Indian Institute of Information Technology, Allahabad
Software Engineering

Instructors: Dr. Sonali Agarwal

SOFTWARE DESIGN SPECIFICATION

FOR

Impedia

A Centralized Appeal Management System

Group Members:

Shreyas Gupta	IIT2019102	iit2019102@iiita.ac.in
Harshdeep Singh Pruthi	IIT2019105	iit2019105@iiita.ac.in
Sarthak Maheshwari	IIT2019117	iit2019117@iiita.ac.in
Garvit Chittora	IIT2019142	iit2019142@iiita.ac.in

Index

Section 1: Introduction

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, collaboration models, object behaviour models, and other supporting requirement information.

Purpose

This document will define the design of ‘Impedia’. It contains detailed figures highlighting the design and flow of the platform for all types of users.

Scope

Here we will describe what features are in the scope of this product and what features are not in the scope.

In Scope:

- Flexible structure so any organization can host their separate version and their admin can add their respective authorities, faculty groups and specify their email domain so students under that domain can join the platform.
- Suggest appropriate authorities or authority groups to students based on their type of appeal and their semester/section.
- Let students make an appeal or send a message to the appropriate authority within the platform.
- Let authorities reply and resolve the appeal within the platform.
- Notify authorities or students about unread appeals/replies through email if enabled.
- Let students create common issues/appeals in the form of petitions and tag concerned authorities, which other students can view and sign, to support the issue/appeal.
- Let concerned authorities view the petition and reply to all the supporting students accordingly.

Out of Scope:

- Conversations initiated by faculties towards students.

- Creation of student groups.

Definitions, Acronyms, and Abbreviations

Abbreviations:

a. SDS: Software Design Specification

Definitions:

a. Authorities: Any faculty member, or any person holding any responsibility or taking decisions in the working of IIITA.
b. Admin: Any person in charge of handling and overseeing the appeal management system.

References

IEEE SDS Format
R. S. Pressman, Software Engineering: A Practitioner's Approach, 5th Ed, McGraw-Hill, 2001.

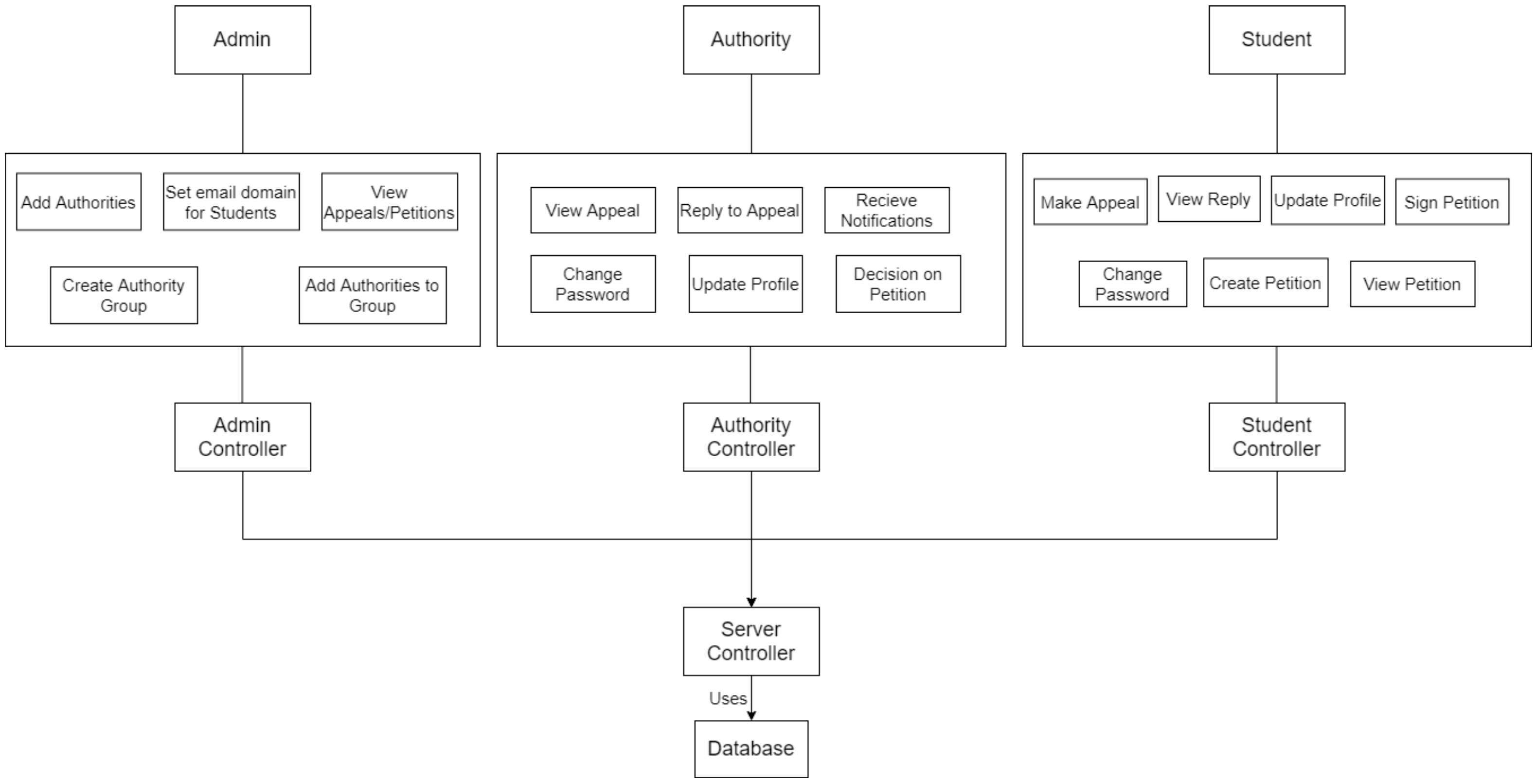
Overview

This SDS is divided into seven sections with various sub-sections. The sections of the Software Design Document are:

1. **Introduction:** describes about the document, purpose, scope of development project definitions and abbreviations used in the document.
2. **Conceptual Architecture/Architecture Diagram:** describes the overview of components, modules, structure and relationships and user interface issues.
3. **Logical Architecture:** describes Logical Architecture Description and Components.
4. **Execution Architecture:** defines the runtime environment, processes, deployment view.
5. **Design Decisions and Trade-offs:** describes the decisions taken along with the reason as to why they were chosen over other alternatives.
6. **Pseudocode for components:** describes pseudocode, as the name indicates.
7. **Appendices:** describes subsidiary matter if any.

