

# **PROJECT REPORT**

Submitted to

**DEPARTMENT OF COMPUTER SCIENCE**



**VIVEKANAND EDUCATION SOCIETY'S  
COLLEGE OF ARTS, SCIENCE AND COMMERCE,  
SINDHI SOCIETY, CHEMBUR, MUMBAI 400071.**

**Tutor web Application**

For Partial Fulfillment for Degree of  
**Bachelor of Science (Computer Science)**  
**Academic Year (2024-25)**

COORDINATOR OF DEPARTMENT

**Mr. Kamlakar Bhopatkar**

COLLEGE GUIDE

**Ms. priya Daniel**

SUBMITTED BY

**Mrityunjay Kumar Pathak**



**VIVEKANAND EDUCATION SOCIETY'S  
COLLEGE OF ARTS, SCIENCE AND COMMERCE**  
**Sindhi Society, Chembur, Mumbai 400071. Phones:**  
**25227514/25227470**  
**NAAC Re-Accredited 'A' Grade**

## **CERTIFICATE**

This is to certify that **Mr. Mrityunjay Kumar Pathak**  
of T.Y.B.Sc (Computer Science) affiliated to  
University of Mumbai has successfully completed a  
project work entitled.

### **Tutor Web Application**

As partial fulfillment of the requirement for the degree  
of B.Sc. (Computer Science) for the academic year  
2022-2023.

COORDINATOR OF DEPARTMENT

**Mr. Kamlakar Bhopatkar**

COLLEGE GUIDE

**Ms. Priya Daniel**

Date: 27-09-2024

Examiner: College Seal

## **Acknowledgement:**

I have great pleasure in presenting this project entitled “**Tutor Web Application**” and I grab this opportunity to convey my immense regards towards all the people who with their invaluable contributions made this project successful.

It gives me great pleasure in presenting this project report. Its justification will never complete if I don't express my vote of thanks to our **V.E.S. College** and **Principal Dr. Anita Kanwar**

I sincerely thank and express my profound gratitude to our Project Guide **Ms. Priya Daniel** for timely and prestigious guidance required for the project completion at each phase of the project development.

I also owe to my friends who have been a constant source of help to solve the problems that cropped up during the development of the project, positive criticism, suggestions, constant support, encouragement and guidance towards the successful completion of the project.

## **INDEX**

<b><u>SR. NO.</u></b>	<b><u>TOPIC PAGE NO.</u></b>
	<b>PRELIMINARY INVESTIGATION</b>
<b>1</b>	<b>Introduction to project 6</b>
<b>2</b>	<b>Description of proposed system 6</b>
<b>3</b>	<b>Description Proposed system and its Modules. 7</b>
<b>4</b>	<b>Advantages of Proposed system 8</b>
<b>5</b>	<b>Technologies Used &amp; Basic Scenarios. 9</b>
	<b>SYSTEM ANALYSIS</b>
<b>6</b>	<b>Use Case Diagram. 11</b>
<b>7</b>	<b>Activity Diagram. 15</b>
<b>8</b>	<b>Database Diagram 20</b>
	<b>SYSTEM IMPLEMENTATION</b>
<b>9</b>	<b>Test Cases 21</b>
<b>10</b>	<b>Screenshots 25</b>
<b>11</b>	<b>Bibliography &amp; References 30</b>

# **PRELIMINARY INVESTIGATION**

## **Introduction to Application**

The Tutor App is an online platform designed to facilitate learning and teaching between students and educators. It allows teachers to create courses, manage assignments, and track student progress, while students can enroll in courses, submit assignments, and monitor their academic performance. The app also provides a search feature to find teachers and courses, enhancing user interaction and accessibility. With a streamlined interface and dynamic features, the Tutor App simplifies education management for both teachers and students.

## **Description of Proposed System**

The Tutor App is a **web-based platform** designed to enhance the interaction between students and educators in a streamlined digital environment. The system allows students to search for and enroll in courses, manage assignments, and track their progress, while educators can create courses, manage student assignments, and monitor student performance. This web application serves as a comprehensive solution for online tutoring, making educational resources more accessible.

## **Workflow and Modules:**

- **Login and Authentication:** Both teachers and students will have secure login functionality, allowing them to access their respective dashboards.
- **Student Management:** Teachers will be able to add, remove, and track students enrolled in their courses.
- **Course Management:** Teachers can create, update, and delete courses. Course details such as the description, duration, and syllabus will be managed through this module.
- **Assignment Management:** Teachers can upload and manage assignments, while students will be able to view, submit, and track the deadlines for their assignments through this module.

