**[DoF3aa]**

**By  
Mohamed Badawy Sayed**

**Mahmoud Ahmed Khalil Ibrahim**

**Ahmed Nageh Abbas  
Mostafa Ahmed Hasan El-Gelany**

**[](https://github.com/Mohamed-badawy-sayed/CMS)**

[**GitHub Repo**](https://github.com/0xBadawy/Dof3aa)

**Dr.**

**Mohamed Ramadan Saady**

**Mohamed Ahmed El-Kholy**

# Table of Contents

Contents

[Table of Contents 1](#_Toc165650582)

[1 Introduction 2](#_Toc165650583)

[2 Requirements 4](#_Toc165650584)

[3 use case scenarios 6](#_Toc165650585)

[4 Dependencies 20](#_Toc165650586)

[5 System Architecture 20](#_Toc165650587)

[6 Diagrams 20](#_Toc165650588)

# 1 Introduction

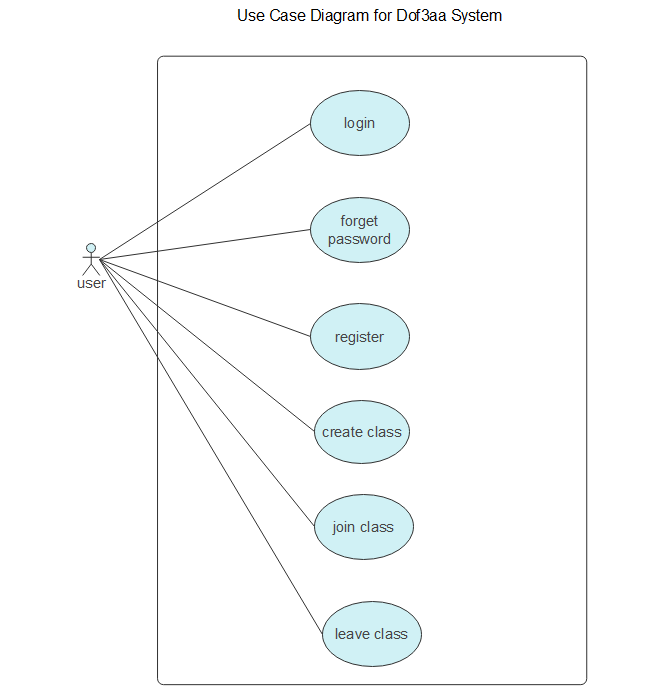
Dof3aa is a web application designed to foster a cohesive and organized community among faculty’s students. It serves as a central hub for all faculty-related updates, materials, and assignments, ensuring that students remain fully informed and engaged with their academic environment.

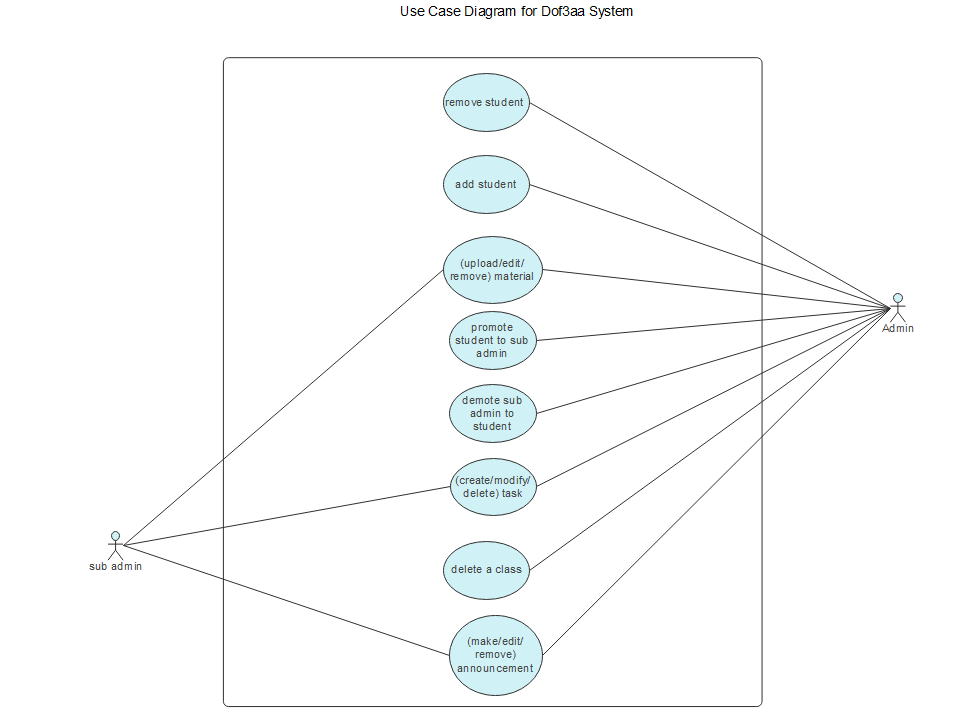
This comprehensive documentation offers a detailed insight into the system, covering its scope, requirements, dependencies, and architecture.

