and notifications.

2.4.2 Invariants

- DegreeFlow does not modify McMaster's official academic records; all recommendations are based on transcript data.
- The system does not alter course enrollment—students must use Mosaic to enroll, drop, or swap courses.
- Transcript parsing strictly follows McMaster's format to ensure accurate data extraction.
- DegreeFlow does not impact how McMaster generates or updates transcripts.
- Course notifications are informational only—students must manually complete enrollment.

3.0 Project Component Diagram



User Login

1. The user logs in using any general email.

Student Portal Workflow

2. Students are directed to their main dashboard:
    - Users are prompted to upload their unofficial transcript.
3. The uploaded unofficial transcript is sent to the PDF Parsing Module, where:
    - The transcript is processed to extract academic history and progress data.
    - The processed data is stored in the Central Database under the student's account.
4. The Degree Progress Tracker utilizes the extracted data and course information from the Undergraduate Calendar to calculate the student's progress and total gpa.
5. The calculated progress is displayed on the UI
6. Students can select a schedule in what if.
7. A detailed schedule is generated based on their selected schedules in what - if
    - Considers user-defined constraints, prerequisites, and real-time course availability.

    - Provides alternative pathways to achieve their academic goals.
8. A student can add a Seat Alert for any course in the schedule (current or next term).
    - The Seat Alert System periodically queries McMaster's Mosaic course enrollment system.
9. When a seat becomes available, the backend triggers the Notification System, sending a real-time alert to the student.
10. Students can also filter search for courses using level and subject code and will display all the courses associated.

4.0 Relationship Between Project Components and Requirements

System Component	Requirements Covered
UI	● P0: Provide a web-based platform where students can upload unofficial transcripts, view degree progress, generate schedules, and receive seat alerts. ● P3: Display completed courses, grades, and cumulative GPA in a clear and structured manner. ● Nonfunctional: Meets security and privacy standards; designed with a focus on clean visuals, ease of use, and basic accessibility.
Seat Alert & Real-Time Notifications	● P1: Notify students about seat availability in full courses through real-time alerts. ● Nonfunctional: Reliability and Timeliness to ensure alerts are accurate and immediate.
PDF Parsing	● P0: Extract relevant academic data from student transcripts for degree tracking. ● Nonfunctional: Accuracy to ensure transcripts are processed correctly and Security to ensure no sensitive student data is leaked.
What If Scenarios	● P3: The front-end allows students to build and modify hypothetical course plans. ● P3: back end organizes selected courses by degree levels / years and integrates requirements from Mosaic Academic Calendar to clearly showcase all course requirements of a given degree. ● Built for responsive interaction and clear visual feedback; optimized for performance and ease of use.
Data Integration with Academic Calendar	● P0: Pull degree requirements, course titles, units, and descriptions from McMaster's Undergraduate Academic Calendar. ● P1: Ensure real-time data updates for accurate academic planning. ● Nonfunctional: Data Consistency and Performance

Degree Progress Tracking	● P0: Display student's total completed units, units remaining, and cumulative GPA. ● Nonfunctional: Designed for quick load times and easy-to-read visuals for consistent usability.

5.0 Project Components

5.1 Degree Progress Tracking

5.1.1 Normal Behaviour

The Degree Progress Tracking component summarizes a student's academic progress using parsed data from their unofficial transcript. Upon upload, the system extracts and analyzes details such as completed units, GPA, and program name. This data is used to calculate progress toward graduation. The dashboard displays key indicators like completed units, required units based on program, and units remaining. Students can easily understand how far they've progressed and how much is left to complete their degree.

5.1.2 Inputs and Outputs

Inputs: The input is a student's unofficial transcript in PDF format. From this file, course codes, units, grades, term details, and the student's program are extracted.
Output: A summary is displayed on the dashboard showing:

- Total completed units
- Program name
- Required units (based on program logic)
- Remaining units to graduate

This output helps students visualize their academic standing. Future iterations may include progress bars showing category-specific completion (e.g., electives, technical electives, minor).

5.1.3 Implementation

Backend:
The backend is implemented using Java Spring Boot. It handles file parsing, encryption, storage, and retrieval of transcript data.

The backend includes the following key functionalities:

- parseTranscript(pdfText, studentId): Extracts structured academic data from a transcript.
- saveOrUpdateTranscript(parsedData): Stores encrypted course and grade data in a PostgreSQL database.
- getTranscript(studentId): Retrieves and decrypts previously uploaded transcript data.

- extractTextFromPdf(file): Converts raw PDF content into parsable text format.

Frontend:
The frontend is implemented in React.js. It enables users to upload their transcript, view a legal consent popup, and visualize a summary of their academic progress.

Key frontend functions include:

- handleSubmit(): Submits the transcript to the backend via an authenticated API call.
- calculateSummary(): Calculates total completed, required, and remaining units based on the student's program.
- The summary is displayed directly on the dashboard once the upload is successful.

5.2.4 Potential Undesired Behaviors

- The user may upload an older version of their unofficial transcript, this may be negated with the checking of timestamps on the pdf file, if it is older then previous upload, the user will be alerted.

5.2 Seat Alert & Real-Time Notifications

5.2.1 Normal Behaviour

The Seat Alert & Real-Time Notifications component enables the students to get notified in case any of the previously full courses get opened for admission. A student can request a notification for particular courses, and when a seat opens up in that course, the system will immediately notify them of their opportunity. This will help make sure the students get into their wanted courses without constantly checking on availability.

