

Project Report for Milestone 2

Project: Course Advising Portal

Student Name: Quhura Fathima

Student UIN: 01275914

Project Description: The Course Advising Portal continues to evolve as a comprehensive platform aimed at enhancing the academic advising process for students at Old Dominion University. Building on the foundational features established in Milestone 1, this milestone introduces significant enhancements focused on the management of course advising functionalities. The portal provides a seamless experience for both students and administrators, ensuring efficient handling of advising tasks while maintaining a high standard of security, user-friendliness, and responsiveness.

Key Features:

1. Admin Functionality:

- **Prerequisite Management:** Admins can now design and create a dedicated prerequisite form that displays courses from levels 100 to 499. This form allows for enabling or disabling courses based on the institution's advising policies.
- **Database Updates:** Admin selections made through the prerequisite form are reflected in the database, ensuring real-time updates to course availability.

2. Student Advising Menu:

- **Access upon Login:** Students can access an Advising menu upon logging in, facilitating direct navigation to advising functionalities.

3. Course Advising History:

- **Display of Advising Records:** A new form is created to display previously submitted advising records, including columns for Date, Term, and Status. If no records are found, the system indicates this clearly.
- **Dynamic Record Management:** Students can create new advising records through a structured form that captures their advising history, including sections for Last Term, Last GPA, Advising Term, prerequisites, and course plans.

4. Dynamic Course Plan and Prerequisites Management:

- **User-Editable Forms:** Students can dynamically add and remove prerequisites and course selections using dropdown menus, ensuring they can tailor their advising plans based on their academic trajectory.
- **Validation Rules:** The system prevents students from selecting courses they have already completed in previous terms, maintaining academic integrity.

5. **Status Management:**

- **Editing Existing Records:** When users click on a record from their advising history, they are redirected to the course advising form pre-populated with existing data. Records marked as 'approved' or 'rejected' are not editable, while those marked as 'pending' can be modified.

6. **Deployment:**

- **Live Demonstration:** The frontend, backend, and database components of the portal are deployed on a server, demonstrating the full functionality of the application in a live environment.

Technologies Used:

1. **Frontend:**

- **React.js:** The portal's frontend is built using React.js, providing a dynamic, responsive, and seamless user interface.
- **Material UI:** Material UI components are utilized to maintain a clean, modern design that adheres to accessibility and usability standards.

2. **Backend:**

- **Node.js and Express.js:** The backend server is developed using Node.js and Express.js, providing a fast, scalable, and flexible architecture for handling user requests, authentication, and database operations.
- **Session Management:** The application employs session-based authentication to securely maintain user sessions.

3. **Database:**

- **MySQL:** Data persistence is managed by MySQL, a reliable and scalable relational database that securely stores user information and advising records.

