Final Report

Elias Emanuel Arriola, Ahmed Hassan, Peter Wahyudianto Madin, Sopheanith Ny - Group #8

CourseMap

Table of Contents

1. Introduction

1.1 Project Overview

Our application aims to provide Computer Science students with crucial insights into assignment time requirements for specific professors and courses. By addressing the uncertainty surrounding workload expectations, we seek to empower students to make informed academic decisions.

1.2 Problem Statement & Objectives

Our core problem is the unknown time commitment required to complete assignments within a course. This lack of transparency directly impacts students' ability to plan effectively and make strategic course selections. Our objective is to build a web application that enables:

- Professors to document and manage assignment details
- Students to contribute and track actual time spent on their assignments
- Facilitating better academic planning and time management
- Provide data-driven insights for course selection

1.3 Motivation & Course Impact

Inspired by our personal academic experiences, this project addresses the challenges students face when navigating complex course workloads. By creating a platform that transparently documents assignment time requirements, we aim to:

- Reduce student stress
- Improve academic productivity
- Provide insights that could potentially influence course design
- Help students make more informed course registration decisions

Our motivations also extend to optimizing learning outcomes based on time allocation. We recognize that students have diverse goals and needs:

- **For some**, courses with more intensive workloads may be appealing, as they associate greater time commitment with deeper learning and skill development.
- **For others**, courses with lighter workloads might be more desirable, allowing them to balance academics with personal goals, extracurricular activities, or other priorities.

Overall, we encourage students to tailor their academic experiences to their unique circumstances, skills, and aspirations, fostering a more effective and personalized approach to learning in increasingly demanding academic environments.

1.4 Learning & Course Concept Integration

Our project demonstrates the comprehensive application of database systems and web development principles learned throughout the course. We integrated multiple technical concepts, including:

Database Design Techniques:

- Implemented a database schema validated to meet Boyce-Codd Normal Form (BCNF) normalization standards
- Created comprehensive database diagrams, including Entity-Relationship (ER) diagrams to visualize data relationships leading to a database schema diagram

Advanced Technical Implementation:

- Developed complex SQL queries that extend beyond basic database operations
    - Created Triggers to automate the updates of the StuAvgTime attribute in the ASSIGNMENTS table when new time records are inserted or updated in the TIME_RECORD table
    - Used cascading updates and deletes to ensure consistency and referential integrity by applying ON DELETE CASCADE and ON UPDATE CASCADE in foreign keys on tables such as COURSE, ASSIGNMENTS, and TIME_RECORD
    - Nested selections implementing complex subqueries, such as identifying students whose time spent on assignments exceeded the average time for that assignment.
    - Set operations to combine results like INTERSECT to filter assignments meeting multiple criteria, like average time and student count thresholds.
- Constructed a web application using a structure similar to course assignments, including:
    - HTML for structure
    - JavaScript for dynamic interactions
    - Node.js for backend functionality

We used Google Cloud Platform (GCP) to host the web application and database. This mirrored real-world deployment scenarios and practices introduced during the course, such as setting up secure database connections, load balancing, and scalability.

Development Workflow:

- Employed GitHub for version control and collaborative development
- Leveraged phpMyAdmin for database management and interaction, reinforcing database administration skills learned in the course

2. Related Work

2.1 Existing Solutions

A couple of platforms attempt to provide students with course and workload insights:

1. **Rate My Professor**
    - Focuses on professor ratings
    - Provides subjective student reviews
    - Limited quantitative data about course workload
2. **DawgPath (University-Specific Platform, for the University of Washington)**

- Provides course grade distributions
- Shows prerequisite mapping
- Offers strategic course planning insights
- Displays median course grades for declared majors
- Concurrent course history

2.2 Comparative Analysis

While existing platforms offer valuable information, they lack:

- Specific, granular time tracking for individual assignments
- Real-time student-contributed data on workload
- A comprehensive view of time investment across different courses and professors

2.3 Feature Comparison Table

Feature	**Rate My Professor**	**DawgPath**	**CouseMap**
Professor Reviews	✅	❌	❌
Course Grade Distribution	❌	✅	❌
Prerequisite Mapping	❌	✅	❌
Concurrent Courses	✅	✅	❌
Assignment Time Tracking	❌	❌	✅
Real-time Student Input	❌	❌	✅
Cross-Course Workload Comparison	❌	❌	✅

2.4 Unique Features

Our application distinguishes itself by:

- Enabling precise assignment time tracking
- Allowing real-time student contributions
- Providing a holistic view of course workload
- Supporting proactive academic planning

3. Design & Implementation

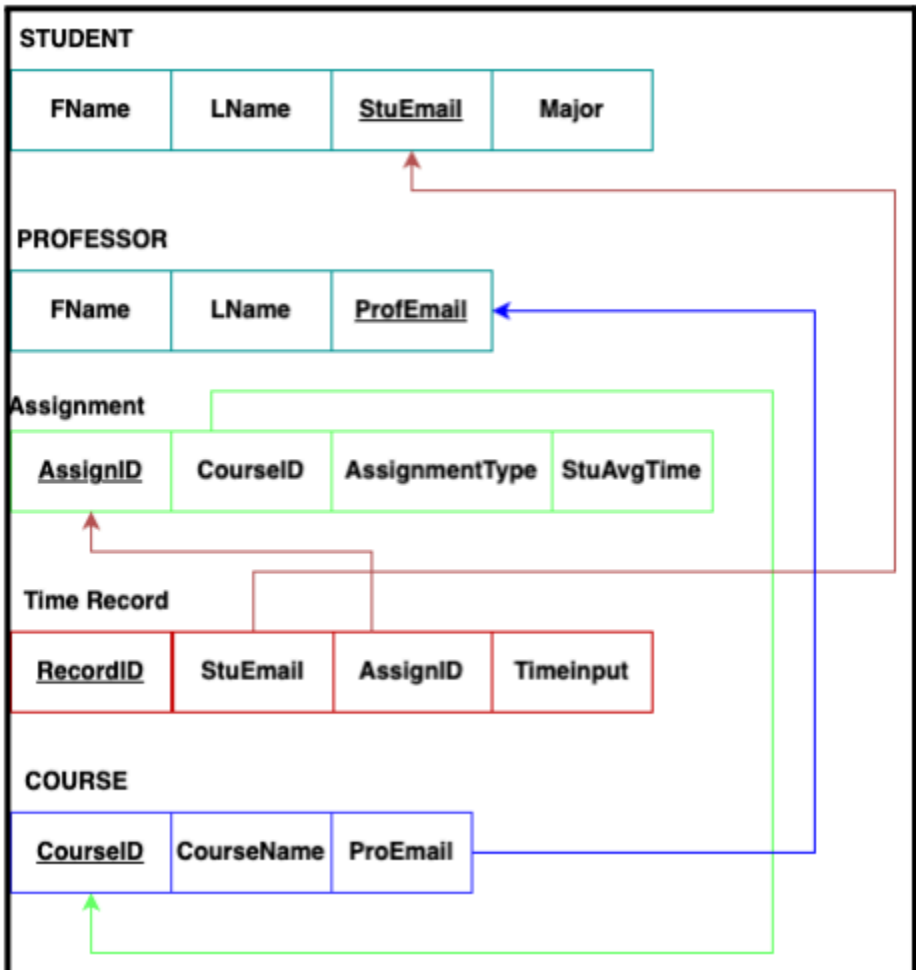
3.1 System Architecture Overview

Our application follows a three-tier architecture:

1. **Presentation Layer (Frontend)**
    - **Technologies:** HTML and JavaScript
    - **Functionality:** Provides the user interface for interacting with the application, allowing users to input, view, drop down, and analyze data on assignment times and workloads
2. **Application Layer (Backend)**
    - **Technologies:** Node.js server
    - **Functionality:** Handles routing, business logic, and communication between the front end and data layer. Ensures data consistency, performs calculations (e.g., average assignment times), and manages API requests.
3. **Data Layer**
    - **Database:** MySQL
    - **Hosting:** Google Cloud Platform (GCP) on a Virtual Machine (VM)
    - **Management:** Data is managed through phpMyAdmin, a web-based tool for administering MySQL databases.
    - **Functionality:** Stores all application data, including student and professor records, course and assignment details, and real-time assignment time inputs. SQL queries, triggers, and constraints ensure data integrity and facilitate analytics.

3.2 Database Design

Database Design Decisions



1. **Key Design Choices:**
    - We made the CourseMap database pretty simple each table has required attributes and no redundancy. Initially, we were going to use Stu_id and Prof_id as primary keys, however, we found that using the emails as primary keys was better since each email is unique and reduces redundancy. This was very helpful when going into the BCNF normalization
2. **Normalization:**
    - All tables comply with the Boyce-Codd Normal Form (BCNF) due to minimal attribute redundancy and elimination of transitive dependencies.
    - No significant redesign was needed, as simplicity was prioritized from the initial stages.

Database Schema

The database schema includes the following tables:

STUDENT Table:

- Stores student details, including names, unique email addresses, and declared majors.
- Includes a CHECK constraint to ensure valid email formatting (%@uw.edu).

PROFESSOR Table:

- Captures professor details with a unique email as the primary identifier.
- Similar email validation ensures a consistent institutional domain.

COURSE Table:

- Links professors to courses using foreign keys.
- Includes cascading actions to maintain referential integrity during updates or deletions.