5.2.2 Inputs and Outputs

*Inputs:*Course information for which the student wants seat availability notifications (course code, term, and class section).Real-time data on course enrollment statuses, queried from McMaster's Mosaic API at regular intervals (every 20–30 seconds).

Outputs:Notifications sent to the student in real time via email when a seat becomes available in a selected course.

5.2.3 Implementation

The Seat Alert system is implemented using Java for backend services and JavaScript for client-side functionality.

Backend Functions:

- The backend implementation uses the following key functions: checkSeatAvailability(courseCode, term, section) queries McMaster's Mosaic API to check seat availability for a specific course. addSubscription(studentId, courseDetails) stores the student’s course subscription preferences in the database. When a seat becomes available, triggerNotification(studentId, courseDetails) initiates the

notification process. The sendNotification(studentId, courseDetails) function sends a real-time email to the student, providing details of the available seat. Finally, generateNotificationContent(courseDetails) creates the notification message, including course-specific details and a direct link to Mosaic for immediate enrollment.

Frontend:

- subscribeToCourse(courseDetails):
  Sends a request to the backend to subscribe to alerts for a course.
- viewSubscriptions(studentId):
  Retrieves the list of current course subscriptions for the student.
- updateNotificationPreferences(preferences):
  Updates the student's notification preferences, such as preferred notification methods.

Notifications:

- Notifications are sent to the student's official McMaster email address using an email integration system Each notification contains detailed information about the course (course code, section, and available seats) and includes a direct link to Mosaic for immediate enrollment.

5.2.4 Potential Undesired Behaviors

- Excessive Query Load:Frequent queries to the Mosaic system (every 20–30 seconds) may create excessive load on the server, potentially causing performance issues or violating the system's usage policies.
- Students may receive duplicate notifications for the same course seat availability if the system fails to recognize previously sent alerts.

5.3 PDF Parsing

5.3.1 Normal Behavior

The PDF Parsing system allows students to upload their unofficial transcripts, extracting relevant academic data to track degree progress and suggest future courses. The system processes:

- Completed courses, grades, and cumulative GPA.
- Term-wise course grouping and co-op status detection.
- Transcript recency validation, warning users if the document is older than three months.

5.3.2 Inputs and Outputs

- Inputs: PDF transcript containing course codes, grades, terms, and issue date.
- Outputs:
  - Extracted data stored in JSON format (structured by term).
  - Recency flag (warning if the transcript is outdated).
  - Course recommendations based on degree requirements.

5.3.3 Implementation

- Backend: Implemented in Java (Spring Boot) using Apache PDFBox for text extraction.
- Data Processing:
  - Extracts course codes, grades, terms, and GPA.
  - Validates formatting, ensuring expected grade scales and course structures.
  - Identifies co-op terms to adjust course planning accordingly.
- Key Classes & Methods:

The backend for PDF parsing is implemented in Java (Spring Boot) using Apache PDFBox for text extraction. The TranscriptParser class handles the extraction and processing of transcript data, using extractText(file) to retrieve raw text from the PDF, parseCourseData(rawText) to identify courses, grades, terms, and GPA, and detectCoopStatus(rawText) to determine co-op terms. The TranscriptValidator class ensures data integrity by checking format consistency with validateTranscriptFormat(rawText), verifying transcript recency with checkRecency(transcriptDate), and ensuring valid grade scales with verifyGradeScale(courseGrades). Finally, the RecommendationEngine class suggests future courses by identifying missing requirements through identifyMissingCourses(completedCourses), recommending electives based on program structure with suggestElectives(completedCourses, program), and generating a structured course plan using generateCoursePlan(completedCourses). This system ensures accurate extraction, validation, and personalized course recommendations to support academic planning.

5.3.4 Potential Undesired Behaviors

- Unreadable PDFs – Scanned images may cause extraction failures.
- Incorrect Parsing – Unusual grading formats or missing course codes could affect accuracy.
- Performance Delays – Large transcripts may slow down processing.
- Security Concerns – Sensitive data must be securely stored and protected.

5.4 What If Scenarios

5.4.1 Normal Behaviour

The What-If Scenarios component enables students to explore alternative academic pathways based on their desired course selections, degree program, and minor choices. This feature helps students understand how their academic decisions impact their graduation timeline and prerequisite fulfillment.

Students can create multiple scenarios to compare different options, ensuring they make informed decisions regarding course selection, electives, and co-op schedules.

5.4.2 Inputs and Outputs

Inputs:

- User-defined constraints, including:
    - Desired course load per semester.
    - Degree program and minor selections.
    - Specific courses they want to take in future terms.
- Real-time course data from McMaster's Mosaic API, including:
    - Course availability.
    - Prerequisite and corequisite information.
    - Semester scheduling constraints.

Outputs:

- Optimized course combinations based on user-defined constraints and prerequisites.
- Expected graduation timeline considering course availability and workload balance.
- Warnings for potential scheduling conflicts or prerequisite violations.

5.4.3 Implementation

Frontend:

- The frontend, functions like createScenario(constraints) send user inputs to the backend for scenario

generation, viewScenarios(studentId) displays saved scenarios for comparison, updateScenario(scenarioId, updates) modifies existing scenarios, and deleteScenario(scenarioId) removes unwanted scenarios. Additionally, comparePathways() allows students to visually analyze multiple scenarios and make informed academic decisions.