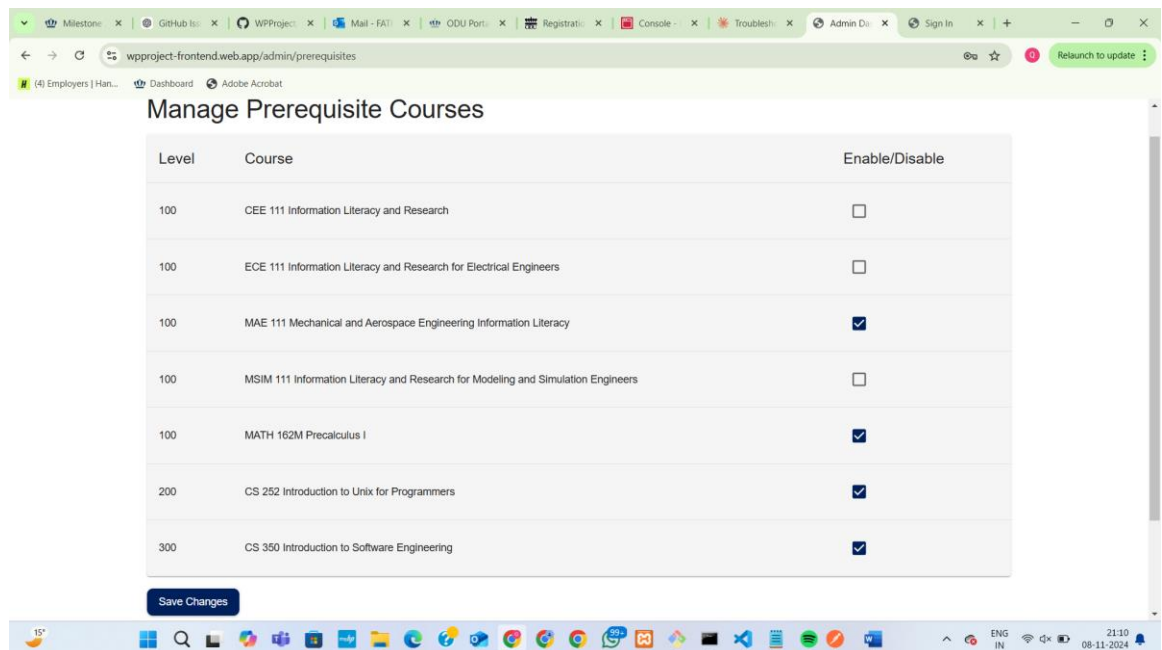
4. **Deployment:**

- **Render:** The backend is deployed on Render, a cloud platform that automates deployment, scaling, and managing applications. Render provides a streamlined process for deploying Node.js applications, allowing for easy management of backend services and database connections.
- **Firebase:** The frontend is deployed using Firebase Hosting, which offers a fast and secure way to host web applications. Firebase Hosting supports single-page applications (SPAs) like those built with React.js and provides features such as automatic SSL certificates and global content delivery.

5. Email Services:

- **Nodemailer:** Email services are implemented using Nodemailer, facilitating OTP verification and notifications to users.

Pre-requisite Form page for admin:



The screenshot shows a web browser window with the URL `wproject-frontend.web.app/admin/prerequisites`. The page title is "Manage Prerequisite Courses". Below the title is a table with three columns: "Level", "Course", and "Enable/Disable". The table contains seven rows of course data. Each row has a checkbox in the "Enable/Disable" column. A "Save Changes" button is located at the bottom left of the table area. The browser's taskbar at the bottom shows the date as 08-11-2024 and the time as 21:10.

Level	Course	Enable/Disable
100	CEE 111 Information Literacy and Research	<input type="checkbox"/>
100	ECE 111 Information Literacy and Research for Electrical Engineers	<input type="checkbox"/>
100	MAE 111 Mechanical and Aerospace Engineering Information Literacy	<input checked="" type="checkbox"/>
100	MSIM 111 Information Literacy and Research for Modeling and Simulation Engineers	<input type="checkbox"/>
100	MATH 162M Precalculus I	<input checked="" type="checkbox"/>
200	CS 252 Introduction to Unix for Programmers	<input checked="" type="checkbox"/>
300	CS 350 Introduction to Software Engineering	<input checked="" type="checkbox"/>

Save Changes

CourseAdvisngForm Page

Admin DashboardHeroku | LoginFirebase CLIODU - Old Dominionwproject-backendMilestone1_ReportHome

localhost:3000/student/advising-form

(71) WhatsAppASUS E-ServiceAmazon.co.uk - Onlin...AgodaExpress VPNMcAfee SecurityLastPass password...Other favorites

Course Advising Form

History

Last Term *
Fall 2029

Last GPA *
3

Advising Term *
Spring 2030

Prerequisites

☐ Please check the box if the student has no prerequisite

100

CEE 111 Information Literacy and Research

Add Prerequisite

Course Plan

400

Add Course Plan

Submit Advising Form

CS 418 Advanced Algorithms

CS 462 Advanced DS Algorithms

CS 498 Operating Systems

15°

Windows taskbar icons

ENG IN21:2308-11-2024

CourseAdvisingHistory Page

Admin DashboardHeroku | LoginFirebase CLIODU - Old Dominionwproject-backendMilestone1_ReportHome

localhost:3000/student/advising-history

(71) WhatsAppASUS E-ServiceAmazon.co.uk - Onlin...AgodaExpress VPNMcAfee SecurityLastPass password...Other favorites