1.2 Scope

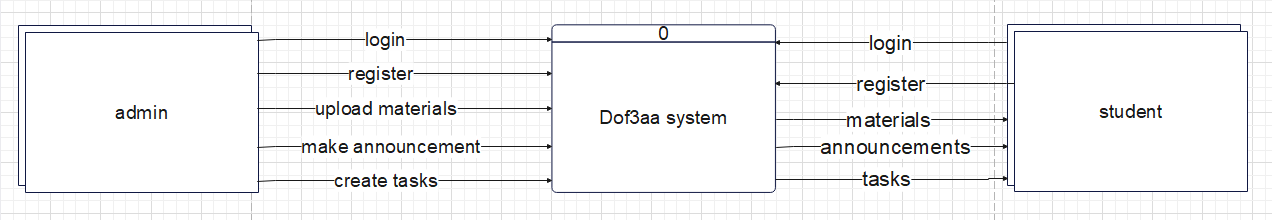
The dof3aa Web Application will include features for both admins and students. Admins will be able to create and manage courses, upload course materials, and add announcements. Students will be able to enroll in courses, and access course materials.

1.2.1 use case diagram





1.2.2 context diagram



# 2 Requirements

**2.1 Functional requirements**

**2.1.1 System Features: Authentication and Authorization**

General Authentication:

* **User Registration & Login**: Users can register for an account and log in. The login process is secured, ensuring that user credentials are protected.

Role-Based Access Control:

* **Role Differentiation**: The system distinguishes between two primary roles: Admin and Student. Each role has unique permissions and access rights within the system.

**2.1.2 Admin Role Capabilities:**

* **Course Management**: Admins have the ability to create new courses within the system or delete courses.
* **Content Management**: Admins can upload, update, and remove course materials as needed.
* **Announcements**: Admins have the authority to create, edit, and delete announcements for courses.
* **Enrollment Oversight**: Admins manage student enrollments, including adding or removing students from courses.
* **Role Assignment**: Admins can change the roles of users, promoting students to admins or demoting admins to students, as necessary.
* **Assignment Management**: Admins can create, modify, and delete assignments within courses and add deadlines to tasks.
* **User Management**: Admins have the power to remove students or other admins from the system, with the exception of the original course creator.

**2.1.3 Student Role Capabilities:**

* **Course Enrollment**: Students can enroll in available courses of their interest.
* **Resource Access**: Students have the ability to browse through course materials, announcements, and assignments.
* **Material Downloads**: Students can download course materials for offline study.
* **Course Withdrawal**: Students have the option to unenroll from courses if they choose.

**2.2 Non-functional requirements:**

**2.2.1 Security:**

* **Data Encryption:** All personal information and user credentials must be securely encrypted to protect against unauthorized access.
* **Access Control**: Implement strict access control measures to ensure that only authenticated students can access their respective courses, materials, and announcements.

**2.2.2 Performance:**

* **Quick Load Times:** The platform should load content and respond to user requests within 2 seconds under typical usage conditions.
* **Scalability:** Capable of scaling dynamically to accommodate increases in user numbers and data volume, supporting at least 10,000 simultaneous student logins.

**2.2.3 Usability:**

* **Intuitive Design:** The platform must feature an intuitive, straightforward design enabling students to find and use features without extensive guidance.
* **Support Resources:** Provide detailed help documentation and resources, allowing students to resolve common issues independently and understand platform functionalities.