Backend:

- The backend implementation includes key functions such as fetchRealTimeCourseData() to retrieve real-time course data from McMaster's Mosaic API, analyzeStudentProgress(studentId) to evaluate the student's academic progress, and generatePathways(studentId, constraints) to create valid academic pathways based on user-defined constraints like course load and program selection. Scenarios are stored using storeScenario(studentId, scenarioDetails) and retrieved later with retrieveScenarios(studentId) for further modifications.

5.4.4 Potential Undesired Behaviors

University changes policies in the future that affect degree requirements. New courses could be added and old courses could be deprecated without updates from the Mosaic API.

5.5 Difficulty Ratings for Each Course

5.5.1 Normal Behaviour

The Course Difficulty Rating component provides students with the ability to assign a difficulty rating from one to five stars for different courses. This feature not only allows students to submit their ratings but also displays the aggregated average rating for each course, which helps other students make informed decisions about course selection. Students can also modify their existing ratings, this ensures their feedback remains current with their course experience.

5.5.2 Inputs and Outputs:

Input: The system accepts student-provided star ratings ranging from one to five stars for each course.

Output: Upon successful rating submission, the system displays the student's individual rating and calculates an updated average rating for the course based on all student submissions. A confirmation message appears to confirm successful rating submission.

5.5.3 Implementation

The frontend is built using HTML, CSS, and JavaScript. User submitting ratings will trigger JavaScript event handlers that manage the rating submission process and update the display in real-time. The backend uses Java Spring Boot framework to handle database operations and RESTful API endpoints for rating submissions and retrievals. The *RatingController* processes requests, validates data through the *RatingService*, and manages interactions with the database.

Key Classes & Methods:

- RatingController.java - Handles API endpoints and requests Routing.
  - ➢ submitRating() - Processes new rating submissions.
  - ➢ gettingRatingByStudentAndCourse() - Retrieves a specific rating by student and course.
  - ➢ getRatingSummary() - Get a summary of rating for the course.

- RatingService.java - Manages business logic.
    - ➢ submitRating(); gettingRatingByStudentAndCourse(), getRatingSummary() for Controller.
    - ➢ validateRatingInput() - Ensures rating is between 1 and 5 stars.
- RatingRepository.java - Handles database interactions.
    - ➢ findByEmailAndCourseCode() - Find a specific rating by student and course.
    - ➢ getAverageRatingsForCourse() - Calculate average rating for a course.
    - ➢ countRatingsByCourseCode() - Count total ratings for a course.

5.5.4 Potential Undesired Behaviors

The primary concern is students might try to submit multiple ratings for the same course, which will potentially skew the average rating. To prevent this, the system implements validation to ensure each student can only maintain one active rating per course at any time.

5.6 Automatic Scheduling

5.6.1 Normal Behaviour

The Automatic Scheduling component generates personalized course timetables based on the student's academic history, degree requirements, and real-time course data. It combines data from PDF Parsing, seat alerts, and scheduling preferences to propose valid schedules. The system avoids time conflicts and prerequisite issues while respecting student constraints like preferred days or credit limits. This component works seamlessly with other modules to deliver timely schedule recommendations.

5.6.2 Inputs and Outputs:

Input: The system reads the student's transcript data, degree requirements, and course availability from Mosaic. It also takes into account user preferences such as no morning classes or constraints for maximum credit loads.

Output: A conflict-free schedule for upcoming terms along with alerts for any issues. If constraints cannot be met, the system may also provide warnings or suggests adjustments.

5.6.3 Implementation

- Backend:
  The backend is built using Java Spring Boot, with the SchedulingService.java class managing the core logic for generating and handling schedules. The SchedulingController.java class sets up the API endpoints for creating new schedules, retrieving a student's schedules, updating scheduling preferences, deleting schedules, and fetching the current active schedule. These endpoints process student data, degree requirements, and real-time course availability from the Mosaic API to produce optimal, conflict-free timetables that meet all prerequisites and user-defined constraints.
- Frontend:
  The frontend is developed with React.js, with a ScheduleGenerator component that allows students to input their scheduling preferences and view the generated schedules. This component communicates with the backend API endpoints to send scheduling requests and receive schedule data.

5.6.4 Potential Undesired Behaviors

API downtime or outdated course data may lead to incomplete or incorrect schedules. Overly strict student constraints can also result in no valid schedule being found. Inaccurate transcript data or outdated degree requirements can cause the scheduler to propose invalid course combinations.