- **Search Module:** Students can search for teachers by name or subject and search for courses by title or description, helping them discover relevant content easily.

## **Advantage of Proposed System**

The Advantage of Proposed System are as follows-

- Less time consuming.
- Convenience and Accessibility
- Students can explore and enroll in courses that suit their interests and needs, while teachers can tailor their course .
- The built-in search feature helps students quickly find relevant teachers and courses.

## **Technologies used**

The Tutor App is built using a modern stack of technologies to ensure scalability, performance, and ease of use:

1. **Frontend:** The user interface is developed using **HTML**, **CSS**, and **JavaScript**, providing a responsive and dynamic user experience across various devices.
2. **Backend:** **Node.js** and **Express.js** are used to create the server-side logic and API endpoints, enabling efficient handling of requests and data processing.
3. **Database:** **MongoDB** is used as the database to store and manage information about users, courses, assignments, and other relevant data in a flexible and scalable way.
4. **File Handling:** The application supports file uploads (e.g., assignments) using **Multer**, allowing students to submit their work securely.
5. **Authentication:** The system uses **JWT (JSON Web Tokens)** for secure and efficient user authentication and session management.
6. **API Communication:** **RESTful APIs** are employed to facilitate communication between the frontend and backend, ensuring seamless data exchange.

# SYSTEM ANALYSIS

## Use Case Diagram

A use case diagram depicts the various operations that a system performs. It contains use cases, actors, and their relationships. Use cases are the sequence of actions that form a single unit of work for an actor. An actor represents a user who is external to the system and interacts with the use case.

### **ELEMENTS OF USE CASE DIAGRAM:**

#### **Actors**

An actor portrays any entity (or entities) that perform certain roles in a given system. The different roles the actor represents are the actual business roles of users in a given system. An actor in a use case diagram interacts with a use case. For example, for modeling a banking application, a customer entity represents an actor in the application. Similarly, the person who provides service at the counter is also an actor. But it is up to you to consider what actors make an impact on the functionality that you want to model. If an entity does not affect a certain piece of functionality that you are modeling, it makes no sense to represent it as an actor. An actor is shown as a stick figure in a use case diagram depicted "outside" the system boundary.

#### **Use Cases**

A use case in a use case diagram is a visual representation of distinct business functionality in a system. The key term here is "distinct business functionality." To choose a business process as a likely candidate for modeling as a use case, you need to ensure that the business process is discrete in nature. As the first step in identifying use cases, you should list the discrete business functions in your problem statement. Each of these business functions can be classified as a potential use case. Remember that identifying use cases is a discovery rather than a creation. As business functionality becomes clearer, the underlying use cases become more easily evident. A use case is shown as an ellipse in a use case diagram.

#### **System Boundary**

A system boundary defines the scope of what a system will be. A system cannot have infinite functionality. So, it follows that use cases



also need to have definitive limits defined. A system boundary of a use case diagram defines the limits of the system. The system boundary is shown as a rectangle spanning all the use cases in the system.

**Relationships: The following relationships can be established among use cases**

- **Extends:** A use case may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label “«extend»”.

