प्रगत संगणन विकास केंद्र
CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING

अॅक्टस
acts

MyACTS

PG-DAC Final Project

Software Requirements Specification

Group 40 - Sep. 2022 batch
241. Alan Rodas Bonjour
242. Sajjan Bhurtyal
243. Grmatsion Kidu Desta
244. Eman Aeroota
245. Fatemah Kheder

1. INTRODUCTION

1.1 PURPOSE OF THIS DOCUMENT

This document describes the MyACTS system requirements and specifications. The system is aimed at ease the life of students and teachers providing a centralized page for most things related to CDAC-ACTS.

1.2 INTENDED AUDIENCE

This document is intended to be read by students and faculty members in the context of the final project of the PG-DAC course of batch September 2022. The document has no significant meaning outside of the college premises.

1.3 PROJECT VISION AND SCOPE

MyACTS purpose is intended as a front page for all students, faculty and administrative personnel. The page will provide access to different “services” (links to third party services and systems, that may or not connect to MyACTS), and a messages and events board. Users of the system will have roles assigned to them, and permissions according to their role. Applications may also connect to the system, having permissions according to the application role, and performing different actions through a JSON API.

The system is intended to be used, at most, inside the CDAC-ACTS premises, as a beta project, as there will be no time to perform production level testing and bug-fixing.

The system may be limited in functionality regarding real usage in the institution, but we aim to provide a basic platform that may be further expanded by other ACTS students, providing more functionality.

1.4 DOCUMENT CONVENTIONS AND NOMENCLATURE

This document uses the following conventions and key names:

MyACTS	The whole application system, including all of its features, modules and related data.
DB	Database (A relational database that holds all the data the application uses and manages)
API / JSON API	A part of the system that allows third party applications to connect to the same. This part of the system provides an HTTP JSON interface for third parties to connect and interact with the system.
Service	A thor party application that may be linked by MyACTS. This application may connect or not with MyACTS through the API.
User	A user of the application. Such a user may be any person or application related to ACTS such as a student, faculty and the administrative personnel. Also third party applications that use the API are considered users.
Permission	A permission is an action inside the system that may or not be performed by a particular user. Permissions include things such as sending a message to another user or a set of users, creating an event, creating users and others.
Role	A role consists of a set of users. A role may have permissions such that any user in that role may inherit such permissions. In such a way, we say that a user has one or more roles that allow him to perform different actions in the system.

2. OVERALL DESCRIPTION

2.1 PRODUCT NEEDS

As students of the CDAC-ACTS community we have detected that communication between different entities of the organization is faulty and error prone. Faculty members have to give their personal contact information to students in order to have a communication channel, but are sometimes not willing or thrilled to do so; students lack channels to communicate to each other officially, except for whatsapp; administration personnel do not have many channels to address students in a clear separate manner or individually, event though some messages are required to be addressed privately; things such as grades, attendance and others need to be published in a bulletin board instead of being able to send them directly to a particular student; schedule need to be published in a bulletin board and sudden changes cannot be addressed or notified properly; Many systems exists that provide different functionalities required by the institution, each own with it's own login credentials and url to be remembered. Additionally, inquiries with different actors of the CDAC-ACTS community have revealed the same flaws happening in other areas that do not involve students.

MyACTS attempts to be a solution to communication of different actors in the campus, allowing for more agile and direct communication. The system will not solve the problem on its own, but will provide a platform for communication and third party application aggregation, that hopes to provide an initial solution to such a problem.

2.2 PRODUCT PERSPECTIVE AND GENERAL FEATURES

MyACTS system may be divided into three modules or sections that cooperate to provide a full, unified experience. Such sections include

- **User Dashboard:**
  The user dashboard is the heart of MyACTS. It provides a simple page that allows users to access all the services registered in the application, as well as viewing the messages and events intended for them. The page is interactive, allowing the user to perform

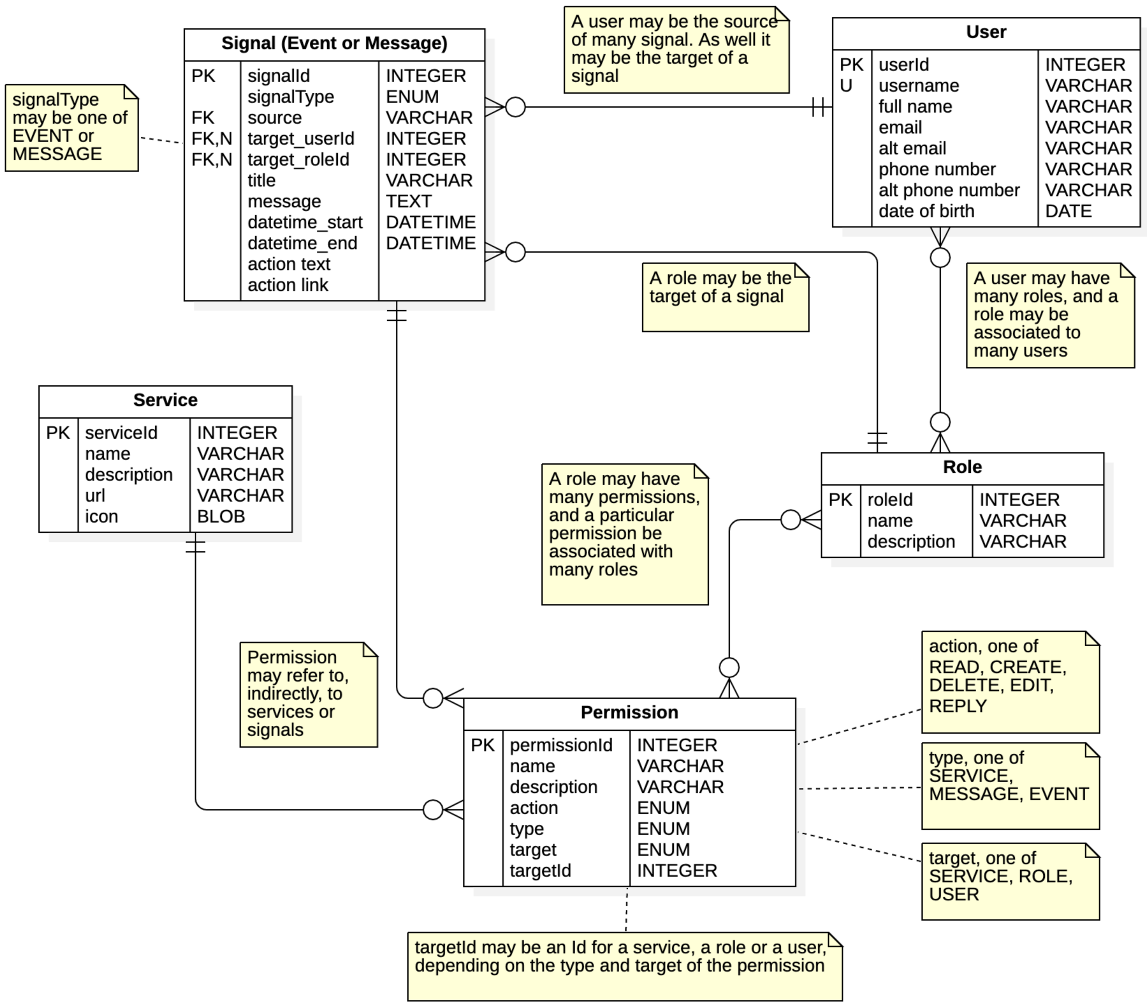
some minimal and simple actions, such as replying to a message, mark it as readed, or other associated actions.

- **Admin panel:**
  The admin panel allows a specialized user, referred as the “admin” (all users with a particular role named as “systemadmin”) to perform configuration actions in the system, such as adding new services, users, roles, and giving permissions to the different users.
- **API:**
  The API is intended for applications that want to connect to the MyACTS system to perform different actions, such as sending a message to a particular user or group. Most importantly, the app will provide a way to authenticate a user, so that any third party application may rely on a single sign-on authority.

All the modules will work with the same database and set of data, thus interacting with each other through the data. Moreover, all parts of the system will be developed under the same project, using the same entities as a base concept.

2.3 PRODUCT ENTITIES AND RELATIONSHIPS

The following **entity–relationship diagram** shows entities involved in the system and their interrelationships. It gives a general idea of how the different entities are connected to each other.


Signal (Event or Message)
PK signalId INTEGER
signalType ENUM
FK source VARCHAR
FK,N target_userId INTEGER
FK,N target_roleId INTEGER
title VARCHAR
message TEXT
datetime_start DATETIME
datetime_end DATETIME
action text
action link
signalType may be one of EVENT or MESSAGE
A user may be the source of many signal. As well it may be the target of a signal
User
PK userId INTEGER
U username VARCHAR
full name VARCHAR
email VARCHAR
alt email VARCHAR
phone number VARCHAR
alt phone number VARCHAR
date of birth DATE
A role may be the target of a signal
A user may have many roles, and a role may be associated to many users
Service
PK serviceId INTEGER
name VARCHAR
description VARCHAR
url VARCHAR
icon BLOB
A role may have many permissions, and a particular permission be associated with many roles
Role
PK roleId INTEGER
name VARCHAR
description VARCHAR
Permission may refer to, indirectly, to services or signals
action, one of READ, CREATE, DELETE, EDIT, REPLY
Permission
PK permissionId INTEGER
name VARCHAR
description VARCHAR
action ENUM
type ENUM
target ENUM
targetId INTEGER
type, one of SERVICE, MESSAGE, EVENT
target, one of SERVICE, ROLE, USER
targetId may be an Id for a service, a role or a user, depending on the type and target of the permission

2.4 ACTORS AND THEIR INTERACTIONS

We distinguish three types of actors that interact with the system, a ***user***, a ***sysadmin*** and an ***application***.

The ***user*** actor refers to regular users of the application. These refer to physical people that are related to CDAC-ACTS. This includes, but is not limited to, students, faculty members, and administration personnel.

This user may perform the following operations:

- Login into the system
- Once logged in, view their dashboard
- Access any services on which they have permissions, depending on their roles.
- View any messages addressed to them or any of their roles.
- View any event addressed to them or any of their roles.
- Perform actions on any message or event addressed to them, if they have the permissions to do so.

The ***sysadmin*** is an actor that acts as the system administrator for the application (although there may be many system administrator users). This entity is a physical person in CDAC-ACTS that is in charge of maintaining the system. Such actors have access to a system administration panel, and can perform the following actions.