ASSIGNMENTS Table:

- Tracks assignment details for each course, with an average time (StuAvgTime) calculated dynamically.

TIME_RECORD Table:

- Record student-reported times for each assignment.
- Implements constraints to enforce positive values and references the STUDENT and ASSIGNMENTS tables.

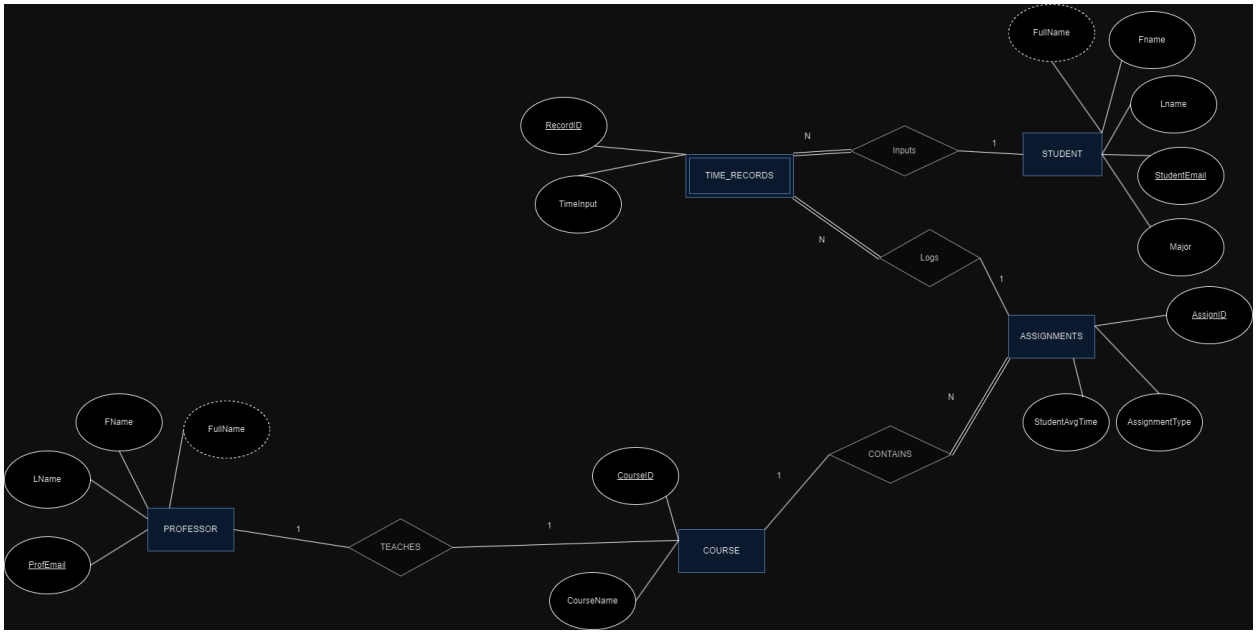
Triggers:

- Two triggers (update_stu_avg_time_insert and update_stu_avg_time_update) ensure the StuAvgTime in the ASSIGNMENTS table is updated whenever new records are inserted or updated in the TIME_RECORD table.

Foreign Keys:

Foreign keys are represented with arrows pointing from child tables to their parent tables:

- COURSE.ProfEmail → PROFESSOR.ProfEmail
- ASSIGNMENTS.CourseID → COURSE.CourseID
- TIME_RECORD.StudentEmail → STUDENT.StuEmail
- TIME_RECORD.AssignID → ASSIGNMENTS.AssignID



The ER diagram reveals the following key relationships:

1. **One-to-Many Student-TimeRecords Relationship:**
    - Each student can have multiple time record entries
    - TimeRecords are linked to individual students through the "Inputs" relationship
    - This enables tracking of individual student engagement and time management
2. **Professor-Course Teaching Relationship:**
    - A one-to-one relationship between professors and courses
    - Each course has exactly one professor assigned
    - Professors are linked through the "TEACHES" relationship
3. **Course-Assignment Structure:**
    - Courses contain multiple assignments (one-to-many relationships)
    - Assignments track various metrics including StudentAvgTime
    - The "CONTAINS" relationship maintains course structure integrity

3.3 Technical Implementation Details

Our code base uses a Mode-View-Controller (MVC) architectural pattern. We stored all of our queries JavaSciprt files in our Controller folder. In our public folder, we have all the queries HTML files. We also have dbconfig.js, server.js, .env, and package. json within our main folder CourseMap.

3.4 Frontend & Backend Integration

- Server-side routing through Express.js
- Direct database query execution via Node.js
- Modular controller approach for managing different application functionalities
- Separation of concerns between data access, logic, and presentation layers

- Index/Home page that links the tables to the respective backend SQL queries

4. Results & Testing

4.1 SQL Query Demonstrations

```
-- Query 1
SELECT CONCAT(P.FName, ' ', P.LName) AS ProfessorName,
    C.CourseName,
    A.AssignmentType,
    A.StuAvgTime
FROM professor P
JOIN course C ON P.ProfEmail = C.ProfEmail
JOIN assignments A ON C.CourseID = A.CourseID;
```

Purpose: Lists each professor's full name, the courses they teach, and the types of assignments in each course along with the average student time spent.

Expected: A table showing the professor's name, the course name, each assignment type in that course, and the average student time spent.

ProfessorName	CourseName	AssignmentType	StuAvgTime
Jane Miller	CSS001	Project	2.18
Jane Miller	CSS001	Assignment	8.91
Jane Miller	CSS001	Quiz	7.2
Jane Miller	CSS001	Project	7.31
Mary Davis	CSS002	Quiz	10.32
Mary Davis	CSS002	Discussion Post	5.53
Mary Davis	CSS002	Project	8.41
Mary Davis	CSS002	Project	7.96
Eyhab Johnson	CSS003	Discussion Post	7.33
Eyhab Johnson	CSS003	Assignment	6.28
Eyhab Johnson	CSS003	Discussion Post	2.4
Eyhab Johnson	CSS003	Project	6.65
Tom Arriola	CSS004	Assignment	6.25
Tom Arriola	CSS004	Quiz	9.69
Tom Arriola	CSS004	Project	6.94
Tom Arriola	CSS004	Quiz	7.31
Noah Emanuel	CSS005	Quiz	9.56
Noah Emanuel	CSS005	Assignment	8.47
Noah Emanuel	CSS005	Discussion Post	8.17
Noah Emanuel	CSS005	Quiz	6.88
Tom Smith	CSS006	Project	3.86
Tom Smith	CSS006	Quiz	8.11
Tom Smith	CSS006	Project	5.48
Tom Smith	CSS006	Project	3.82
Tom Brown	CSS007	Assignment	4.92
Tom Brown	CSS007	Assignment	3.28
Tom Brown	CSS007	Discussion Post	7.47
Tom Brown	CSS007	Project	5.54
Liam Johnson	CSS008	Discussion Post	5.35
Liam Johnson	CSS008	Project	7.24
Liam Johnson	CSS008	Discussion Post	4.58
Liam Johnson	CSS008	Discussion Post	5.83
Monika Johnson	CSS009	Quiz	6.72
Monika Johnson	CSS009	Discussion Post	5.98
Monika Johnson	CSS009	Assignment	7.86
Monika Johnson	CSS009	Project	4.28
Mia Williams	CSS010	Assignment	5.56
Mia Williams	CSS010	Assignment	9.89
Mia Williams	CSS010	Discussion Post	3.78
Mia Williams	CSS010	Discussion Post	4.79

```
-- Query 2
SELECT DISTINCT
  c.CourseName AS "Course Name",
  CONCAT(p.Fname, ' ', p.Lname) AS "Professor Name",
  ROUND(AVG(a.StuAvgTime),2) AS "AvgCourseTime"  -- Calculate overall average time per
course
FROM PROFESSOR p
JOIN COURSE c ON p.ProfEmail = c.ProfEmail
JOIN ASSIGNMENTS a ON a.CourseID = c.CourseID  -- Join with assignments to calculate average
WHERE c.CourseID = ANY (
  SELECT CourseID
  FROM ASSIGNMENTS
  GROUP BY CourseID

  HAVING AVG(StuAvgTime) < 10  -- Filter courses with an average course time < 10 hours.
)
GROUP BY c.CourseName, p.Fname, p.Lname  -- Group by course and professor
ORDER BY c.CourseName ASC;
```

Purpose: Find professors and their courses where the course average time is greater than a specified threshold (default 2 hours)

Expected: A list of professor names, course names, and the course average time where the average course time is less than the threshold In the future users will be able to pass their thresholds.