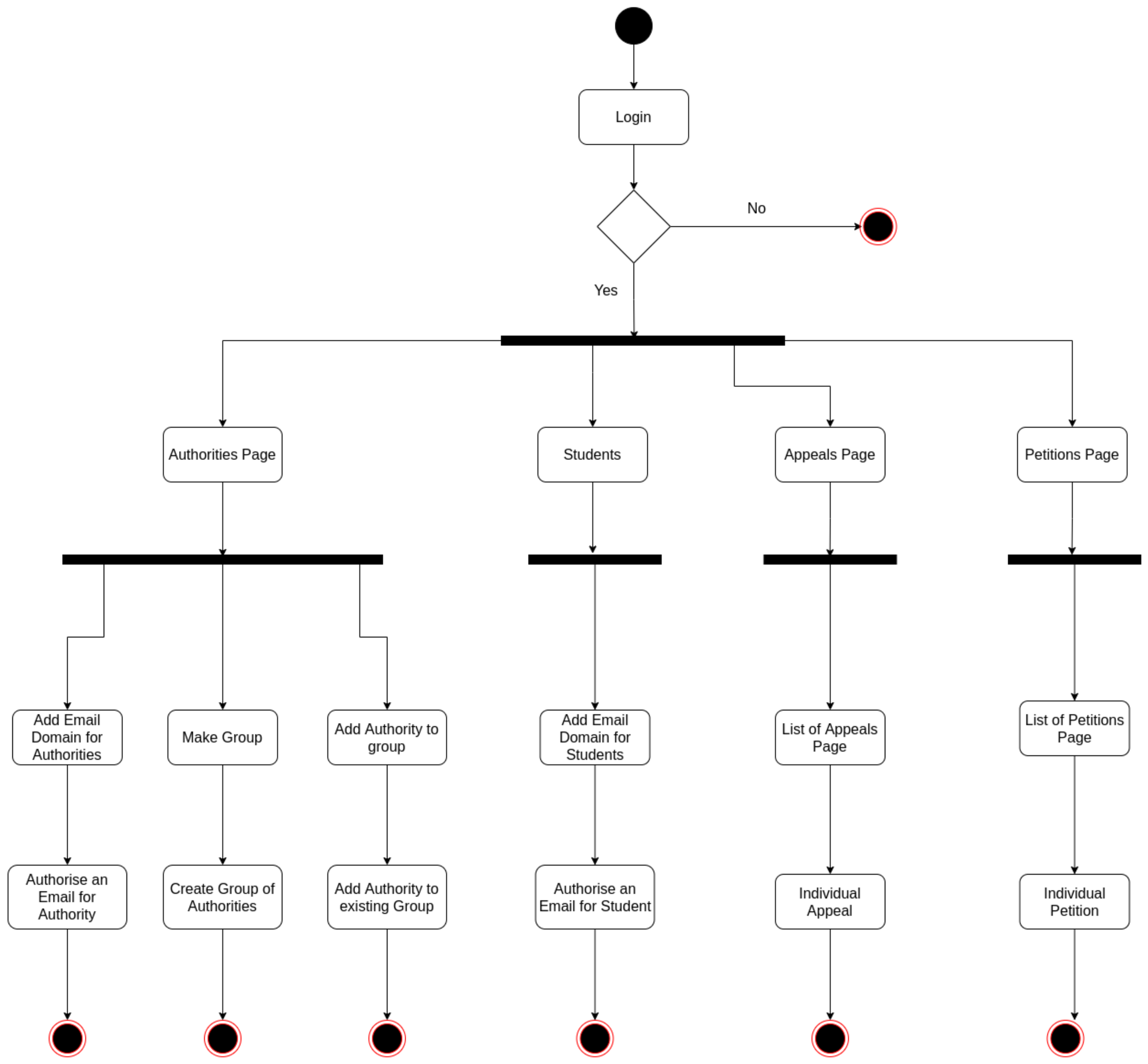
Section 2: Conceptual Architecture/Architecture Diagram

