Development Plan

COMPSCI 4ZP6A: DegreeFlow

Version 0

October 25 , 2024

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

Revision History

Date	**Developer(s)**	**Change**
25-Oct-2024	Aniruddh Arora ,Anupam Raj,Anderson Zhou,Harshit Sharma,Sarah Neelands,Yanchun Wang	Created first draft of document.

1. Team Meeting Plan & Team Communication Plan

To ensure the success of our capstone project, we have established a detailed team meeting and communication plan. We will conduct weekly check-in calls on Microsoft Teams involving all team members. During these meetings, we will review the progress made, discuss any challenges or roadblocks, and plan our next steps to maintain consistent progress. In addition to the weekly meetings, we will maintain an open line of communication via a dedicated group chat on Microsoft Teams to ensure ongoing support, discussion, and collaboration between members outside of formal meetings. This open communication allows the team to seek assistance, share updates, and resolve issues in real time.

For project management, we will utilize GitHub Issues to track tasks, manage workload, and ensure transparency in our workflow.. All code merges into the main branch will undergo a peer review by other team members to maintain quality standards and ensure everyone remains informed.

For document collaboration, we will use Google Docs. This will enable real-time, collaborative editing of documents, allowing all team members to contribute simultaneously, leave comments, and make edits efficiently. The combination of Microsoft Teams for meetings and communication, GitHub for project management, and Google Docs for document collaboration provides us with a comprehensive framework for effective teamwork and successful project completion.

2. Team Member Roles

Functional Requirements:

Functional Requirement Section	**Developer(s)**
Course Search: The system will allow users to search for courses using filters such as department, level, prerequisites, semester, and difficulty.	

The front-end will feature a search bar, filter dropdowns, and display search results to the user. (P0)	Anderson Zhou Yanchun Wang
The back-end will support a search engine and filtering process to handle course queries based on user-selected filters, such as department, level, and prerequisites. (P0)	Anderson Zhou Yanchun Wang
Schedule Generation: The system will automatically generate course schedules based on students' selected courses.	
The front-end will feature a timetable interface with drag-and-drop functionality for user convenience. (P0)	Sarah Neelands
The back-end will implement a scheduling algorithm that generates course schedules for students, including conflict detection based on course times. (P0)	Anupam Raj, Sarah Neelands
Prerequisite Verification: The system will verify whether users meet the prerequisites for their selected courses.	
The front-end will provide real-time warnings if prerequisites are unmet. (P0)	Anderson Zhou Yanchun Wang
The back-end will validate course selections by checking academic history and course requirements. (P0)	Anderson Zhou, Yanchun Wang
Integration with Mosaic: The system will access Mosaic API to find real-time information about courses and schedules.	
The front-end will display real-time data updates and information to users. (P0)	Anupam Raj
The back-end will synchronize with the Mosaic API to retrieve and update data. (P0)	Harshit Sharma,Aniruddh Arora

Transcript Parsing: The system will parse unofficial transcripts uploaded by students to extract academic information.	
The front-end will feature an upload button for the transcript and display course recommendations based on the parsed data. (P0)	Anupam Raj
The back-end will parse the PDF and use a recommendation engine to suggest courses based on the user's academic history. (P0)	Sarah Neelands Harshit Sharma
Secure Login: The system will provide secure authentication for students and admins using OAuth/SAML protocols.	
The front-end will feature a login form with two-factor authentication for enhanced security. (P1)	Anupam Raj
The back-end will handle OAuth/SAML integration and session management for secure user authentication. (P1)	Harshit Sharma,Aniruddh Arora
Seat Alert: The system will notify users when a seat becomes available in a previously full course.	
The front-end will feature a subscribe button and send real-time notifications to users when a seat is available. (P1)	Anupam Raj,Aniruddh Arora
The back-end will track course seat availability and send notifications to subscribed users. (P1)	Anupam Raj Sarah Neelands
Real-Time Notifications: The system will notify users in real time about changes in their schedule or course deviations.	
The front-end will display real-time alerts on the user's dashboard. (P1)	Harshit Sharma Yanchun Wang
The back-end will monitor any schedule changes and trigger notifications for affected users. (P1)	Sarah Neelands,Aniruddh Arora

