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

2.0 Introduction

2.1 Project Purpose

McMaster University students face challenges in planning their academic journey due to difficulties in tracking degree requirements, identifying course conflicts, and ensuring they meet all requisites. This process is time-consuming, error-prone, and requires manual checks across multiple systems like Mosaic, the Academic Calendar, and advisement reports.

DegreeFlow aims to simplify academic planning by providing:

- Track Progress: Progress bars for overall degree completion, electives, technical electives, and minors.
- Conflict-Free Scheduling: A tool to create conflict-free schedules and receive real-time seat availability notifications.
- Personalized Recommendations: Course suggestions based on unofficial transcripts and advisement reports to ensure timely graduation.

DegreeFlow centralizes academic planning, saving students time, reducing errors, and improving the overall experience through tailored recommendations and real-time alerts.

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 Mosaic and 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.
- Course Recommendations: Suggestions based on academic progress and remaining requirements.

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.

2.4 Assumptions and Invariants

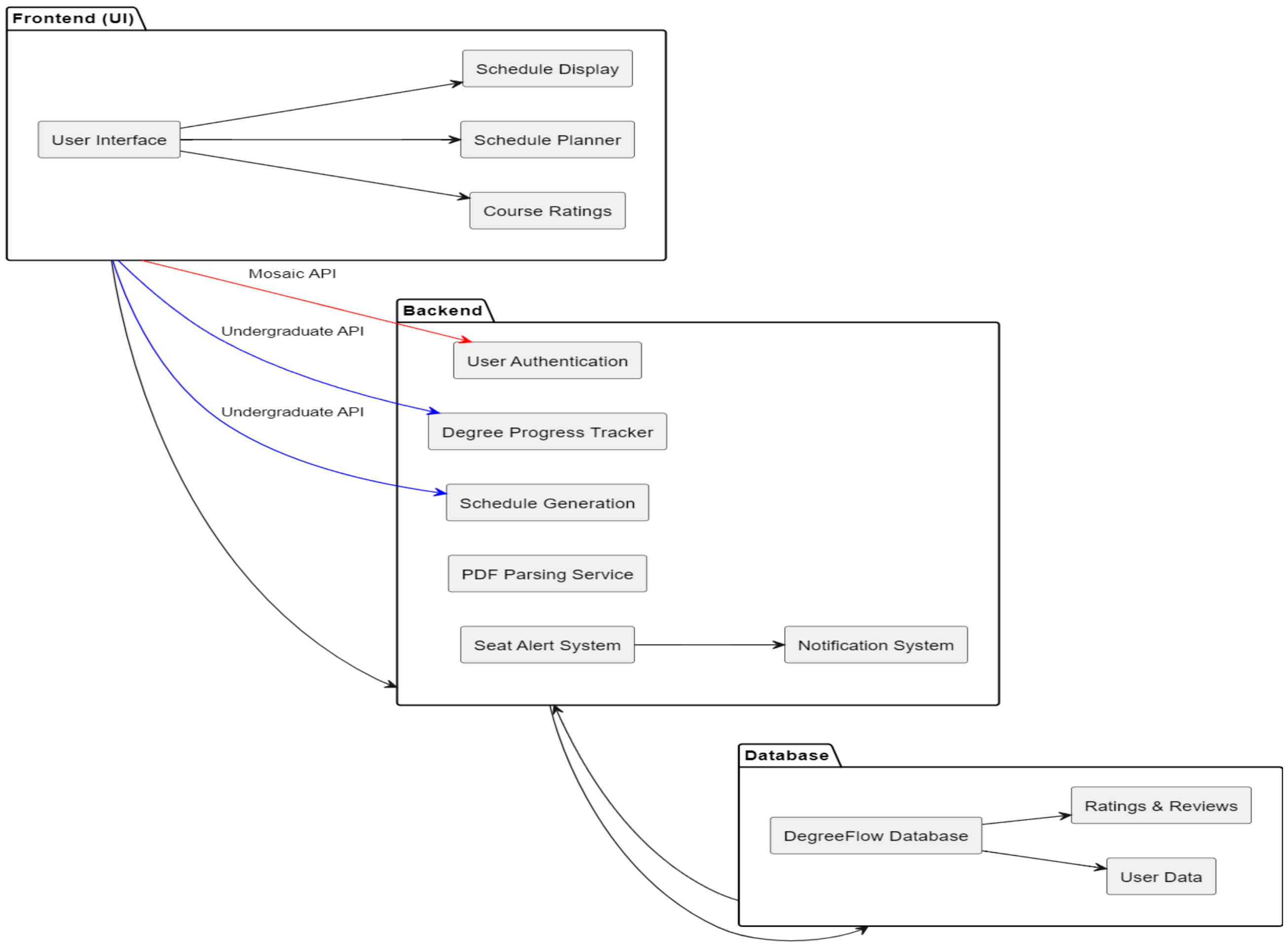
2.4.1 Assumptions

- Students have access to unofficial transcripts in PDF format from Mosaic.
- DegreeFlow integrates with McMaster's Academic Calendar and Mosaic for course information and enrollment data.
- 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).
- 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 & Portal Selection

1. When the user enters the application, they will be directed to a portal selection page where they can choose between
    - Student Portal: For students to manage their degree progress.
    - Admin Portal: For administrators to modify information related to course and degrees.
2. The user logs in using their McMaster email via the Mosaic API.

Student Portal Workflow

3. If the student selects the Student Portal, they are directed to their main dashboard:
    - First-time users are prompted to upload their unofficial transcript.
    - Returning users have their previously saved data displayed.
4. 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.
5. The Degree Progress Tracker utilizes the extracted data and course information from the Undergraduate Calendar to calculate the student's progress.
6. The calculated progress is displayed on the UI
7. Students can generate schedules using the Schedule Generation Algorithm, which:
    - Considers user-defined constraints, prerequisites, and real-time course availability.
    - Provides alternative pathways to achieve their academic goals.
8. If a student saves a generated schedule and sets it as a favorite, they can:
    - Opt to add a Seat Alert for any course in the schedule (current or next term).
9. The Seat Alert System periodically queries McMaster's Mosaic course enrollment system.
10. When a seat becomes available, the backend triggers the Notification System, sending a real-time alert to the student.
11. Students can rate and review courses in the Course Ratings section, where they can:
    - Submit ratings and feedback for courses they have taken.
    - View aggregate average ratings for each course on the UI.
12. All ratings and feedback are stored in the Central Database and accessible for future reference.

Admin Portal Workflow

13. If an admin selects the Admin Portal, they are directed to the Admin Dashboard, which includes:
    - Degree Program Management: Modify available degree paths, course requirements, and prerequisites.
    - Course availability: Modify which courses are available in each term
14. Admins can directly modify system components, such as:
    - Editing Degree Requirements: Adjust required courses, electives, and prerequisites.
    - Editing Course Availability: adjust which courses are available to select for the term.

4.0 Relationship Between Project Components and Requirements

System Component	Requirements Covered
UI	• P0: Provide a web-based platform where students can track their degree progress, manage their course schedule, and receive notifications. • P3: Display historical academic records (e.g., completed courses and grades) in a structured manner. • Nonfunctional: Security & Privacy, Look & Feel, Usability & 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 will provide an interactive scenario builder and detailed reports on potential academic outcomes. • P3: The back-end will simulate academic scenarios and analyse their impact on user progress and graduation. • Nonfunctional: Usability and Humanity, Look and Feel, Performance
Ratings for Each Course	• P3: Students will be able to submit a star rating (1-5 stars) for their enrolled courses. The courses will display average ratings. • Nonfunctional: The rating submission must be immediate and average calculation must be accurate.
Degree Progress Tracking	• P0: Display progress bars for overall degree completion, electives, technical electives, and minors. • Nonfunctional: Performance and Usability to ensure smooth user experience.
Data Integration with Mosaic & Academic Calendar	• P0: Retrieve course information, prerequisites, and enrollment status directly from McMaster University’s official sources. • P1: Ensure real-time data updates for accurate academic planning. • Nonfunctional: Data Consistency and Performance

Automatic Scheduling	● P0: Automatically generate conflict-free course schedules using student given data. ● Nonfunctional: Performance, Reliability, and Timeliness to ensure schedules are generated quickly and accurately.