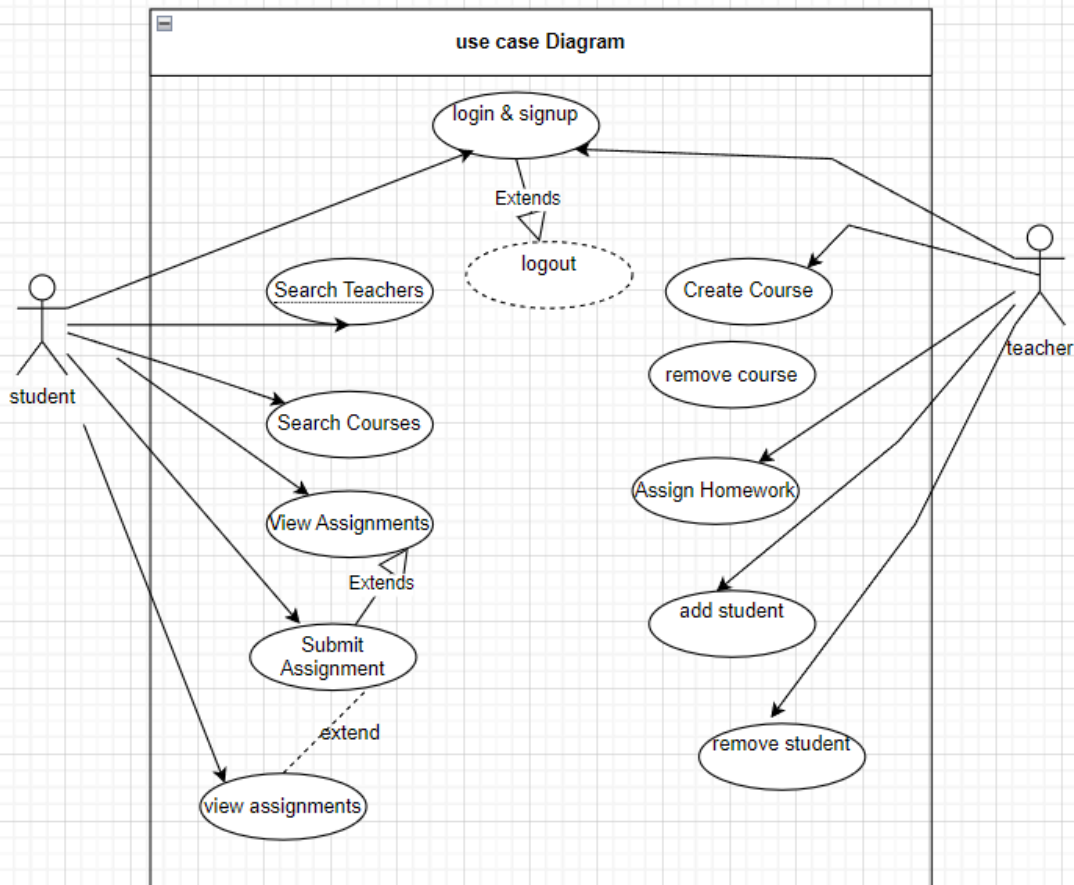
- **Includes:** A use case may include another. Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case. The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behavior from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label “«include»”.

## **Basic Scenario**

Admin

- Register
- Logs in
- Add details of Clients
- Add details of Clients
- Adds Meeting
- Confirms booking of Clients

## Use case diagram



## Activity diagram

### Activity Diagram

An Activity Diagram visually represents the flow of actions or operations that occur within a system. It describes the dynamic aspects of the system by showing the step-by-step sequence of activities or processes, including decisions, parallel actions, and the flow of control. Activity diagrams are often used to model the behavior of a system and

**illustrate how different processes interact.**

## **ELEMENTS OF ACTIVITY DIAGRAM:**

### **1. Activities:**

**Activities represent the individual tasks or actions that take place in the system. They are depicted as rounded rectangles or ovals. An activity can be a simple operation like "login," or it can represent a larger workflow like "submit assignment."**

### **2. Initial Node (Start):**

**This marks the starting point of the workflow. It is represented by a filled black circle and indicates where the process begins.**

### **3. Final Node (End):**

**This marks the end of the workflow. It is represented by a circle with a border around it. The workflow terminates at this point.**

### **4. Decision Node:**

**This is represented by a diamond shape. It indicates a branching in the flow where the process splits based on conditions. The system makes decisions at these nodes, such as whether login credentials are valid or invalid.**

### **5. Fork and Join:**

**Forks and joins are used to represent parallel activities. A fork (represented by a horizontal or vertical bar) splits the flow into multiple activities that occur simultaneously, while a join synchronizes multiple activities back into a single flow.**