Course Name	Professor Name	AvgCourseTime
CSS001	Jane Miller	6.4
CSS002	Mary Davis	8.06
CSS003	Eyhab Johnson	5.66
CSS004	Tom Arriola	7.55
CSS005	Noah Emanuel	8.27
CSS006	Tom Smith	5.32
CSS007	Tom Brown	5.3
CSS008	Liam Johnson	5.75
CSS009	Monika Johnson	6.21
CSS010	Mia Williams	6

```
-- Query 3
SELECT ASSIGNMENTS.AssignID, COURSE.CourseName, ASSIGNMENTS.AssignmentType,
ASSIGNMENTS.StuAvgTime
FROM ASSIGNMENTS
JOIN COURSE ON ASSIGNMENTS.CourseID = COURSE.CourseID
WHERE ASSIGNMENTS.StuAvgTime > 3.0
INTERSECT
SELECT ASSIGNMENTS.AssignID, COURSE.CourseName, ASSIGNMENTS.AssignmentType,
ASSIGNMENTS.StuAvgTime
FROM ASSIGNMENTS
JOIN COURSE ON ASSIGNMENTS.CourseID = COURSE.CourseID
JOIN TIME_RECORD ON ASSIGNMENTS.AssignID = TIME_RECORD.AssignID
GROUP BY ASSIGNMENTS.AssignID, COURSE.CourseName, ASSIGNMENTS.AssignmentType,
ASSIGNMENTS.StuAvgTime
HAVING COUNT(DISTINCT TIME_RECORD.StudentEmail) >= 3;
```

Purpose: Determine the assignments that have an average student time that is greater than 3.0 and have been worked by at least 3 distinct students based on the student's time record.

Expected: It returns the assignment with ID, course name, assignment type, and the average completion time, where the assignment takes over 3 hours to complete an average and has at least 3 students who recorded time for them.

AssignID	CourseName	AssignmentType	StuAvgTime
2	CSS001	Assignment	8.91
4	CSS001	Project	7.31
6	CSS002	Discussion Post	5.53
9	CSS003	Discussion Post	7.33
13	CSS004	Assignment	6.25
14	CSS004	Quiz	9.69
16	CSS004	Quiz	7.31
26	CSS007	Assignment	3.28
27	CSS007	Discussion Post	7.47
29	CSS008	Discussion Post	5.35
30	CSS008	Project	7.24
31	CSS008	Discussion Post	4.58
32	CSS008	Discussion Post	5.83
33	CSS009	Quiz	6.72
38	CSS010	Assignment	9.89
40	CSS010	Discussion Post	4.79

-- Query 4

```
SELECT PROFESSOR.FName, PROFESSOR.LName, PROFESSOR.ProfEmail,
        COUNT(DISTINCT ASSIGNMENTS.AssignID) AS Total_Assingments,
    ROUND(AVG(ASSIGNMENTS.StuAvgTime),2) AS AverageAssingmentTime
FROM PROFESSOR
JOIN COURSE ON PROFESSOR.ProfEmail = COURSE.ProfEmail
JOIN ASSIGNMENTS ON COURSE.CourseID = ASSIGNMENTS.CourseID
GROUP BY PROFESSOR.ProfEmail, PROFESSOR.FName, PROFESSOR.LName
ORDER BY AverageAssingmentTime DESC;
```

Purpose: It determines how many assignments each professor has and the average time students take to complete the assignments.

Expected: It returns a list of professors, including their name, and email, along with the total number of assignments they have and the average time students spend on that assignment.

FName	LName	ProfEmail	Total_Assingments	AverageAssingmentTime
Noah	Emanuel	nemanuel5@uw.edu	4	8.27
Mary	Davis	mdavis2@uw.edu	4	8.06
Tom	Arriola	tarriola4@uw.edu	4	7.55
Jane	Miller	jmiller1@uw.edu	4	6.4
Monika	Johnson	mjohnson9@uw.edu	4	6.21
Mia	Williams	mwilliams10@uw.edu	4	6.01
Liam	Johnson	ljohnson8@uw.edu	4	5.75
Eyhab	Johnson	ejohnson3@uw.edu	4	5.66
Tom	Smith	tsmith6@uw.edu	4	5.32
Tom	Brown	tbrown7@uw.edu	4	5.3

-- Query 5

```
SELECT P.FName AS ProfessorName, P.LName, C.CourseName, A.AssignID, A.StuAvgTime
FROM PROFESSOR P, COURSE C, ASSIGNMENTS A
```

```
WHERE P.ProfEmail = C.ProfEmail
AND C.CourseID = A.CourseID
AND A.StuAvgTime > 5
ORDER BY A.StuAvgTime DESC;
```

Purpose: Find all professors who teach courses with assignments where the average student time exceeds 5 hours

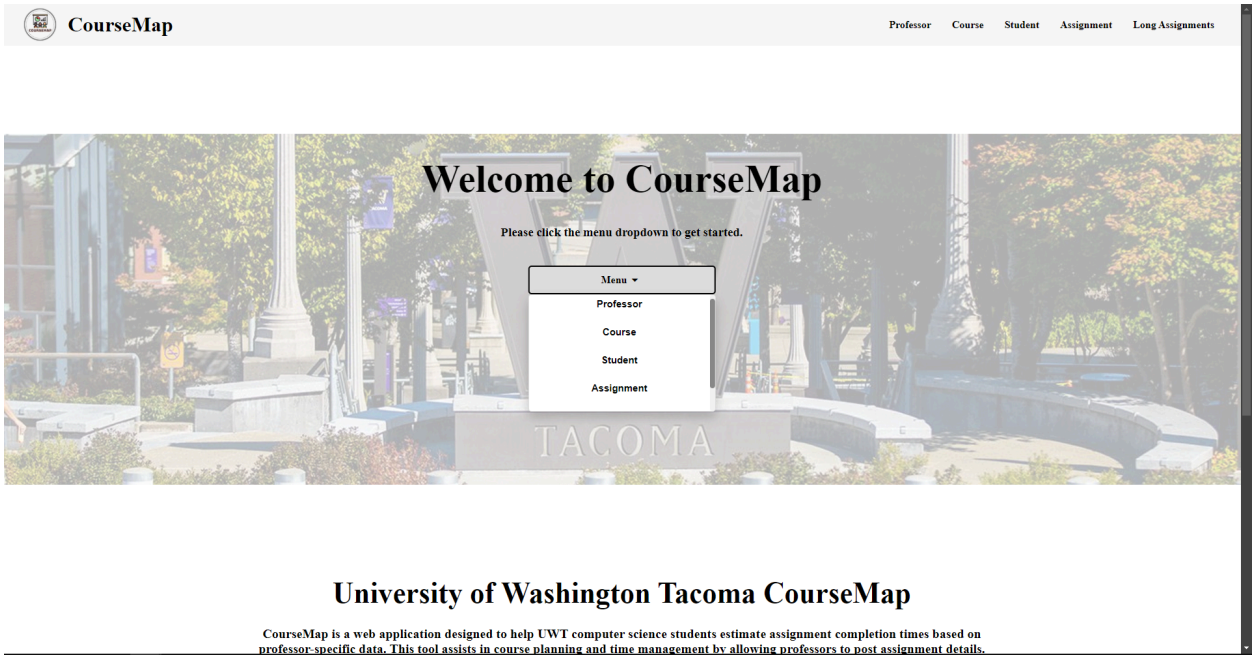
Expected: Shows professor names and course details where assignments take longer than 5 hours on average, sorted by average time

ProfessorName	LName	CourseName	AssignID	StuAvgTime
Mary	Davis	CSS002	5	10.32
Mia	Williams	CSS010	38	9.89
Tom	Arriola	CSS004	14	9.69
Noah	Emanuel	CSS005	17	9.56
Jane	Miller	CSS001	2	8.91
Noah	Emanuel	CSS005	18	8.47
Mary	Davis	CSS002	7	8.41
Noah	Emanuel	CSS005	19	8.17
Tom	Smith	CSS006	22	8.11
Mary	Davis	CSS002	8	7.96
Monika	Johnson	CSS009	35	7.86
Tom	Brown	CSS007	27	7.47
Eyhab	Johnson	CSS003	9	7.33
Jane	Miller	CSS001	4	7.31
Tom	Arriola	CSS004	16	7.31
Liam	Johnson	CSS008	30	7.24
Jane	Miller	CSS001	3	7.2
Tom	Arriola	CSS004	15	6.94
Noah	Emanuel	CSS005	20	6.88
Monika	Johnson	CSS009	33	6.72
Eyhab	Johnson	CSS003	12	6.65
Eyhab	Johnson	CSS003	10	6.28
Tom	Arriola	CSS004	13	6.25
Monika	Johnson	CSS009	34	5.98
Liam	Johnson	CSS008	32	5.83
Mia	Williams	CSS010	37	5.56
Tom	Brown	CSS007	28	5.54
Mary	Davis	CSS002	6	5.53
Tom	Smith	CSS006	23	5.48
Liam	Johnson	CSS008	29	5.35