- Login into the system
- View the system administration panel with all their sections:
  - Users:
    - List all users in the system
    - Create a new user in the system
    - Edit a user in the system
    - Delete a user in the system
    - Associate a user to one or more roles
    - View all services associated to a user
  - Roles:
    - List all roles in the system

        - Create a new role in the system
        - Edit a role in the system
        - Delete a role in the system
        - Delete a role and all their associated users from the system
        - Add permissions to a role
        - Remove permissions to a role
        - View all services associated to a role
    - Permissions
        - List all permissions in the system (not service related)
    - Services
        - List all services in the system
        - Create a new service in the system
        - Edit a service in the system
        - Delete a service from the system
        - Allow a role to access a particular service
        - Allow a user to access a particular service
    - Statistics
        - List reports of the number of users, divided by role
        - List reports of messages sent by different users

An ***application*** is an actor that consists of a third party application. Such applications may connect to the system (current scope will only provide user/password authentication, but public/private key authentication may be provided in the future) and perform actions onto the system. The following actions may be provided:

- Enquire if a user token is valid (that is, if authenticate a user through the application)
- Send messages to a particular user or role (according to permissions)
- Send events to a particular user or role (according to permissions)
- Edit a particular message or event previously sent (according to permissions)
- Delete a particular message or event previously sent (according to permissions)
- List all previously sent messages
- List all previously sent event
- Receive replies to messages from other users (according to permissions)

2.5 OPERATING ENVIRONMENT

Operating environment for MyACTS requires the following

- Operating system: Windows.
- database: MySQL
- Server system on IIS
- Client with modern web browser

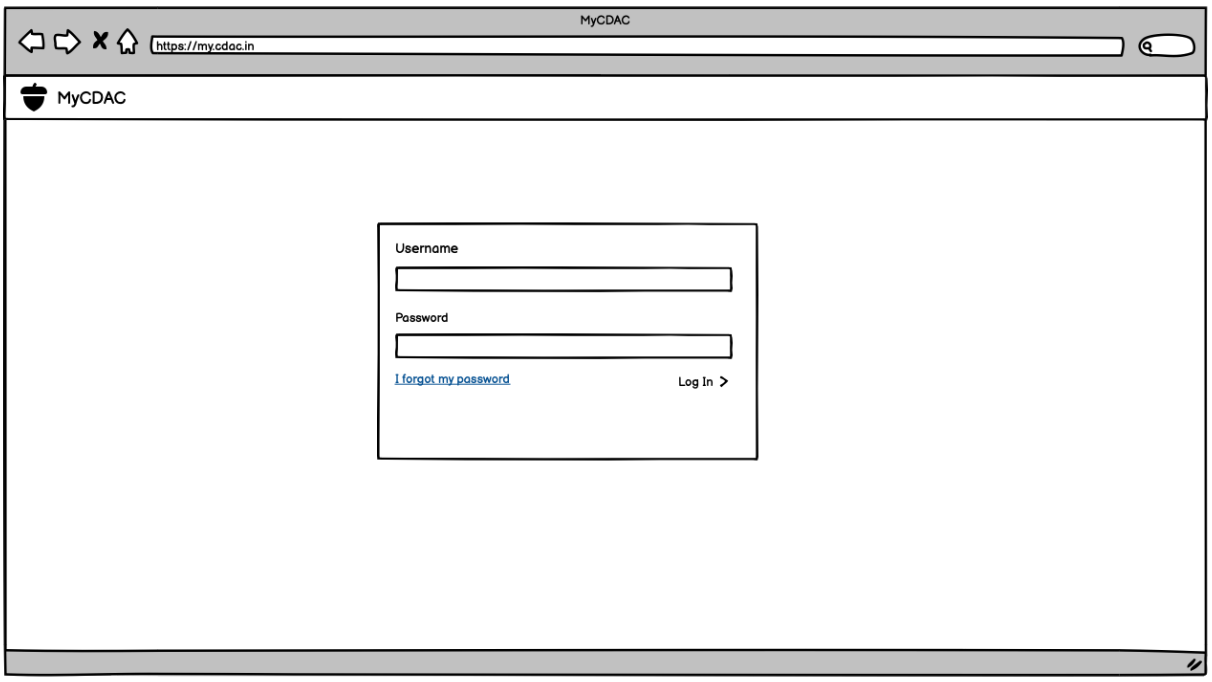
3. FUNCTIONAL REQUIREMENTS

In this section we will provide some mockups of the system, use cases and diagrams specifying a more in depth overview of the system.

3.1 USER REQUIREMENTS

3.1.1 USER LOGIN

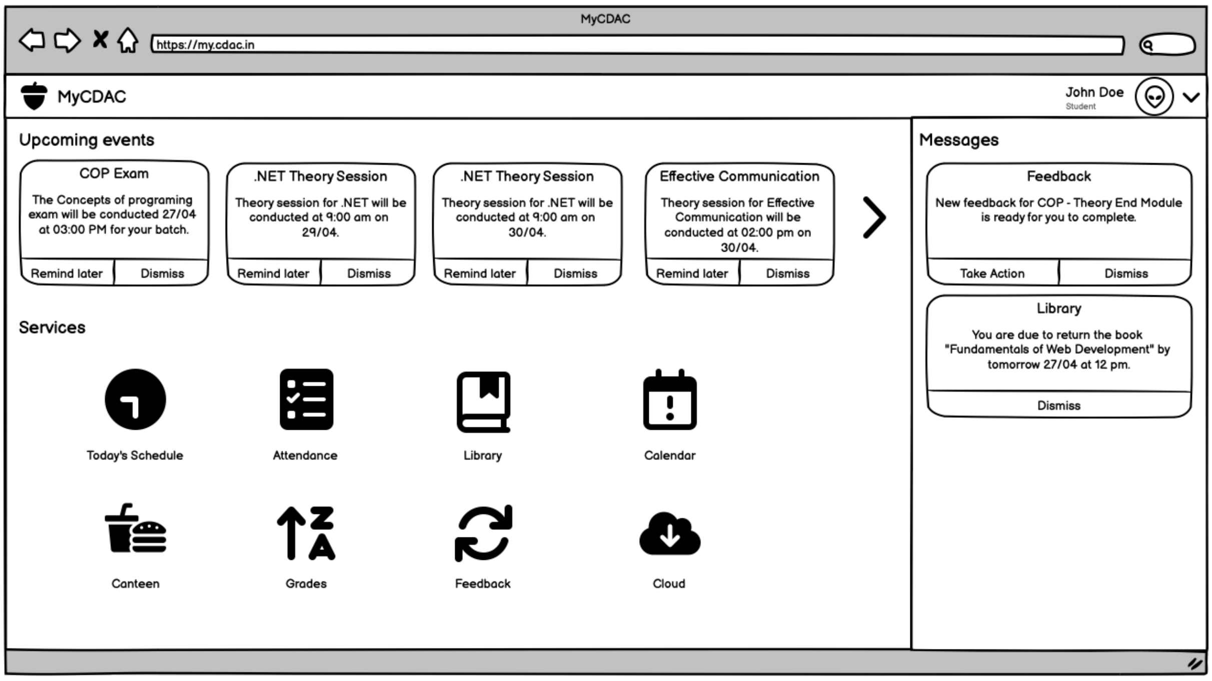
Given I'm a user of the application, when I first reach the application then I should be presented with a login screen that asks for my username and password.



Mockup of login screen

Scenario 1:

If I introduce my username and password correctly (according to the values stored by the application), I should be presented with the dashboard screen.



COP Exam
The Concepts of programing exam will be conducted 27/04 at 03:00 PM for your batch.
Remind later | Dismiss

.NET Theory Session
Theory session for .NET will be conducted at 9:00 am on 29/04.
Remind later | Dismiss

.NET Theory Session
Theory session for .NET will be conducted at 9:00 am on 30/04.
Remind later | Dismiss

Effective Communication
Theory session for Effective Communication will be conducted at 02:00 pm on 30/04.
Remind later | Dismiss

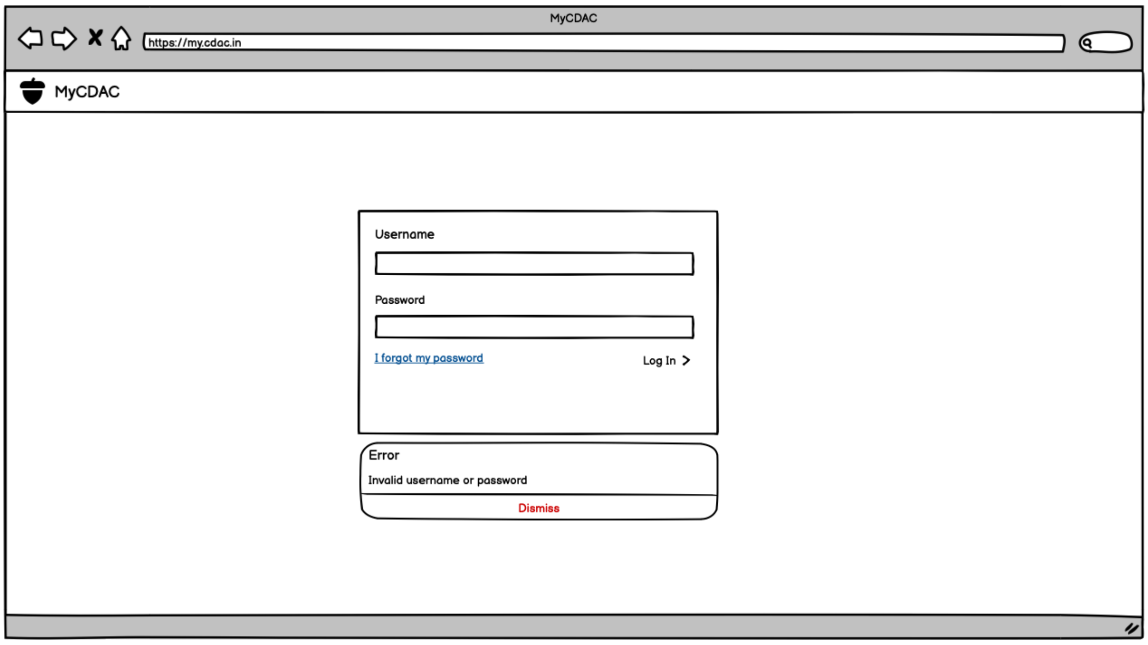
Feedback
New feedback for COP - Theory End Module is ready for you to complete.
Take Action | Dismiss

Library
You are due to return the book "Fundamentals of Web Development" by tomorrow 27/04 at 12 pm.
Dismiss

Mockup of dashboard screen

<u>Scenario 2:</u>

If either my username or password are incorrect, the same login screen with an error message stating “Invalid username or password” should be displayed.