5.0 Project Components

5.1 Degree Progress Tracking

5.1.1 Normal Behaviour

The Degree Progress Tracking component extracts the data from the students unofficial transcript that the pdf parsing component gathered to display the students progress towards the completion of their degree in the form of a progress bar. The display will show a progress bar for the total degree completion, elective completion, technical elective completion, and if the student has a minor, its completion as well. Users will not be able to edit the data directly to maintain accuracy.

5.1.2 Inputs and Outputs

Inputs: The inputs will be the students unofficial transcript pdf, which will have the course codes and their units extracted, along with their program and plan. These extracted courses will be categorized as requisite, elective, and technical elective, and minor courses if the student declares it.

Output: The data of the course categorization will be displayed as a progress bar. There will be a progress bar for total degree completion, and category specific completion (elective, requisite, technical elective, and minor if applicable)

5.1.3 Implementation

Backend:
The backend is implemented in Java Spring Boot, handling:

- The backend implementation includes the following key functions: extractCourseDetails(transcript): Extracts course details from the parsed transcript. categorizeCourses(courseList): Categorizes courses into core, elective, and technical elective categories. storeAndRetrieveData(studentId, courseData): Stores and retrieves data from a PostgreSQL database. calculateProgress(studentId): Determines the students current progress towards their degree.

Frontend:
The front-end is built with React.js:

- The frontend implementation includes the following key functions: displayDashboard(studentId): Displays a dashboard with progress bars for each category. checkTranscriptStatus(studentId): Warns the student if the transcript data is outdated.

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.

- There may be changes to the course requirements, however since these changes will be updated in the academic calendar, the categorization will reflect these changes.

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 Ratings for Each Course

5.5.1 Normal Behaviour

The Course Rating component provides students with the ability to assign a rating from one to five stars for their completed or current 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. The student's enrollment status and course registration history are required to validate their eligibility to rate specific courses.

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.
    - ➢ gettingRatingsByStudent() - Retrieves student rating history.
    - ➢ getCourseAvgRating() - Calculates course's average rating.
- RatingService.java - Manages business logic.
    - ➢ validateEnrollmentStatus() - Checks student eligibility.
    - ➢ updateCourseAvg() - Recalculates course's rating average.
    - ➢ preventDuplicateRatings() - Ensures one rate per student.
- RatingRepository.java - Handles database interactions.
    - ➢ findByCourseId() - Retrieves course ratings.
    - ➢ findByStudentId() - Retrieves student rating history.
    - ➢ updateRaing() - Modifies existing ratings.

5.5.4 Potential Undesired Behaviors

The primary concern is the possibility of students attempting to rate courses they haven't taken, which could lead to inaccurate and misleading course ratings. This is addressed through strict course enrollment verification before allowing rating submissions. Additionally, 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