Architecture Diagram

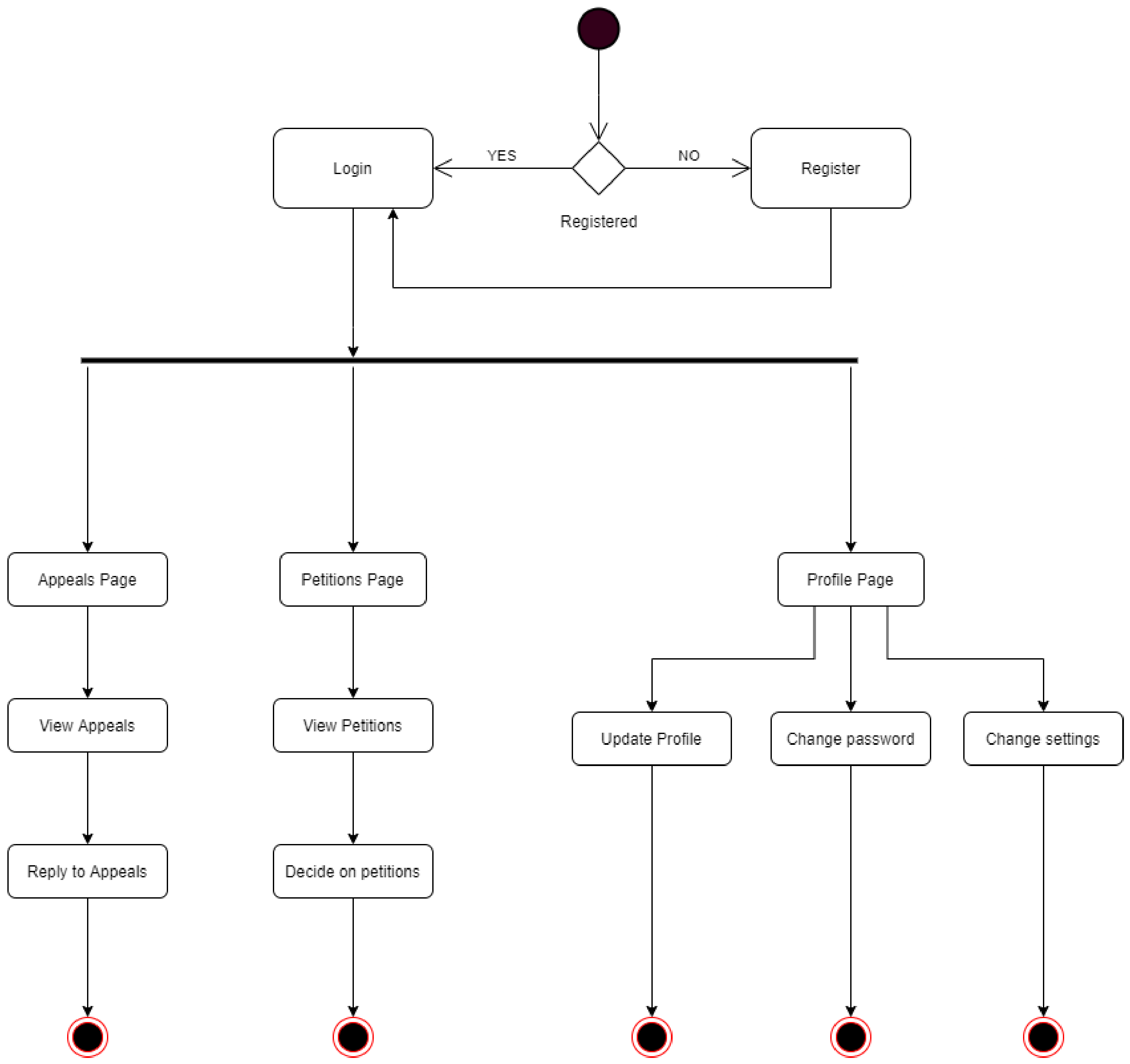


Activity Diagram
(Next Page)

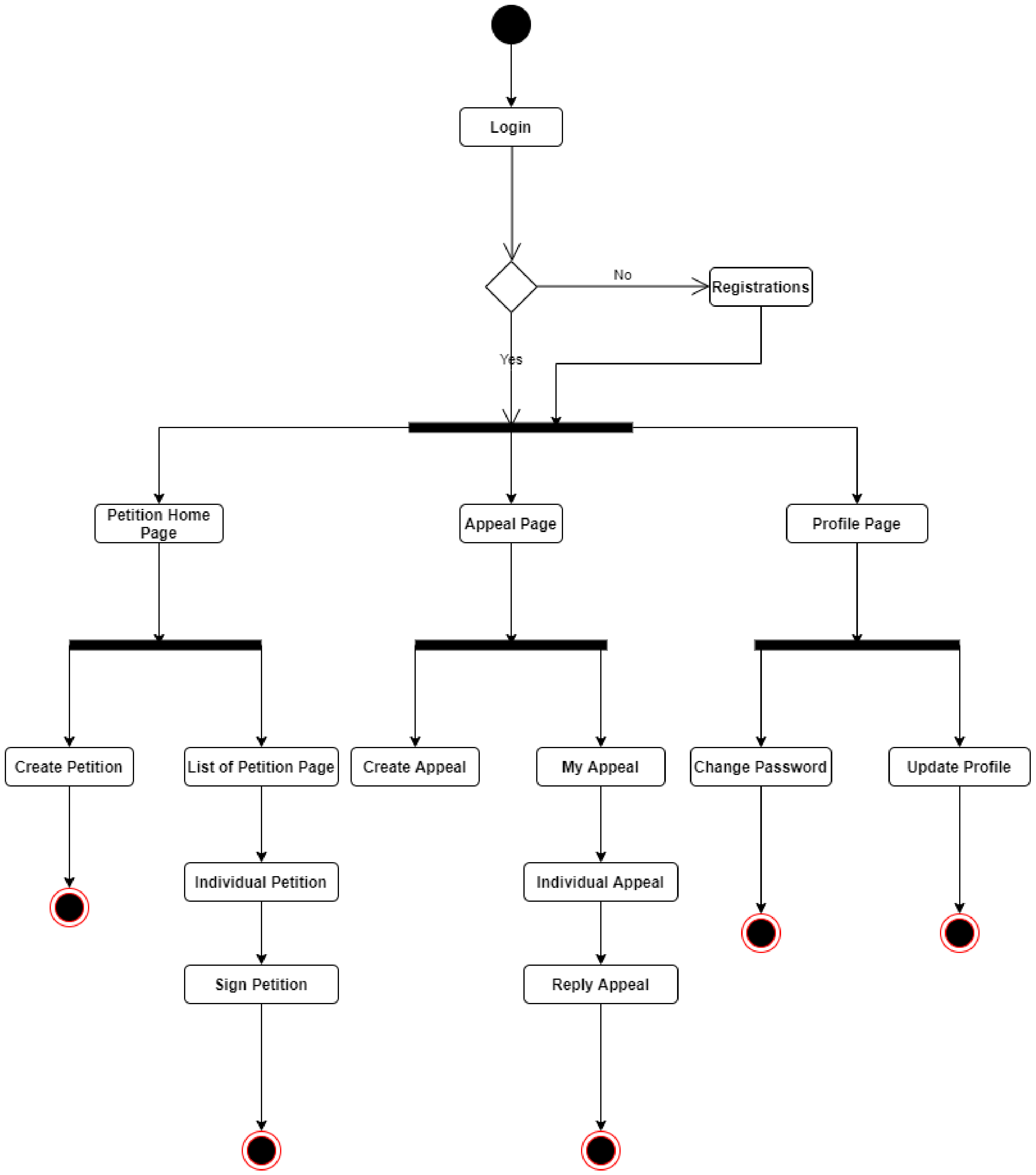
Admin


Login
No
Yes
Authorities Page
Students
Appeals Page
Petitions Page
Add Email Domain for Authorities
Make Group
Add Authority to group
Add Email Domain for Students
List of Appeals Page
List of Petitions Page
Authorise an Email for Authority
Create Group of Authorities
Add Authority to existing Group
Authorise an Email for Student
Individual Appeal
Individual Petition

Authority


Login
YES
NO
Register
Registered
Appeals Page
Petitions Page
Profile Page
View Appeals
View Petitions
Update Profile
Change password
Change settings
Reply to Appeals
Decide on petitions

Student


Login
No
Registrations
Yes
Petition Home Page
Appeal Page
Profile Page
Create Petition
List of Petition Page
Create Appeal
My Appeal
Change Password
Update Profile
Individual Petition
Individual Appeal
Sign Petition
Reply Appeal