**2.2.4 Reliability:**

* **High Availability:** Achieve 99.9% uptime, not including planned maintenance windows, to ensure students can always access their materials and announcements.
* **Consistent Data Integrity:** Employ mechanisms to ensure that uploaded materials and entered data are consistently validated, preventing errors or corruption.
* **Robust Backup and Recovery:** Regularly back up data with a reliable recovery strategy in place to quickly restore service in case of system failure or data loss.

**2.2.5 Maintainability:**

* **Modular Architecture:** Design the system with a modular approach to simplify updates, maintenance, and the rollout of new features.
* **Comprehensive Documentation:** Maintain detailed documentation of the system’s architecture, codebase, and APIs to facilitate efficient maintenance and future enhancements.
* **Future-proof Technologies:** Choose widely supported and contemporary technologies for development to ensure long-term viability and community support.

# 3 use case scenarios

|  |  |
| --- | --- |
| case name: | register |
| case ID: | **Op1** |
| Actor(s): | **user** |
| Description: | **User register to system** |
| Triggering: | **User want to create an account** |
| Preconditions : | **User should have an email** |
| Main path : | 1. **Enter email** 2. **Enter password** 3. **Receive link on entered account** 4. **Verify account** |
| Alter Path : | **If the email is already use you have to use another email** |
| Post conditions: | **User have an account on the system** |
| Priority: | **Medium** |

|  |  |
| --- | --- |
| case name: | Log in |
| case ID: | **Op2** |
| Actor(s): | **user** |
| Description: | **User log in to the system** |
| Triggering: | **User wants use his account** |
| Preconditions : | **User should have an account** |
| Main path : | 1. **Enter email** 2. **Enter password** 3. **Verify email and password** |
| Alter Path : | **If the email or password is wrong show the message “wrong email or password”** |
| Postcondition: | **User logged in the system** |
| Priority: | **high** |

|  |  |
| --- | --- |
| case name: | Forget password |
| case ID: | **Op3** |
| Actor(s): | **user** |
| Description: | **The user forgot the password** |
| Triggering: | **The user forgot the password and wants to change it** |
| Preconditions : | **User should have an account** |
| Main path : | 1. **Enter email** 2. **Receive link on email** 3. **Enter link** 4. **Set new password** |
| Postcondition: | **User changed the password** |
| Priority: | **high** |

|  |  |
| --- | --- |
| case name: | Join class |
| case ID: | **Op4** |
| Actor(s): | **user** |
| Description: | **User join a class via class id** |
| Triggering: | **User want to join class** |
| Preconditions : | **The user is logged in** |
| Main path : | 1. **Enter the class section** 2. **Click join the class** 3. **Enter class id** |
| Alter Path : | **if id is invalid show “invalid id”.**  **if user already joint to the class show “already joint in the class”.** |
| Postcondition: | **User joined the class** |
| Priority: | **low** |

|  |  |
| --- | --- |
| case name: | Create class |
| case ID: | **Op5** |
| Actor(s): | **user** |
| Description: | **User creates a new class** |
| Triggering: | **User wants to create a new class** |
| Preconditions : | **The user is logged in** |
| Main path : | 1. **Click Create a class** 2. **Enter name, photo, and id** |
| Alter Path : | **if id is used show “this id is already used”** |
| Postcondition: | **New class created** |
| Priority: | **low** |

|  |  |
| --- | --- |
| case name: | leave class |
| case ID: | **Op6** |
| Actor(s): | **user** |
| Description: | **User leaves a class** |
| Triggering: | **A user wants to leave a class** |
| Preconditions : | **The user already in the class** |
| Main path : | 1. **Enter class** 2. **Enter settings** 3. **Leave class** |
| Alter Path : | **If the user is the admin show “admin can’t leave class”** |
| Postcondition: | **Class left successfully** |
| Priority: | **low** |

|  |  |
| --- | --- |
| case name: | Remove a student |
| case ID: | **Op7** |
| Actor(s): | **Admin** |
| Description: | **Admin wants to remove a student from the class** |
| Triggering: | **Admin wants to remove a student from the class** |
| Preconditions : | **admin is the creator of the class** |
| Main path : | 1. **Enter class** 2. **Browse users of the class** 3. **Select user** 4. **Remove user** |
| Alter Path : | **If the admin tries to remove itself show “admin can’t remove itself”** |
| Postcondition: | **User removed from the class** |
| Priority: | **medium** |

