

Team 6 - Milestone #5

Andrew Wang, Keyu Lin, Lexie King, Shrey Bahadur

Project Overview

Repository: https://github.com/lhking1122/326_Course_Recommendation_System.git

Issues: https://github.com/lhking1122/326_Course_Recommendation_System/issues

Milestones:

Description: The user will input the courses they have already taken, and optionally any academic or career interests they have, and the app will recommend future courses they might be interested in. The recommendations will also consider prerequisite/eligibility requirements to ensure only available courses are suggested. A visual CS major course progress map will show completed courses, current courses, and suggested future courses to help students plan their academic path.

Key Features:

- User Input: Students can enter courses taken, interests, and career goals.
- Course Recommendation: The system recommends appropriate courses based on inputs and provides course details (e.g., prerequisites, credits, schedule, etc.).
- Course Schedule Builder: Students can select courses and generate schedules, the system will indicate time conflicts and provide suggestions for adjustments.
- Course description search up: Students can search up interested courses through the course details tab.
- Course Reviews: Students can look at and leave reviews for courses

Roles

Lexie King:

- Role: Project Manager
- Issues: General UI Design, Past Course Storage, Course Details, Course Recommendation Home Page

Shrey Bahadur:

Role: Note Taker

Issues: Course Review Page, User login data,

Andrew Wang:

Role: Integration Manager

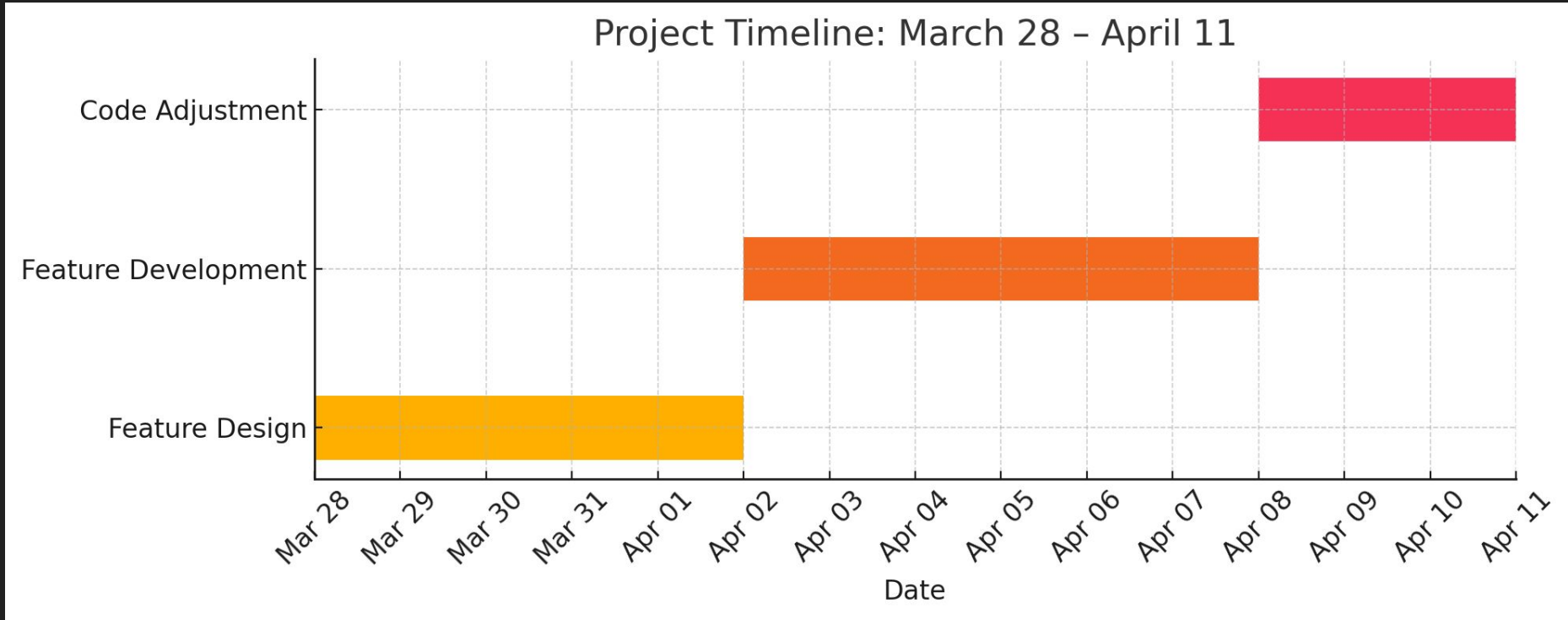
Issues: Ensuring UI, data structures, and the various pages are compatible with each other. Course Progress Chart page.