User Interface Issues

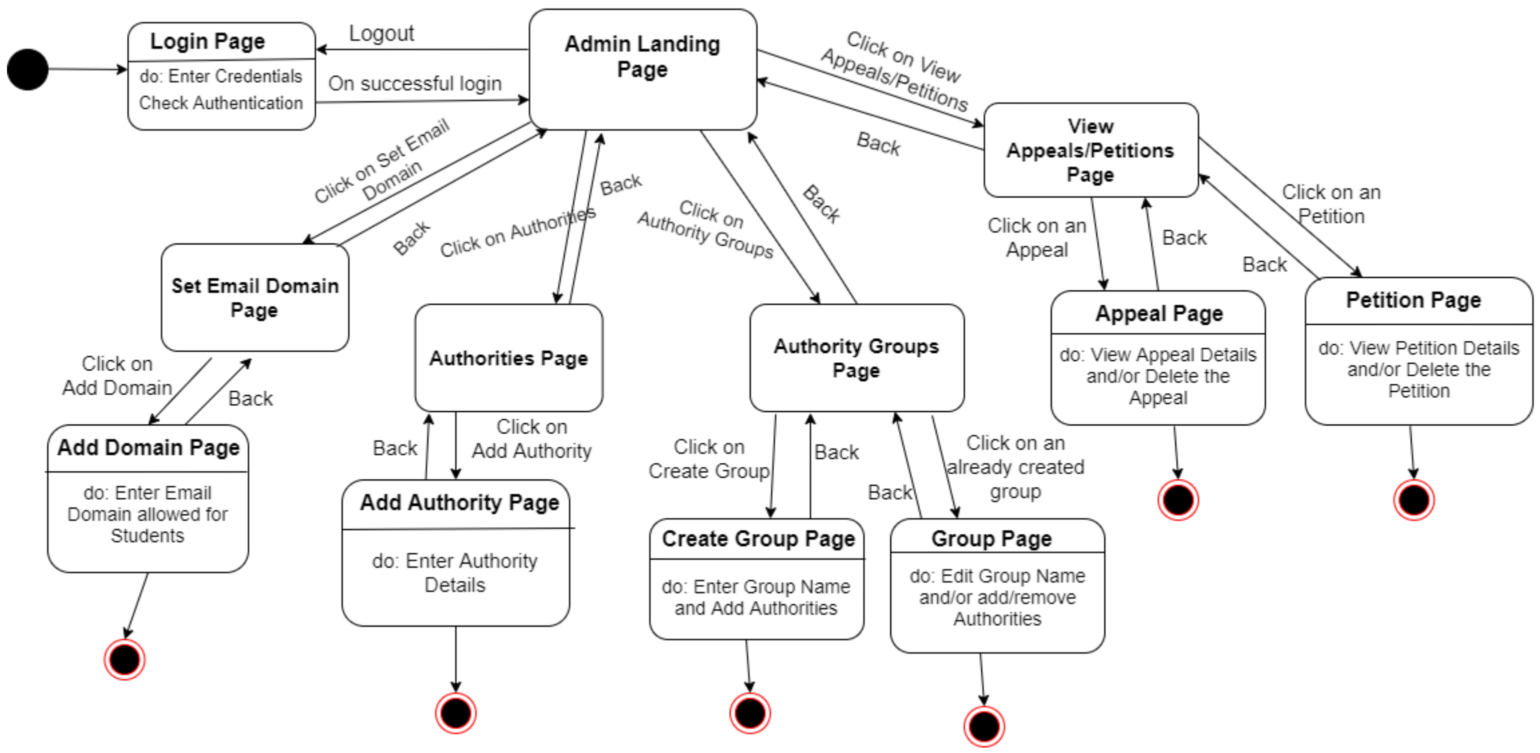
This section will address User Interface issues as they apply to the following hypothetical users of the Appeal Management System, Impedia.

- User A is a 20-year-old female, a student of IIIT Allahabad, IIIrd year, who is fairly comfortable with technology. She is proficient with using most common computer applications. Since User A is familiar with web applications, Impedia, will use common user interface conventions. For example, links between screens will use ordinary, easy to understand descriptions such as “Login”, ”View Profile”, ‘Make Appeal” etc. To maintain consistency, any other links will also appear in the bottom half of the screen.
- User B is a 37-year-old male, A staff member of IIIT Allahabad Administration. Although comfortable using web applications, he has never used any portal for appeal management, like Impedia. Since User B is not fond of such technology oriented applications, it is imperative that he be given clear on-screen directions. Consequently, the user interface has been tried to be such that grabs the attention of the user. It has been taken care that users should not consider this application as a burden/difficult to use. On-clicks task implementation has been done in Impedia, so that the admin (User B, in this case), has an ease in updating authority details and student details, and monitoring appeals via Mieten. Color combination has been so chosen, that allows the user to read all of the text on the screen in direct sunlight. Text size is reasonably larger and, therefore, more readable.
- User C is a 31-year-old female, A faculty member of IIIT Allahabad. She is well versed with using web apps and therefore, Impedia would be easily operated by her. Again, information fetching would be a few clicks away with minimal/no extra efforts.

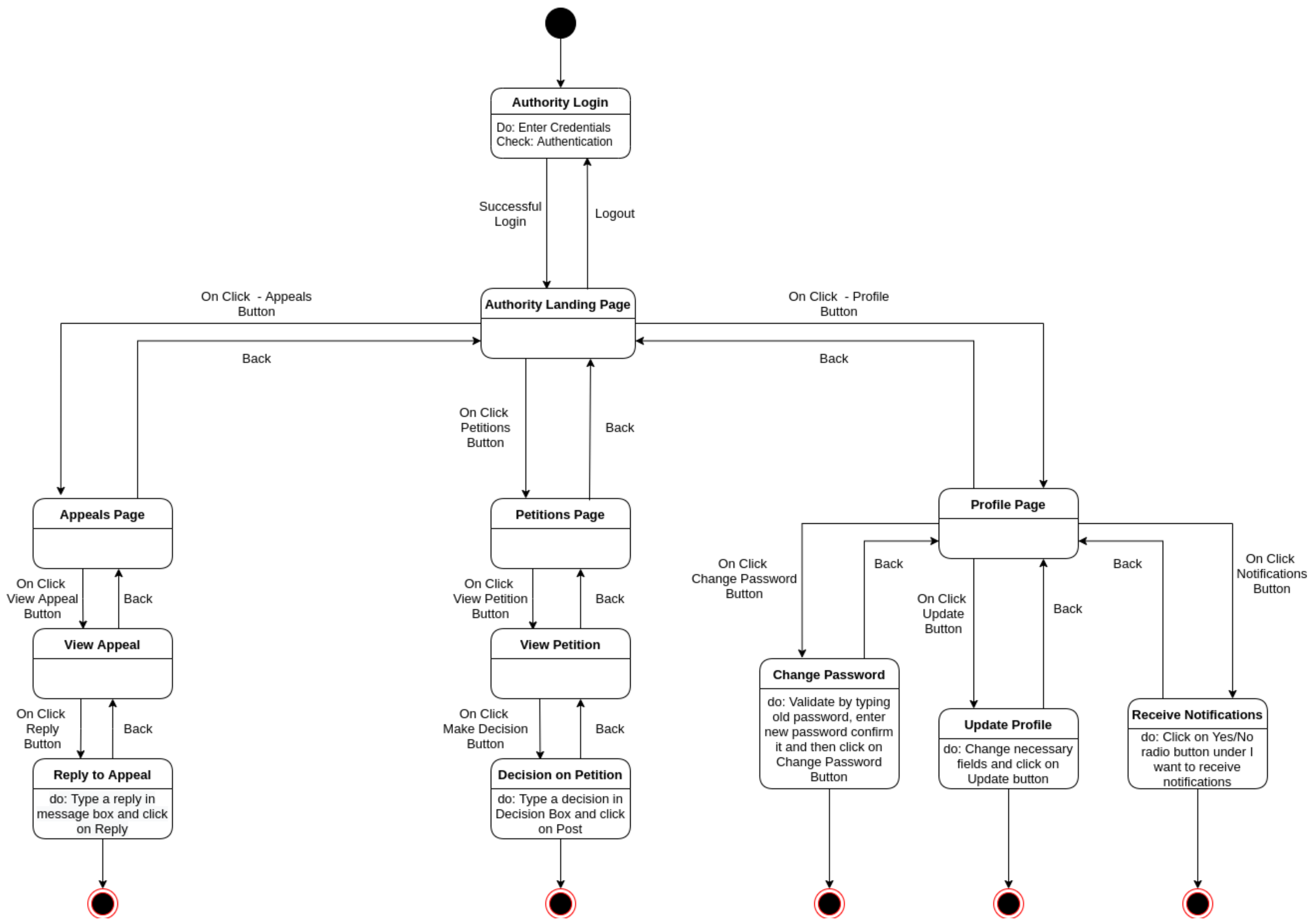
Section 3: Logical Architecture (State diagram, Activity diagram, ER diagram, Data flow diagram)