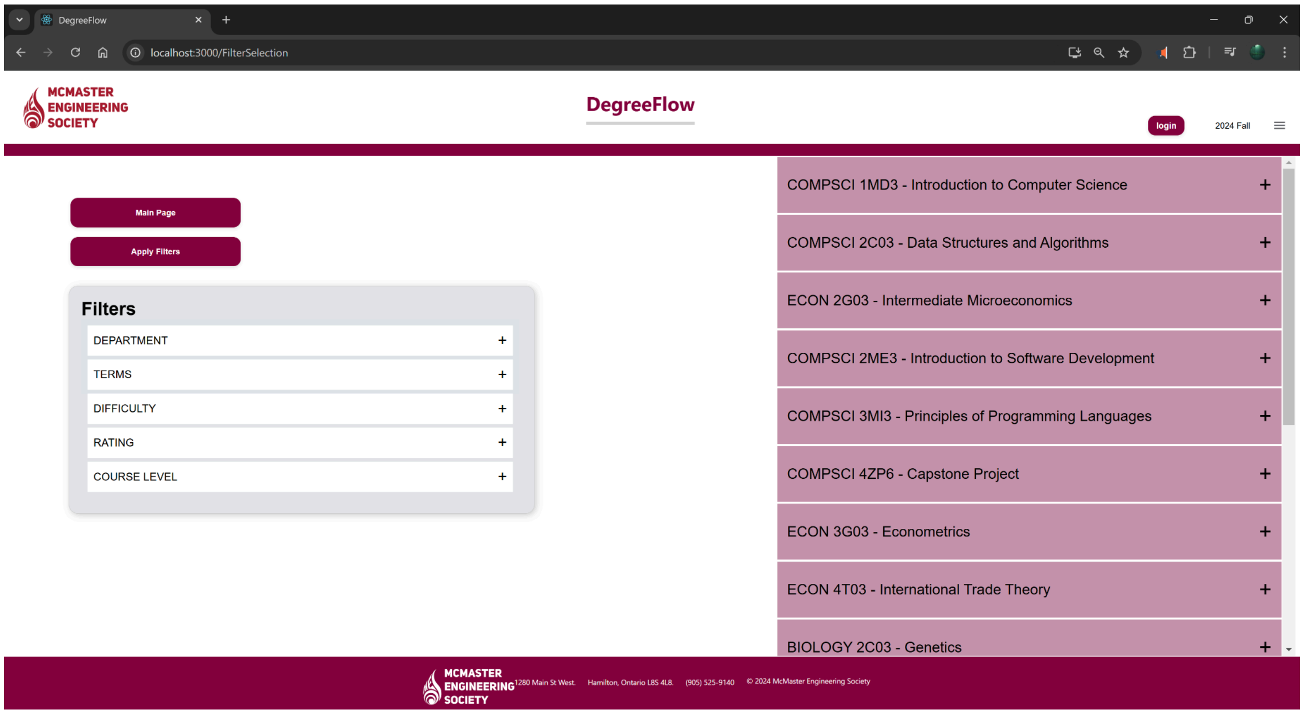
API downtime or outdated course data may lead to incomplete or incorrect schedules. Overly strict student constraints can also result in no valid schedule being found. Inaccurate transcript data or outdated degree requirements can cause the scheduler to propose invalid course combinations.

6.0 User Interface

The user interface for DegreeFlow is a web-based application designed to streamline academic planning for students. It features an intuitive dashboard that allows students to track their degree progress, upload and parse unofficial transcripts, and monitor seat availability in real-time. Students can set up alerts for full courses, explore different academic paths through "What-If Scenarios," and view detailed progress bars that reflect their completion status. The interface is built with React.js and includes interactive tables for displaying data and charts for visualizing degree progress. The overall design aims to provide a smooth, user-friendly experience to make academic planning more efficient.