4.2 Frontend Interface Showcase & User Flow

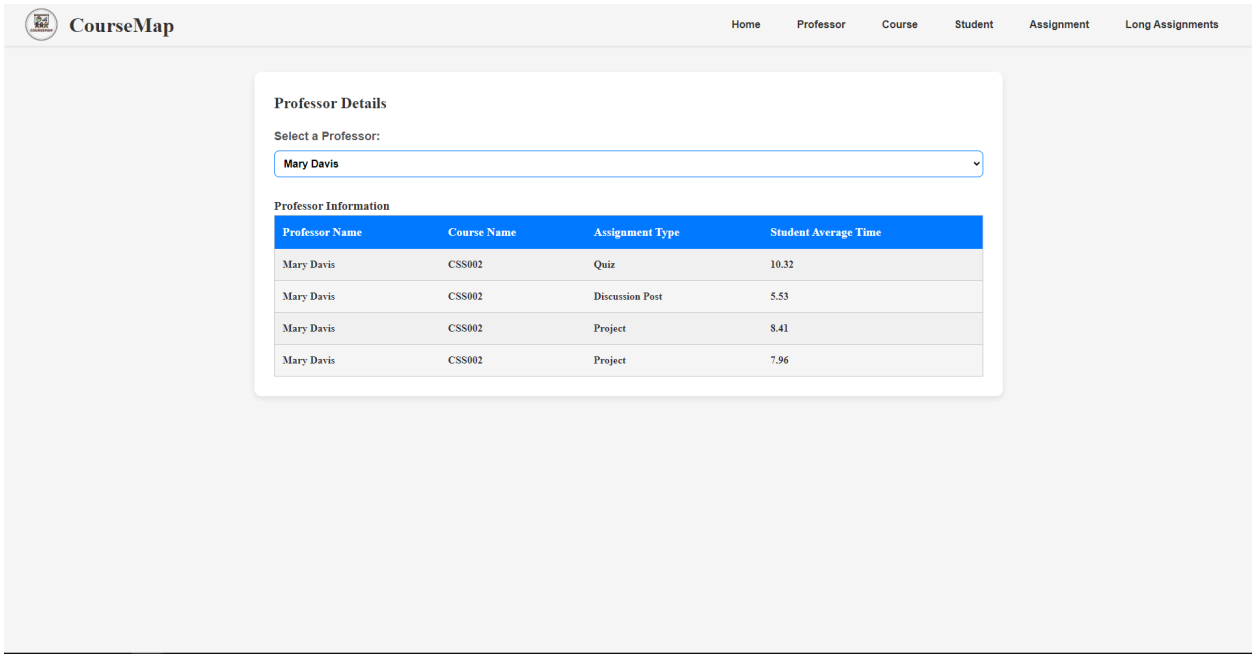
The current version of the web application focuses on providing a simple, informative dashboard for both students and professors to view historical data on average time spent on course assignments. There is no login functionality or ability for users to input or update the data directly. Instead, the application relies on a curated dataset that has been surveyed and aggregated from previous student experiences.

Home Page



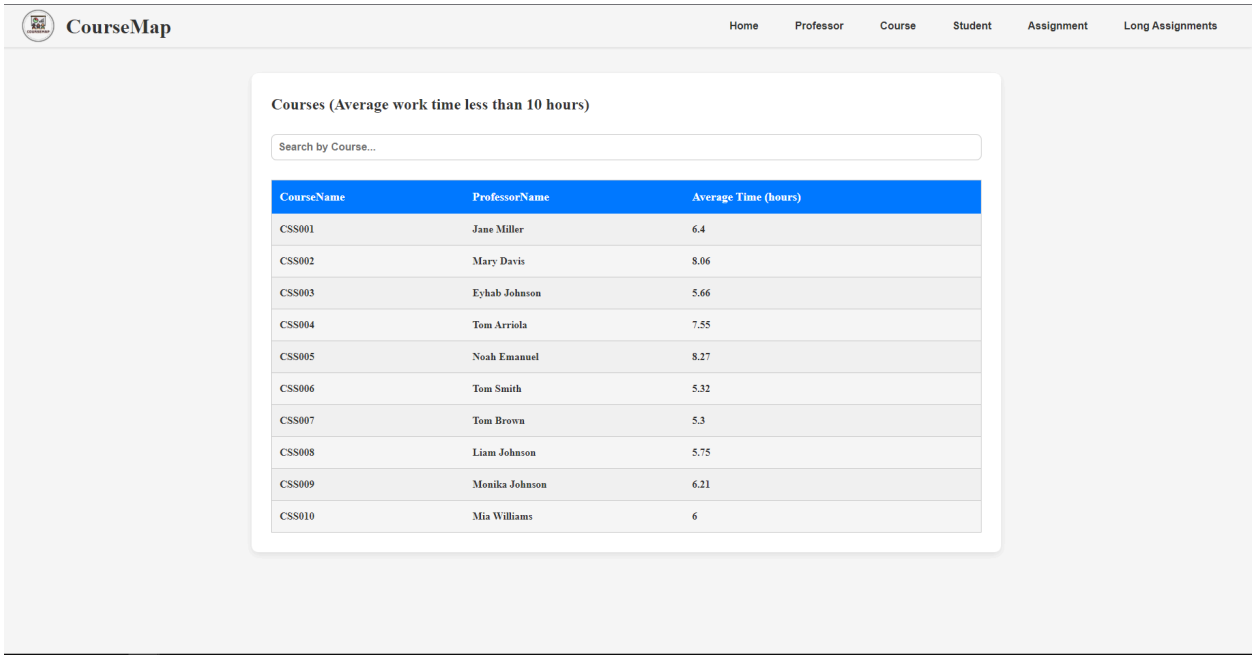
On this home page users can use the drop-down menu to navigate to five different assignment statistics. Alternatively, users can use the nav bar on the top right to switch to different page statistics.

Professor Page



The Professor Details page provides data on a selected professor's assignment. Users, including students, professors, and academic advisors, can view the professor's name, the courses they teach, the assignment types, and the average time students have spent on each assignment.

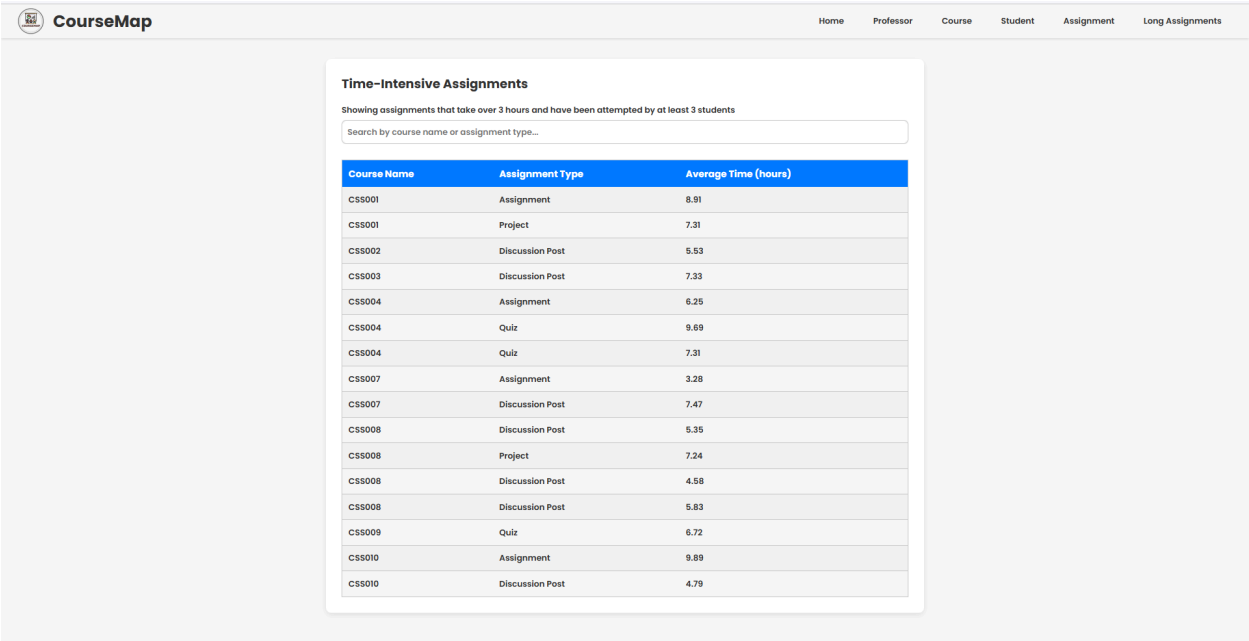
Course Page



The Courses (Average work time less than 10 hours) page allows users to search for courses where the average student time is under 10 hours. The page displays the course name, professor name, and average time students spend on the coursework, enabling users to quickly identify less time-intensive courses.

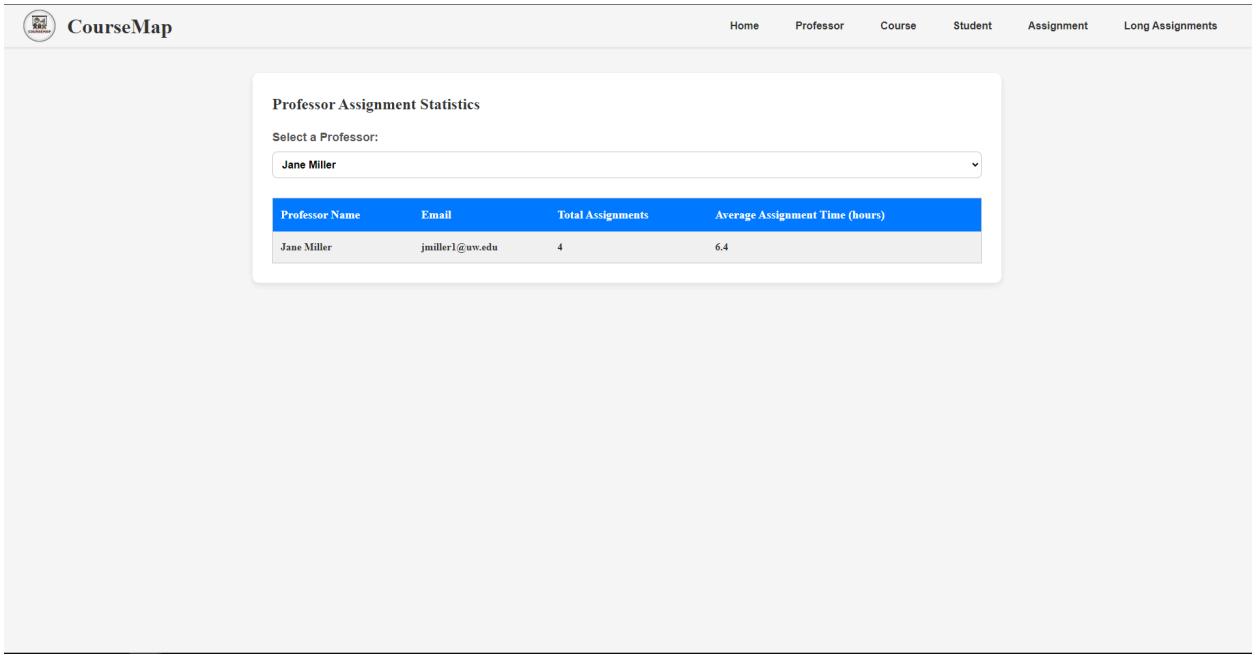
Users can filter their results by course name making it easier to find specific courses of interest.