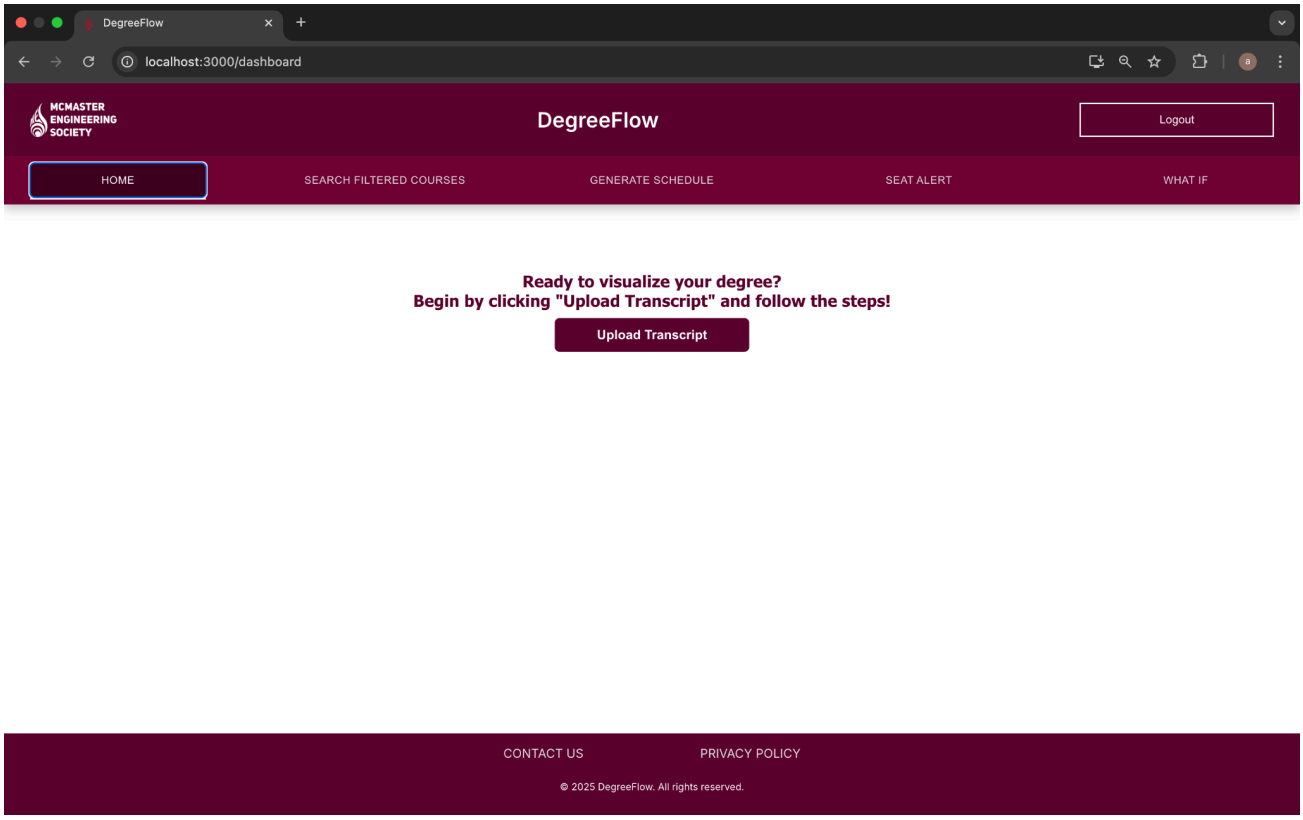
6.0 User Interface

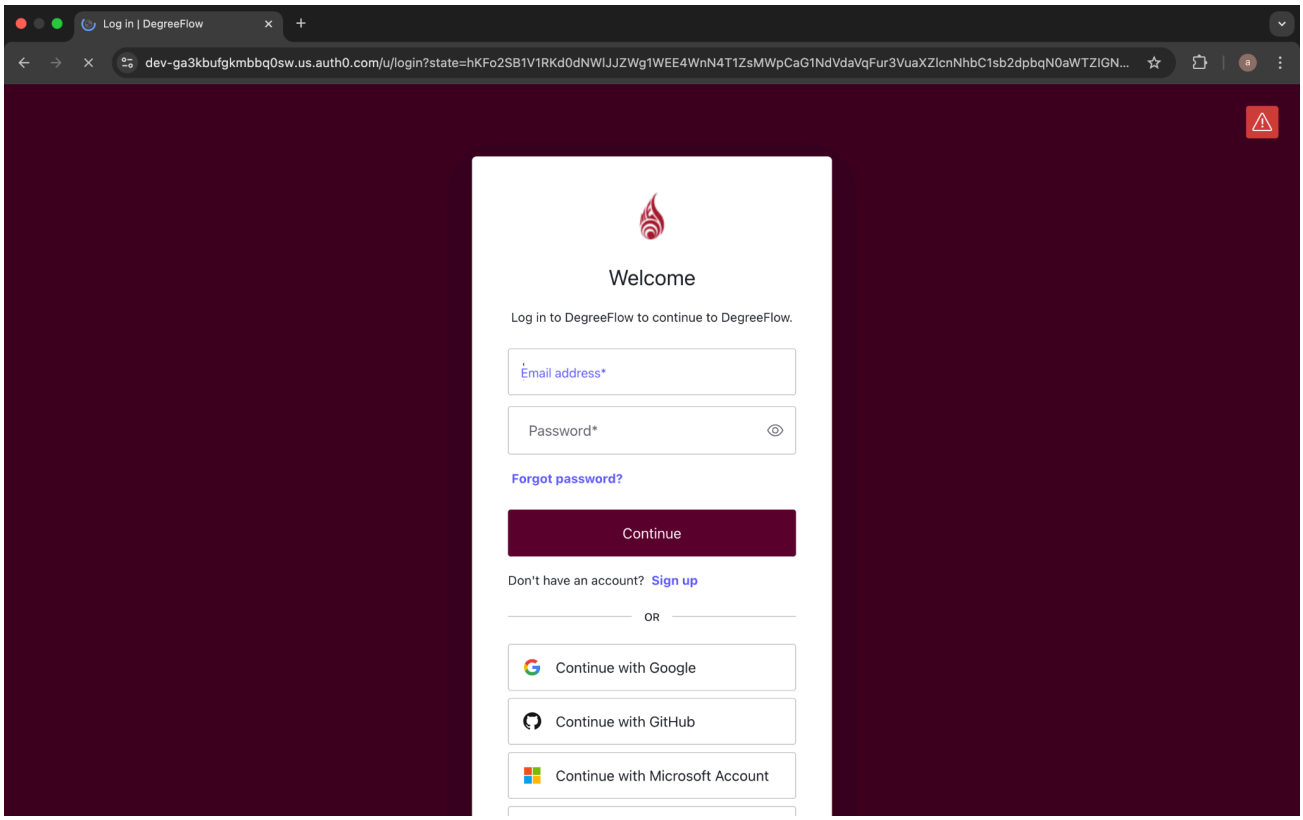
The user interface for DegreeFlow is a web-based application designed to support students in planning and managing their academic journey. Built using React.js, the interface features a clean and intuitive dashboard where students can:

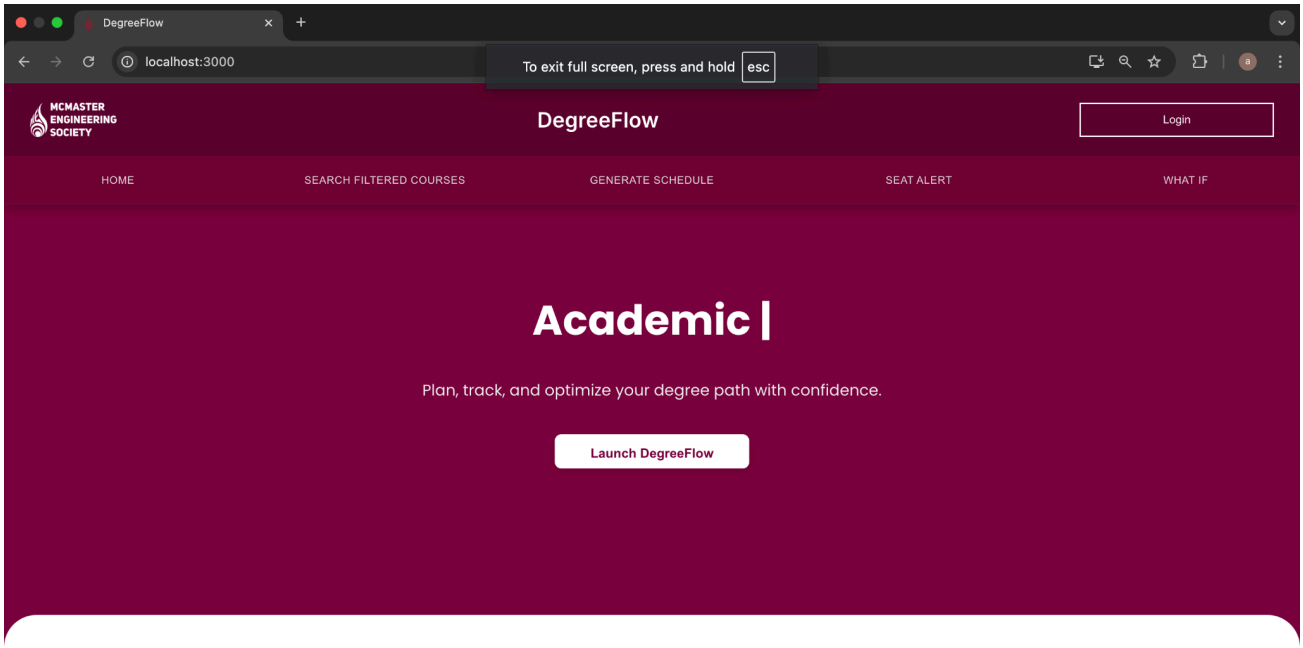
- Upload their unofficial transcript and view parsed academic data.
- Track cumulative GPA, total completed units, and remaining units required for graduation.
- Receive seat availability alerts for selected courses.
- Explore potential course plans using the What-If scenario builder.
- Generate detailed, semester-by-semester schedules based on selected plans and available course offerings.