Keyu Lin:

Role: User Experience Tester

Issues: User Data Structure, Course Recommendation Home Page, User Experience Tester

Historical Timeline



Lexie - Assigned tasks & Pull Request

- Project manager
- General UI Design
- Course Recommendation Home Page UI
- Course Details Page UI
 - Course Data Structure
 - Course search within Course Details Page
- User Data Structure
- Past Course Storage
- Course Recommendation Algorithm

https://github.com/voidpls/326_Course_Recommendation_System/pull/68

https://github.com/voidpls/326_Course_Recommendation_System/pull/67

https://github.com/voidpls/326_Course_Recommendation_System/pull/51

https://github.com/voidpls/326_Course_Recommendation_System/pull/38

https://github.com/voidpls/326_Course_Recommendation_System/pull/34

Lexie - Completed Task Description

- General UI Design
 - I changed all UI for the separate html pages from hyperlinks to Multi-View for page switching.(A. D. 1-2 points, completed, fully functional, branch: homepage)
 - Changed header to a universal header, changed header style.
- Course Details Page UI
 - Course Data Structure
 - Uploaded course list json files and helper js functions to merge and sort all course description information.
 - Course search within Course Details Page
 - The page shows full course list by default, and search box allows users to search by course title, course number, or keywords. (C. 1-2 points, completed, fully functional, branch: course description page)
 - Course filter
 - Allow user to filter courses by course level (100+, 200+,...,800+). (C. 1-2 points, completed, fully functional, branch: course description page)
 - Allow user to filter courses by offered semester (Fall, Spring). (C. 1-2 points, completed, fully functional, branch: course description page)
 - Added RateMyProfessor link to instructor names (C. 1-2 points, completed, fully functional, branch: course description page)
 - Made all prerequisite courses buttons, when user clicks it, it will display that specific prereq course. (C. 1-2 points, completed, fully functional, branch: course description page).
 - Added a back button so user can navigate back to where they were.

Lexie Work Summary - commits

```
commit 6a727b6768504b7bf7e0495f64b4a676db7a575e
Author: lexie <lhking@umass.edu>
Date: Thu Apr 10 16:00:55 2025 -0400
```

Added RateMyProfessor links to instructor names. Changed all Prerequisites courses to buttons so when user clicks it, it will display the specific clicked course. Added back button, so when user clicks it, it will bring user back to where they were before (including scrolling to the position user was at).

```
commit 4b23fa7c5144750ea945c14580ef7f1f1839b4ff
Merge: e766a15 2efcb72
Author: lhking1122 <lexiek1122@gmail.com>
Date: Fri Apr 4 00:14:48 2025 -0400
```

Merge pull request #51 from lhking1122/homepage

Changed file structure. (hyper-links into buttons between tab for multiview). Changed all css ids so each id is specific to each html file.

```
commit 58df63d5d37a39228b704b17c63a6e99a46d5989
Merge: b492fcd b8b817a
Author: lhking1122 <lexiek1122@gmail.com>
Date: Thu Apr 10 16:39:35 2025 -0400
```