Student Page



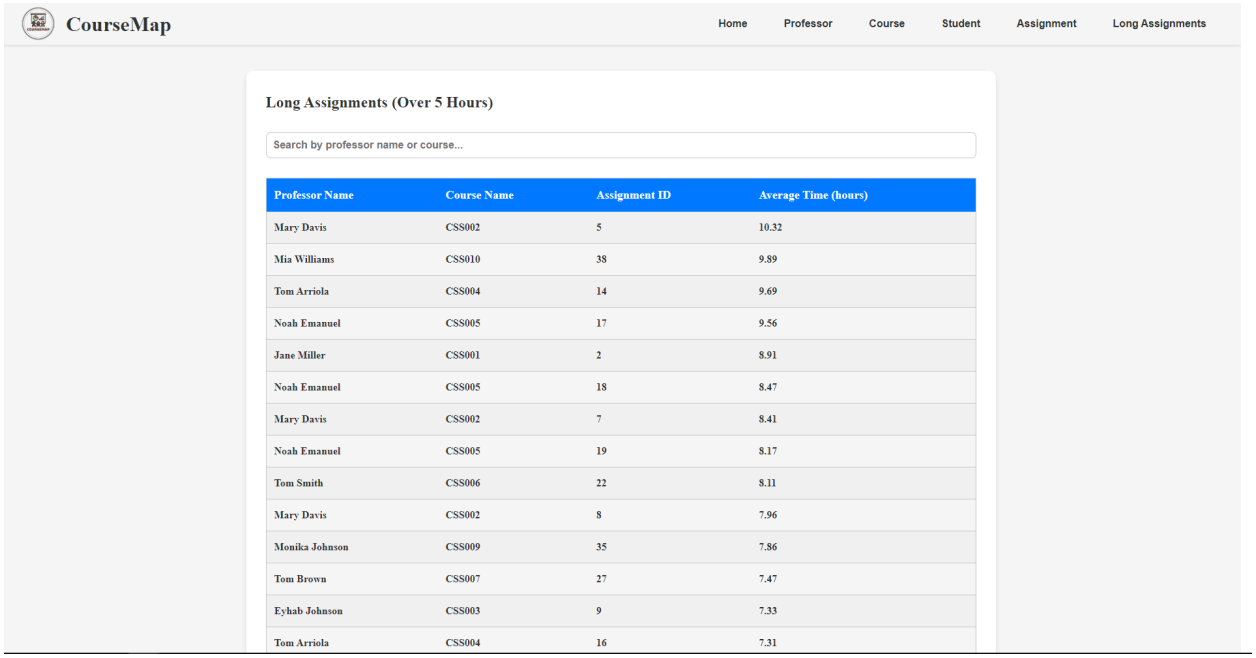
The Time-Intensive Assignments page displays assignments that take over 3 hours on average and have been attempted by at least 3 students. Users can search by course name or assignment type. The table shows the Course Name, Assignment Type, and Average Time (in hours) for these demanding assignments.

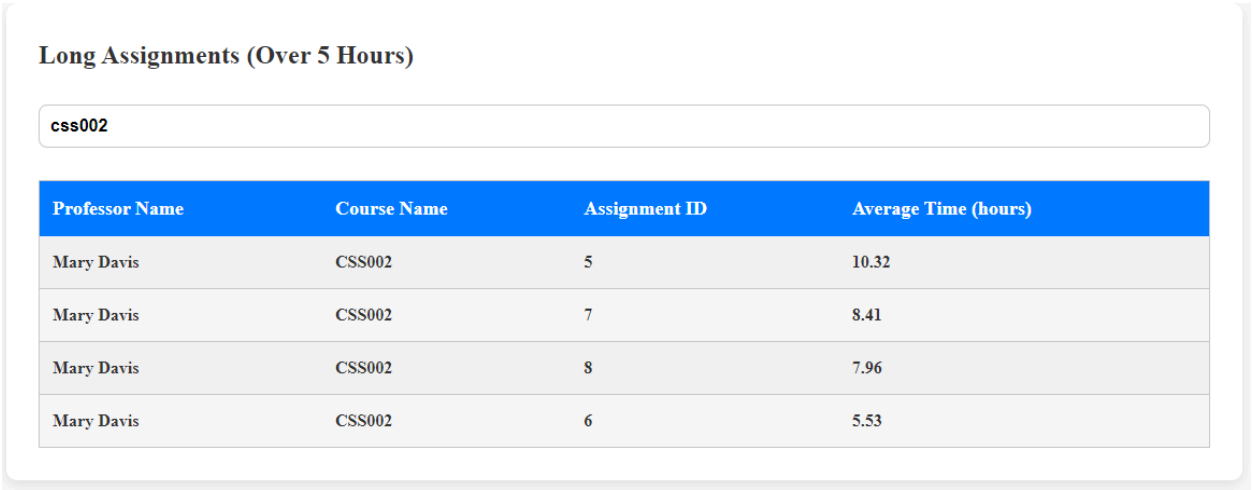
Assignment Page



The Professor Assignment Statistics page displays the selected professor's details, including their name, email, total assignments, and the average time students spent on those assignments. Users select which professor details they want to view.

Long Assignment Page





The Long Assignments (Over 5 Hours) page displays a list of assignments that take students over 5 hours on average to complete. Users can search by professor name or course.

5. Conclusions

5.1 Summary of Achievements

CourseMap successfully demonstrated the practical application of database design and web development principles. The project resulted in a comprehensive relational database with five interconnected tables, containing over a hundred data entries. We developed a custom web interface and implemented ten sophisticated SQL queries, of which five were prominently featured. The database achieved Boyce-Codd Normal Form (BCNF) normalization, ensuring data integrity and minimal redundancy.

5.2 Project Contributions

As a team, we collaborated to create CourseMap, an innovative solution addressing the need for transparent assignment time tracking for Computer Science students. Our project combined technical skills in database design, web development, and data analysis to build a platform that provides valuable insights into course workloads. By creating a user-friendly interface and implementing complex database queries, we developed a tool to help students make more informed academic decisions.

As for individual contributions Elias created both query1 and query2 .js and .html files. Sopheanith implemented query3.html, index.html, and both index.css and query1.css. Peter created query4.html and query5.html. Lastly, Ahmed implemented query3.js, query4.js and query5.js.

5.3 Future Work Possibilities

Our current version of CourseMap has a strong foundation for future updates and development. In the future, we would like to implement user authentication with distinct login screens for professors, students, and academic advisors. We aim to expand the database to support more complex relationships, such as

allowing professors to teach multiple courses and different course sections. Additional features we would like to include were more advanced queries, expanded search functionality, and direct user data input capabilities.

