[DoF3aa]

By

Mohamed Badawy Sayed

Mahmoud Ahmed Khalil Ibrahim

Ahmed Nageh Abbas
Mostafa Ahmed Hasan El-Gelany

github.com/0xBadawy/Dof3aa

Dr.

Mohamed Ramadan Saady

Table of Contents

Contents

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.

1.1 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, announcements, and tasks.

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.
- **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:	User Registration
case ID:	Op1
Actor(s):	User
Description:	User registers for a new account on the platform.
Triggering:	User intends to create a new account.
Preconditions :	User possesses a valid email address.
Main path :	1. User enters their email address. 2. User sets a password. 3. A verification link is sent to the provided email address. 4. User verifies their account through the link.
Alter Path :	If the email address is already in use, prompt the user to choose a different one.
Post conditions:	User successfully creates an account on the platform.
Priority:	Medium

case name:	User Login
case ID:	Op2
Actor(s):	User
Description:	User logs into their account.
Triggering:	User wishes to access their account.
Preconditions :	User has a registered account.
Main path :	1. User enters their email address. 2. User inputs their password. 3. System verifies the provided credentials.
Alter Path :	If the email or password is incorrect, display a "Wrong email or password" message.
Postcondition:	User successfully logs into the platform.
Priority:	High

case name:	Forgot Password
case ID:	Op3
Actor(s):	User
Description:	User initiates the password reset process.
Triggering:	User forgets their password and seeks to reset it.
Preconditions :	User has a registered account.
Main path :	1. User enters their email address. 2. User receives a password reset link via email. 3. User follows the link to set a new password.
Postcondition:	User successfully changes their password.
Priority:	High

case name:	Join Class
case ID:	Op4
Actor(s):	User
Description:	User joins a class using a class ID.
Triggering:	User intends to join a specific class.
Preconditions :	User is logged into their account.
Main path :	1. User navigates to the class section. 2. User selects "Join a Class". 3. User enters the class ID.
Alter Path :	If the ID is invalid show “invalid ID”. if user already joint to the class show “already joint in the class”.
Postcondition:	User successfully joins the class.
Priority:	Low

case name:	Create Class
case ID:	Op5
Actor(s):	User
Description:	User creates a new class.
Triggering:	User wants to establish a new class.
Preconditions :	User is logged into their account.
Main path :	1. User selects "Create a Class". 2. User provides class name, photo, and ID.
Alter Path :	If the ID is already in use show “This ID is already used”
Postcondition:	New class is successfully created.
Priority:	low

case name:	Leave Class
case ID:	Op6
Actor(s):	User
Description:	User exits a class.
Triggering:	User intends to leave a particular class.
Preconditions :	User is currently enrolled in the class.
Main path :	1. User accesses the class. 2. User enters class settings. 3. User selects "Leave Class".
Alter Path :	If the user is the admin, display "Admin can't leave class" message.
Postcondition:	User successfully exits the class.
Priority:	low

case name:	Remove a Student
case ID:	Op7
Actor(s):	Creator
Description:	Creator removes a student from the class.
Triggering:	Creator decides to remove a student from the class.
Preconditions :	Admin is the creator of the class and admin is logged in.
Main path :	1. Creator accesses the class. 2. Creator navigates to the list of class users. 3. Creator selects the student to remove. 4. Creator confirms the removal action.v
Alter Path :	If the Creator tries to remove themselves, display " Creator can't remove itself" message.
Postcondition:	Student is successfully removed from the class.
Priority:	Medium

case name:	(Upload/Edit/Remove) Material
case ID:	Op8
Actor(s):	Creator, Admin
Description:	Admin or Creator manages class materials.
Triggering:	Admin or Creator intends to (upload/edit/remove) class materials.
Preconditions :	User is either an admin or Creator of the class.
Main path :	1. User accesses the class. 2. User navigates to the materials section. 3. User performs the desired action (upload/edit/remove).
Postcondition:	Material is (added/edited/removed) successfully.
Priority:	Medium

case name:	(Upload/Edit/Remove) Announcement
case ID:	Op9
Actor(s):	Admin, Creator
Description:	Admin or Creator manages class announcements.
Triggering:	Admin or Creator intends to (upload/edit/remove) class announcements.
Preconditions :	User is either an admin or Creator of the class.
Main path :	1. User accesses the class. 2. User navigates to the announcements section. 3. User performs the desired action (upload/edit/remove).
Postcondition:	Announcement is (added/edited/removed) successfully.
Priority:	Medium

case name:	(Upload/Edit/Remove) Task
case ID:	Op10
Actor(s):	Admin, Creator
Description:	Admin or Creator manages class tasks.
Triggering:	Admin or Creator intends to (upload/edit/remove) class tasks.
Preconditions :	User is either an admin or Creator of the class.
Main path :	1. User accesses the class. 2. User navigates to the tasks section. 3. User performs the desired action (upload/edit/remove).
Postcondition:	Task is (added/edited/removed) successfully.
Priority:	Medium

case name:	Promote Student to Admin
case ID:	Op11
Actor(s):	Creator
Description:	Creator promotes a student to the Admin role.
Triggering:	Creator decides to elevate a student's role to Admin.
Preconditions :	Admin is the creator of the class and the student is enrolled.
Main path :	1. Creator accesses the class. 2. Creator navigates to the user management section. 3. Creator selects the student to promote. 4. Creator promotes the selected student to sub-admin.
Alter Path :	If the student is already an admin or Creator, display "This student is already an admin or Creator" message.
Postcondition:	Student is successfully promoted to Admin.
Priority:	Medium

case name:	Demote Admin to Student
case ID:	Op12
Actor(s):	Creator
Description:	Creator demotes an Admin to a regular student.
Triggering:	Admin decides to downgrade an Admin's role to student.
Preconditions :	Admin is the creator of the class and the sub-admin is enrolled.
Main path :	1. Creator accesses the class. 2. Creator navigates to the user management section. 3. Creator selects the sub-admin to demote. 4. Creator demotes the selected sub-admin to student.
Alter Path :	If the user is already a student, display "This user is already a student" message.
Postcondition:	Admin is successfully demoted to student.
Priority:	Medium