7.0 Appendix

7.1 UI/UX Design

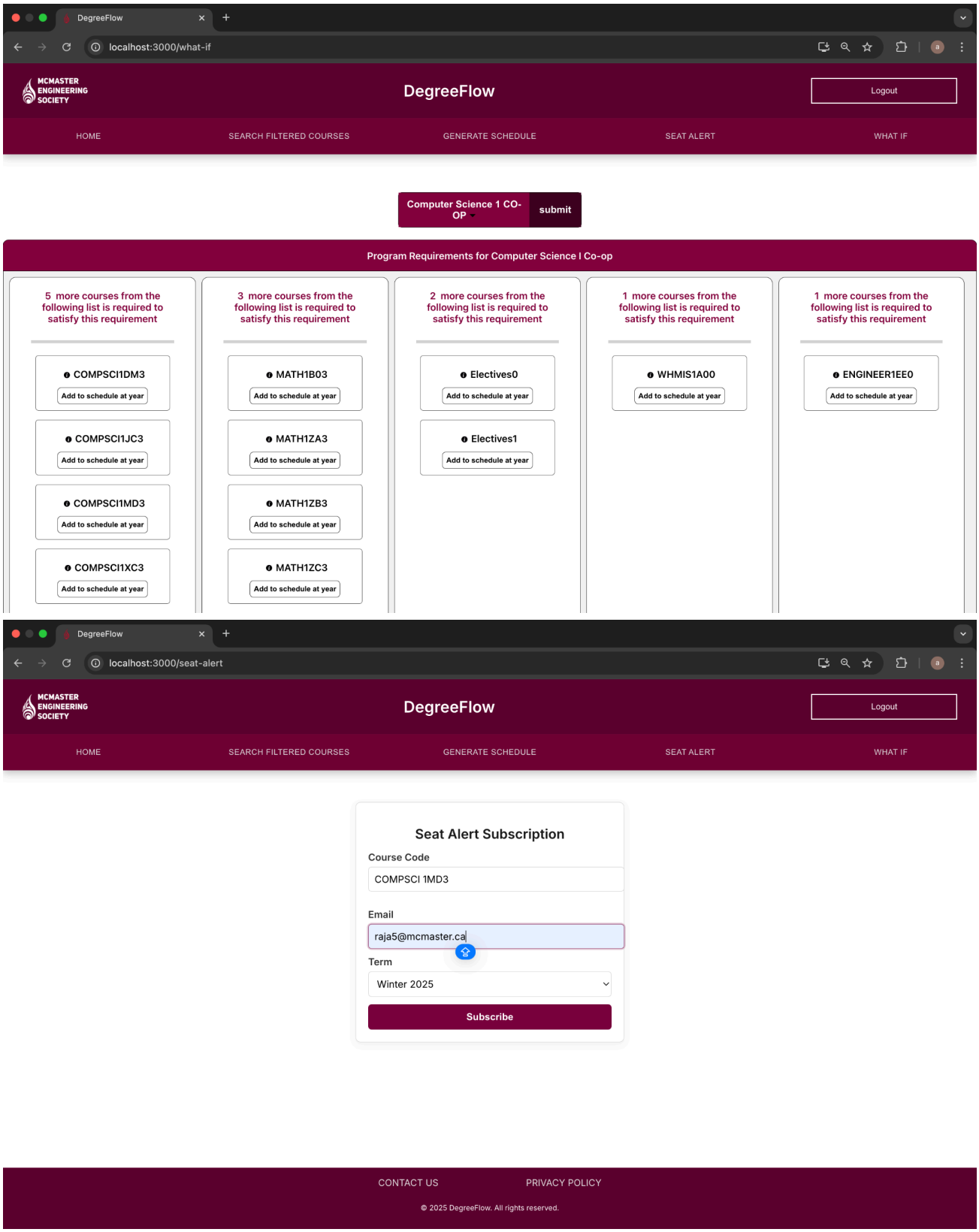


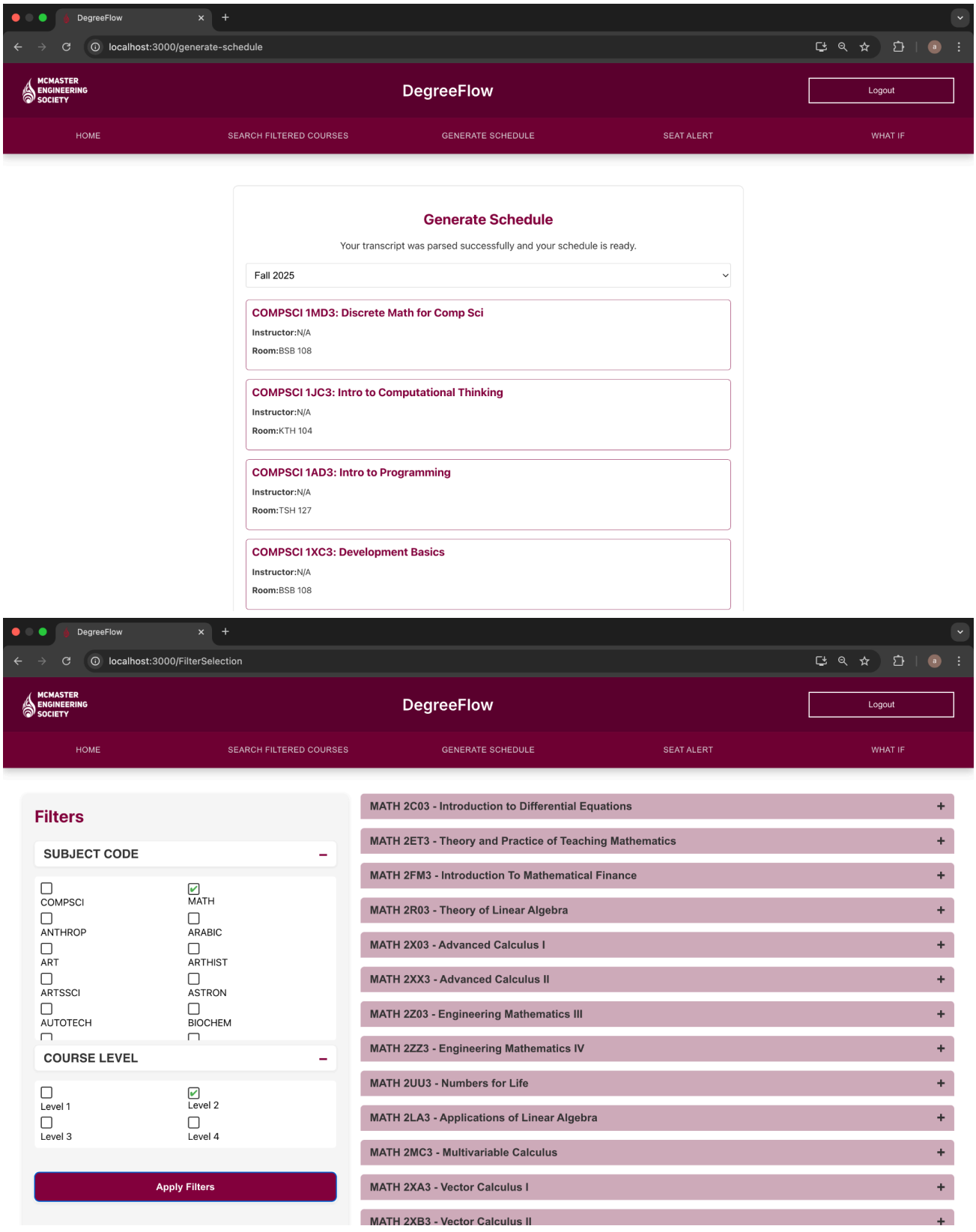


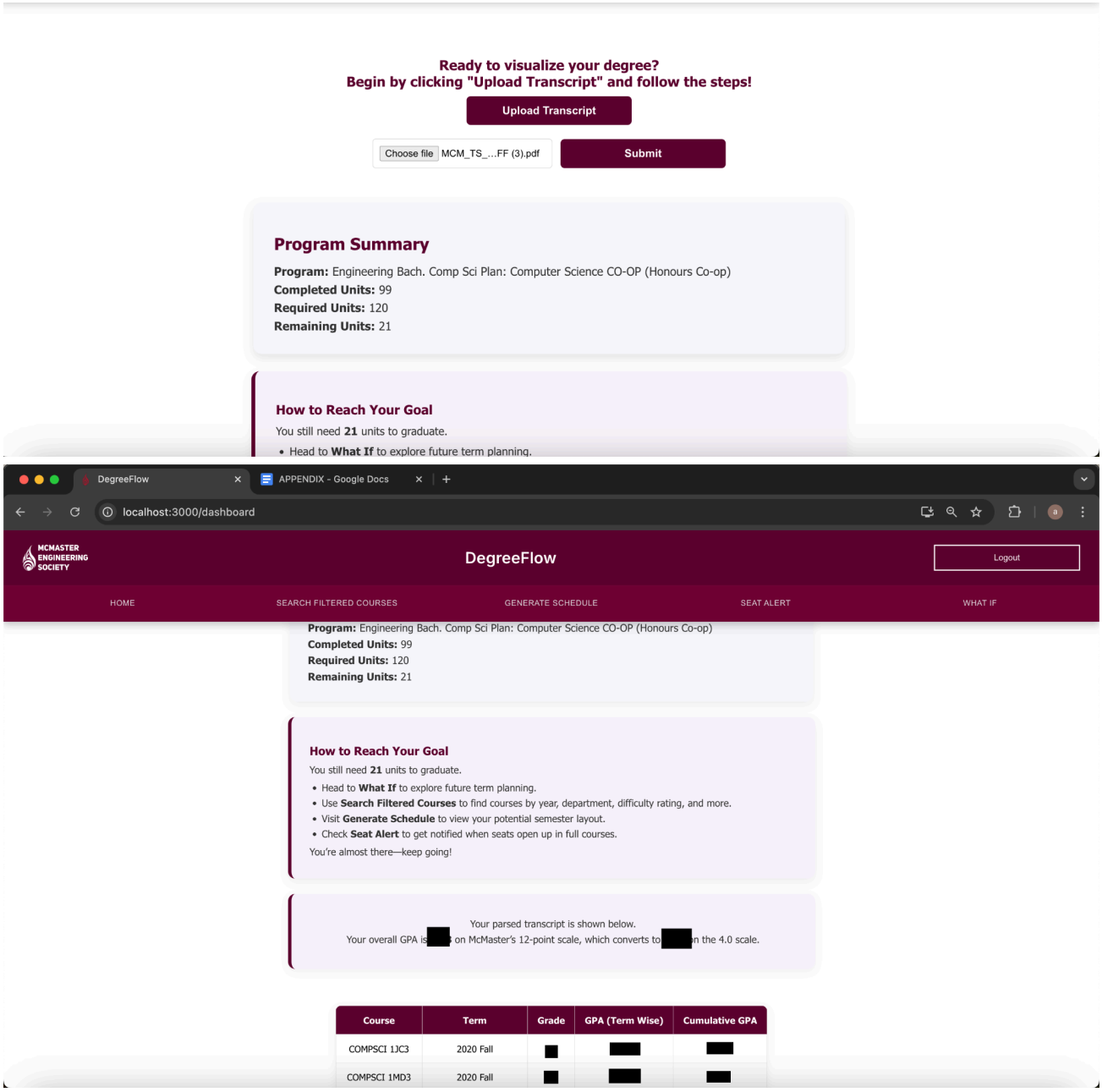


What is DegreeFlow?

DegreeFlow is your intelligent academic co-pilot, purpose-built for McMaster students. Whether you are planning a minor, cross-checking electives, or racing toward graduation,DegreeFlow gives you a complete, personalized overview of your degree journey — all in one place.