7.0 Appendix

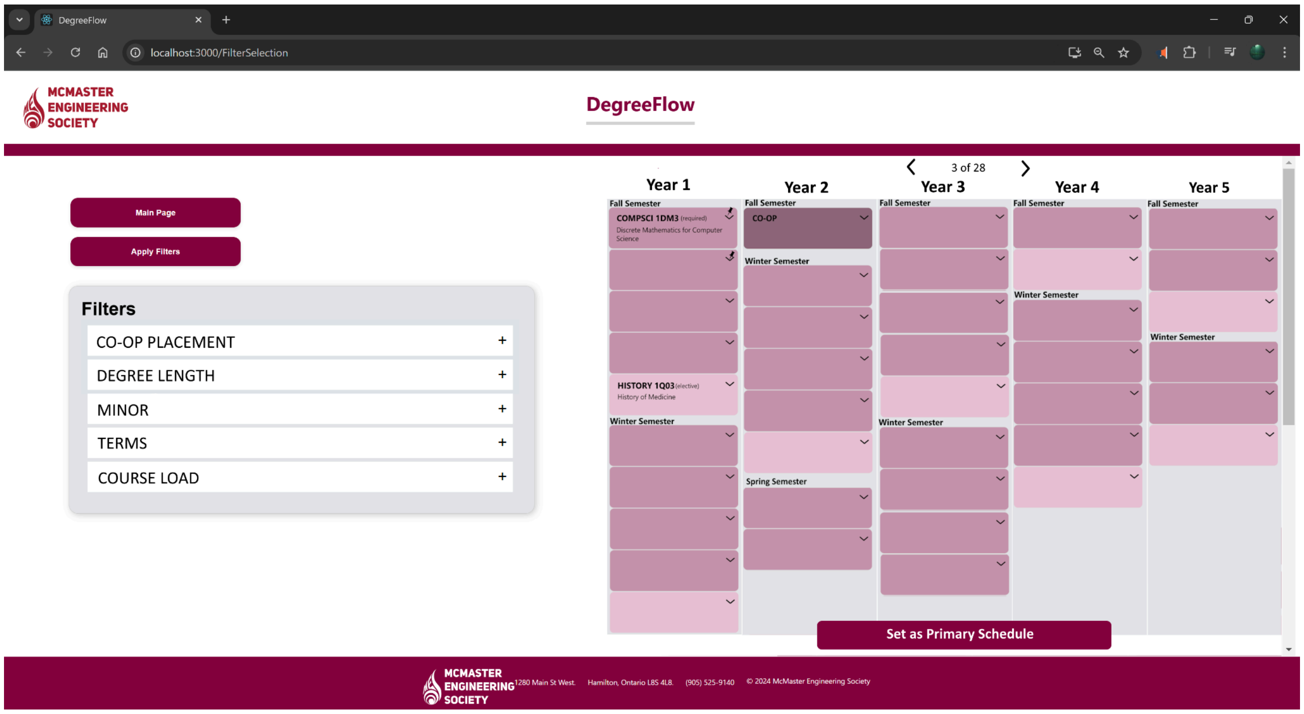
7.1 UI/UX Design



Filters
- DEPARTMENT +
- TERMS +
- DIFFICULTY +
- RATING +
- COURSE LEVEL +

- COMPSCI 1MD3 - Introduction to Computer Science +
- COMPSCI 2C03 - Data Structures and Algorithms +
- ECON 2G03 - Intermediate Microeconomics +
- COMPSCI 2ME3 - Introduction to Software Development +
- COMPSCI 3MI3 - Principles of Programming Languages +
- COMPSCI 4ZP6 - Capstone Project +
- ECON 3G03 - Econometrics +
- ECON 4T03 - International Trade Theory +
- BIOLOGY 2C03 - Genetics +

MCMASTER ENGINEERING SOCIETY 1280 Main St West. Hamilton, Ontario L8S 4L8. (905) 525-9140 © 2024 McMaster Engineering Society