State Diagram:

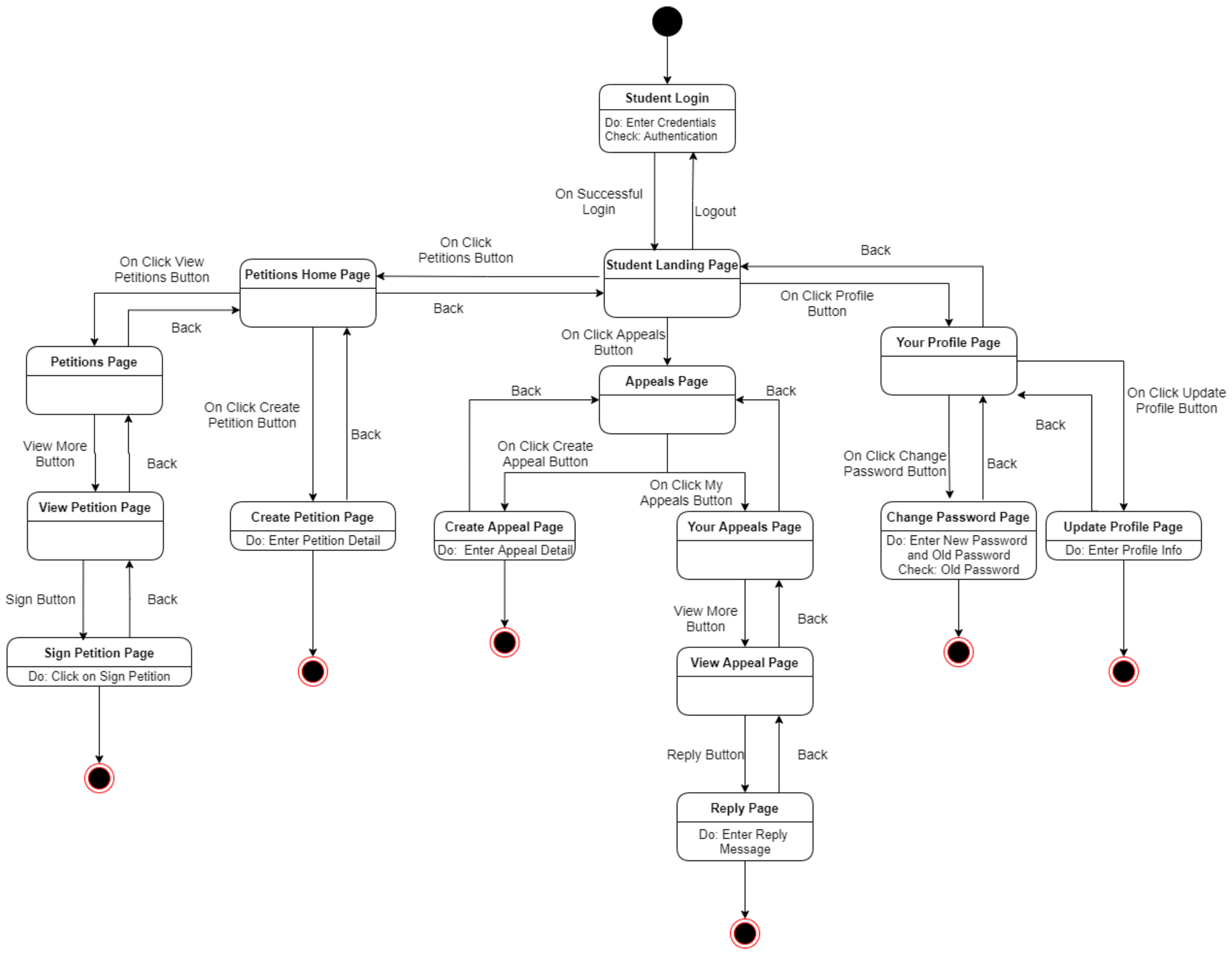
Admin State Diagram


Login Page
do: Enter Credentials
Check Authentication
Logout
On successful login
Admin Landing Page
Click on View Appeals/Petitions
Back
View Appeals/Petitions Page
Click on an Petition
Back
Petition Page
do: View Petition Details and/or Delete the Petition
Click on an Appeal
Back
Appeal Page
do: View Appeal Details and/or Delete the Appeal
Click on Set Email Domain
Back
Set Email Domain Page
Click on Add Domain
Back
Add Domain Page
do: Enter Email Domain allowed for Students
Click on Authorities
Back
Authorities Page
Click on Add Authority
Back
Add Authority Page
do: Enter Authority Details
Click on Authority Groups
Back
Authority Groups Page
Click on Create Group
Back
Create Group Page
do: Enter Group Name and Add Authorities
Click on an already created group
Back
Group Page
do: Edit Group Name and/or add/remove Authorities