Course Advising History

Date	Term	Status
2024-11-07T05:00:00.000Z	Spring 2025	Pending
2024-11-07T05:00:00.000Z	Spring 2030	Pending
2024-11-06T05:00:00.000Z	Spring 2027	Approved
2024-11-06T05:00:00.000Z	Spring 2028	Rejected

OLD DOMINION UNIVERSITY

Norfolk, VA 23529

Contact us & mailing information

Directions to campus

CONNECT WITH #ODU

USEFUL LINKS

Apply to ODU

Give to ODU

15°

Windows taskbar icons

ENG IN21:2508-11-2024

Update Advising Form Page

The screenshot shows a web browser window with the URL `localhost:3000/student/advising-form?advisingTerm=Spring%202025&status=Pending`. The page title is "Course Advising Form". It contains several input fields and buttons:

- History:** A section with three input fields: "Last Term" (containing "Fall 2024"), "Last GPA" (containing "4"), and "Advising Term" (containing "Spring 2025").
- Prerequisites:** A section with a checkbox "Please check the box if the student has no prerequisite" and a button "Add Prerequisite".
- Course Plan:** A section with two dropdown menus. The first dropdown is set to "500" and the second to "Data Science Fundamentals". The second dropdown is set to "600" and the second to "Artificial Intelligence". There is a button "Add Course Plan" below these.
- Update Advising Form:** A blue button at the bottom of the form.

The browser's taskbar at the bottom shows various application icons and the system clock indicating 21:26 on 08-11-2024.

Backend Deployment

The screenshot shows the Render dashboard for a project named "WPProject-backend". The main section displays the deployment status as "Live" with a green checkmark. The deployment was completed on November 8, 2024, at 5:11 PM. The deployment log is visible, showing the following steps:

- Nov 8 05:11:52 PM: Server running on port 10000
- Nov 8 05:11:52 PM: Database connected
- Nov 8 05:11:52 PM: Admin user already exists.
- Nov 8 05:12:02 PM: Generated OTP for qhuraefathima56@gmail.com: 839791
- Nov 8 05:12:37 PM: OTP sent to qhuraefathima56@gmail.com: 250 2.0.0 OK 1731103957 6a1883df08f44-6d3961ed26asm24196176d9.38 - gsmtp
- Nov 8 05:12:37 PM: Session after setting user: Session {
- Nov 8 05:12:37 PM: cookie: {
- Nov 8 05:12:37 PM: path: '/',
- Nov 8 05:12:37 PM: _expires: 2024-11-09T22:12:36.764Z,
- Nov 8 05:12:37 PM: originalMaxAge: 86400000,
- Nov 8 05:12:37 PM: httpOnly: true,
- Nov 8 05:12:37 PM: secure: true
- Nov 8 05:12:37 PM: },
- Nov 8 05:12:37 PM: user: { email: 'qhuraefathima56@gmail.com', isAdmin: 1 }

The dashboard also shows a sidebar with navigation links: Events, Logs, Disks, Environment, Shell, Previews, Jobs, Metrics, Scaling, and Settings. The bottom of the screen shows the Windows taskbar with various application icons and the system clock indicating 23:41 on 08-11-2024.