|  |  |
| --- | --- |
| case name: | add a student |
| case ID: | **Op8** |
| Actor(s): | **Admin** |
| Description: | **Admin wants to add a student to the class** |
| Triggering: | **Admin wants to add a student to the class** |
| Preconditions : | **admin is the creator of the class** |
| Main path : | 1. **Enter class** 2. **Enter users** 3. **Click Add user** 4. **Add user email** |
| Alter Path : | **If user already joined the class show “user already exists”**  **If email is not found in the system or user disabled add property show “this email may not exist on the system”** |
| Postcondition: | **Student will be added** |
| Priority: | **medium** |

|  |  |
| --- | --- |
| case name: | (Upload/edit/remove) material |
| case ID: | **Op9** |
| Actor(s): | **Admin, sub-admin** |
| Description: | **Admin wants to (upload/edit/remove) material** |
| Triggering: | **Admin wants to (upload/edit/remove) material** |
| Preconditions : | **User is admin or sub-admin** |
| Main path : | 1. **Enter class** 2. **Enter material section** 3. **Click (add/edit/remove) material** |
| Postcondition: | **Material (added/edited/removed) successfully** |
| Priority: | **medium** |

|  |  |
| --- | --- |
| case name: | (Upload/edit/remove) announcement |
| case ID: | **Op10** |
| Actor(s): | **Admin, sub-admin** |
| Description: | **Admin wants to (upload/edit/remove) announcement** |
| Triggering: | **Admin wants to (upload/edit/remove) announcement** |
| Preconditions : | **User is admin or sub-admin** |
| Main path : | 1. **Enter class** 2. **Enter announcements section** 3. **Click (add/edit/remove) announcements** |
| Postcondition: | **announcement (added/edited/removed) successfully** |
| Priority: | **medium** |

|  |  |
| --- | --- |
| case name: | (Upload/edit/remove) task |
| case ID: | **Op11** |
| Actor(s): | **Admin, sub-admin** |
| Description: | **Admin wants to (upload/edit/remove) task** |
| Triggering: | **Admin wants to (upload/edit/remove) task** |
| Preconditions : | **User is admin or sub-admin** |
| Main path : | 1. **Enter class** 2. **Enter tasks section** 3. **Click (add/edit/remove) tasks** |
| Postcondition: | **tasks (added/edited/removed) successfully** |
| Priority: | **medium** |

|  |  |
| --- | --- |
| case name: | Promote student to sub-admin |
| case ID: | **Op12** |
| Actor(s): | **Admin** |
| Description: | **Admin wants to promote a student** |
| Triggering: | **Admin wants to promote a student** |
| Preconditions : | **admin is the creator of the class .**  **the student joined the class.** |
| Main path : | 1. **Enter class** 2. **Enter users section** 3. **Select user** 4. **Promote to sub-admin** |
| Alter Path : | **If user is already admin or sub-admin show “this student is already admin or sub-admin”** |
| Postcondition: | **Student will be promoted** |
| Priority: | **medium** |

|  |  |
| --- | --- |
| case name: | demote sub-admin to student |
| case ID: | **Op13** |
| Actor(s): | **Admin** |
| Description: | **Admin wants to demote a sub-admin** |
| Triggering: | **Admin wants to demote a sub-admin** |
| Preconditions : | **admin is the creator of the class .**  **the sub-admin joined the class.** |
| Main path : | 1. **Enter class** 2. **Enter users section** 3. **Select sub-admin** 4. **demote to student** |
| Alter Path : | **If user is already student show “this user is already student”** |
| Postcondition: | **Sub-admin will be demoted** |
| Priority: | **medium** |

|  |  |
| --- | --- |
| case name: | Delete class |
| case ID: | **Op14** |
| Actor(s): | **Admin** |
| Description: | **Admin wants to delete class** |
| Triggering: | **Admin wants to delete class** |
| Preconditions : | **admin is the creator of the class.** |
| Main path : | 1. **Enter class** 2. **settings** 3. **click delete class** |
| Alter Path : | **If user is not the creator show “only admin can delete the class”** |
| Postcondition: | **Class will be deleted** |
| Priority: | **high** |