Merge pull request #68 from voidpls/Course-Details-Page

Course details added filters by course level (100+, 200+,). added filters by offered semester (Spring, Fall). sorted course description list from lowest course number to largest course number.


```
commit b8b817a65da27378bae2f7daf6e4d95481578300
Author: lexie <lhking@umass.edu>
Date: Thu Apr 10 16:37:27 2025 -0400
```

sorted course description list from lowest course number to largest course number.

```
commit 145771116a72f17383ee145a4aa975f888cbb174
Author: lexie <lhking@umass.edu>
Date: Mon Mar 31 13:25:49 2025 -0400
```

Added course_details_page.js, which provides the search feature and displays the course list from json file. Added helper js files that helps to filter and merge course list json files.

Lexie - Screen Shots of UI Implementation #1

 **Course Recommender**

[Home](#) [Course Details Page](#) [My Course Progress](#) [Course Reviews](#) [My Profile](#)

Input Your Information
Interests

Year in College

Course Recommendations
Course Title 1
Short description of course 1 goes here.

Course Title 2
Short description of course 2 goes here.

© 2025 Course Recommender. All rights reserved.

Lexie - Screen Shots of UI Implementation #2



Course Recommender

[Home](#)[Course Details Page](#)[My Course Progress](#)[Course Reviews](#)[My Profile](#)

Search Courses

Filter by Course Level

☐ 100☐ 200☐ 300☐ 400☐ 500

Filter by Semester

☐ Fall☐ Spring

CICS 109: Introduction to Data Analysis in R

Instructors: [Jasper McChesney](#)

Description: An introduction to data analysis in the open-source R language, with an emphasis on practical data work. Topics will include data wrangling, summary statistics, modeling, and visualization. Will also cover fundamental programming concepts including data types, functions, flow of control, and good programming practices. Intended for a broad range of students outside of computer science. Some familiarity with statistics is expected. 1 credit.

Prerequisites: N/A

Credits: 1

Offered Semester: Spring

CICS 110: Foundations of Programming

Instructors: [Cole Reilly](#), [Jacob Urisman](#), [Brett Mullins](#), [Prit Pritam Shah](#), [Aadam Anish Lokhandwala](#)

Description: An introduction to computer programming and problem solving using computers. This course teaches you how real-world problems can be solved computationally using programming constructs and data abstractions of a modern programming language. Concepts and techniques covered include variables, expressions, data types, objects, branching, iteration, functions, classes, and methods. We will also cover how to translate problems into a sequence of instructions, investigate the fundamental operation of a computational system and trace program execution and memory, and learn how to test and debug programs. No previous programming experience required. (Gen. Ed. R2) Prerequisite: R1 (or a score of 15 or higher on the math placement test Part A), or one of the following courses: MATH 101&102 or MATH 104 or MATH 127 or MATH 128 or MATH 131 or MATH 132. 4 credits.

Prerequisites: R1 (or a score of 15 or higher on the math placement test Part A), or one of the following courses: MATH 101&102 or MATH 104 or MATH 127 or MATH 128 or MATH 131 or MATH 132

Credits: 4

Lexie - Screenshots of Code Snippet and Explanation

```
document.addEventListener('DOMContentLoaded', () => {

    divHomePage = document.getElementById('HomePage');
    fetch("homepage.html")
        .then(res => res.text())
        .then(data => {
            divHomePage.innerHTML = data;
            divHomePage.style.display = "block";
        });

    divCourseDetailsPage = document.getElementById('CourseDetailsPage');
    fetch("course_details_page.html")
        .then(res => res.text())
        .then(data => {
            divCourseDetailsPage.innerHTML = data;
            divCourseDetailsPage.style.display = "none";
            fetch("/src/Services/course-details-page/sorted_full_course_list.json")
                .then(response => response.json())
                .then(data => {
                    courses = data;
                    displayCourses(courses);    // Show all courses
                    setupSearch();              // Enable live search
                })
                .catch(error => console.error("Failed to load course data:", error));
        });
});
```