Frontend Deployment

```
MINGW64~/Users/qana/OneDrive/Documents/GitHub/WPProject/frontend
Counting objects: 100% (17/17), done.
Delta compression using up to 12 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 717.06 KiB | 5.31 MiB/s, done.
Total 10 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/qfath001/WPProject.git
235cb004b..b7eb1745b main -> main

qurra@happy_lappy MINGW64 ~/OneDrive/Documents/GitHub/WPProject/frontend (main)
$ npm run build
> react-scripts build
Creating an optimized production build...
Compiled successfully.

File sizes after gzip:
  152.27 kB build/static/js/main.8717c2c8.js

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
  serve -s build

Find out more about deployment here:
  https://cra.link/deployment

qurra@happy_lappy MINGW64 ~/OneDrive/Documents/GitHub/WPProject/frontend (main)
$ firebase deploy
--- Deploying to 'wproject-frontend'...
i deploying hosting
i hosting[wproject-frontend]: beginning deploy...
i hosting[wproject-frontend]: found 7 files in build
i hosting: upload complete
+ hosting[wproject-frontend]: file upload complete
+ hosting[wproject-frontend]: finalizing version...
+ hosting[wproject-frontend]: version finalized
+ hosting[wproject-frontend]: releasing new version...
+ hosting[wproject-frontend]: release complete
+ Deploy complete!

Project Console: https://console.firebase.google.com/project/wproject-frontend/overview
Hosting URL: https://wproject-frontend.web.app

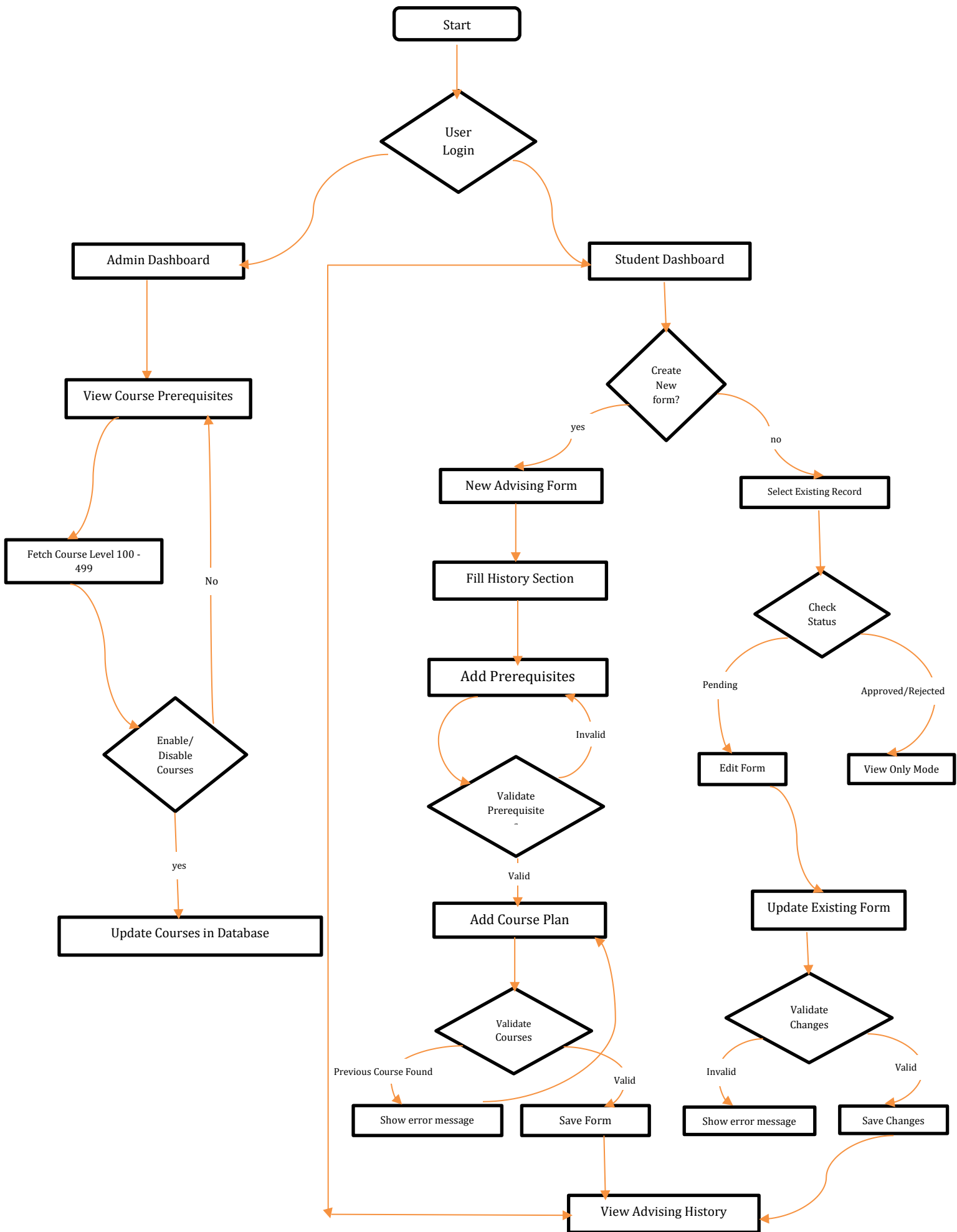
qurra@happy_lappy MINGW64 ~/OneDrive/Documents/GitHub/WPProject/frontend (main)
$
```