# 4 Dependencies

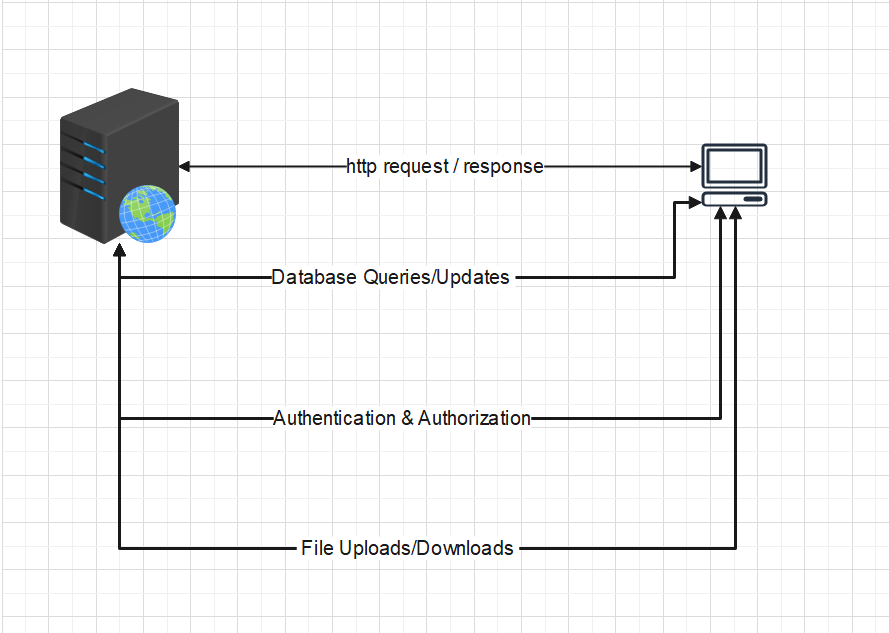
The application may depend on third-party services for features such as authentication.

# 5 System architecture

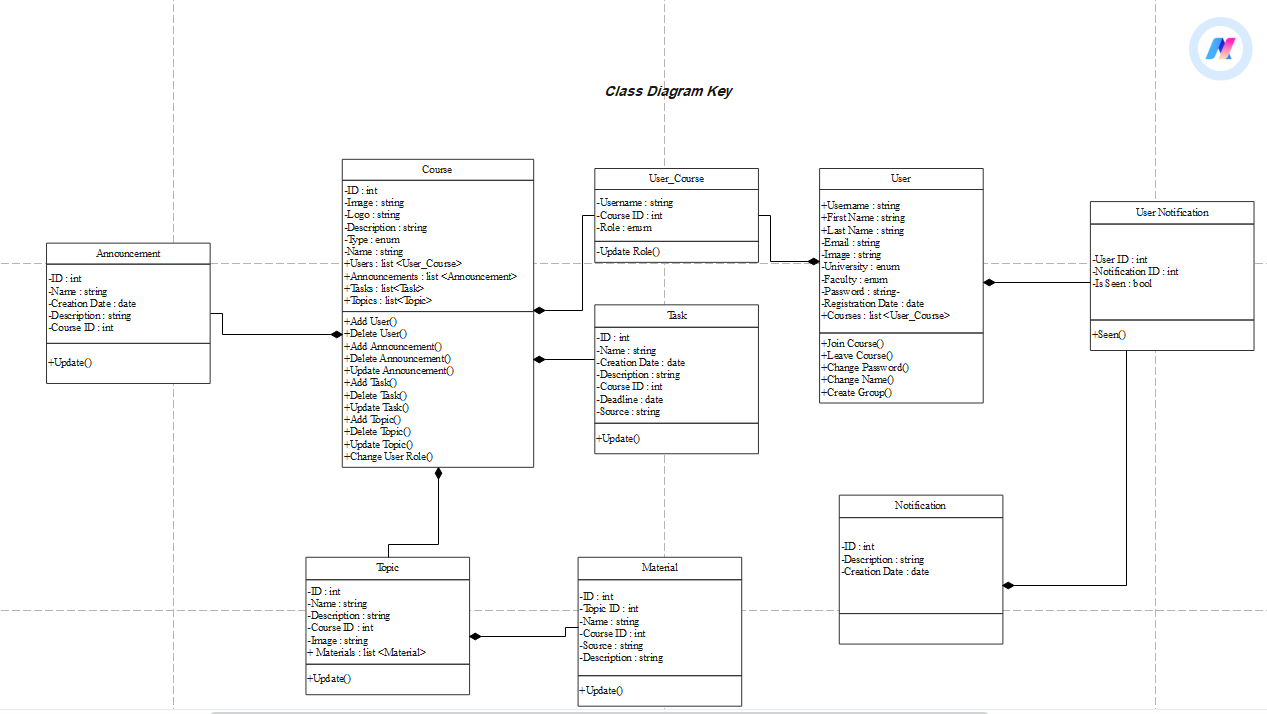
The dof3aa Web Application will be built using a client-server architecture. The front end will be developed using modern web technologies such as HTML5, CSS3, and JavaScript. The back end will be implemented using a server-side framework such as Node.js, with a not only relational database (e.g., Mongo DB) for data storage.