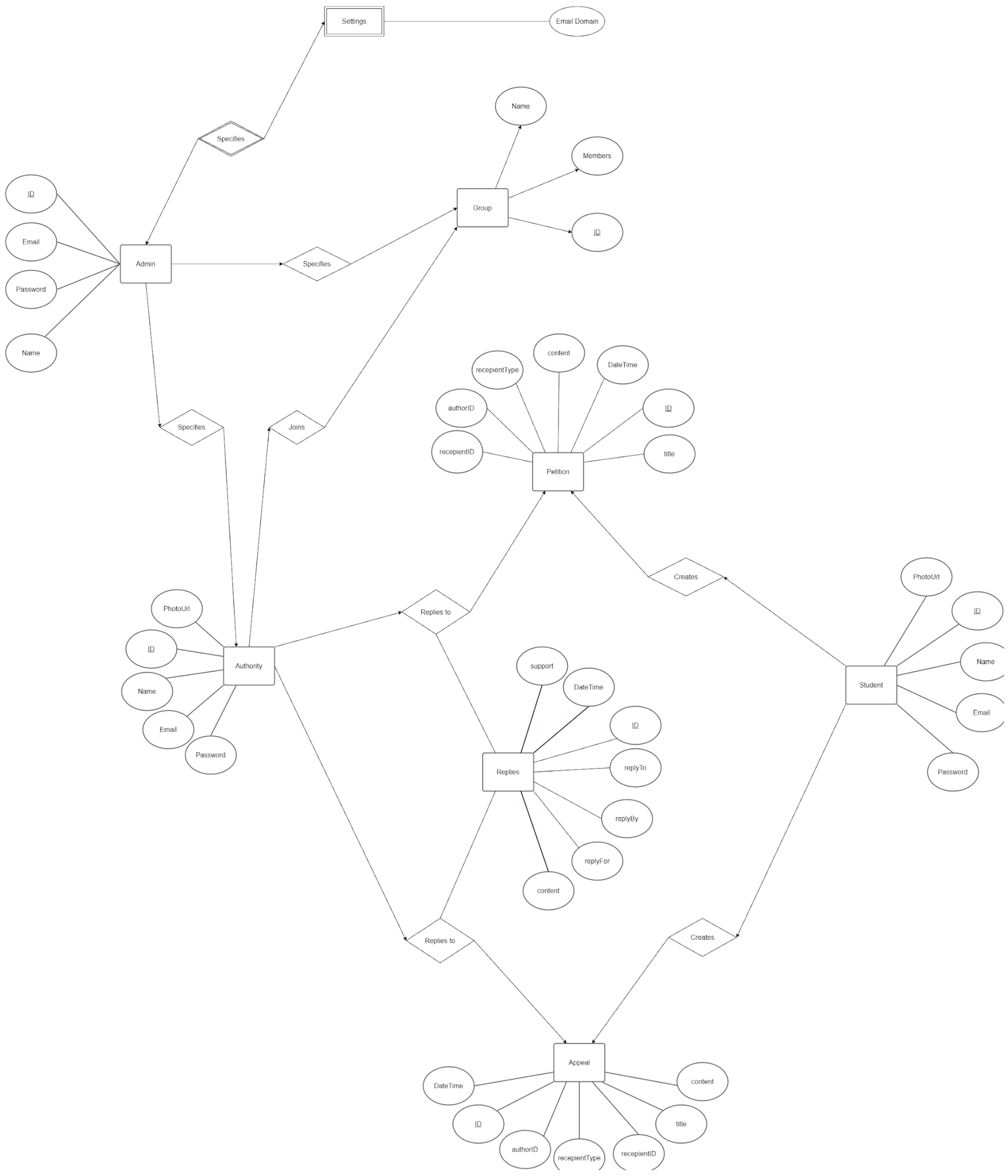
Authority State Diagram


Authority Login
Do: Enter Credentials
Check: Authentication
Successful Login
Logout
Authority Landing Page
On Click - Appeals Button
Back
On Click - Profile Button
Back
On Click Petitions Button
Back
Appeals Page
On Click View Appeal Button
Back
View Appeal
On Click Reply Button
Back
Reply to Appeal
do: Type a reply in message box and click on Reply
Petitions Page
On Click View Petition Button
Back
View Petition
On Click Make Decision Button
Back
Decision on Petition
do: Type a decision in Decision Box and click on Post
Profile Page
On Click Change Password Button
Back
On Click Update Button
Back
Back
On Click Notifications Button
Change Password
do: Validate by typing old password, enter new password confirm it and then click on Change Password Button
Update Profile
do: Change necessary fields and click on Update button
Receive Notifications
do: Click on Yes/No radio button under I want to receive notifications