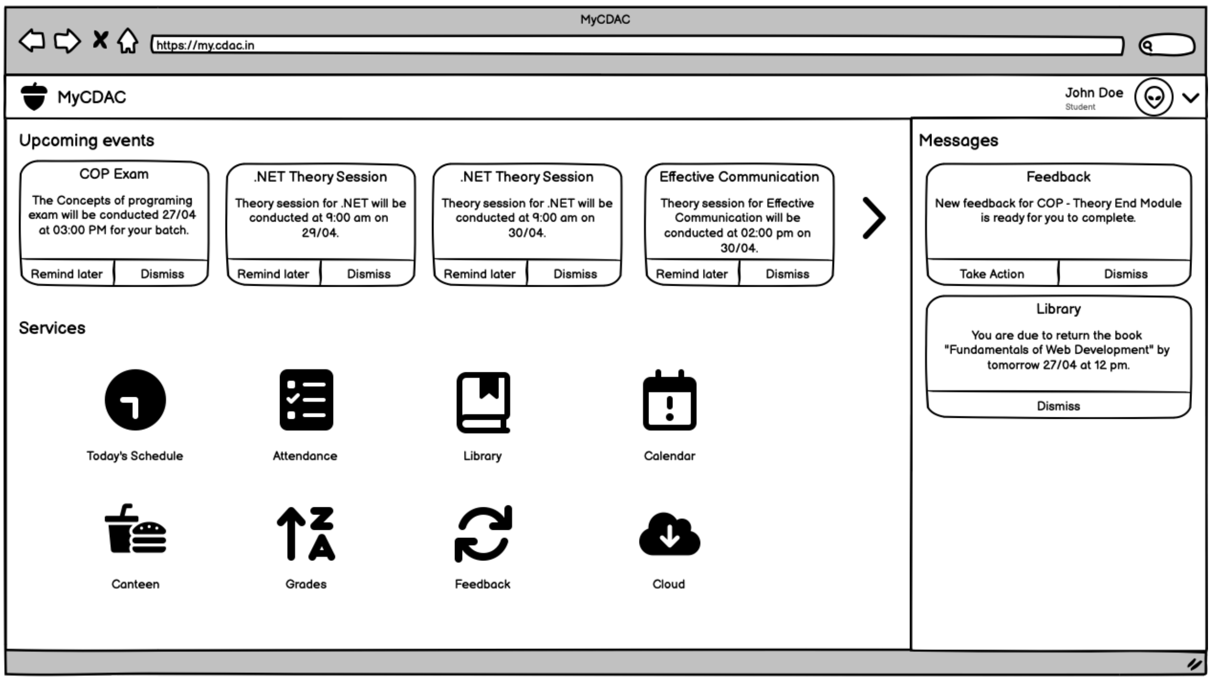


Error
Invalid username or password
Dismiss

Mockup of login screen with error message

3.1.2 DASHBOARD SCREEN

Given I'm a user of the application and I have logged in to the application, then I should be presented with the dashboard screen.



COP Exam
The Concepts of programing exam will be conducted 27/04 at 03:00 PM for your batch.
Remind later | Dismiss

.NET Theory Session
Theory session for .NET will be conducted at 9:00 am on 29/04.
Remind later | Dismiss

.NET Theory Session
Theory session for .NET will be conducted at 9:00 am on 30/04.
Remind later | Dismiss

Effective Communication
Theory session for Effective Communication will be conducted at 02:00 pm on 30/04.
Remind later | Dismiss

Feedback
New feedback for COP - Theory End Module is ready for you to complete.
Take Action | Dismiss

Library
You are due to return the book "Fundamentals of Web Development" by tomorrow 27/04 at 12 pm.
Dismiss

Mockup of dashboard screen

Then, the dashboard screen should:

- Present in the list referred to as "messages" all messages addressed to me as a user.
- Present in the list referred to as "messages" all messages address at any one of my roles.
- Regardless of if the message is directed to the user or some of his roles, all messages are displayed equally in the same list.
- Each message in the "messages" list should display the title of the message, the message itself, and the dismiss button.

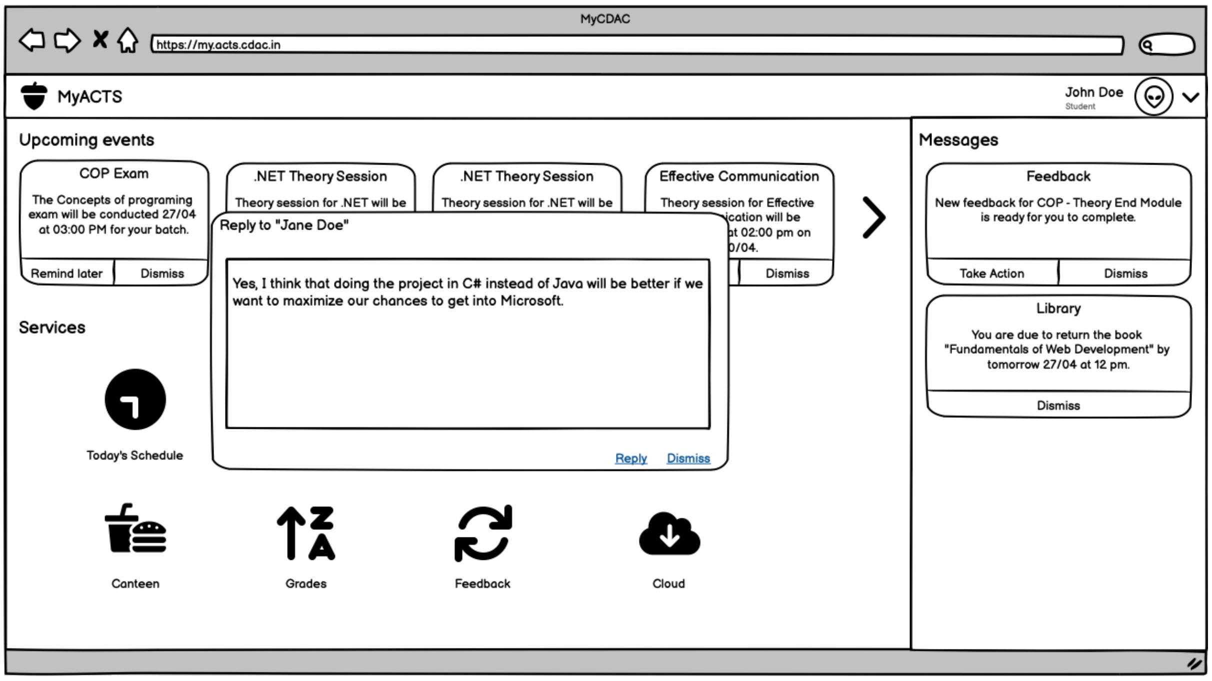
- If the user has the reply message permission, and the message is replyable, then a reply button should appear.
- Messages that have a particular associated action should present an additional button with the action name, and that just links to a webpage or application.
- Present in the list referred to as “events” all events addressed to me as a user.
- Present in the list referred to as “events” all events addressed to any one of my roles.
- Each event in the “events” list should present a button referred to as “dismiss”.
- Each event in the “events” list should present a button referred to as “remind me later”.
- A list of services identified as “services” should be presented to the user. For a service to be included in the list, any of the roles of the user should have read access to the service.
- When a user clicks on a service, a new tab/windows on the browser should open, pointing to the clicked service webpage.
- On the top, a title bar with the CDAC-ACTS logo on one side and the user name and avatar should be displayed.
- Upon clicking on the username or avatar, a popup window with options “Edit my profile”, “Change password” and “Log out” should appear.

<u>Scenario 1: Dismissing a message</u>

On the dashboard a user should be able to click on the “dismiss” button of a message. On clicking on the dismiss button, the message should be marked as read, thus never again be presented to the user in subsequent accesses to the dashboard. Moreover, the message in question should disappear instantly from the list of messages.

<u>Scenario 2: Replying to a message</u>

On the dashboard a user should be able to click on the “reply” button of a message. The reply button should only be presented to a user that has sufficient permissions as to reply messages, and, if the message has been marked as replyable by the sender. On clicking on the reply button, a textbox should be presented as a popup to the user, allowing him to enter the text of the reply. The popup should contain two buttons, the “dismiss” button (to cancel the action) and the “reply” button (to effectively send the reply).


MyCDAC
https://my.acts.cdac.in
MyACTS
John Doe
Student
Upcoming events
COP Exam
The Concepts of programing exam will be conducted 27/04 at 03:00 PM for your batch.
Remind later | Dismiss
.NET Theory Session
Theory session for .NET will be
.NET Theory Session
Theory session for .NET will be
Effective Communication
Theory session for Effective ...ication will be ...at 02:00 pm on ...0/04.
Dismiss
Reply to "Jane Doe"
Yes, I think that doing the project in C# instead of Java will be better if we want to maximize our chances to get into Microsoft.
Reply | Dismiss
Services
Today's Schedule
Canteen
Grades
Feedback
Cloud
Messages
Feedback
New feedback for COP - Theory End Module is ready for you to complete.
Take Action | Dismiss
Library
You are due to return the book "Fundamentals of Web Development" by tomorrow 27/04 at 12 pm.
Dismiss

Mockup of dashboard screen while replying to a message

<u>Scenario 3: Taking action to a message</u>

On some particular messages, the user may be requested to take an action. Actions refer simply to access a third party application. Thus, taking action implies simply following a link.

In the scenario in which a user is presented with a message that has an action, if the user clicks on the “take action” button, then a new window or tab should be open in the browser, whose address would match that of the action of the message.

Scenario 4: Dismissing an event

On the dashboard a user should be able to click on the “dismiss” button of an event. On clicking on the dismiss button, the event should be marked as read, thus never again be presented to the user in subsequent accesses to the dashboard. Moreover, the event in question should disappear instantly from the list of events.

Scenario 5: Remind later of an event

On the dashboard a user should be able to click on the “remind me later” button of an event. On clicking on the remind me later button, the event should be marked as to be reminded later. The event in question should disappear instantly from the list of events for the remaining time of this user session. When the user logs again or a day has passed, then the event should again be presented to the user.

Scenario 6: Accessing a service

On the dashboard a user should be able to click on any of the service icons or names. When clicking on any service, a new window or tab should be opened in the browser, whose address should match that of the service.

3.2 APPLICATION REQUIREMENTS

3.2.1 APPLICATION LOGIN

Given I'm the designer of an application that is going to access MyACTS, I should be able to send a request with the following signature to the application URL to authenticate myself.

```
url: /api/login
method: POST
body:
{
    “username”: “<MyUsername>”
    “password”: “<MyPassword>”
}
```

Where <MyUsername> and <MyPassword> correspond to the actual username or password that I was assigned by a system administrator to access the MyACTS system.

Scenario 1: Username and password are correct

If the given username and password match the one registered in the application MyACTS for a valid username and password, the following response should be returned.

```
status: 200
body:
<Token>
```

Where Token corresponds to a uniquely generated JWT token for this particular session. The token can later be used as an authentication method for subsequent requests on behalf of the application.

Scenario 2: Username and password are incorrect

If the given username and password do not match the one registered in the application MyACTS for a valid username and password, the following response should be returned.

```
status: 401
body:
{
    “status”: 401,
    “type”: “Unauthorized”,
    “message”: “The username and password do not match”
}
```

3.2.2 ACCESSING SERVICES VIA AUTHENTICATION

Given I’m an application actor, and that I have performed the login by using appropriate credentials and got a token, then, all subsequent requests I send should include:

```
header:
authentication: bearer <Token>
```

If the given token is a valid token, and has not expired, then, the appropriate response should be returned, depending on the actual action being performed. In case of invalid token, or a token that has expired, then, the following response should be returned.

```
status: 401
body:
{
    “status”: 401,
    “type”: “Unauthorized”,
    “message”: “The token is invalid or expired”
}
```

3.2.3 SENDING A MESSAGE

Given I’m an application actor, and that I have performed the login by using appropriate credentials and got a token, then, I should be able to send a message to a user or role by sending a request to /api/messages.

Scenario 1: Sending a message to a user

The request to sending a message to a user should have the following signature:

```
url: /api/messages
method: POST
header:
authentication: bearer <Token>
body:
{
    “user”: <UserId>,
    “title”: “<MessageTitle>”,
    “message”: “<MessageBody>”,
    “action_text”: “<MessageActionText>”,
    “action_url”: “<MessageActionURL>”
}
```

Where <UserId> is the Id of a valid user in the system, <MessageTitle> is the title that is going to be used for the message, and <MessageBody> is the text of the actual message. These three elements are not null required elements. <MessageActionText> and <MessageActionURL> are optional, and correspond to a possible additional action to be taken by the user when reading this message.