The screenshot shows the 'Course Recommender' web application. At the top is a dark red navigation bar with the application name and a logo. Below it is a secondary dark red bar containing navigation links: Home, Course Details Page, My Course Progress, Course Reviews, and My Profile. The main content area is white and divided into two columns. The left column, titled 'Search Courses', features a search input field and a 'Search' button. Below these are two filter sections: 'Filter by Course Level' with radio buttons for 100, 200, 300, 400, and 500; and 'Filter by Semester' with radio buttons for Fall and Spring. The right column displays course details for 'CICS 109: Introduction to Data Analysis in R'. It lists the instructor as Jasper McChesney, provides a detailed description of the course, states prerequisites as N/A, and indicates 4 credits. Below this, it shows details for 'CICS 110: Foundations of Programming', listing multiple instructors, a detailed description, prerequisites (R1 or a score of 15 or higher on the math placement test Part A, or one of the following courses: MATH 101&102 or MATH 104 or MATH 127 or MATH 128 or MATH 131 or MATH 132), and 4 credits.

For milestone 4, we wrote all of our tabs separately and connected them with links. So I now changed everything to multi-view, making tabs buttons and using js to fetch individual tab htmls. JavaScript fetches and injects these pages into the correct sections of the DOM, enabling a multi-view setup where different parts of the application can be shown or hidden as needed. This modular design keeps the overall UI cleaner, easier to maintain, and allows for more flexibility when updating individual pages without affecting the entire structure.

Lexie - Challenges and Insights

Changing the UI to multi-view was definitely a big challenge. Since I made the change after we incorporated js files to our project, there was a big hassle to debug conflicts between js files for each page and the js in index.html, it took a while to modify all the js files so every page would display correctly. And we also had individual css files for each page, after I made the UI multi-view, I realized that all the css has conflicts since there was a lot of redundant ids, so I had to go and change all the CSS file tags and ids to make sure all the style was correct.

Working in a collaborative environment taught me the importance of clear communication and planning, especially when making major changes like switching to a multi-view UI. Since multiple team members were working on different JavaScript and CSS files, it was crucial to coordinate changes carefully to avoid conflicts. I learned that early discussions about file structures, naming conventions, and dependencies could save a lot of time and confusion later. Collaboration also helped me realize that documenting changes and helping teammates understand updates is just as important as writing the code itself. Overall, teamwork emphasized the value of flexibility, patience, and proactive problem-solving when working on shared projects.

Lexie - Future Improvements & Next Steps

Future improvements: I would definitely try to plan the format of the webpage from the start, changing the structure mid-way to multi-view is a pain.

Next Steps: Next I'm going to work on User data storage, and the recommendation algorithm. I haven't done much for the homepage UI yet since it will require displaying course recommendations. And I will try to add a few more features to the homepage if possible (interactive charts and stuff maybe).

Lexie - Links to Issues

[Course Data Structure](#)

[Search Feature](#)

[Display Courses](#)

[Filter by Semester](#)

[Filter by course level](#)

[Change UI to Multi-View](#)

[Changed Header](#)

[Add RateMyProfessor Link to Instructor names](#)

[Make Prereq courses clickable](#)

[Add back button, including scrolling to where the user was](#)

Andrew Work Summary

For this milestone I did the following:

- I made the course selection flowchart fully functional - inputs are captured and can be outputted back.
- Added local persistence to the course selection form.
 - First localStorage then migrated to indexedDB.
- Added the pre-2023 flowchart as well. This can be toggled via a multiview UI selection button. Both flowcharts are fully functional forms.
- Fixed bugs to allow integration with the page's overall multiview UI

(ALL OF MY WORK WAS DONE UNDER THE 'course_selection' BRANCH)

Andrew Work Summary

The issues that I resolved for this milestone are:

Capture user flowchart inputs:

https://github.com/voidpls/326_Course_Recommendation_System/issues/29

Add pre-2023 flowchart:

https://github.com/voidpls/326_Course_Recommendation_System/issues/30

Switch course selection local persistence from localStorage to indexeddb:

https://github.com/voidpls/326_Course_Recommendation_System/issues/71

Locally persist which flowchart the user selected:

https://github.com/voidpls/326_Course_Recommendation_System/issues/70

(WIP, as thought of this feature last-minute)

Andrew Work Summary

PRs closed:

https://github.com/voidpls/326_Course_Recommendation_System/pull/44

https://github.com/voidpls/326_Course_Recommendation_System/pull/54

https://github.com/voidpls/326_Course_Recommendation_System/pull/55

https://github.com/voidpls/326_Course_Recommendation_System/pull/73 (Accidentally committed to main, so missing commits. Commit below:)

Switch from localstorage to indexeddb API for storing form data



voidpls committed 51 minutes ago · ✓ 1 / 1

83f0547



Andrew Work Features

Feature: capture user inputs (complete)

The flowchart form captures all of the user's inputted fields. This includes toggle-able fields and also text-input fields. It returns the inputs as an array, which can be passed to the backend later.

This uses form validation (**feature B**) in which non-valid inputs (in this case only blank text fields, are ignored). The form uses click and keyup event listeners to toggle fields (**feature C**).

I believe that this is a 1 point basic feature, as it uses dynamic content updating.

BS-CS Tracking Form for Major Requirements

Effective Fall 2016

Four Math Courses		Two Introductory CS Courses		
MATH 131 Calc I	MATH 132 Calc II	CS 121 Intro to Problem Solving	CS 187 Data Structures	
MATH 233 Multivariate Calc OR STAT 515 Stats I	Four CS Core Courses			
MATH 235 Linear Algebra	CS 220 Programming Methodology	CS 230 Comp Sys Principles		
	CS 240 Reasoning Under Uncertainty	CS 250 Intro to Computation		
Eight Upper-Level CS Courses				
CS 311 Algorithms	CS 300+ 326	CS 300+ 383	CS 300+ 	CS 320 OR CS 326
CS 300+ 	CS 400+ 	CS 400+ 	CS 400+ 	CICS 305 Social Issues in Comp. (OR other JYW)
Submit Selected Courses				
Selected Courses: MATH131, MATH132, MATH233_or_STAT515, MATH235, CS121, CS187, CS220, CS230, CS240, CS250, CS_300_elective_1: 326, CS_300_elective_2: 383, CS320_or_CS326, CICS305				

Andrew Work Features

Feature: multiview UI + pre-2023 flowchart (complete)

The user can toggle between which flowchart to use via buttons (**feature D**). Upon clicking, the selected form (.html component) is loaded using fetch (**feature F**) and dynamically inserted using javascript DOM manipulation (**feature A**).

I believe that this is a 2 point basic feature, as it makes use of 2 different techniques: multiview UI and dynamic content updates.

Which course track are you on?

Computer Science BS 2023-Now

Computer Science BS Pre-2023

Andrew Work Features

Feature: data persistence (complete)

User inputs are remembered by the site using IndexedDB (**feature E**). This allows form inputs to be kept upon reloading/changing pages. Upon loading the form, persisted data is dynamically entered into the form.

I believe this is a 3 point advanced feature, as it uses IndexedDB and dynamic content updates. I created a helper function for IndexedDB and fully utilized the Promise functionality that the IndexedDB API provides, and building my own promises with it:

```
// get selected courses (promise)
const selectedCourses = new Promise((resolve, reject) => {
  const request = store.get('selectedCourses');
  request.onsuccess = () => resolve(request.result ? request.result.data : []);
  request.onerror = () => reject(request.error);
});

// get course inputs (promise)
const courseInputs = new Promise((resolve, reject) => {
  const request = store.get('courseInputs');
  request.onsuccess = () => resolve(request.result ? request.result.data : {});
  request.onerror = () => reject(request.error);
});

// resolve database get promises
const [selectedIds, inputValues] = await Promise.all([selectedCourses, courseInputs]);
```