# 6 Diagrams

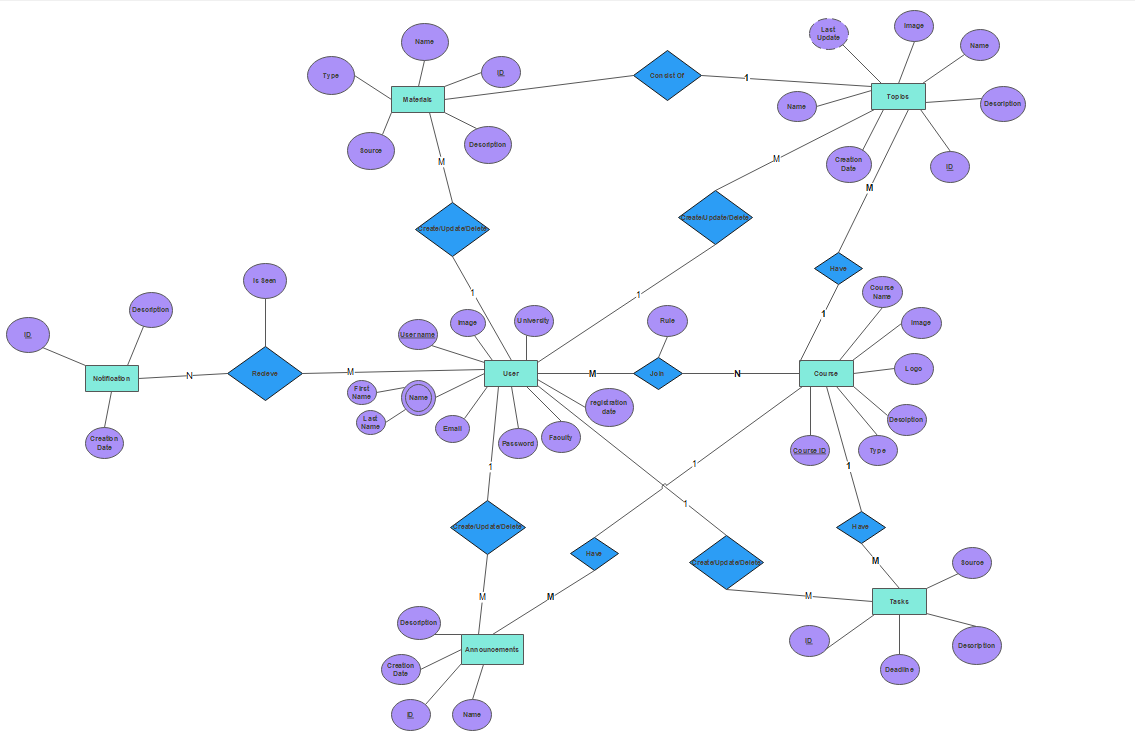
6.1 architecture diagram



6.2 Class diagram



6.3 ERD diagram



6.4 Schema