If the user Id given does not correspond to a valid user of the system, or if only one of the arguments “action_text” and “action_url” is given, then the following response is returned.

```
status: 400
body:
{
    “status”: 400,
    “type”: “Bad Request”,
    “message”: “The message is not properly formed or the receiver
Id is invalid”
}
```

In case the message is properly formed, then the following response should be returned

```
status: 200
body:
{
    “messageId”: <MessageId>,
    “user”: <UserId>,
    “title”: “<MessageTitle>”,
    “message”: “<MessageBody>”,
    “action_text”: “<MessageActionText>”,
    “action_url”: “<MessageActionURL>”
}
```

That is, the same message sent by the user, with the added ID that corresponds to the new message.

Scenario 2: Sending a message to a role

The request to sending a message to a role should have the following signature:

```
url: /api/messages
method: POST
header:
authentication: bearer <Token>
body:
{
    “role”: <RoleId>,
    “title”: “<MessageTitle>”,
    “message”: “<MessageBody>”,
    “action_text”: “<MessageActionText>”,
    “action_url”: “<MessageActionURL>”
}
```

Where <RoleId> is the Id of a valid role in the system, <MessageTitle> is the title that is going to be used for the message, and <MessageBody> is the text of the actual message. These three elements are not null required elements. <MessageActionText> and <MessageActionURL> are optional, and correspond to a possible additional action to be taken by the user when reading this message.

If the role Id given does not correspond to a valid role of the system, or if only one of the arguments “action_text” and “action_url” is given, then the following response is returned.

```
status: 400
body:
{
    “status”: 400,
    “type”: “Bad Request”,
    “message”: “The message is not properly formed or the receiver
Id is invalid”
}
```

In case the message is properly formed, then the following response should be returned

```
status: 200
body:
{
    “messageId”: <MessageId>,
    “role”: <RoleId>,
    “title”: “<MessageTitle>”,
    “message”: “<MessageBody>”,
    “action_text”: “<MessageActionText>”,
    “action_url”: “<MessageActionURL>”
}
```

That is, the same message sent by the user, with the added ID that corresponds to the new message.

Scenario 3: Invalid Requests and insufficient permissions

If the request sended has both a user and a role field in their bodies, then the following response should be sent to the user.

```
status: 400
body:
{
    “status”: 400,
    “type”: “Bad Request”,
    “message”: “The message is not properly formed or the receiver
Id is invalid”
}
```

If the user sending the request does not have enough privileges to send a request to a particular user or role, the following answer should be sent.

```
status: 403
body:
{
    “status”: 403,
    “type”: “Forbidden”,
    “message”: “You do not have permissions to send such a
message”
}
```

3.2.4 CREATING AN EVENT

Given I’m an application actor, and that I have performed the login by using appropriate credentials and got a token, then, I should be able to create an event for users to see, either because I’m targeting such a user, or the role of that user. This can be achieved through a request to /api/events.

Scenario 1: Creating an event for a user

The request to creating an event only for a particular user should have the following signature:

```
url: /api/events
method: POST
header:
authentication: bearer <Token>
body:
{
    “user”: <UserId>,
    “title”: “<EventTitle>”,
    “message”: “<EventBody>”,
    “start”: “<StartDateTimeOfEvent>”,
    “end”: “<EndtDateTimeOfEvent>”
}
```

Where <UserId> is the Id of a valid user in the system, <EventTitle> is the title that is going to be used for the event, <EventBody> is the text of the actual event, <StartDateTimeOfEvent> is a date time element for when the event starts, <EndDateTimeOfEvent> is a date time element for when the event ends. All the elements are not null, required.

If the user Id given does not correspond to a valid user of the system, or any required element is not given, then the following response is expected.

```
status: 400
body:
{
    “status”: 400,
    “type”: “Bad Request”,
    “message”: “The event is not properly formed or the receiver
Id is invalid”
}
```

In case the event is properly formed, then the following response should be returned

```
status: 200
body:
{
    “eventId”: <EventId>,
    “user”: <UserId>,
    “title”: “<EventTitle>”,
    “message”: “<EventBody>”,
    “start”: “<StartDateTimeOfEvent>”,
    “end”: “<EndtDateTimeOfEvent>”
}
```

That is, the same event sent by the user, with the added ID that corresponds to the new event.

<u>Scenario 2: Creating an event for a role</u>

The request to create an event for all users in a role should have the following signature:

```
url: /api/events
method: POST
header:
authentication: bearer <Token>
body:
{
    “user”: <UserId>,
    “title”: “<EventTitle>”,
    “message”: “<EventBody>”,
    “start”: “<StartDateTimeOfEvent>”,
    “end”: “<EndtDateTimeOfEvent>”
}
```

Where <UserId> is the Id of a valid user in the system, <EventTitle> is the title that is going to be used for the event, <EventBody> is the text of the actual event, <StartDateTimeOfEvent> is a date time element for when the event starts, <EndDateTimeOfEvent> is a date time element for when the event ends. All the elements are not null, required.

If the user Id given does not correspond to a valid user of the system, or any required element is not given, then the following response is expected.

```
status: 400
body:
{
    “status”: 400,
    “type”: “Bad Request”,
    “message”: “The event is not properly formed or the receiver
Id is invalid”
}
```

In case the event is properly formed, then the following response should be returned

```
status: 200
body:
{
    “eventId”: <EventId>,
    “role”: <RoleId>,
    “title”: “<EventTitle>”,
    “message”: “<EventBody>”,
    “start”: “<StartDateTimeOfEvent>”,
    “end”: “<EndtDateTimeOfEvent>”
}
```

That is, the same event sent by the user, with the added ID that corresponds to the new event.

Scenario 3: Invalid Requests and insufficient permissions

If the request sended has both a user and a role field in their bodies, then the following response should be sent to the user.

```
status: 400
body:
{
    “status”: 400,
    “type”: “Bad Request”,
    “message”: “The event is not properly formed or the receiver
Id is invalid”
}
```

If the user sending the request does not have enough privileges to send a request to a particular user or role, the following answer should be sent.

```
status: 403
body:
{
    “status”: 403,
    “type”: “Forbidden”,
    “message”: “You do not have permissions to create such an
event”
}
```

3.2.5 READING MESSAGES

Given I’m an application actor, and that I have performed the login by using appropriate credentials and got a token, then, I should be able to read all messages that I have previously sent through the following request.

```
url: /api/messages
method: GET
header:
authentication: bearer <Token>
```

The result should be a list of all messages sent by the user. If the user does not have permissions to read messages then the following should be returned as response.

```
status: 403
body:
{
    “status”: 403,
    “type”: “Forbidden”,
    “message”: “You do not have permissions to read messages”
}
```

3.2.6 READING A PARTICULAR MESSAGE

Given I'm an application actor, and that I have performed the login by using appropriate credentials and got a token, then, I should be able to read all messages that I have previously sent through the following request.

```
url: /api/messages/<Id>
method: GET
header:
authentication: bearer <Token>
```

The result should be a single message sent by the user.

If the message does not exist, or if it's a message sent by a different user, or if the user does not have enough permissions to read messages, then the following should be returned as a response.

```
status: 403
body:
{
    “status”: 403,
    “type”: “Forbidden”,
    “message”: “You do not have permissions to read messages”
}
```

3.2.7 READING EVENTS

Given I'm an application actor, and that I have performed the login by using appropriate credentials and got a token, then, I should be able to read all events that I have previously sent through the following request.

```
url: /api/events
method: GET
header:
authentication: bearer <Token>
```

The result should be a list of all events sent by the user. If the user does not have permissions to read events then the following should be returned as response.

```
status: 403
body:
{
    “status”: 403,
    “type”: “Forbidden”,
    “message”: “You do not have permissions to read messages”
}
```

3.2.8 READING A PARTICULAR EVENT

Given I’m an application actor, and that I have performed the login by using appropriate credentials and got a token, then, I should be able to read all events that I have previously created through the following request.

```
url: /api/messages/<Id>
method: GET
header:
authentication: bearer <Token>
```

The result should be a single event created by the user.

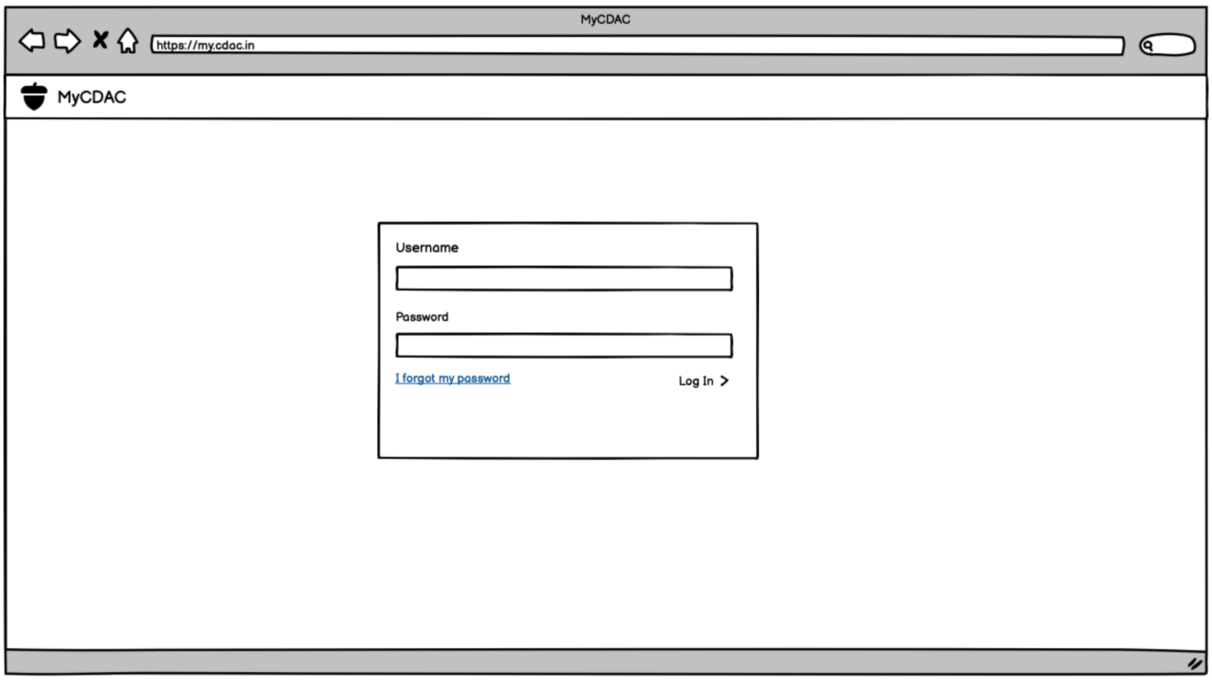
If the event does not exist, or if it’s an event sent by a different user, or if the user does not have enough permissions to read events, then the following should be returned as a response.

```
status: 403
body:
{
    “status”: 403,
    “type”: “Forbidden”,
    “message”: “You do not have permissions to read events”
}
```