0	"courseInputs"	▼ {id: 'courseInputs', data: {...}} ▶ data: {CS_300_elective_1_input: id: "courseInputs"
1	"selectedCourses"	▼ {id: 'selectedCourses', data: Ar ▶ data: (9) ['MATH132', 'CICS110' id: "selectedCourses"

Andrew Work Code

This code is the primary logic for my multiview UI switching (demonstrated before)/

This piece of logic helps make the overall UI more simplistic and user friendly, as the user can easily toggle between which UI they specifically need, without having to see the other UI after switching.

I originally had issues with initiating the loaded flowchart, but updating my `initFlowchart()` method to be more universal fixed it.

```
async function changeCourse(btns, btn) {
  if (btn.classList.contains("course-progress-chart-button-selected")) return

  // remove selected from all selection btns
  Array.from(btns).forEach(btn => btn.classList.remove('course-progress-chart-button-selected'));

  // add selected to new btn
  btn.classList.add('course-progress-chart-button-selected');

  const parent = document.getElementById('course-flowchart-container');
  if (btn.id === 'flowchart-2023') {
    const cs_bs_2023_flowchart = await fetch('Components/course_progress_chart/cs_bs_2023_flowchart.html');
    const html = await cs_bs_2023_flowchart.text();

    parent.innerHTML = html;

  } else if (btn.id === 'flowchart-2016') {
    const cs_bs_2016_flowchart = await fetch('Components/course_progress_chart/cs_bs_2016_flowchart.html');
    const html = await cs_bs_2016_flowchart.text();

    parent.innerHTML = html;
  }

  await initFlowchart();
}
```

Andrew Work Challenges and Insights

One of the biggest challenges was switching from `localStorage` to `indexedDB`. I had initially implemented form persistence using `localStorage`, but later rewrote my persistence code to work with `indexedDB`. I was originally hesitant to use `indexedDB` because of the very complicated API compared to `localStorage`, but creating a helper function for initializing the `indexedDB` database made it a lot easier.

Also, using `indexedDB` Promises to my advantage, instead of fighting against it for being unfamiliar, took more time but ultimately made the integrating `indexedDB` process a very positive change for my code.

In terms of being in a team environment, it's often difficult as you don't have access to everything going on in the big picture. However, putting each member's pages/features together and seeing it all work was very satisfying.

Andrew Work Next Steps

I have an unfinished feature from this milestone, as I only thought of it at the last minute - persisting which form/UI the user has chosen. I'll implement this for the next milestone.

For the next milestone, I plan to have my form persist through the back-end, so that users can login on a different browser and still access their saved form. I also plan to have the form submission push the submitted input data to the server (POST endpoint), for which can be used later on for the user profile and for course recommendations.

https://github.com/voidpls/326_Course_Recommendation_System/issues/79

https://github.com/voidpls/326_Course_Recommendation_System/issues/78

Shrey Work Summary

This milestone I worked on making the course review page dynamically load in content. The courses displayed are all now coming from json data we have and when a course is selected it is displayed properly. I also added a dropdown that enables the user to add a review On top of that I also made some minor UI changes.

Course List - (A) Dynamic updates 1-2 points (C) Event Handling 1-2 points
Selected Course Header - (A) Dynamic updates 1-2 points
Add Review Dropdown/Button - (C) Event Handling 1-2 points
Review List - (A) Dynamic updates 1-2 points

Shrey - Commits and PRs and Issues

Commits:

shows selected course



shreyBaha committed 2 hours ago

dynamically loads in courses to review page



shreyBaha committed 6 hours ago

Merge pull request [#77](#) from voidpls/shrey/course_review



shreyBaha authored 10 minutes ago · ✓ 1 / 1

add review feature added



shreyBaha committed 11 minutes ago · ✓ 1 / 1

PR Link: https://github.com/voidpls/326_Course_Recommendation_System/pull/77

Issues Links:

https://github.com/voidpls/326_Course_Recommendation_System/issues/19?issue=voidpls%7C326_Course_Recommendation_System%7C33