Personalized Course Recommendations: The system will provide personalized course recommendations based on academic history and transcript parsing.	
The front-end will display suggested courses and a personalized academic pathway visualization. (P2)	Sarah Neelands, Anupam Raj
The back-end will utilize recommendation algorithms and data analysis to generate tailored course suggestions. (P2)	Sarah Neelands, Aniruddh Arora, Harshit Sharma
Course Ratings: The system will allow users to rate the courses they have taken and view community ratings.	
The front-end will feature input fields for rating submission and display average ratings for each course. (P3)	Anupam Raj,Aniruddh Arora
The back-end will store and manage course ratings and feedback provided by users. (P3)	Anupam Raj,Aniruddh Arora
User Personalization: The system will enable users to personalize their experience by saving preferences such as favorite courses or departments.	
The front-end will feature a settings page for users to manage their preferences. (P3)	Anderson Zhou Yanchun Wang
The back-end will store user preferences, manage user profiles, and apply these preferences during course searches and recommendations. (P3)	Anderson Zhou Yanchun Wang
Advanced What-If Scenarios: The system will allow users to explore alternative academic paths and assess their impact on graduation.	
The front-end will provide an interactive scenario builder and detailed reports on potential academic outcomes. (P3)	Yanchun Wang,Sarah Neelands

The back-end will simulate academic scenarios and analyse their impact on user progress and graduation. (P3)	Harshit Sharma,Anupam Raj, Anderson Zhou

Non-Functional Requirements:

Non-Functional Requirement Section	**Developer(s)**
Look and Feel	Sarah Neelands,Aniruddh Arora
Usability and Humanity	Sarah Neelands, Harshit Sharma
Performance	Anupam Raj, Anderson Zhou, Harshit Sharma
Security	Anupam Raj, Harshit Sharma,Aniruddh Arora
Legal	Anupam Raj, Yanchun Wang,Aniruddh Arora

Project Coordinator/Manager: Harshit Sharma

3.Workflow Plan

A) Github plan

The Main branch will contain the final version of the project and feature branches will be used to test or change code. We will be using Pull requests to merge new code into the main branch. This will allow the team to review the code, make changes, and test the code before merging it into the main branch, reducing the risk of errors. We will also be using Github issues to keep track of ideas or problems.
Basic workflow:

1. Create a new branch when you start working on a new feature or when fixing a problem.
2. Commit changes to that branch.
3. Create a pull request when you are done and ready to review.
4. Once reviewed, merge the pull request into the main branch.

B) Agile methods

We will be using an agile approach for our project. This involves breaking down tasks into smaller more manageable pieces and regularly checking progress to ensure we are staying on track and making needed improvements. We will use github projects to manage our tasks, and track them using issues. Task will be organised into two week sprints with weekly meetings online to share progress, help, and make sure we are meeting deadlines.

C) Location of stored data

Storage for regular data (student, course information, etc) will be in a cloud based database. We will use a secure cloud base solution like Azure SQL to store generated academic pathways, and course related data. That way no personal student information will be stored.

D) Location of algorithm processing:

Scheduling and recommendations algorithms will be run on cloud based servers, likely through a platform like Azure so the system can handle large computations in real time.

E) Tools / methods used to achieve requirements

<u>Course scheduling:</u>

The system will use java based algorithms for scheduling, the student schedules will be based on course availability, prerequisites, and student preferences. The system is expected to generate at least one valid course schedule, within 3 seconds.

<u>Real time notifications:</u>

Twilio API will be used for sms alerts, the backend of the notifications will be through java and the front end of the notification will use javascript. Notifications will be sent within 3 seconds when changes occur.