2. Milestone Accomplishments

Fulfilled	Feature#	Specification
Yes	1	Design and create the prerequisite form for admin, displaying courses from levels 100 to 499 with fields: Level, Course, Enable/Disable
Yes	2	Update database based on the admin's selection of prerequisites.
Yes	3	Implement an Advising menu accessible upon student login.
Yes	4	Design and create the 'Course Advising History' form to display previously submitted records or indicate no records with columns Date, Term, Status.
Yes	5	Develop a form for creating new 'Course Advising' with sections: History, prerequisites, and course plan.

Yes	6	Implement the History section with fields: Last Term, Last GPA, Advising Term.
Yes	7	Enable dynamic addition of rows for prerequisites section with fields: Level, Course Name (dropdowns).
Yes	8	Enable dynamic addition of rows for the course section with fields: Level, Course Name (dropdowns).
Yes	9	Implement rules for course selection, preventing addition of previously taken courses.
Yes	10	Implement functionality to save new entries for display in 'Course Advising History' with a Pending status.
Yes	11	Redirect users to Course Advising form with selected record pre-populated; editable if status is 'pending'.
Yes	12	Deploy frontend, backend, and database on a server; demonstrate from live server.

3. Architecture



Flow of architecture

1. Admin Portal Functionality:

- **Design and Create Prerequisite Form:**

- **Frontend:**

- PrerequisiteForm.js: This component fetches all courses with levels between 100 and 499 from the backend. It allows the admin to view, enable, or disable courses using a checkbox interface.
 - Axios requests are used to communicate with backend APIs to fetch and update course data.

- **Backend:**

- adminPrerequisite.js and adminRoutes.js: These modules handle API endpoints for fetching courses, updating enable/disable status, and ensuring secure access using verifyAdmin middleware.
 - verifyAdmin middleware ensures that only authenticated admins can access the prerequisite form functionalities.

- **Database Update:**

- Courses in the database are stored with their enabled/disabled status. The courses table is updated when the admin toggles the status, which is reflected through SQL queries.

2. Student Portal Functionality:

- **Advising Menu and History:**

- **Frontend:**

- CourseAdvisingHistory.js: This component fetches and displays the advising history records for the logged-in student. It shows the date, term, and status of previous submissions, and allows users to click on records with 'Pending' status for editing.
 - Navigation is handled using React Router.

- **Backend:**

- `advisingRoutes.js`: Provides an API endpoint for fetching the advising history based on the student's email from the session, ensuring only authenticated users can access their data.
- **Creating and Managing Advising Forms:**
 - **Frontend:**
 - `CourseAdvisingForm.js`: This form allows students to create or edit advising forms with three main sections:
 - **History Section**: Captures the last term, last GPA, and advising term.
 - **Prerequisites Section**: Enables dynamic addition of rows to select prerequisites. The form validates prerequisites and prevents duplicates.
 - **Course Plan Section**: Allows dynamic addition of courses with validation rules to prevent selecting previously taken courses.
 - State management is used for form fields, and form submission is handled via Axios requests.
 - **Backend:**
 - `advisingRoutes.js`: Handles form submission, validation, updating records, and ensuring that courses taken in previous terms cannot be re-added.
 - `isAuthenticated` middleware ensures that only logged-in users can submit or edit forms.
 - The backend checks for duplicates and verifies prerequisites before inserting data into the database.
 - SQL queries are used to manage records in `advising_history`, `course_plan`, and `prerequisites` tables.
- **Rules and Validation for Course Selection:**
 - **Backend Validation:**
 - On form submission, the backend ensures that:
 - Advising forms are unique for each term.