Student State Diagram

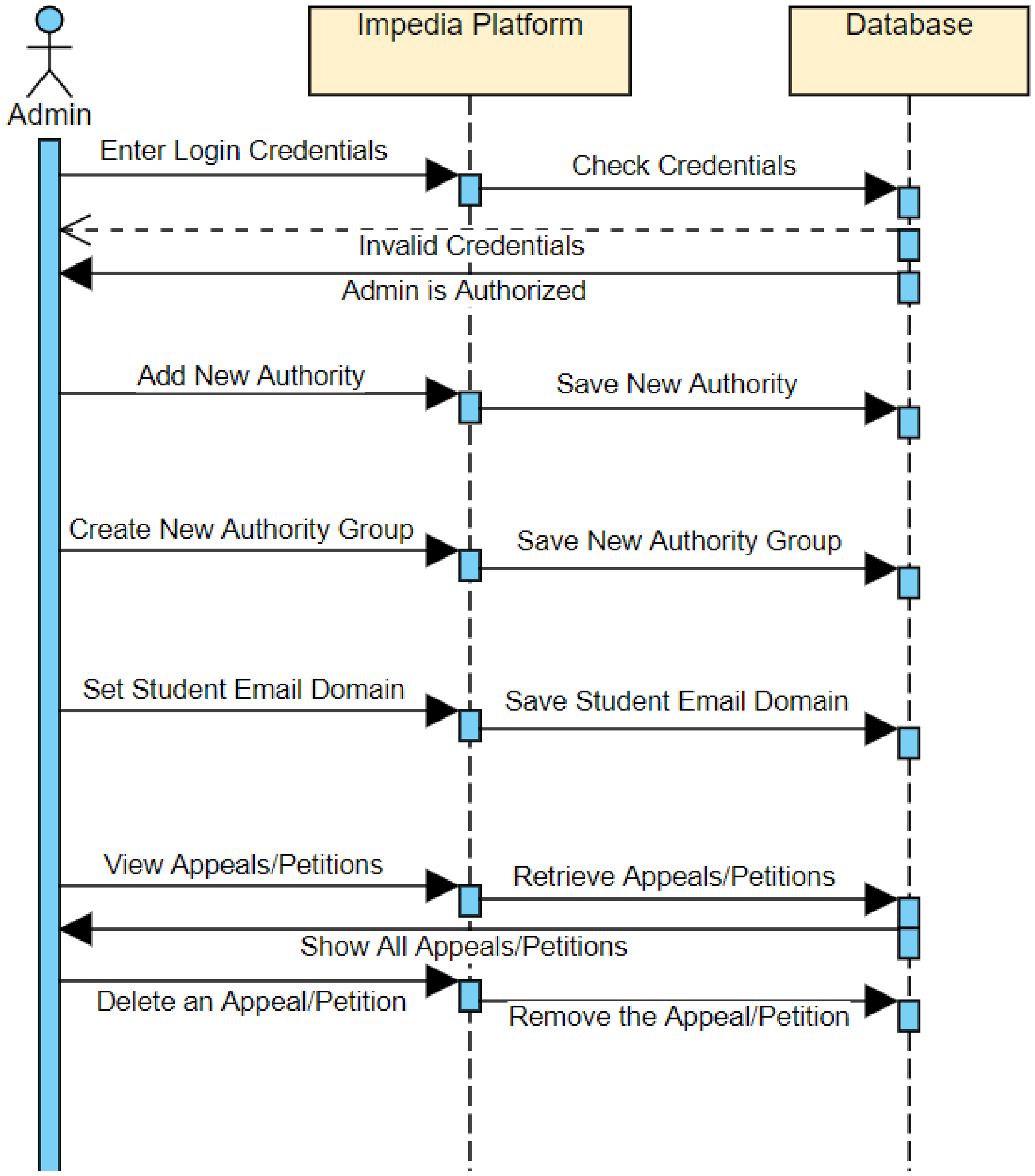


ER Diagram:

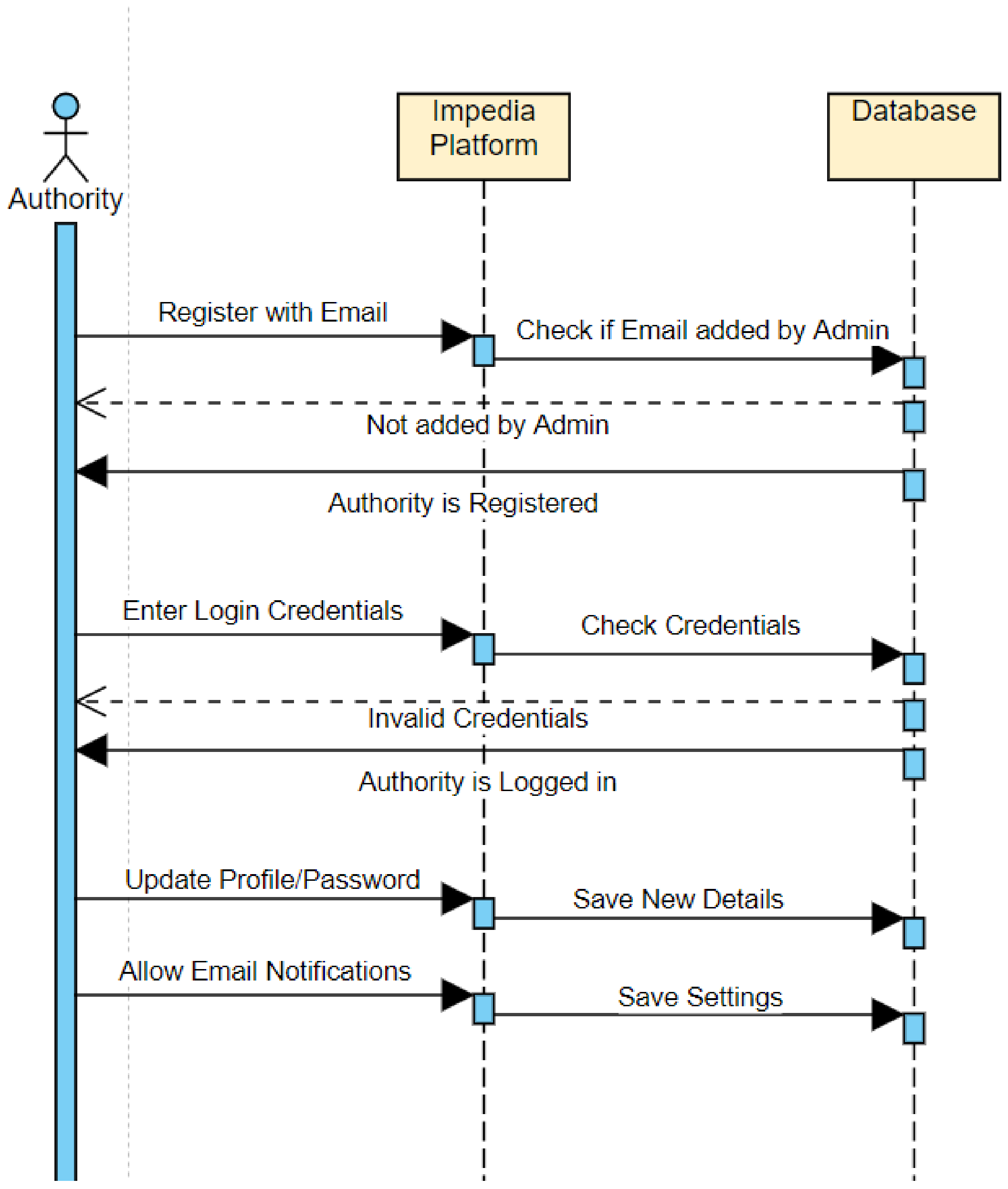

Settings
Email Domain
Specifies
Name
Members
Group
ID
ID
Email
Password
Name
Admin
Specifies
Specifies
Joins
recepientType
content
DateTime
authorID
ID
recepientID
title
Petition
Creates
PhotoUrl
ID
Name
Email
Password
Authority
Replies to
support
DateTime
ID
replyTo
replyBy
replyFor
content
Replies
Student
PhotoUrl
ID
Name
Email
Password
Replies to
Creates
Appeal
DateTime
content
ID
title
authorID
recepientType
recepientID