3.3 SYSTEM ADMINISTRATOR REQUIREMENTS

3.3.1 APPLICATION LOGIN

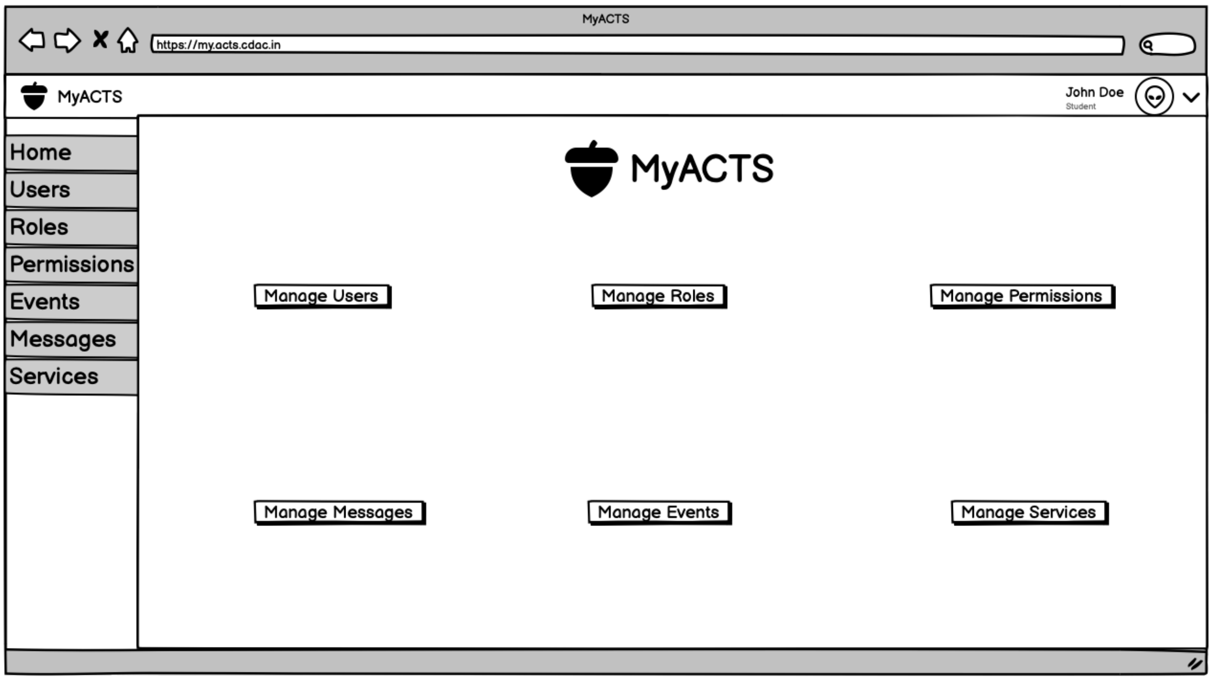
Given I'm a system administrator of the application, when I first reach the application then I should be presented with a login screen that asks for my username and password.



Mockup of login screen

Scenario 1: Login is correct

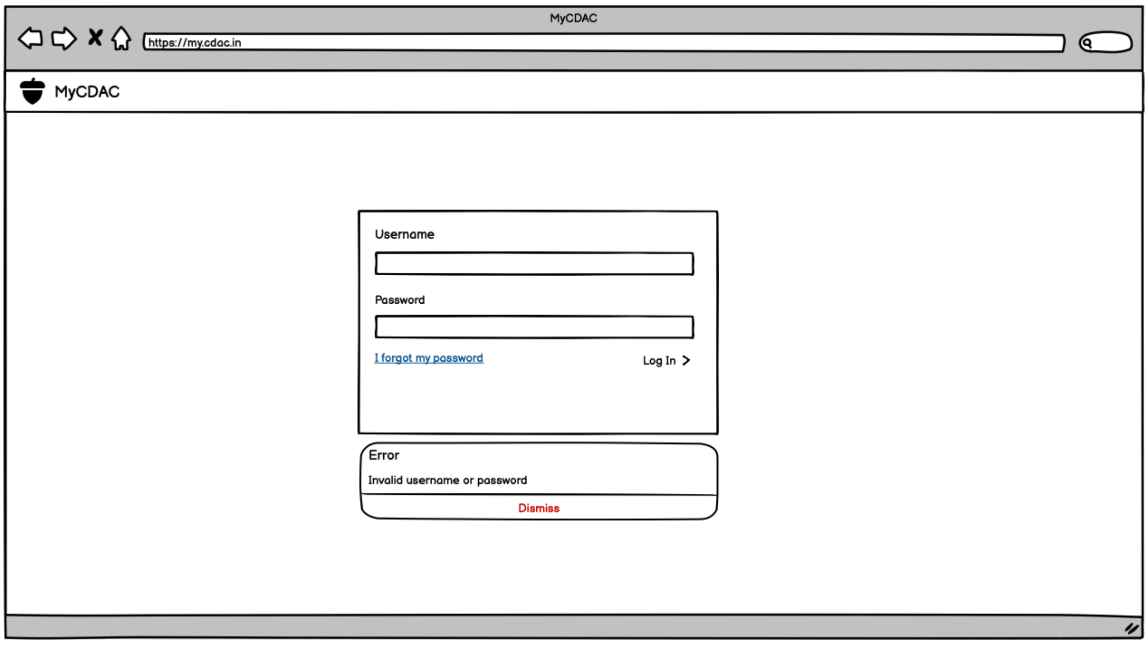
If I introduce my username and password correctly (according to the values stored by the application), I should be presented with the system administration panel screen.



Mockup of system administrator panel

Scenario 2: Login is incorrect

If either my username or password are incorrect, the same login screen with an error message stating "Invalid username or password" should be displayed.

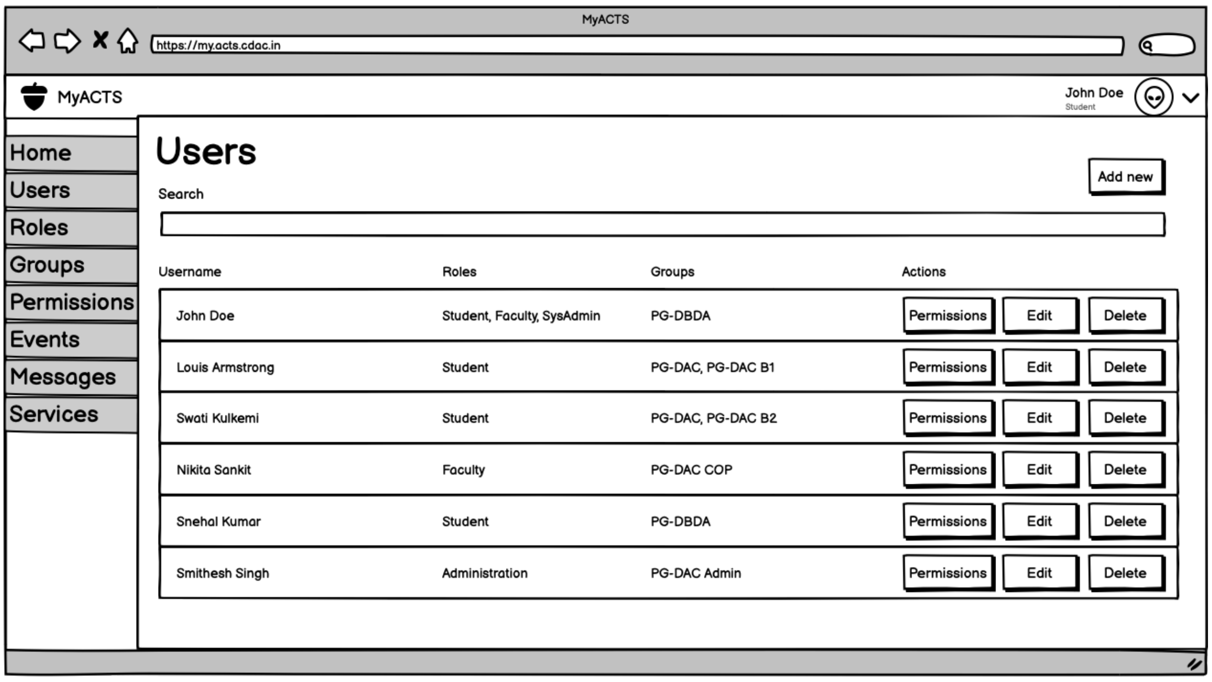


Mockup of login screen with error message

3.3.2 CREATING A NEW USER

Given I'm a system administrator of the application and I have logged to the application, have been presented with the administration panel, I want to be able to create new users in the application.

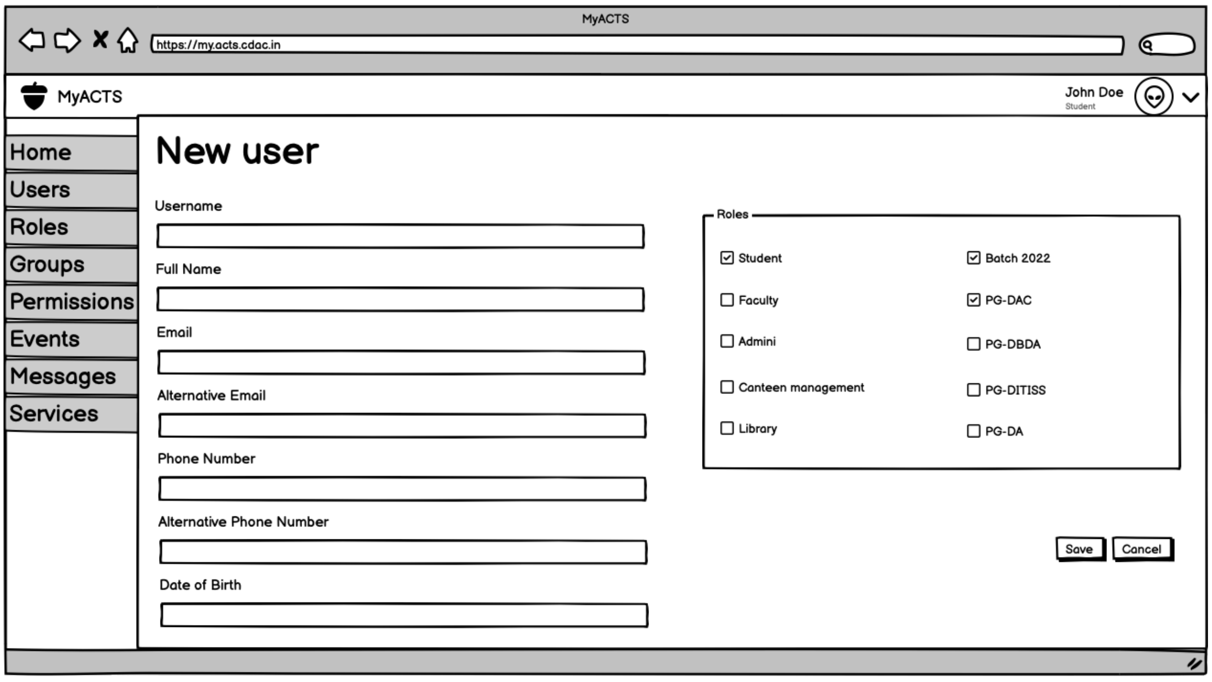
For this, I will go to the “Manage Users” section, from the main actions, or from the sidebar, and be presented with a screen as follows.