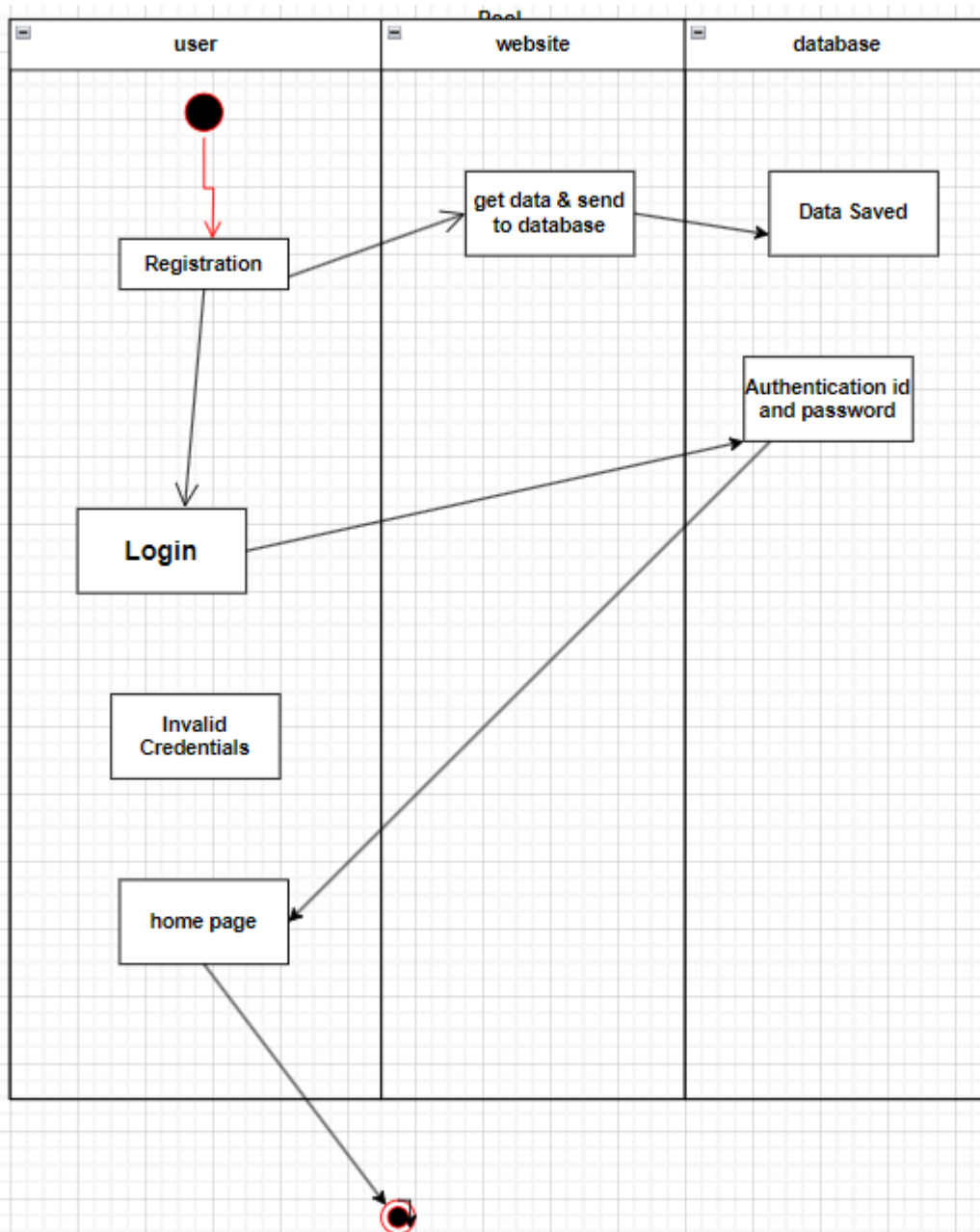
### **6. Transitions:**

**Arrows represent transitions between activities, showing the flow from one activity to the next. They indicate the sequence of execution within the workflow.**