5.4 Lessons Learned

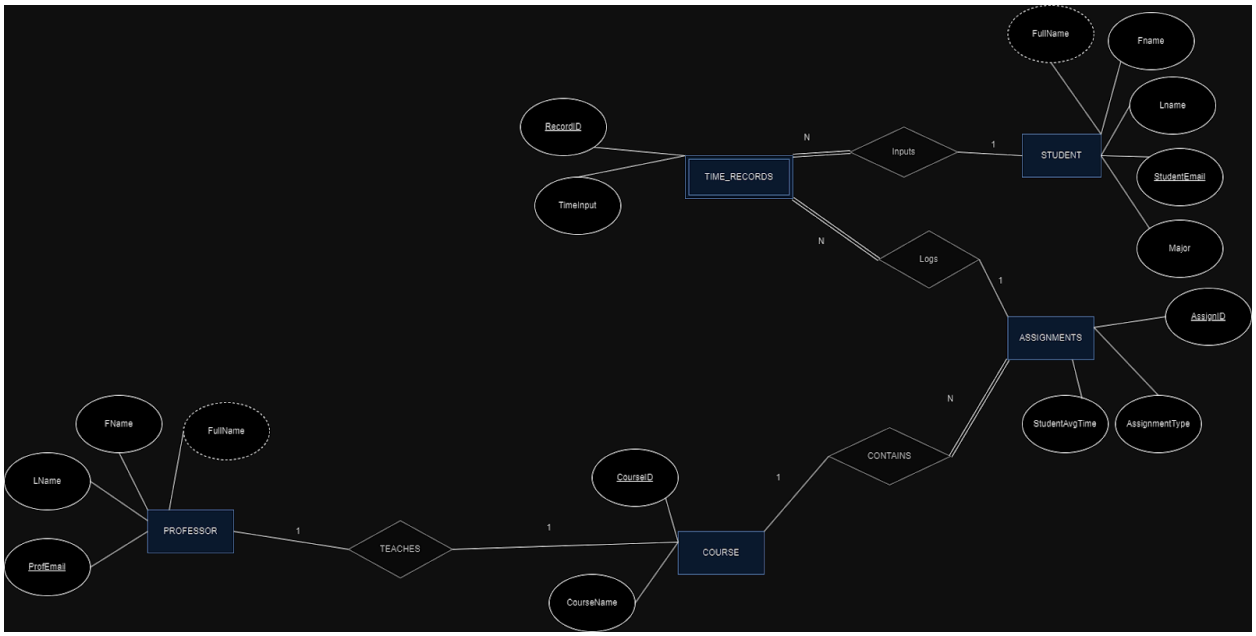
Through this project, our team gained hands-on experience in critical areas of software and database development. We developed a deeper understanding of relational database design, including normalization techniques and complex SQL querying. The project provided practical experience with front-end technologies like Node.js, JavaScript, and HTML, as well as cloud deployment using Google Cloud Platform. We learned the importance of systematic database design, advanced query construction, and the integration of front-end and back-end technologies.

6. References

Bootswatch. (n.d.). *Free themes for Bootstrap*. Retrieved [December 1, 2024], from https://bootswatch.com/.

Appendices

A. Appendix A: ER Diagram



B. Appendix B: Normalization Proof

Normalization Proof:

Student Table

STUDENT (Fname, LName, StuEmail, Major):

Primary Key: StuEmail

Functional Dependencies:

- StuEmail → {Fname, LName, Major}
- There are no other non-trivial FDs

Why Functional Dependency is Justified:

- A student email (StuEmail) uniquely identifies a student.
- StuEmail uniquely determines Fname (first name) and LName (last name).
- StuEmail serves as a natural key because emails are unique to students.

BCNF Proof:

- The only determinant is StuEmail, which is the primary key.
- A table is in BCNF if all determinants are candidate keys. Since StuEmail is the primary key and there are no other determinants, the table is in BCNF.

Professor Table

PROFESSOR (FNAME, LNAME, ProfEmail)

Primary Key: ProfEmail

Functional Dependencies:

- ProfEmail → {Fname, Lname}
- There are no other non-trivial FDs

  BCNF Proof:

Why Functional Dependency is Justified:

- A professor's email (ProfEmail) uniquely determines their first name (Fname) and last name (LName).
- This is logical because emails are unique identifiers for professors.

BCNF Proof:

- The only determinant is ProfEmail, which is the primary key.
- Since all determinants are candidate keys and there are no other functional dependencies, the table is in BCNF.

Course Table

COURSE (CourseID, CourseName, ProfEmail)

Primary Key: CourseID

Functional Dependencies:

- CourseID → {CourseName, ProfEmail}
- There are no other functional dependencies

Why Functional Dependency is Justified:

- CourseID is a unique identifier for courses.
- A specific CourseID uniquely determines the course name (CourseName) and the professor's email (ProfEmail).
- There is no relationship between CourseName and ProfEmail outside of their shared dependency on CourseID.

BCNF Proof:

- The only determinant is CourseID, which is the primary key.
- Since CourseID is a candidate key and there are no other functional dependencies, the table is in BCNF.

Assignments Table

ASSIGNMENTS (AssignID, CourseID, AssignmentType, StuAvgTime)

- Primary Key: AssignID

Functional Dependencies:

- AssignID → {CourseID, AssignmentType, StuAvgTime}
- There are no other non-trivial FDs

Why Functional Dependency is Justified:

- AssignID uniquely identifies each assignment.
- Each assignment has exactly one CourseID, one AssignmentType, and one StuAvgTime (average student time).
- There are no dependencies between CourseID, AssignmentType, and StuAvgTime outside of their shared dependency on AssignID.

BCNF Proof:

- The only determinant is AssignID, which is the primary key.
- Since AssignID is a candidate key and there are no other functional dependencies, the table is in BCNF.

TimeRecord Table

TIME_RECORD Table(RecordID, StuEmail, AssignID, Timeinput)

Primary Key: RecordID

Functional Dependencies:

- RecordID → {StuEmail, AssignID, Timeinput}
- There are no other non-trivial FDs

Why Functional Dependency is Justified:

- Each record in the TIME_RECORD table represents a specific time input by a student for a particular assignment.
- RecordID uniquely identifies the student email (StuEmail), assignment ID (AssignID), and time input (Timeinput).

The RecordID is the natural key that encapsulates all this information.

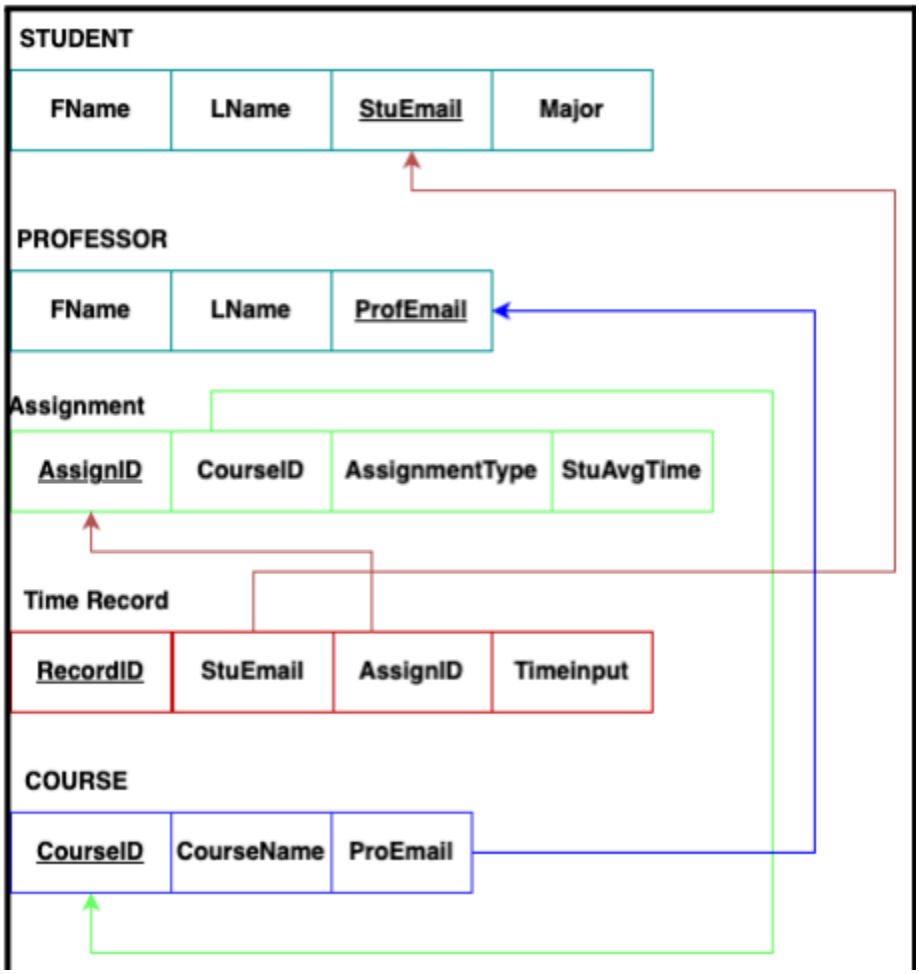
BCNF Proof:

- The only determinant is RecordID, which is the primary key.
- Since RecordID is a candidate key and there are no other functional dependencies, the table is in BCNF.

Conclusion:

The entire **COURSEMAP** database is in BCNF because each relation schema {STUDENTS, PROFESSOR, COURSE, ASSIGNMENTS, TIME_RECORD} is in BCNF

C. Appendix C: Relational Schema Diagram



D. Appendix D: SQL Query Screenshots

```sql
SELECT CONCAT(P.FName, ' ', P.LName) AS ProfessorName,
       C.CourseName,
       A.AssignmentType,
       A.StuAvgTime
FROM professor P
JOIN course C ON P.ProfEmail = C.ProfEmail
JOIN assignments A ON C.CourseID = A.CourseID;
```

Query 1

```sql
SELECT DISTINCT
    c.CourseName AS "Course Name",
    CONCAT(p.Fname, ' ', p.Lname) AS "Professor Name",
    ROUND(AVG(a.StuAvgTime),2) AS "AvgCourseTime"  -- Calculate overall average time per course
FROM PROFESSOR p
JOIN COURSE c ON p.ProfEmail = c.ProfEmail
JOIN ASSIGNMENTS a ON a.CourseID = c.CourseID  -- Join with assignments to calculate average
WHERE c.CourseID = ANY (
    SELECT CourseID
    FROM ASSIGNMENTS
    GROUP BY CourseID

    HAVING AVG(StuAvgTime) < 10  -- Filter courses with an average course time < 10 hours.
)
GROUP BY c.CourseName, p.Fname, p.Lname  -- Group by course and professor
ORDER BY c.CourseName ASC;
```