Username	Roles	Groups	Actions
John Doe	Student, Faculty, SysAdmin	PG-DBDA	Permissions Edit Delete
Louis Armstrong	Student	PG-DAC, PG-DAC B1	Permissions Edit Delete
Swati Kulkemi	Student	PG-DAC, PG-DAC B2	Permissions Edit Delete
Nikita Sankit	Faculty	PG-DAC COP	Permissions Edit Delete
Snehal Kumar	Student	PG-DBDA	Permissions Edit Delete
Smithesh Singh	Administration	PG-DAC Admin	Permissions Edit Delete

Mockup of system administrator panel for user management

On this screen I will press the Add User button, as to be presented with the create new user screen.



New user

Username

Full Name

Email

Alternative Email

Phone Number

Alternative Phone Number

Date of Birth

Roles

✓ Student	✓ Batch 2022
☐ Faculty	✓ PG-DAC
☐ Admini	☐ PG-DBDA
☐ Canteen management	☐ PG-DITISS
☐ Library	☐ PG-DA

Mockup of system administrator panel when creating a new user

Scenario 1: Cancel button pressed

If the user presses the cancel button, then the list of user is presented, and no new user is saved in the application.

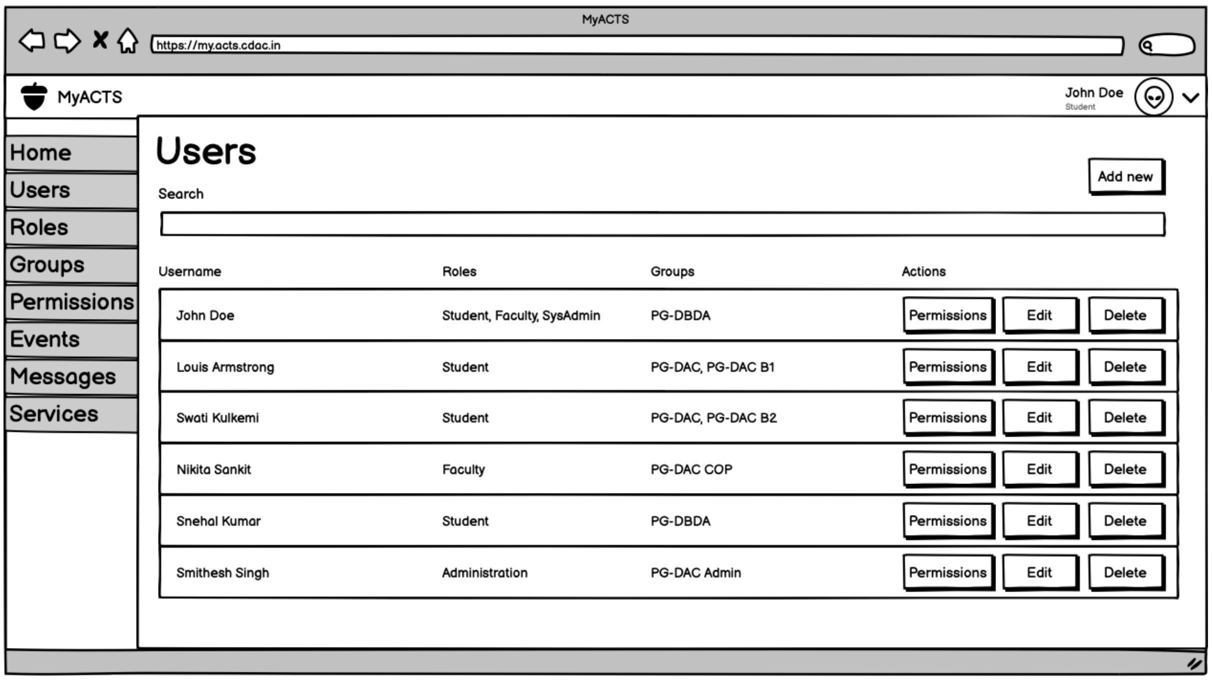
Scenario 2: Save button pressed

If the user presses the save button, and all the required fields are filled (which include all but Alternate Email, Alternate Phone Number and Date of Birth) then the new user is saved in the application. The default password for every user is the same as the username and can only be changed by the user created. Then the list of users is presented, with the information of the new user added.

3.3.3 EDITING A USER

Given I'm a system administrator of the application and I have logged to the application, have been presented with the administration panel, I want to be able to edit new users in the application.

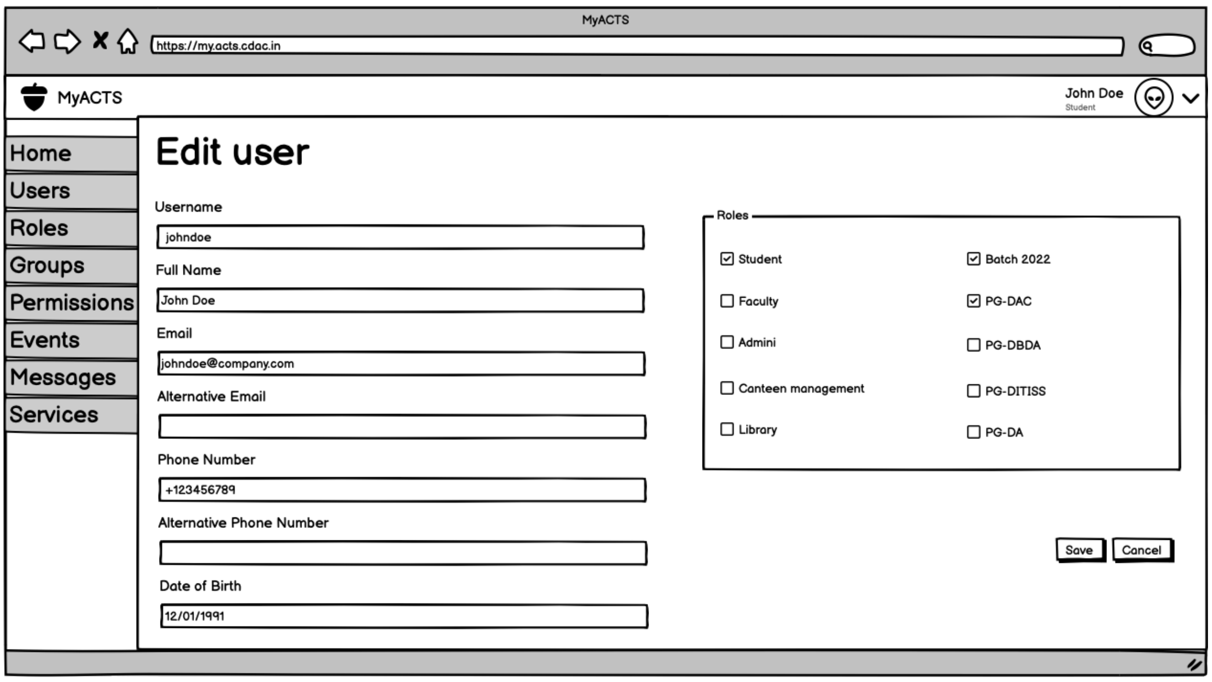
For this, I will go to the “Manage Users” section, from the main actions, or from the sidebar, and be presented with a screen as follows.



Username	Roles	Groups	Actions
John Doe	Student, Faculty, SysAdmin	PG-DBDA	Permissions Edit Delete
Louis Armstrong	Student	PG-DAC, PG-DAC B1	Permissions Edit Delete
Swati Kulkemi	Student	PG-DAC, PG-DAC B2	Permissions Edit Delete
Nikita Sankit	Faculty	PG-DAC COP	Permissions Edit Delete
Snehal Kumar	Student	PG-DBDA	Permissions Edit Delete
Smithesh Singh	Administration	PG-DAC Admin	Permissions Edit Delete

Mockup of system administrator panel for user management

On this screen I will press the Edit button on the row of the user I want to edit, as to be presented with the edit user screen.



Edit user

Username: johndoe
Full Name: John Doe
Email: johndoe@company.com
Alternative Email:
Phone Number: +123456789
Alternative Phone Number:
Date of Birth: 12/01/1991

Roles

☑ Student	☑ Batch 2022
☐ Faculty	☑ PG-DAC
☐ Admini	☐ PG-DBDA
☐ Canteen management	☐ PG-DITISS
☐ Library	☐ PG-DA

Mockup of system administrator panel when editing a user

The field for username is read only field, all other fields are editable.

<u>Scenario 1: Cancel button pressed</u>

If the user presses the cancel button, then the list of user is presented, and the user is not edited.

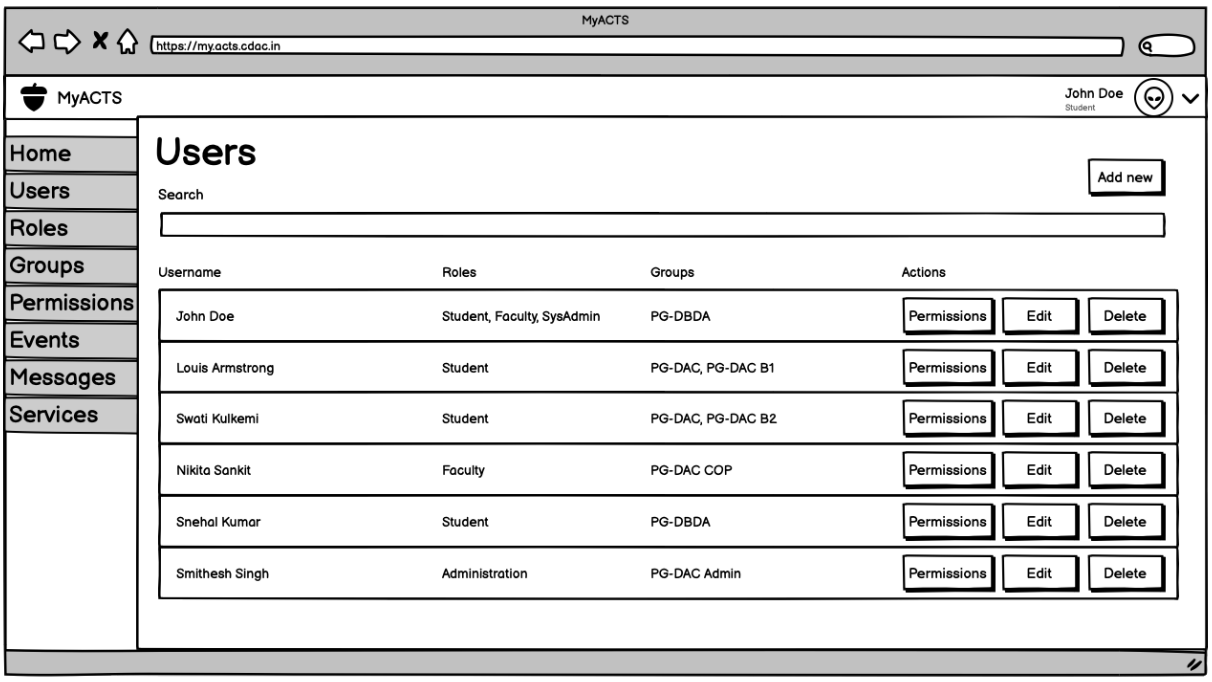
<u>Scenario 2: Save button pressed</u>

If the user presses the save button, and all the required fields are filled (which include all but Alternate Email, Alternate Phone Number and Date of Birth) then the user's new data is saved in the application. The password for the user is not changed. Then the list of users is presented, with the information of the edited user changed.