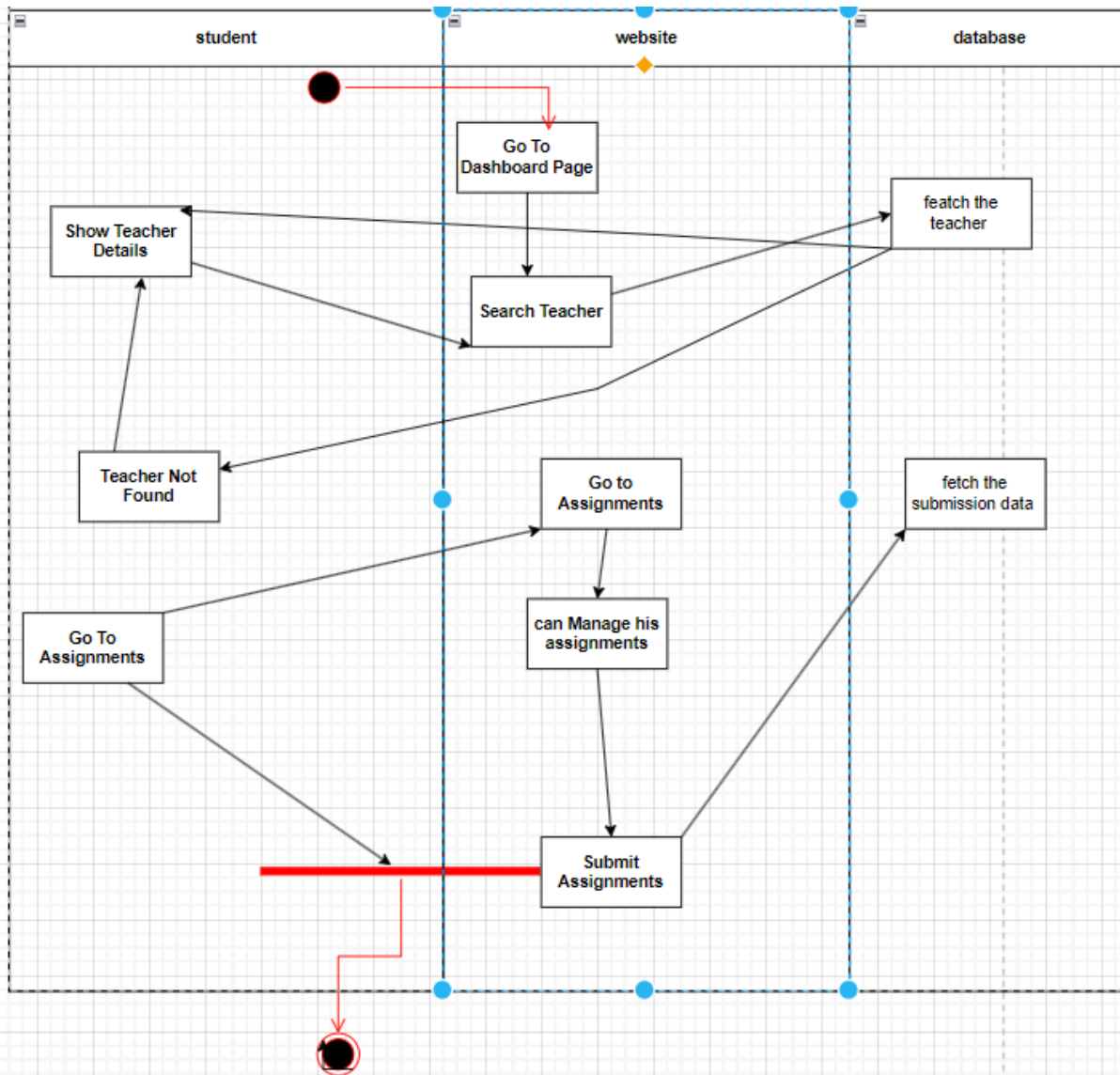
### **7. Swimlanes:**

**Swimlanes are used to group related activities by the actor or system that performs them. They help clarify the responsibility of different actors, such as students, teachers, or the system itself.**