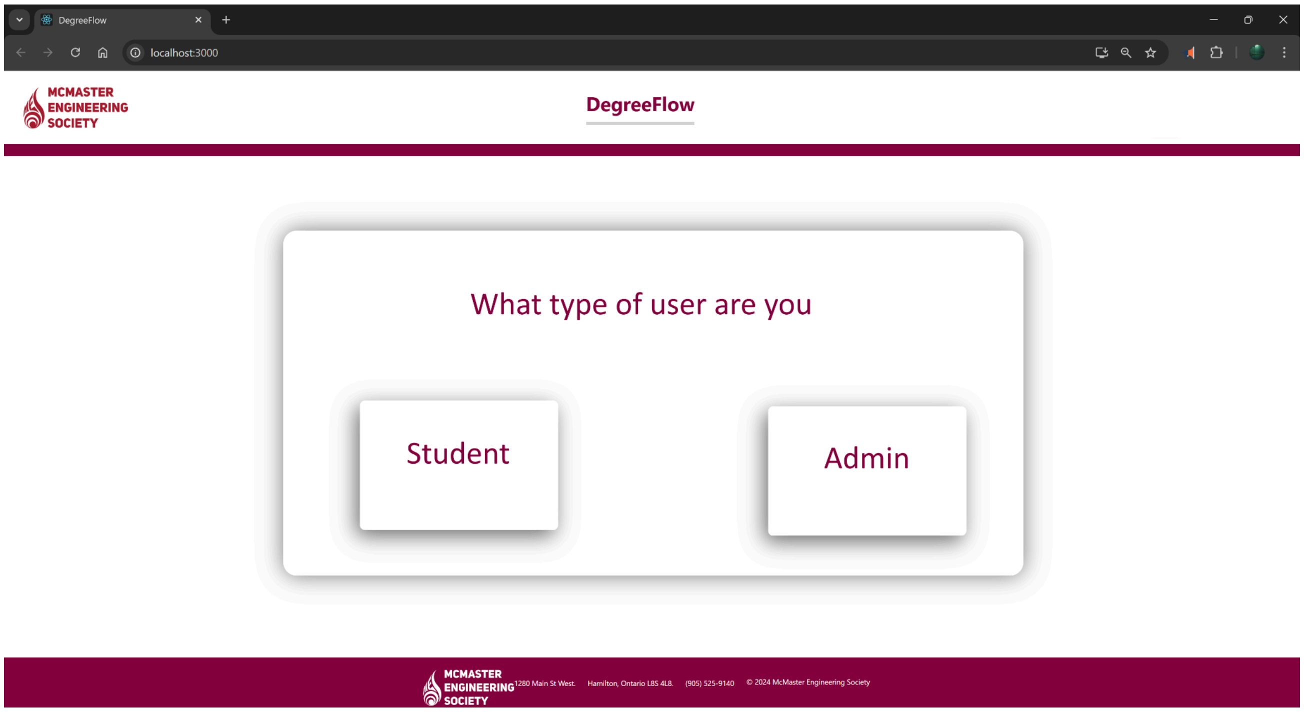


Filters
- CO-OP PLACEMENT +
- DEGREE LENGTH +
- MINOR +
- TERMS +
- COURSE LOAD +

COMPSCI 1DM3 (required) Discrete Mathematics for Computer Science

HISTORY 1Q03 (elective) History of Medicine

MCMASTER ENGINEERING SOCIETY 1280 Main St West. Hamilton, Ontario L8S 4L8. (905) 525-9140 © 2024 McMaster Engineering Society


DegreeFlow
localhost:3000
MCMASTER ENGINEERING SOCIETY
DegreeFlow
What type of user are you
Student
Admin
MCMASTER ENGINEERING SOCIETY 1280 Main St West. Hamilton, Ontario L8S 4L8. (905) 525-9140 © 2024 McMaster Engineering Society

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	<ul><li>POST /api/transcript/upload – Uploads the student's transcript PDF for processing.</li><li>GET /api/transcript/{studentId} – Retrieves the extracted transcript data for a specific student.</li><li>GET /api/transcript/validate/{studentId} – Checks if the transcript is older than three months and returns a warning if outdated.</li><li>GET /api/transcript/courses/{studentId} – Returns a list of completed courses with grades.</li><li>GET /api/transcript/co-op-status/{studentId} – Checks if the student has participated in a co-op term.</li><li>GET /api/transcript/missing-requirements/{studentId} – Identifies required courses that have not been completed.</li><li>GET /api/transcript/recommendations/{studentId} – Suggests potential courses for the next term based on academic history.</li></ul>
What If Scenarios	<ul><li>POST /api/whatif/create – Accepts user constraints and generates a What-If Scenario.</li><li>GET /api/whatif/retrieve/{scenarioId} – Fetches the scenario details, including courses and timeline.</li><li>GET /api/whatif/retrieve-all/{studentId} – Retrieves all stored What-If Scenarios for the student.</li><li>GET /api/whatif/recommendations/{studentId} – Suggests alternative course combinations and pathways.</li><li>POST /api/whatif/update/{scenarioId} – Allows students to modify an existing What-If Scenario.</li><li>DELETE /api/whatif/delete/{scenarioId} – Deletes a stored What-If Scenario.</li></ul>

Ratings for Each Course	● POST /api/courses/rating - Submit or update a rating for a specific course. ● GET /api/courses/{courseId}/rating - Retrieve rating details for a course. ● GET /api/students/{studentId}/ratings - Get all ratings submitted by a student.
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.