Query 2

```sql
SELECT ASSIGNMENTS.AssignID, COURSE.CourseName, ASSIGNMENTS.AssignmentType, ASSIGNMENTS.StuAvgTime
FROM ASSIGNMENTS
JOIN COURSE ON ASSIGNMENTS.CourseID = COURSE.CourseID
WHERE ASSIGNMENTS.StuAvgTime > 3.0
INTERSECT
SELECT ASSIGNMENTS.AssignID, COURSE.CourseName, ASSIGNMENTS.AssignmentType, ASSIGNMENTS.StuAvgTime
FROM ASSIGNMENTS
JOIN COURSE ON ASSIGNMENTS.CourseID = COURSE.CourseID
JOIN TIME_RECORD ON ASSIGNMENTS.AssignID = TIME_RECORD.AssignID
GROUP BY ASSIGNMENTS.AssignID, COURSE.CourseName, ASSIGNMENTS.AssignmentType, ASSIGNMENTS.StuAvgTime
HAVING COUNT(DISTINCT TIME_RECORD.StudentEmail) >= 3;
```

Query 5

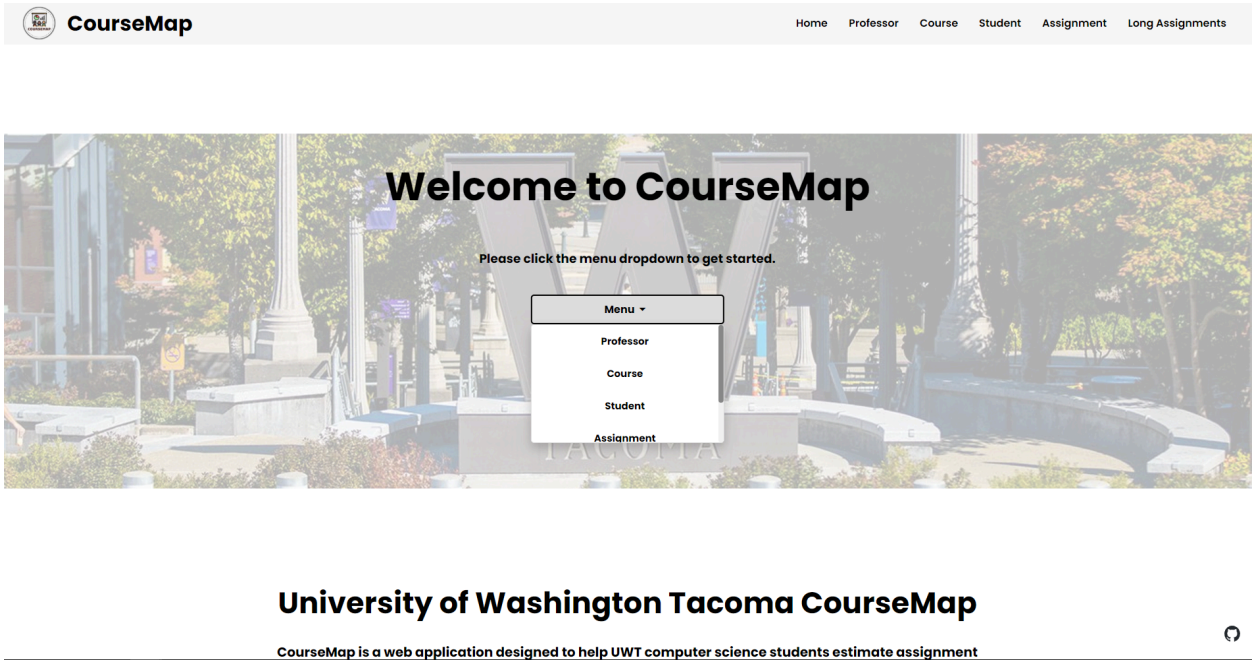
```sql
SELECT PROFESSOR.FName, PROFESSOR.LName, PROFESSOR.ProfEmail,
        COUNT(DISTINCT ASSIGNMENTS.AssignID) AS Total_Assingments,
       ROUND(AVG(ASSIGNMENTS.StuAvgTime),2) AS AverageAssingmentTime
FROM PROFESSOR
JOIN COURSE ON PROFESSOR.ProfEmail = COURSE.ProfEmail
JOIN ASSIGNMENTS ON COURSE.CourseID = ASSIGNMENTS.CourseID
GROUP BY PROFESSOR.ProfEmail, PROFESSOR.FName, PROFESSOR.LName
ORDER BY AverageAssingmentTime DESC;
```

Query 7

```
SELECT P.FName AS ProfessorName, P.LName, C.CourseName, A.AssignID, A.StuAvgTime
FROM PROFESSOR P, COURSE C, ASSIGNMENTS A
WHERE P.ProfEmail = C.ProfEmail
AND C.CourseID = A.CourseID
AND A.StuAvgTime > 5
ORDER BY A.StuAvgTime DESC;
```

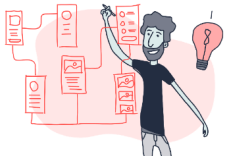
Query 10

E. Appendix E: Interface Screenshots



University of Washington Tacoma CourseMap

CourseMap is a web application designed to help UWT computer science students estimate assignment completion times based on professor-specific data. This tool assists in course planning and time management by allowing professors to post assignment details.