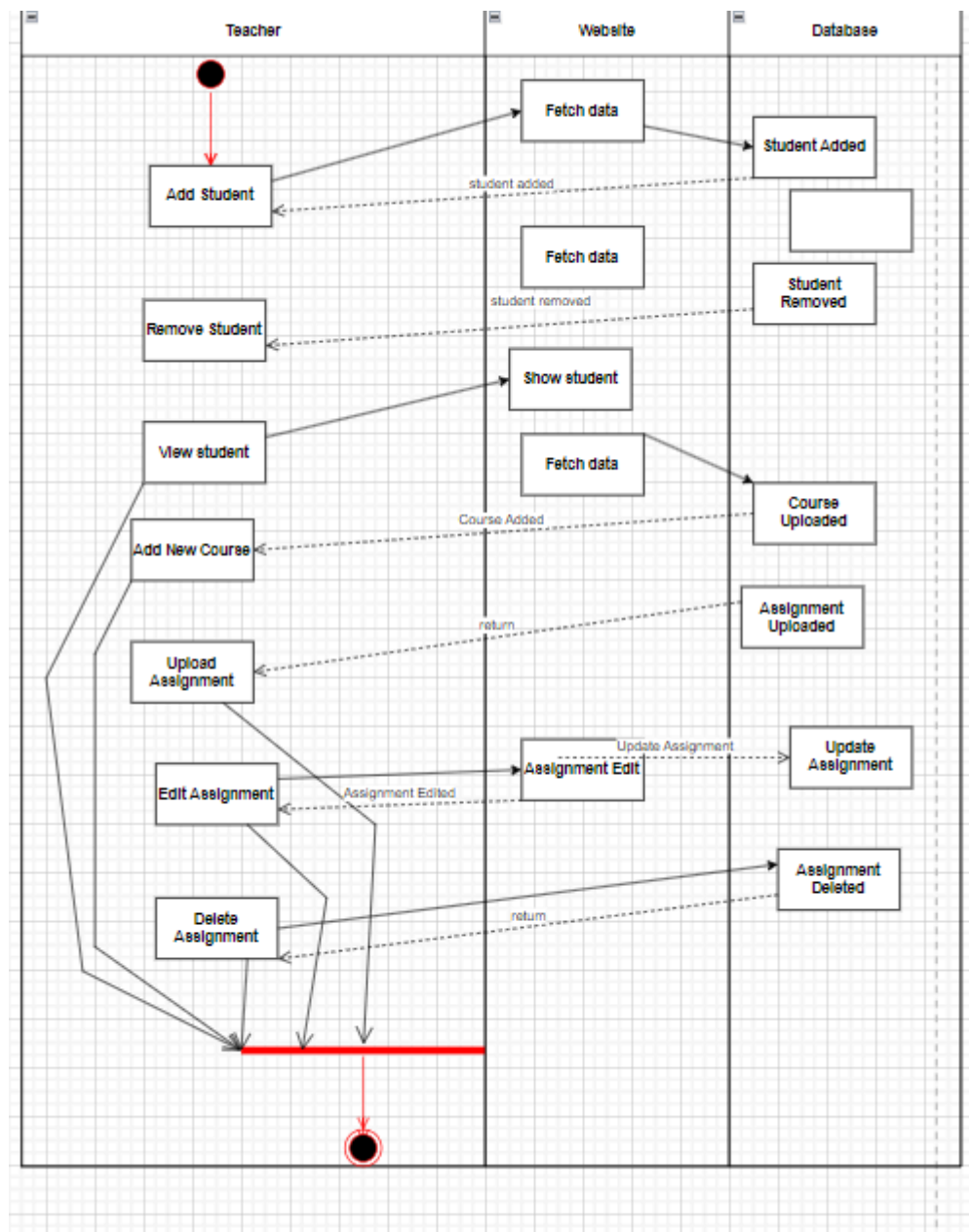
## User login



## Students Activity:



## TEACHER ACTIVITY:



# Database Diagram

## Database Diagram Explanation

A Database Diagram visually represents the structure of a database. It includes tables, columns, relationships between tables, primary keys, and foreign keys. This diagram helps understand how data is organized and how various entities in the system are related.

## Elements of a Database Diagram:

### Tables

Each table represents an entity in the system (such as Students, Teachers, or Courses). The table consists of rows (data entries) and columns (fields or attributes). A table is the foundation of relational databases.

### Columns

Columns (or fields) define the properties of the entity represented by the table. Each column stores specific information, such as a student's name, a course title, or a teacher's email. Each column has a defined data type (e.g., string, integer).

### Primary Key

The Primary Key is a unique identifier for each record in a table. It ensures that no two rows in the table have the same identifier. For example, a **studentID** in the Students table uniquely identifies every student.

### Foreign Key

A Foreign Key establishes a relationship between two tables. It links the primary key of one table to a corresponding column in another table. For example, a **courseID** in the **Enrollments** table might link to the primary key **courseID** in the **Courses** table, indicating the course a student is enrolled in.

### Relationships

Relationships define how tables interact with each other. A one-to-many relationship means one record in a table can relate to multiple records

in another table (e.g., a Teacher can teach multiple Courses). A many-to-many relationship might be modeled through an intermediary table (e.g., students and courses via an Enrollment table).

Student
+ Name
+class
+Previous Year Grade
+Email
+Password
+Mobile Number
+Parent Mobile number
+

Teachers
+First Name
+Last Name
+Email
+Password
+Education
+ Certificate
+Photo
+subject
+Date Registered
+

Course
+Course Name
+teacher
+Start Date
+End Date
+Delete Course
+Student

assignment
+Assignment Title
+Assignment description
+Assignment File



# Sequence Diagram:

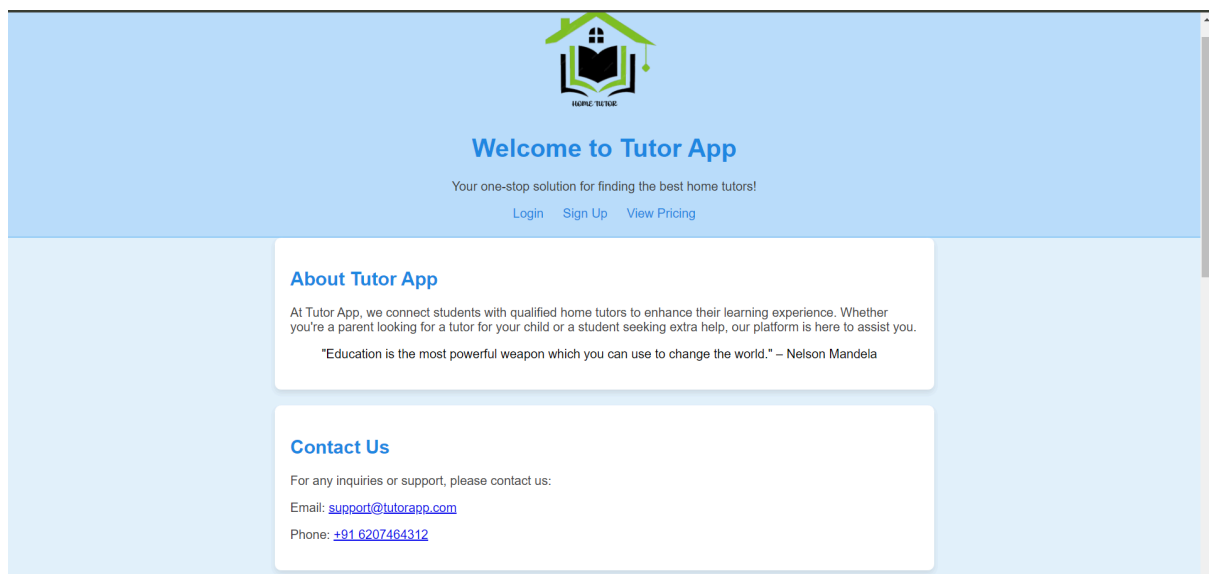
## Test Cases:

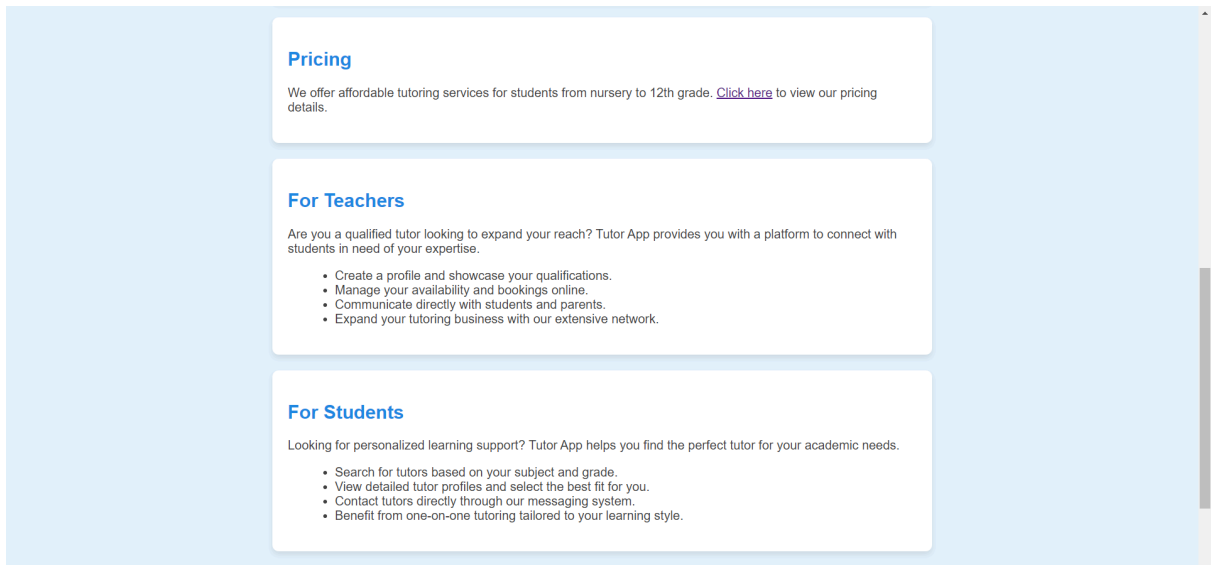
Test Case ID	Module	Form	Test Conditions	Steps Input Test data	Expected output	Actual Output	Status
TC 1	Register	User Register	To register user should submit all the data requires	Fill the data and tap on submit	it will take the data and register data	Registers user to the tutor App	Pass
TC 2	Login	User Login	To login user should give email and password. Tutor App will authenticate it	Fill the Email and Password	it will give access to dashboard.	Logs in to the Dashboard	Pass

TC 3	Your Assignment page	View Assignments	To see the assignments given to students	Go to student Assignment Folder there	it will show Assignments given	It shows assignments	Pass
TC 4	Manage Assignment page	Give Assignments	To check Assignment is uploaded or not	Give title upload oneDocument and submit	it will upload the assignment	It uploads assignments	Pass
TC 5	Manage Students	Add Student	To check the student is added	Fill the email And course after it press submit	a student will be added to the course with a notification	Goes to property activity	Pass