- No duplicate courses are added.
- Courses previously taken are not added again.
- Data integrity is maintained using SQL queries and conditional logic.

3. **Status-Based Behavior for Advising Records:**

- **Frontend:**

- CourseAdvisingForm.js checks the status of the record (e.g., 'Pending', 'Approved', 'Rejected'). If the status is 'Approved' or 'Rejected', the form fields are disabled to prevent further edits.
- Records with 'Pending' status allow editing and saving.

- **Backend:**

- The backend ensures that only 'Pending' records can be modified. Updates to 'Approved' or 'Rejected' records are blocked.

4. **Deployment of Frontend, Backend, and Database:**

- **Frontend Deployment:**

- The React frontend is deployed on Firebase Hosting or an equivalent platform.

- **Backend Deployment:**

- The Express backend is deployed on Render (or any preferred platform).

- **Database Configuration:**

- MySQL is configured with remote access enabled, allowing communication between the deployed backend and database.

- **API Communication:**

- Axios is used for secure communication between the frontend and backend, with CORS and authentication measures in place.

5. Database Design

Advising History Table

Field	Type	Key	Example
id	int(11)	Primary	1
student_email	varchar(255)	Foreign	student@example.com
term	varchar(50)		Fall 2024
date_submitted	date		15-02-2024
status	varchar(50)		Pending
last_term	varchar(50)		Spring 2023
last_gpa	decimal(3,2)		3.75
prerequisites	text		[{"level": "200", "courseName": "Calculus II"}]
course_plan	text		[{"level": "300", "courseName": "Physics"}]

Courses Table

Field	Type	Key	Example
id	int(11)	Primary	1
level	varchar(10)		100
course_name	varchar(255)		Introduction to Programming
enabled	tinyint(1)		1

Course Catalog Table

Field	Type	Key	Example
id	int(11)	Primary	1
level	int(11)		200
course_name	varchar(255)		Database Systems
enabled	tinyint(1)		1

Course Plan Table

Field	Type	Key	Example
id	int(11)	Primary	1
advising_history_id	int(11)	Foreign	5
course_level	varchar(10)		300
course_name	varchar(255)		Linear Algebra

Prerequisites Table

Field	Type	Key	Example
id	int(11)	Primary	1
advising_history_id	int(11)	Foreign	3
course_level	varchar(10)		200
course_name	varchar(255)		Calculus I

6. Implementation

1. Design and create the prerequisite form for admin, displaying courses from levels 100 to 499 with fields: Level, Course, Enable/Disable

- **File(s):** PrerequisiteForm.js, adminPrerequisite.js, adminRoutes.js
- **Frontend:**
 - **Component:** PrerequisiteForm.js
 - Fetches courses from the backend and displays a table with fields for Level, Course, and a checkbox to enable/disable the course.
 - Uses Axios to make GET requests to fetch course data and POST requests to update the status.
 - **Backend:**
 - **Route:** /admin/prerequisites (GET) in adminPrerequisite.js
 - Fetches courses between levels 100-499 from the courses table.
 - **Route:** /admin/prerequisites/update (POST) in adminPrerequisite.js
 - Updates the enabled status of a course in the courses table.
 - **Middleware:** verifyAdmin ensures only admin users can access these routes.

2. Update database based on the admin's selection of prerequisites

- **File(s):** PrerequisiteForm.js, adminPrerequisite.js
- **Frontend:**

- When an admin changes the enable/disable status of a course, the change is tracked and sent via a POST request upon clicking the "Save" button.
- **Backend:**
 - Updates the courses table in the database with the new enabled status based on the admin's selection.