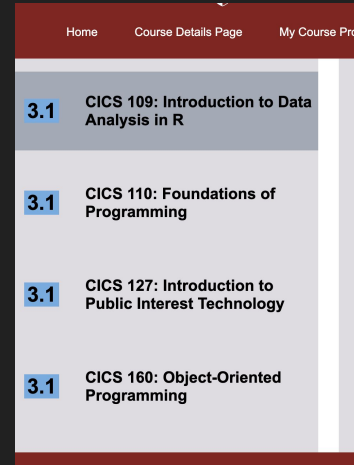
https://github.com/voidpls/326_Course_Recommendation_System/issues/19

https://github.com/voidpls/326_Course_Recommendation_System/issues/17

Shrey - Code snippet

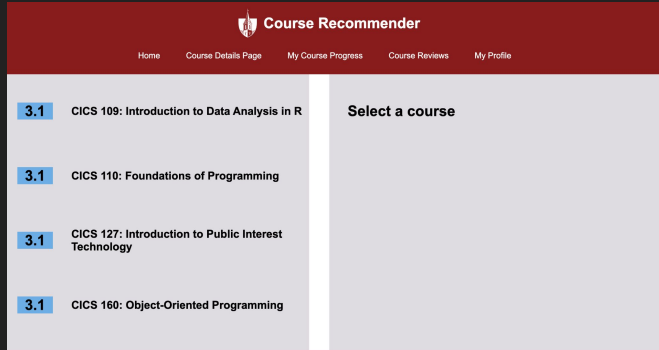
This code snippet corresponds to the list of courses shown below and is dynamically generated when the page is loaded

```
async function loadCourses(){
  let courseListElement = document.getElementById("course-list")
  console.log(courseListElement)
  let resoonse = await fetch("Services/course-details-page/complete_course_list.json")
  let courseList = await resoonse.json()
  console.log(courseList)
  for(let i = 0; i < courseList.length; i++){
    let course = courseList[i]
    let course_item = document.createElement("li")
    course_item.className = "course-item"
    course_item.innerHTML = `<div class="rating-box"><h1 class="rating-text">3.1</h1></div>`
    //course_item.onclick = courseClick
    course_item.addEventListener("click", function (e) {courseClick(course)}, false)
    course_item.params = course
    courseListElement.appendChild(course_item)
  }
}
```