3.3.4 DELETING A USER

Given I'm a system administrator of the application and I have logged to the application, have been presented with the administration panel, I want to be able to edit new users in the application.

For this, I will go to the “Manage Users” section, from the main actions, or from the sidebar, and be presented with a screen as follows.



Username	Roles	Groups	Actions		
John Doe	Student, Faculty, SysAdmin	PG-DBDA	Permissions	Edit	Delete
Louis Armstrong	Student	PG-DAC, PG-DAC B1	Permissions	Edit	Delete
Swati Kulkemi	Student	PG-DAC, PG-DAC B2	Permissions	Edit	Delete
Nikita Sankit	Faculty	PG-DAC COP	Permissions	Edit	Delete
Snehal Kumar	Student	PG-DBDA	Permissions	Edit	Delete
Smithesh Singh	Administration	PG-DAC Admin	Permissions	Edit	Delete

Mockup of system administrator panel for user management

On this screen I will press the Delete button on the row of the user I want to delete, as to be presented with a confirmation dialog that has the caption “Are you sure you want to delete the user?” and the buttons “Yes” and “No”.

<u>Scenario 1: No button pressed</u>

If the user presses the No button, then the list of user is presented, and the user is not deleted.

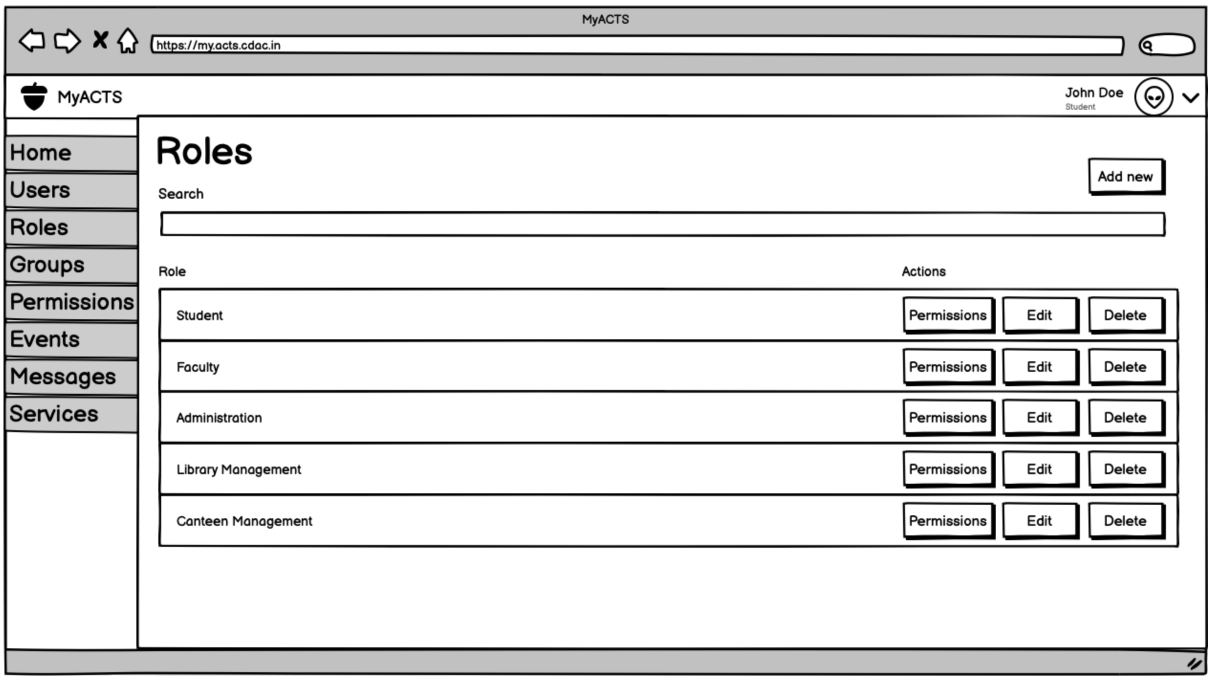
<u>Scenario 2: Yes button pressed</u>

If the user presses the yes button, then the user is deleted from the application. Then the list of users is presented, with the deleted user not shown.

3.3.4 CREATING A NEW ROLE

Given I'm a system administrator of the application and I have logged to the application, have been presented with the administration panel, I want to be able to create new roles in the application.

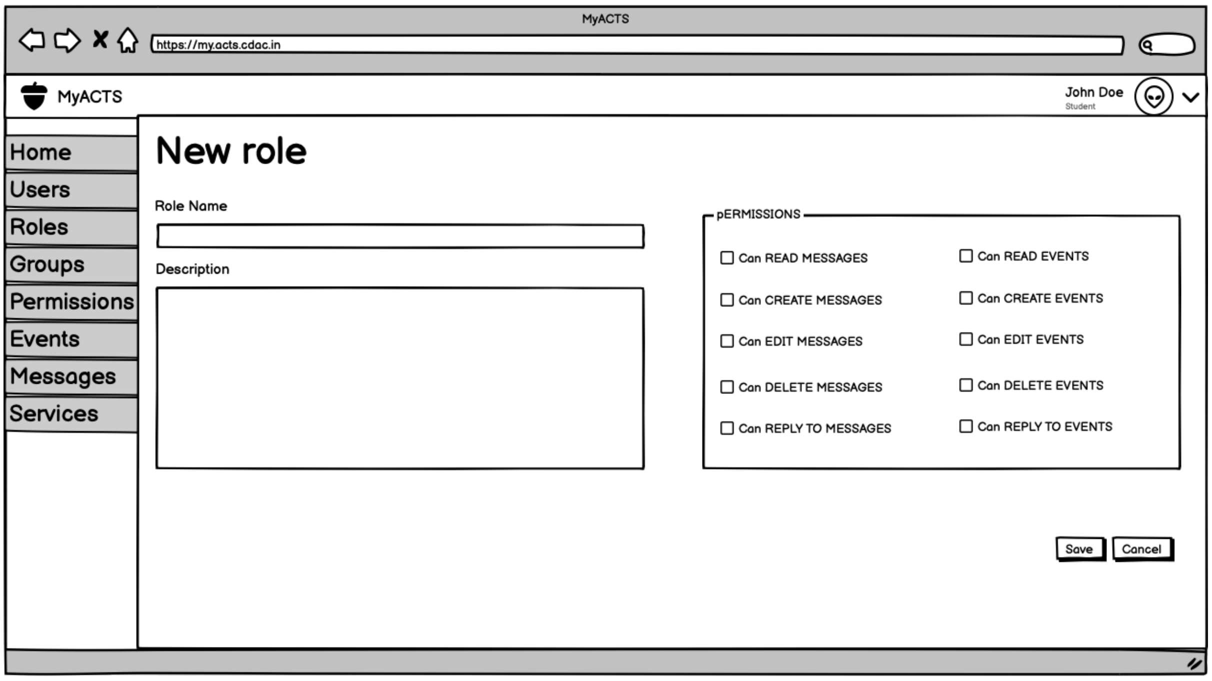
For this, I will go to the “Manage Roles” section, from the main actions, or from the sidebar, and be presented with a screen as follows.



Role	Actions		
Student	Permissions	Edit	Delete
Faculty	Permissions	Edit	Delete
Administration	Permissions	Edit	Delete
Library Management	Permissions	Edit	Delete
Canteen Management	Permissions	Edit	Delete

Mockup of system administrator panel for role management

On this screen I will press the Add New button, as to be presented with the create new role screen.



Mockup of system administrator panel when creating a new role

Scenario 1: Cancel button pressed

If the user presses the cancel button, then the list of roles is presented, and no new role is saved in the application.

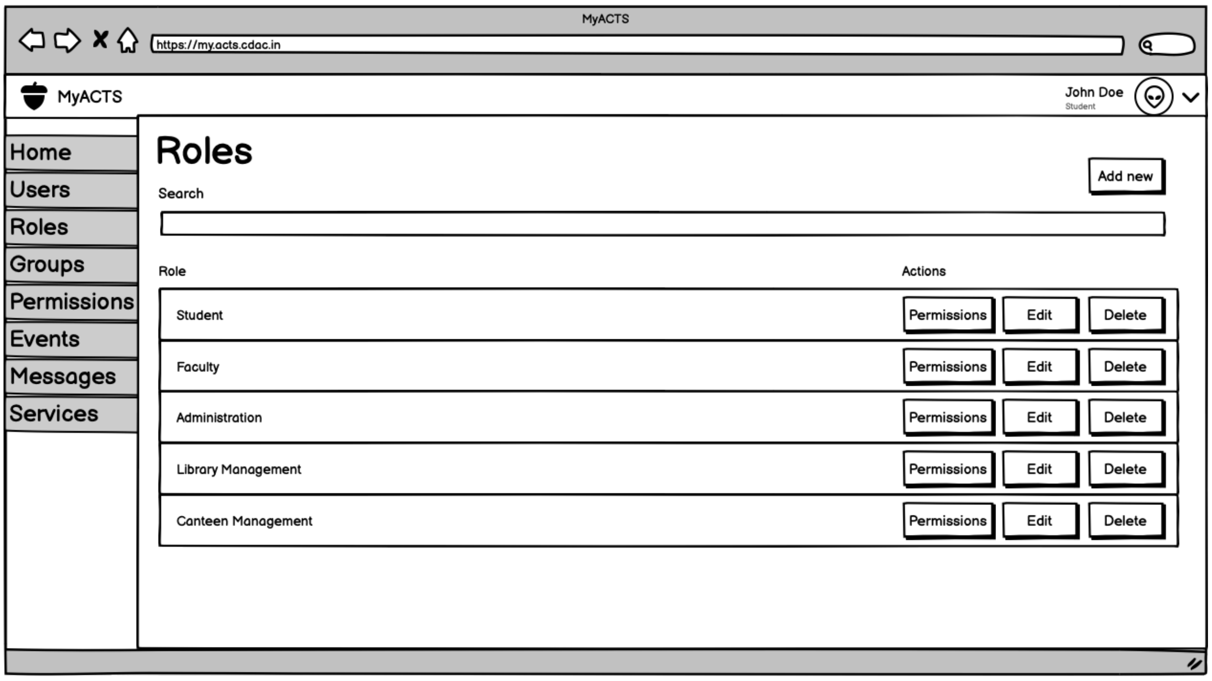
Scenario 2: Save button pressed

If the user presses the save button, and all the required fields are filled (which only include the name) then the new role is saved in the application. Then the list of roles is presented, with the information of the new role added.