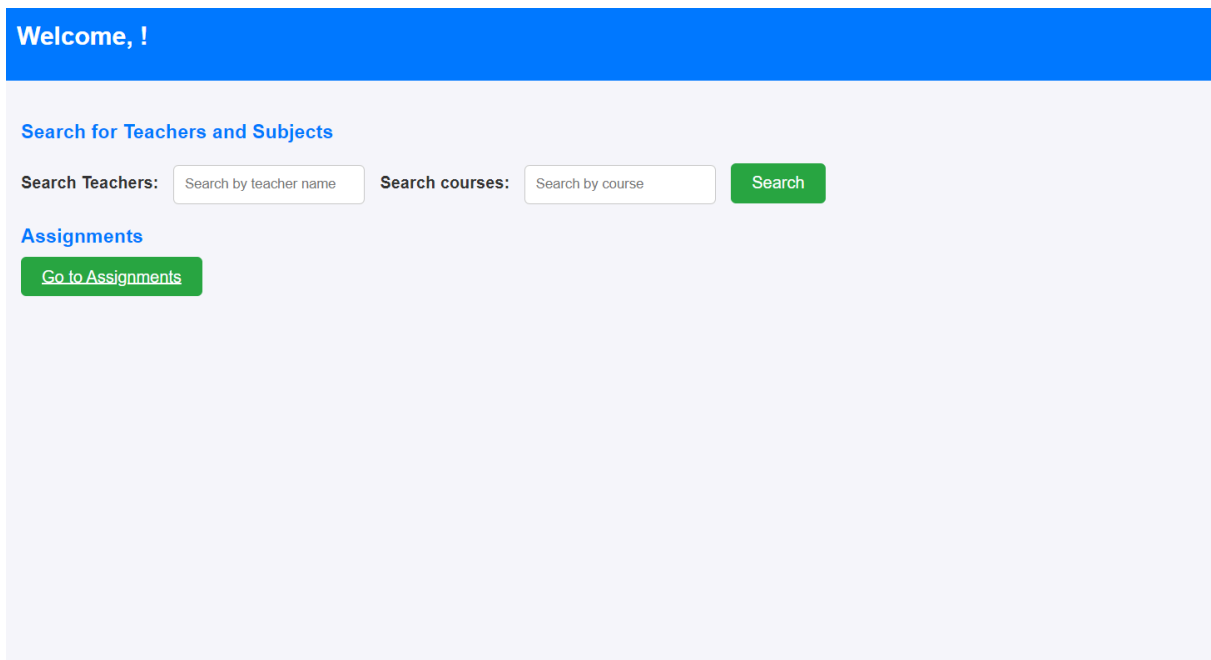
## ScreenShots:

## Index:






## Student Dashboard:



## Teacher Dashboard:

Welcome, !

 Profile  
Menu

#### Your Dashboard

[Manage Your Assignments](#)

#### Messages from Students

## Assignment pages :

### Assignments Given to You

[View All Assignments](#)

### Progress Tracking

[View Your Progress](#)

### Submit Your Assignment

Assignment Title:

Upload Assignment:

No file chosen

Notes (optional):

Manage Assignments

Assignments Folder

View All Assignments

Upload New Assignment

Edit Assignment

Delete Assignment

Students Folder

View All Students

Add New Student

Remove Student

Tasks Given to Students

View Assigned Tasks


Assign New Task

Edit Assigned Task

Delete Assigned Task

add a new course

Student adding :



Add New Student

New Student Form

Email:

mrityunjaypathak2111@gmail.com

Course Name:

photography

Add Student

© 2024 Tutor App. All rights reserved.

**Bibliography & References**

Websites:

- <https://www.mongodb.com>
- <https://www.tutorialspoint.com>
- <https://www.w3schools.com>
- <https://nodejs.org/en>

## Api's :

### **Student Registration API** (POST /api/students/register)

- Registers a new student in the system.

### **Student Login API** (POST /api/students/login)

- Authenticates a student and generates a session token.

### **Teacher Registration API** (POST /api/teachers/register)

- Registers a new teacher in the system.

### **Teacher Login API** (POST /api/teachers/login)

- Authenticates a teacher and generates a session token.

### **Create Course API** (POST /api/courses)

- Allows teachers to create a new course.

### **Fetch Courses API** (GET /api/courses)

- Fetches a list of courses. Supports query parameters to search by course name.

### **Fetch Teachers API** (GET /api/teachers)

- Fetches a list of teachers. Supports query parameters to search by teacher name.

### **Fetch Single Teacher API** (GET /api/teacher/:username)

- Fetches specific details of a teacher by their username.

### **Course Enrollment API** (POST /api/courses/:courseId/enroll)

- Allows a student to enroll in a course.

### **Assignment Submission API** (POST /api/assignments/submit)

- Enables students to submit assignments for their enrolled courses.

**Teacher's Courses API** (GET /api/teachers/:teacherId/courses)

- Fetches the courses associated with a specific teacher.

**Student Dashboard Data API** (GET /api/students/:studentId/dashboard)

- Retrieves data like enrolled courses and assignments for a specific student.