3. Implement an Advising menu accessible upon student login

- **File(s):** CourseAdvisingHistory.js
- **Frontend:**
 - Displays a menu accessible to students once logged in, allowing them to view their advising history.
- **Backend:**
 - Ensures routes related to advising are protected by isAuthenticated middleware, restricting access to logged-in students only.

4. Design and create the "Course Advising History" form to display previously submitted records or indicate no records

- **File(s):** CourseAdvisingHistory.js, advisingRoutes.js
- **Frontend:**
 - CourseAdvisingHistory.js fetches and displays advising records for a student in a table format with columns: Date, Term, and Status.
 - Clicking on a "Pending" record redirects the user to CourseAdvisingForm.js for editing.
- **Backend:**
 - **Route:** /student/advising-history (GET) in advisingRoutes.js
 - Retrieves advising history records for the logged-in student using their session email.

5. Develop a form for creating a new "Course Advising" form with three sections: History, Prerequisites, and Course Plan

- **File(s):** CourseAdvisingForm.js
- **Frontend:**
 - Form divided into three sections:

- **History:** Fields for Last Term, Last GPA, and Advising Term.
- **Prerequisites:** Allows dynamic row addition with Level and Course Name dropdowns.
- **Course Plan:** Allows dynamic row addition with Level and Course Name dropdowns.

6. Implement the History section with fields: Last Term, Last GPA, Advising Term

- **File(s):** CourseAdvisingForm.js
- **Frontend:**
 - Fields for Last Term, Last GPA, and Advising Term, which are validated before form submission.
 - Changes to these fields are not allowed when editing a record (if status is 'Approved' or 'Rejected').

7. Enable dynamic addition of rows for prerequisites section with fields: Level, Course Name (both fields will be dropdown)

- **File(s):** CourseAdvisingForm.js
- **Frontend:**
 - Dynamic rows with dropdowns for Level and Course Name.
 - The dropdowns fetch data from the backend and filter courses based on the selected level.

8. Enable dynamic addition of rows for the course section with fields: Level, Course Name (both fields will be dropdown)

- **File(s):** CourseAdvisingForm.js
- **Frontend:**
 - Similar functionality as the prerequisites section, allowing dynamic addition of course rows.

9. Implement rules for course selection, preventing the addition of courses previously taken in the previous terms

- **File(s):** CourseAdvisingForm.js, advisingRoutes.js
- **Frontend:**

- Prevents selection of courses already taken in previous terms by validating against takenCourses.
- **Backend:**
 - Validates that courses submitted in the form have not been taken in previous terms using a query that checks previous advising records.

10. Implement functionality to save new entries so that newly created records are displayed in the "Course Advising History" form with a Pending status

- **File(s):** CourseAdvisingForm.js, advisingRoutes.js
- **Frontend:**
 - On form submission, new entries are added with a 'Pending' status.
- **Backend:**
 - **Route:** /advising/submit-advising (POST) in advisingRoutes.js
 - Handles form submission, inserts a new record in advising_history, and sets the status to 'Pending'.

11. When a user clicks on any record displayed in the "Course Advising History" form, they should be redirected to the Course Advising form with the selected record pre-populated

- **File(s):** CourseAdvisingHistory.js, CourseAdvisingForm.js, advisingRoutes.js
- **Frontend:**
 - Clicking on a record in CourseAdvisingHistory.js redirects to CourseAdvisingForm.js with pre-populated data.
 - Editing is allowed only if the status is 'Pending'.
- **Backend:**
 - **Route:** /advising/advising-history/:advisingTerm (GET) in advisingRoutes.js
 - Fetches data for a specific advising term.

12. Deploy your Frontend, Backend, and Database on server and your demo should be demonstrated from the live server (For Milestone 2 and Milestone 3)

- **Deployment Steps:**
 - **Frontend:** Deploy on platforms like Firebase Hosting or Netlify.

- **Backend:** Deploy on Render or Heroku with proper database connections.
- **Database:** Configure MySQL to allow remote connections for the backend server.