DegreeFlow
localhost:3000/what-if
MCMASTER ENGINEERING SOCIETY
DegreeFlow
Logout
HOME
SEARCH FILTERED COURSES
GENERATE SCHEDULE
SEAT ALERT
WHAT IF
Computer Science 1 CO-OP
submit
Program Requirements for Computer Science I Co-op
5 more courses from the following list is required to satisfy this requirement
COMPSCI1DM3
COMPSCI1JC3
COMPSCI1MD3
COMPSCI1XC3
3 more courses from the following list is required to satisfy this requirement
MATH1B03
MATH1ZA3
MATH1ZB3
MATH1ZC3
2 more courses from the following list is required to satisfy this requirement
Electives0
Electives1
1 more courses from the following list is required to satisfy this requirement
WHMIS1A00
1 more courses from the following list is required to satisfy this requirement
ENGINEER1EE0
Add to schedule at year
DegreeFlow
localhost:3000/seat-alert
MCMASTER ENGINEERING SOCIETY
DegreeFlow
Logout
HOME
SEARCH FILTERED COURSES
GENERATE SCHEDULE
SEAT ALERT
WHAT IF
Seat Alert Subscription
Course Code
COMPSCI 1MD3
Email
raja5@mcmaster.ca
Term
Winter 2025
Subscribe
CONTACT US
PRIVACY POLICY
© 2025 DegreeFlow. All rights reserved.


DegreeFlow
localhost:3000/generate-schedule
MCMASTER ENGINEERING SOCIETY
DegreeFlow
Logout
HOME
SEARCH FILTERED COURSES
GENERATE SCHEDULE
SEAT ALERT
WHAT IF
Generate Schedule
Your transcript was parsed successfully and your schedule is ready.
Fall 2025
COMPSCI 1MD3: Discrete Math for Comp Sci
Instructor:N/A
Room:BSB 108
COMPSCI 1JC3: Intro to Computational Thinking
Instructor:N/A
Room:KTH 104
COMPSCI 1AD3: Intro to Programming
Instructor:N/A
Room:TSH 127
COMPSCI 1XC3: Development Basics
Instructor:N/A
Room:BSB 108
DegreeFlow
localhost:3000/FilterSelection
MCMASTER ENGINEERING SOCIETY
DegreeFlow
Logout
HOME
SEARCH FILTERED COURSES
GENERATE SCHEDULE
SEAT ALERT
WHAT IF
Filters
SUBJECT CODE
COMPSCI
MATH
ANTHROP
ARABIC
ART
ARTHIST
ARTSSCI
ASTRON
AUTOTECH
BIOCHEM
COURSE LEVEL
Level 1
Level 2
Level 3
Level 4
Apply Filters
MATH 2C03 - Introduction to Differential Equations
MATH 2ET3 - Theory and Practice of Teaching Mathematics
MATH 2FM3 - Introduction To Mathematical Finance
MATH 2R03 - Theory of Linear Algebra
MATH 2X03 - Advanced Calculus I
MATH 2XX3 - Advanced Calculus II
MATH 2Z03 - Engineering Mathematics III
MATH 2ZZ3 - Engineering Mathematics IV
MATH 2UU3 - Numbers for Life
MATH 2LA3 - Applications of Linear Algebra
MATH 2MC3 - Multivariable Calculus
MATH 2XA3 - Vector Calculus I
MATH 2XB3 - Vector Calculus II



7.2 API

Principal Component	API Endpoints
Degree Progress Tracking	• POST /api/transcript/upload – Uploads and processes the student's transcript • GET /api/progress/{studentId} – Retrieves the degree progress of a specific student • GET /api/missing-requirements/{studentId} – Returns missing required courses

	• GET /api/progress/warnings/{studentId} – Checks for outdated transcripts and other warnings
Seat Alert & Real-Time Notifications	• **POST /api/courses/subscribe** – Allows students to subscribe to alerts for a specific course. • **GET /api/seats/status** – Fetches the latest seat availability for the requested course. • **POST /api/notifications/send** – Sends notifications to the student when a seat is available.
PDF Parsing	• POST /api/transcript/upload – Uploads the student's transcript PDF for processing. • GET /api/transcript/{studentId} – Retrieves the extracted transcript data for a specific student. • GET /api/transcript/validate/{studentId} – Checks if the transcript is older than three months and returns a warning if outdated. • GET /api/transcript/courses/{studentId} – Returns a list of completed courses with grades. • GET /api/transcript/co-op-status/{studentId} – Checks if the student has participated in a co-op term. • GET /api/transcript/missing-requirements/{studentId} – Identifies required courses that have not been completed. • GET /api/transcript/recommendations/{studentId} – Suggests potential courses for the next term based on academic history.
What If Scenarios	• GET /api/degree/requirement – Accepts degree code and returns degree requirement • GET /api/degree/degreeName - returns list of degree code and degree name

	• POST /api/degree/addSchedule - Saves user customized What-If scenario into database
Ratings for Each Course	• POST /api/ratings - Submit or update a rating for a specific course. • GET /api/ratings/summary/{courseCode} - Retrieve rating details for a course. • GET /api/ratings/student/{email}/course/{courseCode} - Get a specific rating by student and course.
Automatic Scheduling	• POST /api/schedules/{studentId}/generate - Generates schedules for a student identified by scheduleId by following constraints found through their academic progress and preferences. • GET /api/schedules/{studentId}- Retrieves all schedules associated with a specific student identified by studentId. • PUT /api/schedules/{scheduleId}/preferences - Updates the scheduling preferences for an existing schedule identified by scheduleId. • DELETE /api/schedules/{scheduleId} - Deletes a specific schedule identified by scheduleId. • GET /api/schedules/current/{studentId} - Fetches the current active schedule for a student identified by studentId.