<u>Course recommendations:</u>

A rule based recommendation algorithm implemented in java, will suggest optimal courses based on prerequisites, degree progress, and students grades. Recommendations will be generated within 3 seconds of user input.

<u>Degree process tracking:</u>

Information from McMaster systems (like academic calendar API) and students provided unofficial transcript, will be used to track students progress in their degree. The java backend will process this information and the front end displaying this information will use javascript. Degree progress will be displayed within 3 seconds of user input.

<u>Authentication and user management:</u>

Azure SSO via SAML will be integrated with the java backend for a secure login. The authentication process will be completed in under 3 seconds.

4.Proof of Concept Demonstration Plan

The main risk for the success of the **DegreeFlow** project is ensuring that the user interface is intuitive and that the system handles data accurately and securely. To address this risk, the proof-of-concept will focus on demonstrating core features primarily through the user interface. It will include a mock login system to simulate user access and a basic dashboard to display key information, such as parsed transcript data. We will also showcase an interactive degree planner for visualizing completed courses and prerequisites, and use UI mockups to represent course ratings. By focusing on these elements, the PoC will demonstrate that the user interface is functional, easy to navigate, and effective in supporting academic planning.

5. Technology

a. Specific Programming Language (Frontend and Backend), Coding Environment, and Unit Testing Framework

- **Frontend:** We will use JavaScript for the frontend, with Visual Studio Code (VS Code) as the primary coding environment due to its extensive support for JavaScript (and React, if used) and its rich ecosystem of extensions for debugging and linting.
- **Backend:** The backend will be developed using Java and Spring Boot with Spring Tool Suite (STS) as the coding environment. STS is optimized for Spring development, offering preconfigured settings and tools that streamline the setup, navigation, and management of Spring Boot applications.
- **Unit Testing:** We will incorporate unit testing using TestNG for the backend. TestNG's advanced testing features, like flexible test configurations and dependency control, align well with our Spring Boot-based project and support efficient test management.

b. Machine Learning (ML) Libraries

- Machine Learning is not included in this project's scope, as we are focusing on AI features that don't require ML-based predictive modeling.

c. GPU Usage and Other Relevant Technology Aspects

- **GPU Usage**: A GPU is not required for this project, as the core functionalities (e.g., database queries, user request handling, scheduling) are CPU-efficient.
- **Additional Relevant Technologies:**
  - **Database:** We will use a cloud-based SQL database (e.g., Azure SQL) for secure data storage and management.
  - **Notifications:** Twilio API will handle real-time SMS alerts (e.g., seat availability), enhancing the user experience.
  - **Authentication:** Azure SSO with SAML will provide secure, university-standard login for students and administrators.
  - **CI/CD:** GitHub Actions will automate testing and deployment, maintaining code quality and consistency.