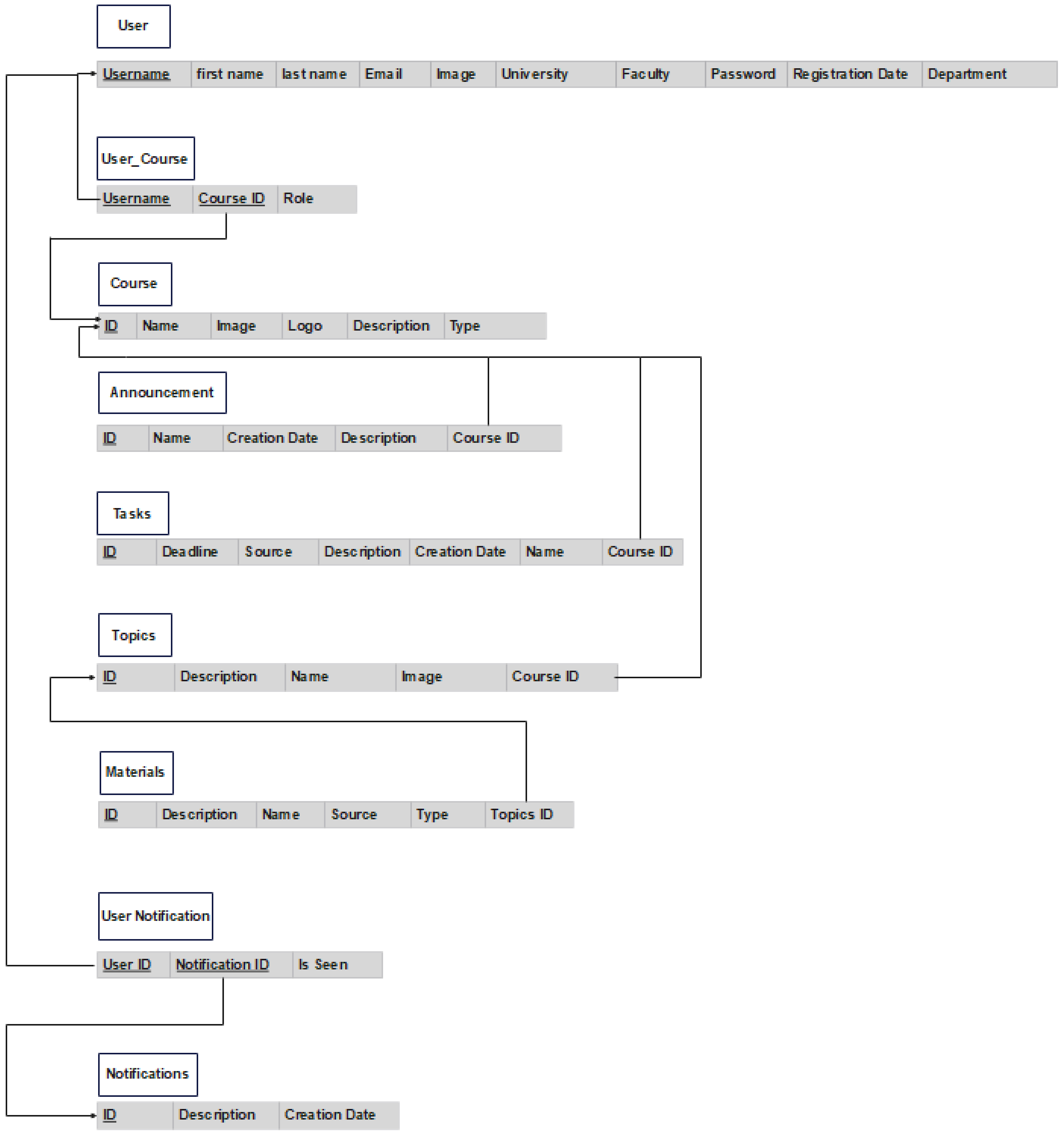
case name:	Delete Class
case ID:	Op13
Actor(s):	Creator
Description:	Creator deletes a class.
Triggering:	Creator decides to remove a class from the platform.
Preconditions :	Admin is the creator of the class.
Main path :	1. Creator accesses the class. 2. Creator navigates to class settings. 3. Creator selects the option to delete the class. 4. Creator confirms the deletion action.
Alter Path :	If the user is not the creator, display "Only Creator can delete the class" message.
Postcondition:	Class is successfully deleted from the platform.
Priority:	High

case name:	Download Materials
case ID:	Op14
Actor(s):	User
Description:	User downloads materials
Triggering:	User decided to download materials from the platform to use it.
Preconditions :	User is logged in and joined to the class
Main path :	1. User accesses the class. 2. User navigates to the materials section. 3. User selects materials and click “Download”.
Postcondition:	Materials downloaded successfully.
Priority:	Low

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.

Schema


User
Username | first name | last name | Email | Image | University | Faculty | Password | Registration Date | Department
User_Course
Username | Course ID | Role
Course
ID | Name | Image | Logo | Description | Type
Announcement
ID | Name | Creation Date | Description | Course ID
Tasks
ID | Deadline | Source | Description | Creation Date | Name | Course ID
Topics
ID | Description | Name | Image | Course ID
Materials
ID | Description | Name | Source | Type | Topics ID
User Notification
User ID | Notification ID | Is Seen
Notifications
ID | Description | Creation Date