3.3.5 EDITING A ROLE

Given I'm a system administrator of the application and I have logged to the application, have been presented with the administration panel, I want to be able to edit roles in the application.

For this, I will go to the “Manage Roles” section, from the main actions, or from the sidebar, and be presented with a screen as follows.


MyACTS

https://my.acts.cdac.in

MyACTS

John Doe
Student

Home
Users
Roles
Groups
Permissions
Events
Messages
Services

Roles

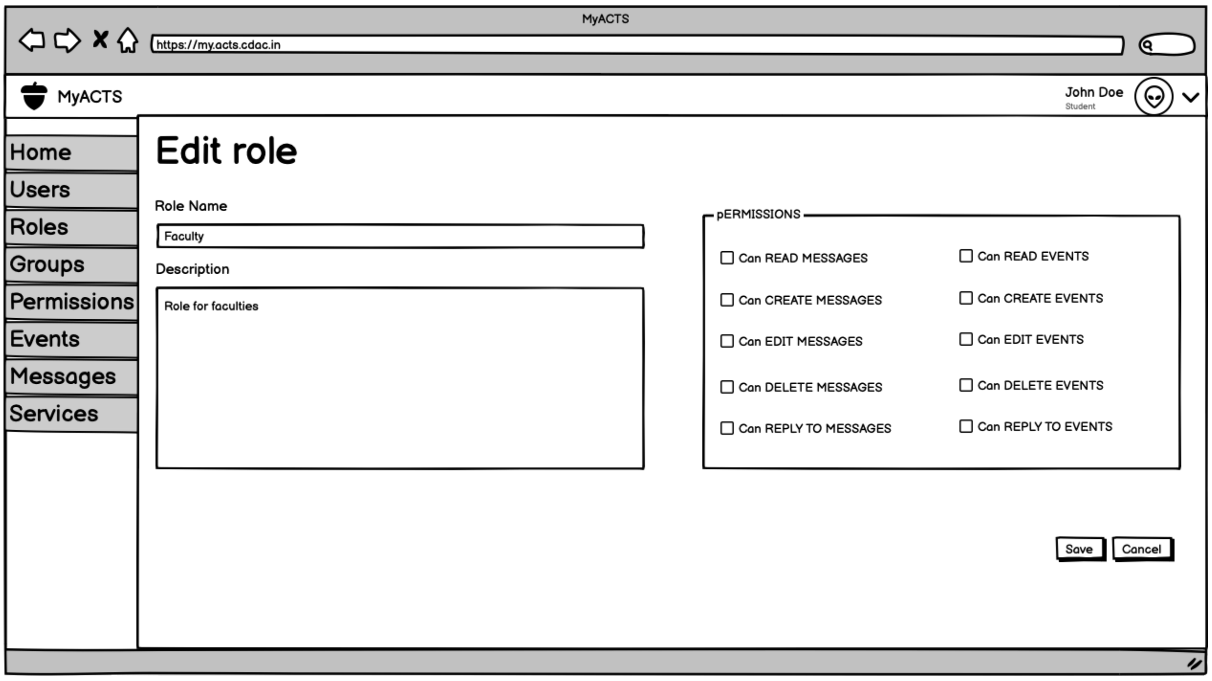
Add new

Search

Role	Actions
Student	Permissions Edit Delete
Faculty	Permissions Edit Delete
Administration	Permissions Edit Delete
Library Management	Permissions Edit Delete
Canteen Management	Permissions Edit Delete

Mockup of system administrator panel for roles management

On this screen I will press the Edit button on the row of the role I want to edit, as to be presented with the edit role screen.


MyACTS

https://my.acts.cdac.in

MyACTS

John Doe
Student

Home
Users
Roles
Groups
Permissions
Events
Messages
Services

Edit role

Role Name: Faculty

Description: Role for faculties

pERMISSIONS

- ☐ Can READ MESSAGES
- ☐ Can CREATE MESSAGES
- ☐ Can EDIT MESSAGES
- ☐ Can DELETE MESSAGES
- ☐ Can REPLY TO MESSAGES
- ☐ Can READ EVENTS
- ☐ Can CREATE EVENTS
- ☐ Can EDIT EVENTS
- ☐ Can DELETE EVENTS
- ☐ Can REPLY TO EVENTS

Save Cancel

Mockup of system administrator panel when editing a role

Scenario 1: Cancel button pressed

If the user presses the cancel button, then the list of roles is presented, and the role is not edited.

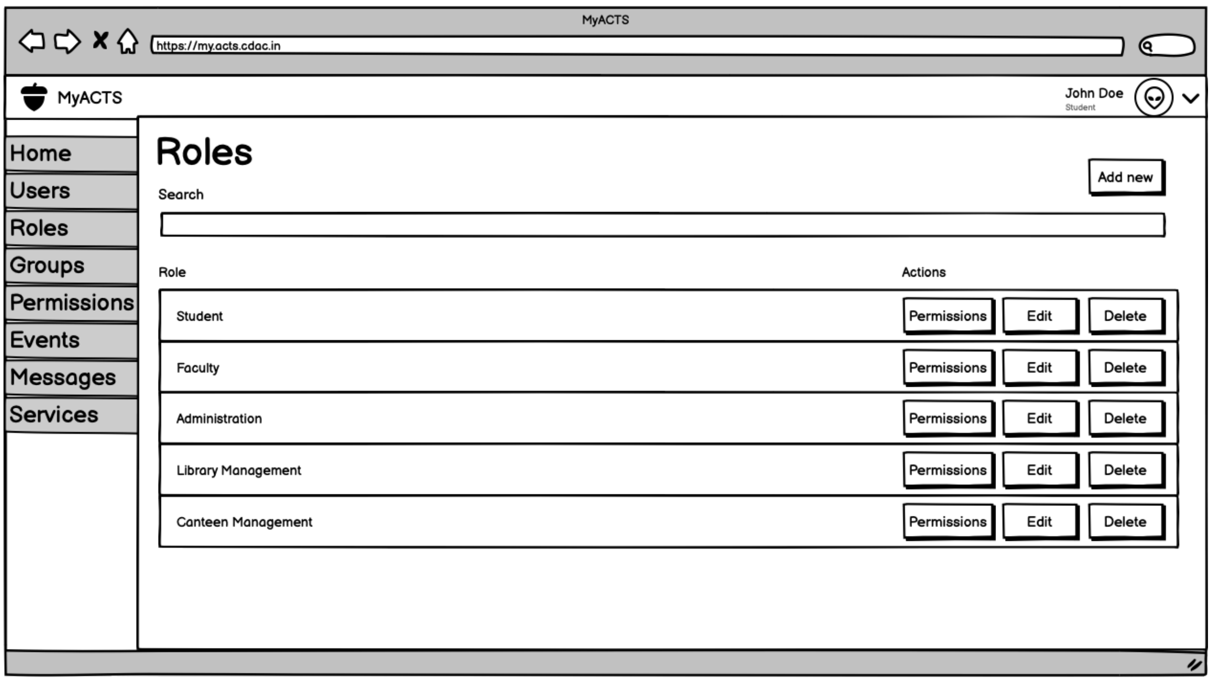
Scenario 2: Save button pressed

If the user presses the save button, and all the required fields are filled (which include only the name) then the role's new data is saved in the application. Then the list of roles is presented, with the information of the edited role changed.

3.3.6 DELETING A ROLE

Given I'm a system administrator of the application and I have logged to the application, have been presented with the administration panel, I want to be able to delete roles in the application.

For this, I will go to the "Manage Roles" section, from the main actions, or from the sidebar, and be presented with a screen as follows.



MyACTS

https://my.acts.cdac.in

MyACTS — John Doe, Student

Home | Users | Roles | Groups | Permissions | Events | Messages | Services

Roles

Search

Add new

Role	Actions		
Student	Permissions	Edit	Delete
Faculty	Permissions	Edit	Delete
Administration	Permissions	Edit	Delete
Library Management	Permissions	Edit	Delete
Canteen Management	Permissions	Edit	Delete

Mockup of system administrator panel for permissions management

On this screen I will press the Delete button on the row of the role I want to delete, as to be presented with a confirmation dialog that has the caption “Are you sure you want to delete the role?” and the buttons “Yes” and “No”.

<u>Scenario 1: No button pressed</u>

If the user presses the No button, then the list of roles is presented, and the role is not deleted.

<u>Scenario 2: Yes button pressed</u>

If the user presses the yes button, then the role is deleted from the application. Then the list of roles is presented, with the deleted role not shown.

4. TECHNOLOGIES AND REQUIREMENTS

4.1 SOFTWARE INTERFACES

- Development languages:
  - .NET Core will provide the platform for the software. It's multi-platform support and it's write once run everywhere approach make it a good option for modern software development.
  - C# is going to be the language used for this project. As a high level language it provides all the features that developers need to create the software in a fast manner, and as a statically typed language, it provides robustness to the application.
  - ASP.MVC will act as the language/framework in which to provide the user interface. This language will allow for fast and secure building of web interfaces, as well as API interfaces with thorough documentation.
  - Entity Framework will act as the framework for the persistence layer. The database independence provided by the tool, and the
- Database technologies:
  - The database will be the relational database MySQL. The simplicity of the database, the multi-platform support of the same make it a great option for development. As Entity Framework will abstract away the underlying database, changing the same on the future should not represent problems if so decided.

4.2 HARDWARE INTERFACES

- A powerful enough server running Windows is to be provided for hosting and running the application. Such server should be connected to the network 24/7 and be hosted in any domain with an SSL certificate.

- A database server with at least 4GB of storage space is to be provided. Such server may be in the machine as the application server.
- Clients will require a modern browser. On this particular development, only latest versions of Chromium will be officially supported.

5. NONFUNCTIONAL REQUIREMENTS

5.1 PERFORMANCE REQUIREMENTS

The system should behave with reasonable performance, providing users a response in a non-blocking time frame. That is, a request via the browser should provide a response in a few milliseconds, thus not making the user wait in blank screens. The same requirement applies for third party application responses. Yet, no specific response time or performance is initially required.

Regarding database management, it’s expected that data is non-redundant, by providing normalized tables. No requirements for auditing or redundancy is considered at this step, so only minimal tables and data ought to be created.

5.2 SAFETY REQUIREMENTS

No database redundancy is considered at this state. Database backups and server maintenance is out of the scope of this system.

5.3 SECURITY REQUIREMENTS

Regarding security, the system needs to support HTTPS transactions through the applications. Sensitive data such as passwords need to be stored in a secure encrypted manner. Only username + password login is intended at this stage. Additional login methods that involve two step authentication or similar security features are not considered at this point.

5.4 SOFTWARE QUALITY ATTRIBUTES

- **AVAILABILITY:** The system should be live, available on a 24/7 basis, without downtime for maintenance or additional tasks.
- **CORRECTNESS:** The messages and events intended for a particular user should only be visible for such a user. No action should be performed by a user if it doesn't have the right permissions to perform it.
- **MAINTAINABILITY:** The system will not have support from the makers. All the support is intended to be perform by the system administrators and devops teams in CDAC, while hosting this project.
- **USABILITY:** The number of concurrent users for the system should match the number of CDAC-ACTS members at a particular point in time. Even though the hardware places an important role, the system should scale properly to support the number of tens of thousands of concurrent users.