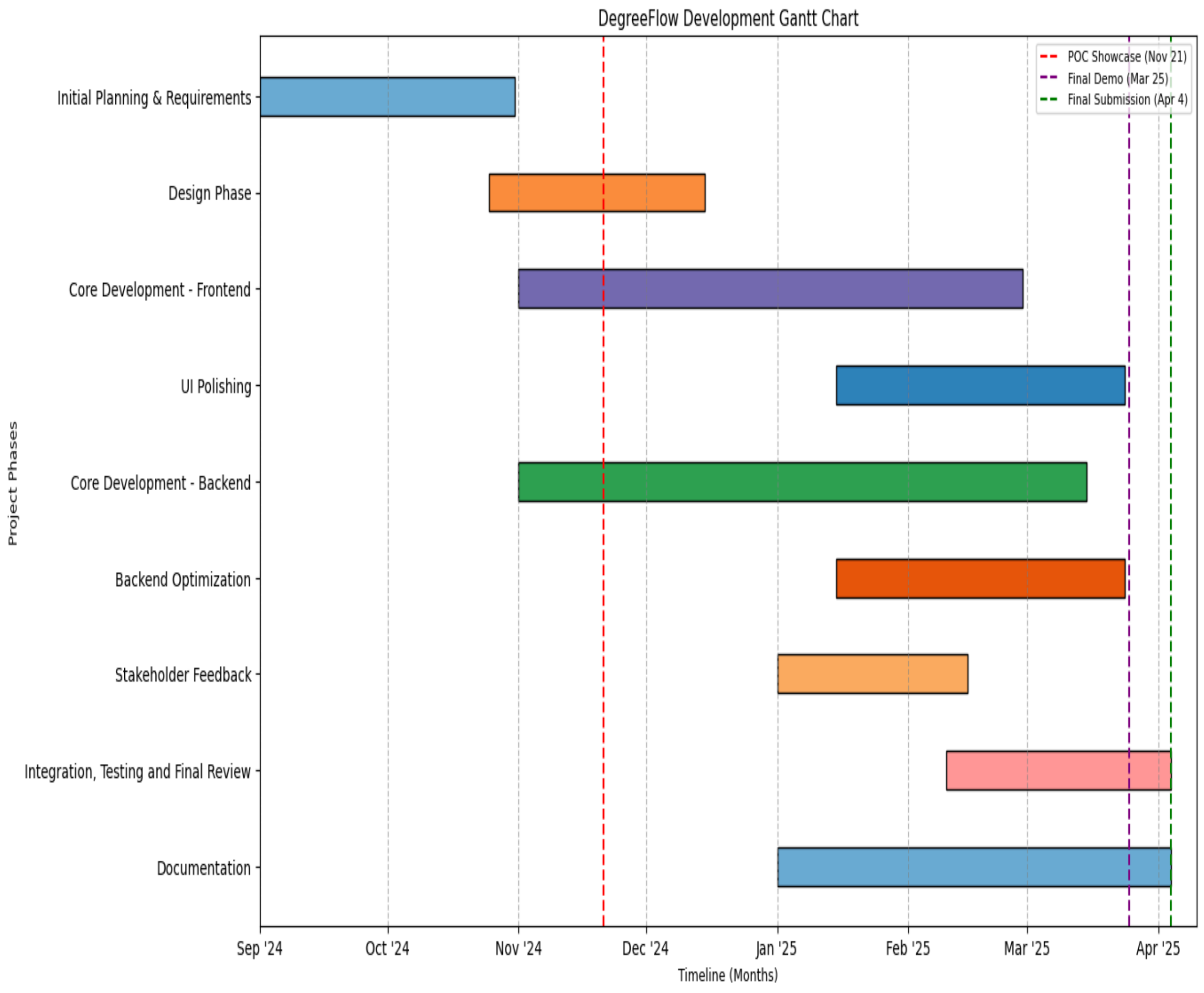
6.Coding Standard

Frontend (JavaScript): We will follow the Airbnb JavaScript Style Guide to maintain code consistency, readability, and best practices across the frontend codebase. We'll use ESLint with Airbnb's configuration to enforce these standards and ensure high code quality.
Backend (Java with Spring Boot): We will adhere to the Google Java Style Guide for the backend. This includes consistent naming conventions, proper indentation, and organized

imports, which will be checked using the Google Java Format tool and integrated into our IDEs.

Unit Testing (TestNG): All unit tests will be stored in a separate `test` directory within the backend project structure. Each major backend component will have a corresponding test file to keep tests organized and to follow best practices in test isolation.

7.Project Scheduling


DegreeFlow Development Gantt Chart
POC Showcase (Nov 21)
Final Demo (Mar 25)
Final Submission (Apr 4)
Project Phases
Initial Planning & Requirements
Design Phase
Core Development - Frontend
UI Polishing
Core Development - Backend
Backend Optimization
Stakeholder Feedback
Integration, Testing and Final Review
Documentation
Sep '24
Oct '24
Nov '24
Dec '24
Jan '25
Feb '25
Mar '25
Apr '25
Timeline (Months)