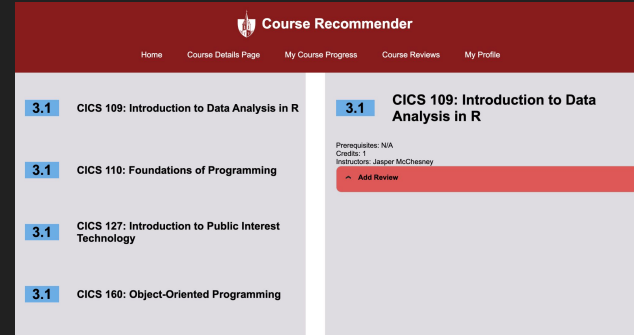


Shrey - UI Screenshots

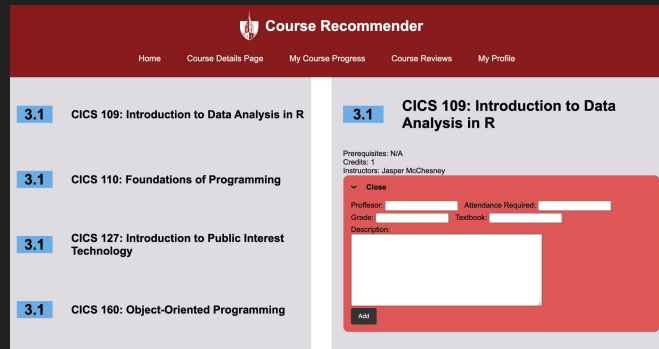
When you first land



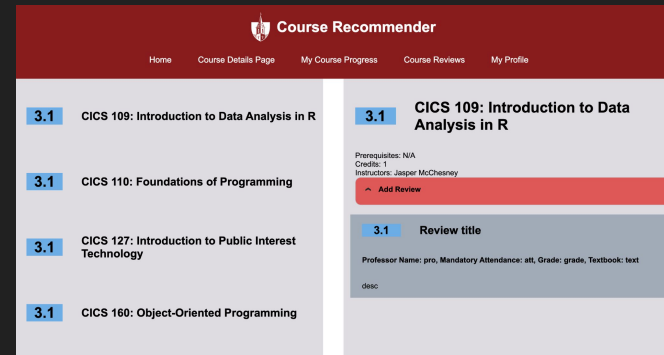
When you select a course



Adding a review



Review Added



Shrey - Challenges and Key takeaways

Challenges: I had some trouble getting my js file to work at first as I wasn't properly establishing it in the index.html file so that took me a bit of time. I also had some trouble with the css for the add review dropdown. I also had trouble with adding a function to an event listener with parameters.

Key Takeaways: Since we were changing the way we were loading pages, we had to collaborate with each other to make sure all of our pages were loading in correctly. I learned that it's important to communicate with your team on application wide changes to make sure everyone can catch up/update what they need to quickly.

Shrey - Future improvements

I need to touch up the css of the file as well as add a couple of js features I forgot (adding review title to dropdown and form validation). Next up I will be looking to implement the backend by doing the user login info schema and protection as well as connecting the backend to the frontend of the review page.

Keyu Work Summary

Change the text box under the interests module on the profile page to a multi-select box (supports selecting multiple interests). Implemented the effect of checking and unchecking the selections in multi-select box. Implemented the effect of save button, after clicking the save button, the data will be updated in the profile module and also saved in indexedDB.

Keyu - Pull requests

https://github.com/voidpls/326_Course_Recommendation_System/pull/50

https://github.com/voidpls/326_Course_Recommendation_System/pull/69

Interests

Select interests...

AI

Machine Learning

Data Science

Web Development

Cybersecurity

Cache Storage

Cookies

Indexed DB

- http://127.0.0.1:5501
 - CourseRecommenderDB (default)
 - userProfile

Local Storage

Session Storage

Filter Items

Key	Value
1	{"id":1,"name":"123","email":"321","phone":"888","gradYear":"2026","contact":"Phone","interests":"AI,Data Science"}

Filter values

1:{"id":1,"name":"123","em...sts":"AI,Data Science"}

Parsed Value

1:Object

- id:1
- name:"123"
- email:"321"
- phone:"888"
- gradYear:"2026"
- contact:"Phone"
- interests:"AI,Data Science"
- proto:Object

Keyu - Code snippet

```
// Handle option selection
options.forEach(option => {
  option.addEventListener('click', function() {
    const value = this.getAttribute('data-value');

    if (selectedInterests.includes(value)) {
      // Remove if already selected
      selectedInterests = selectedInterests.filter(item => item !== value);
      this.classList.remove('selected');
    } else {
      // Add if not selected
      selectedInterests.push(value);
      this.classList.add('selected');
    }

    updateSelectedTags();
    updateHiddenInput();
  });
});
```

The code is mainly for the implementation of some of the functions of the multi-select box (“checked” and “unchecked” related events and effects)

Keyu - Challenges

Due to the lack of familiarity with the use of indexedDB, the process of writing code often encountered errors. Even if the code runs smoothly afterward, the data can not be correctly stored in indexedDB. After perfecting the data storage, the data could not be read correctly. Finally, after spending a lot of time on modification and testing, the relevant functions were basically implemented.

Keyu - Next steps

The functionality of the profile page has basically been implemented, the next plan is to continuously test and improve some of the details of this page, such as the input content restrictions