Track Your Progress Your Way

Master Your Schedule

Optimize Your Workflow

CourseMap

Home | Professor | Course | Student | Assignment | Long Assignments

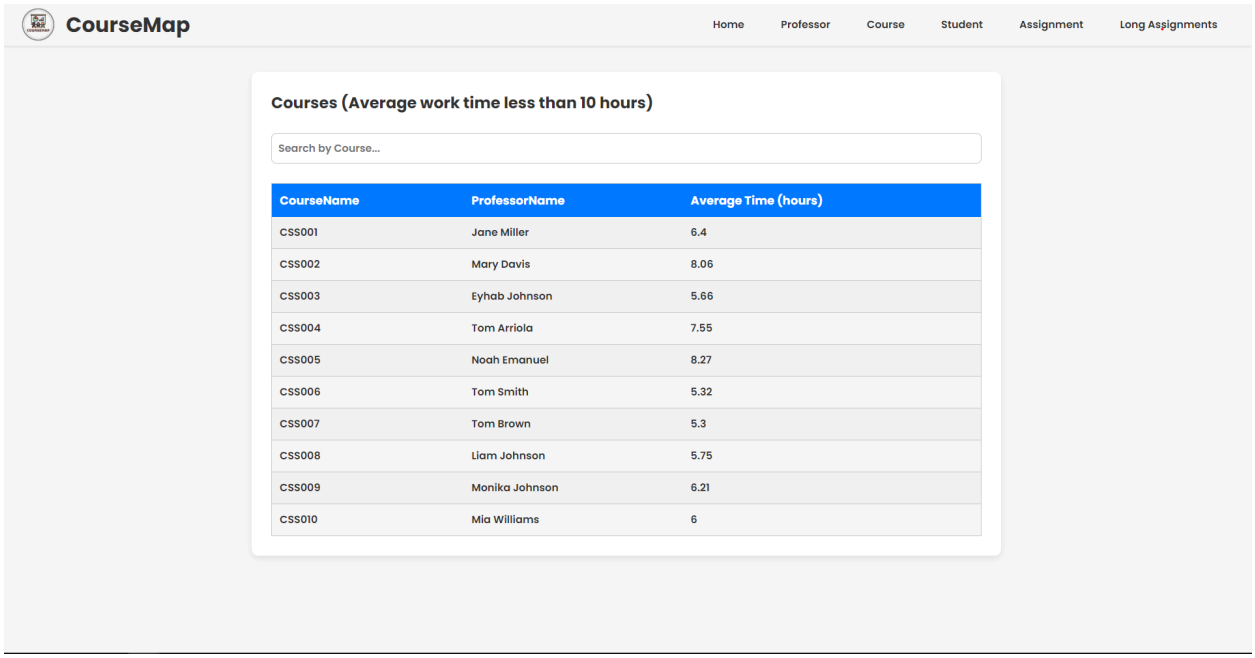
Professor Details

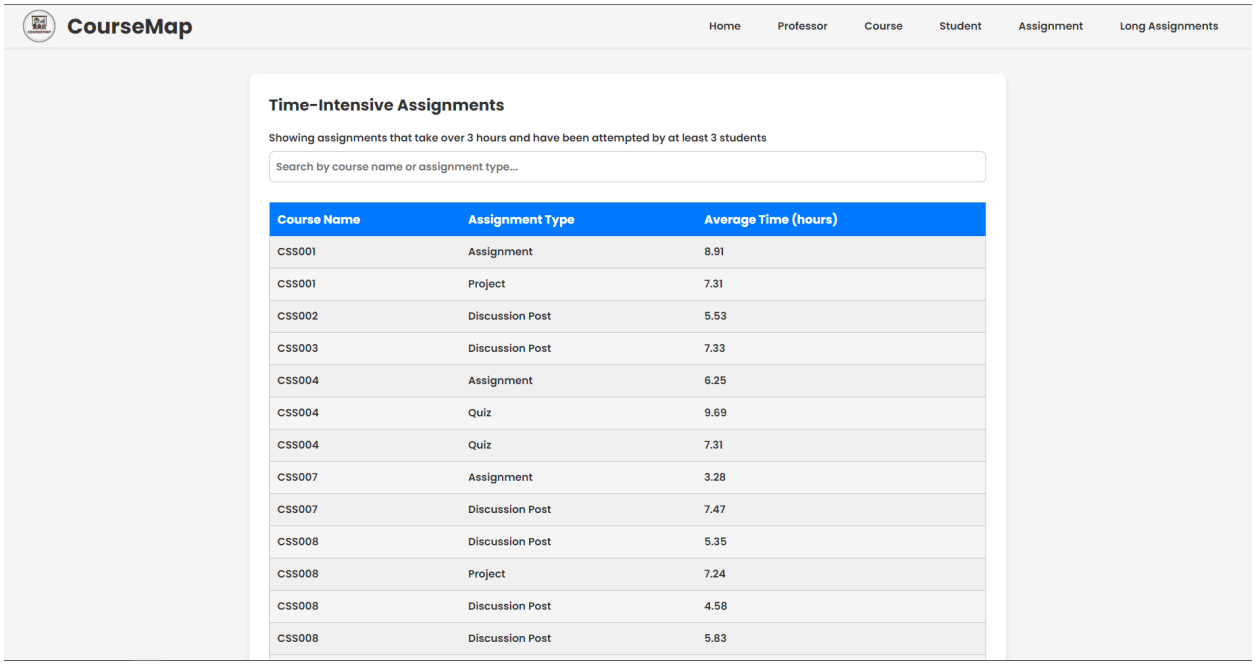
Select a Professor:

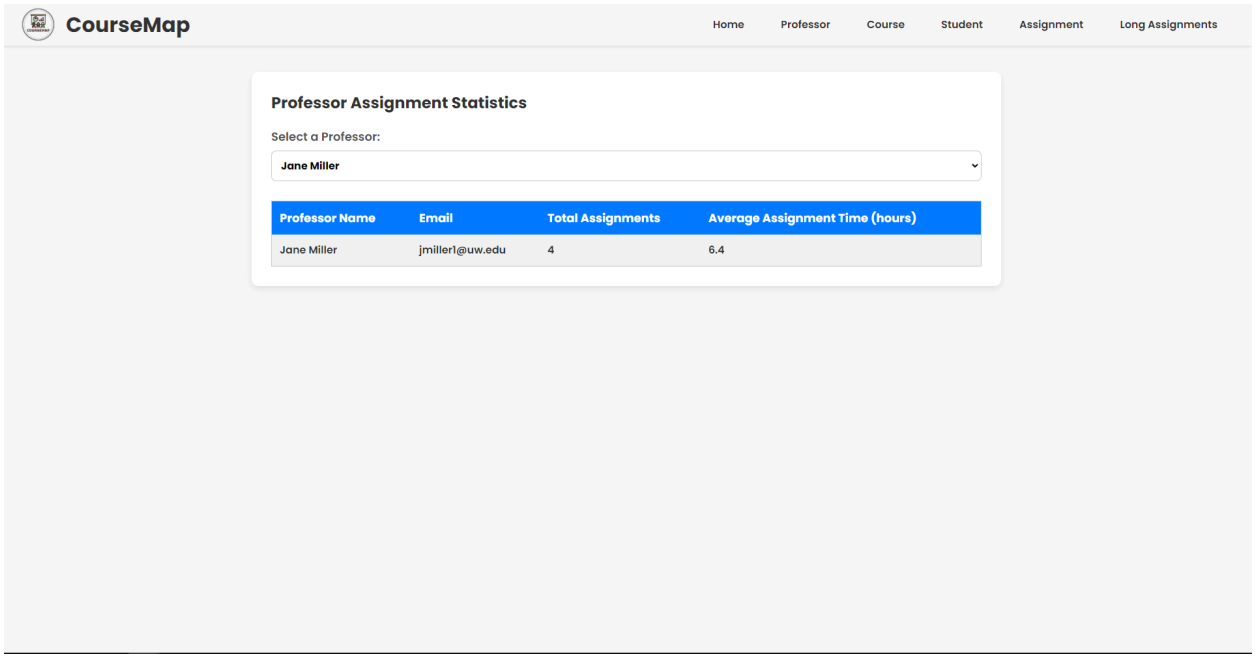
Tom Smith

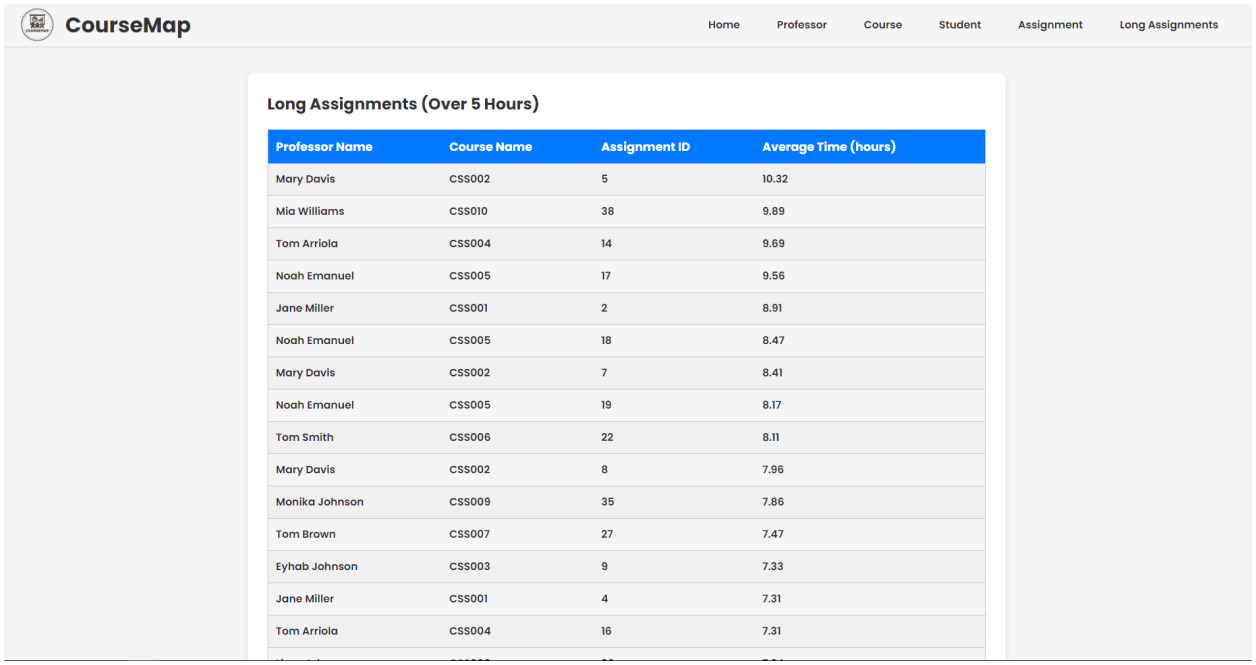
Professor Information

Professor Name	Course Name	Assignment Type	Student Average Time
Tom Smith	CSS006	Project	3.86
Tom Smith	CSS006	Quiz	8.11
Tom Smith	CSS006	Project	5.48
Tom Smith	CSS006	Project	3.82


CourseMap
Home
Professor
Course
Student
Assignment
Long Assignments
Courses (Average work time less than 10 hours)
Search by Course...
CourseName
ProfessorName
Average Time (hours)
CSS001 Jane Miller 6.4
CSS002 Mary Davis 8.06
CSS003 Eyhab Johnson 5.66
CSS004 Tom Arriola 7.55
CSS005 Noah Emanuel 8.27
CSS006 Tom Smith 5.32
CSS007 Tom Brown 5.3
CSS008 Liam Johnson 5.75
CSS009 Monika Johnson 6.21
CSS010 Mia Williams 6


CourseMap
Home
Professor
Course
Student
Assignment
Long Assignments
Time-Intensive Assignments
Showing assignments that take over 3 hours and have been attempted by at least 3 students
Search by course name or assignment type...
Course Name
Assignment Type
Average Time (hours)
CSS001 Assignment 8.91
CSS001 Project 7.31
CSS002 Discussion Post 5.53
CSS003 Discussion Post 7.33
CSS004 Assignment 6.25
CSS004 Quiz 9.69
CSS004 Quiz 7.31
CSS007 Assignment 3.28
CSS007 Discussion Post 7.47
CSS008 Discussion Post 5.35
CSS008 Project 7.24
CSS008 Discussion Post 4.58
CSS008 Discussion Post 5.83





Professor Name	Course Name	Assignment ID	Average Time (hours)
Mary Davis	CSS002	5	10.32
Mia Williams	CSS010	38	9.89
Tom Arriola	CSS004	14	9.69
Noah Emanuel	CSS005	17	9.56
Jane Miller	CSS001	2	8.91
Noah Emanuel	CSS005	18	8.47
Mary Davis	CSS002	7	8.41
Noah Emanuel	CSS005	19	8.17
Tom Smith	CSS006	22	8.11
Mary Davis	CSS002	8	7.96
Monika Johnson	CSS009	35	7.86
Tom Brown	CSS007	27	7.47
Eyhab Johnson	CSS003	9	7.33
Jane Miller	CSS001	4	7.31
Tom Arriola	CSS004	16	7.31