Sequence Diagrams

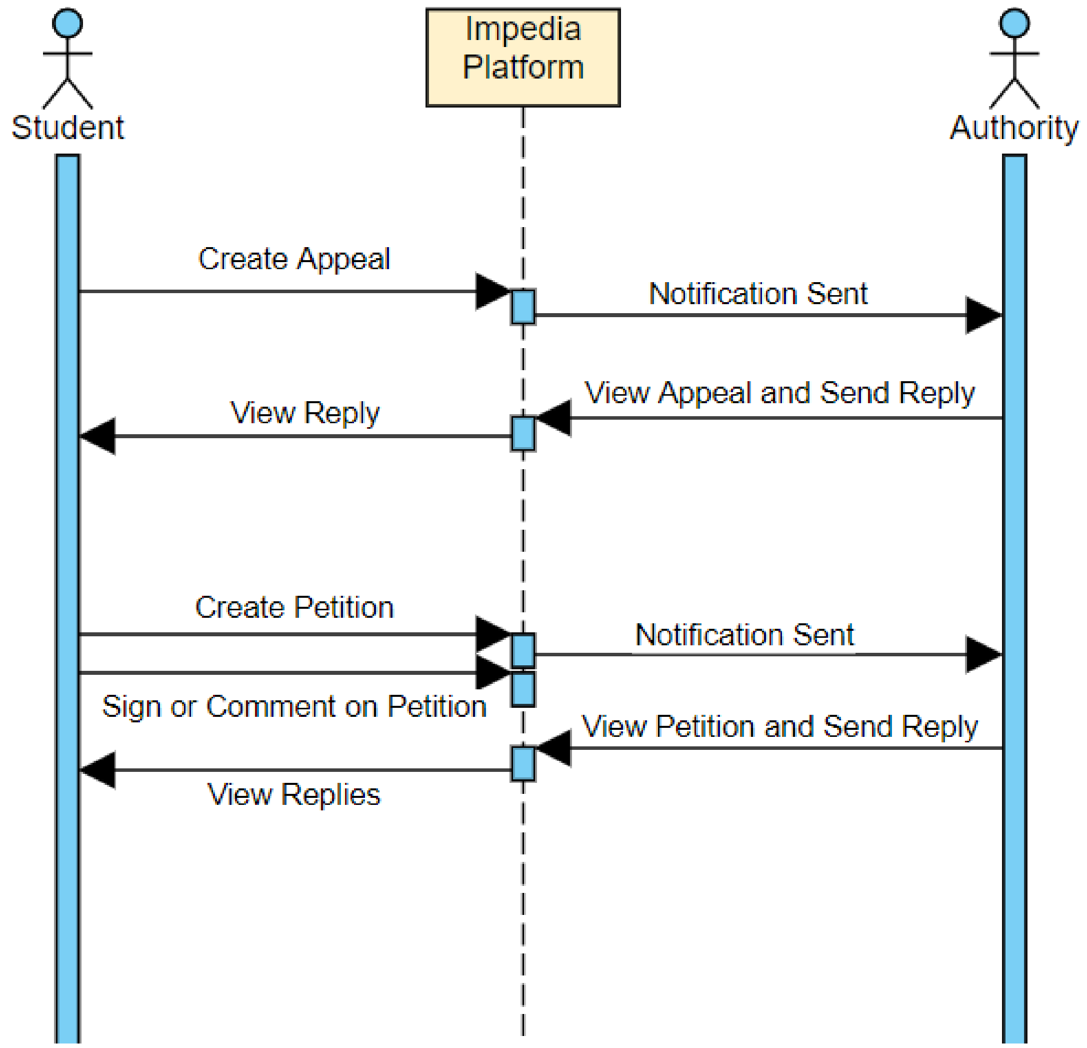
Admin sequence diagram:



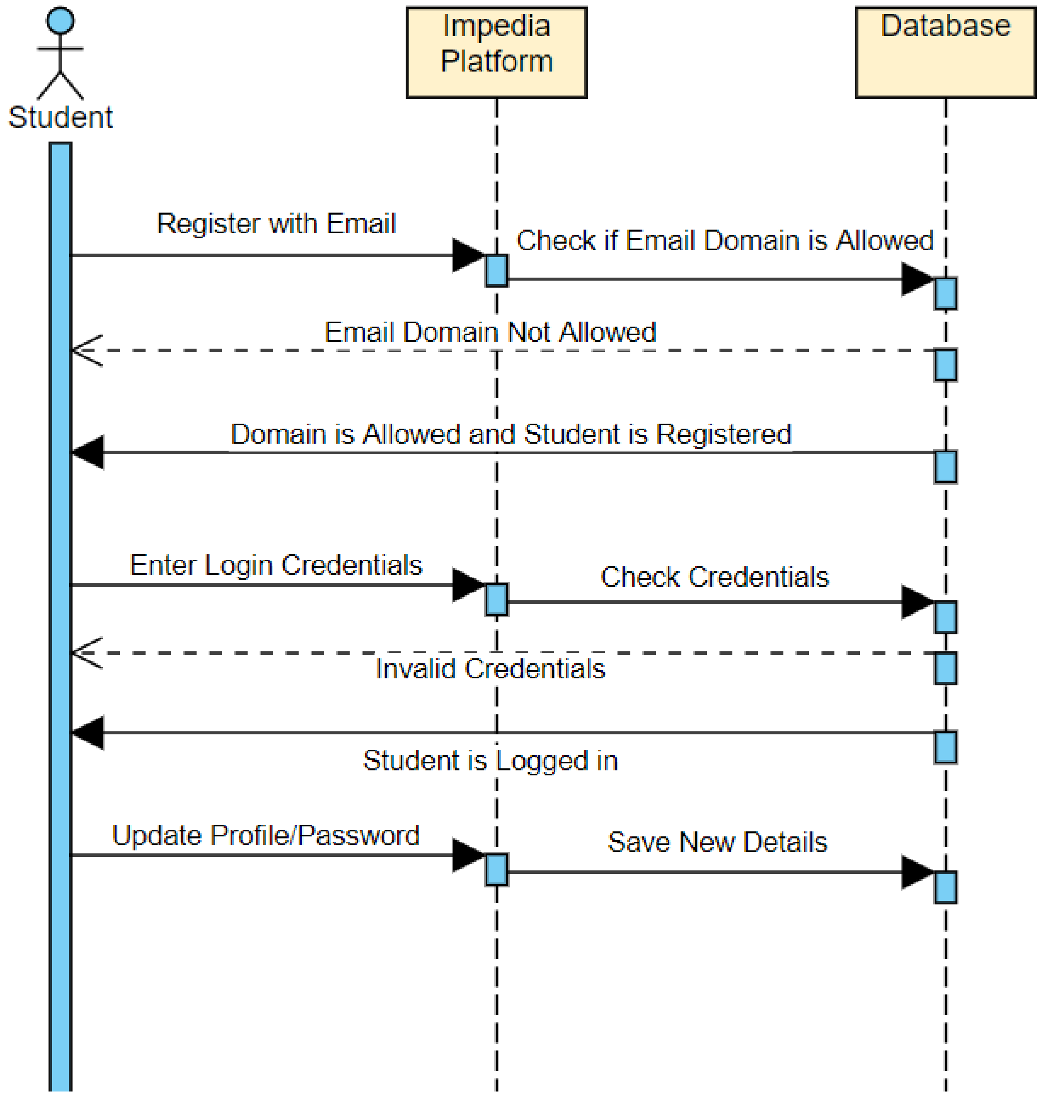
Authority sequence diagram:



Student-authority sequence diagram:

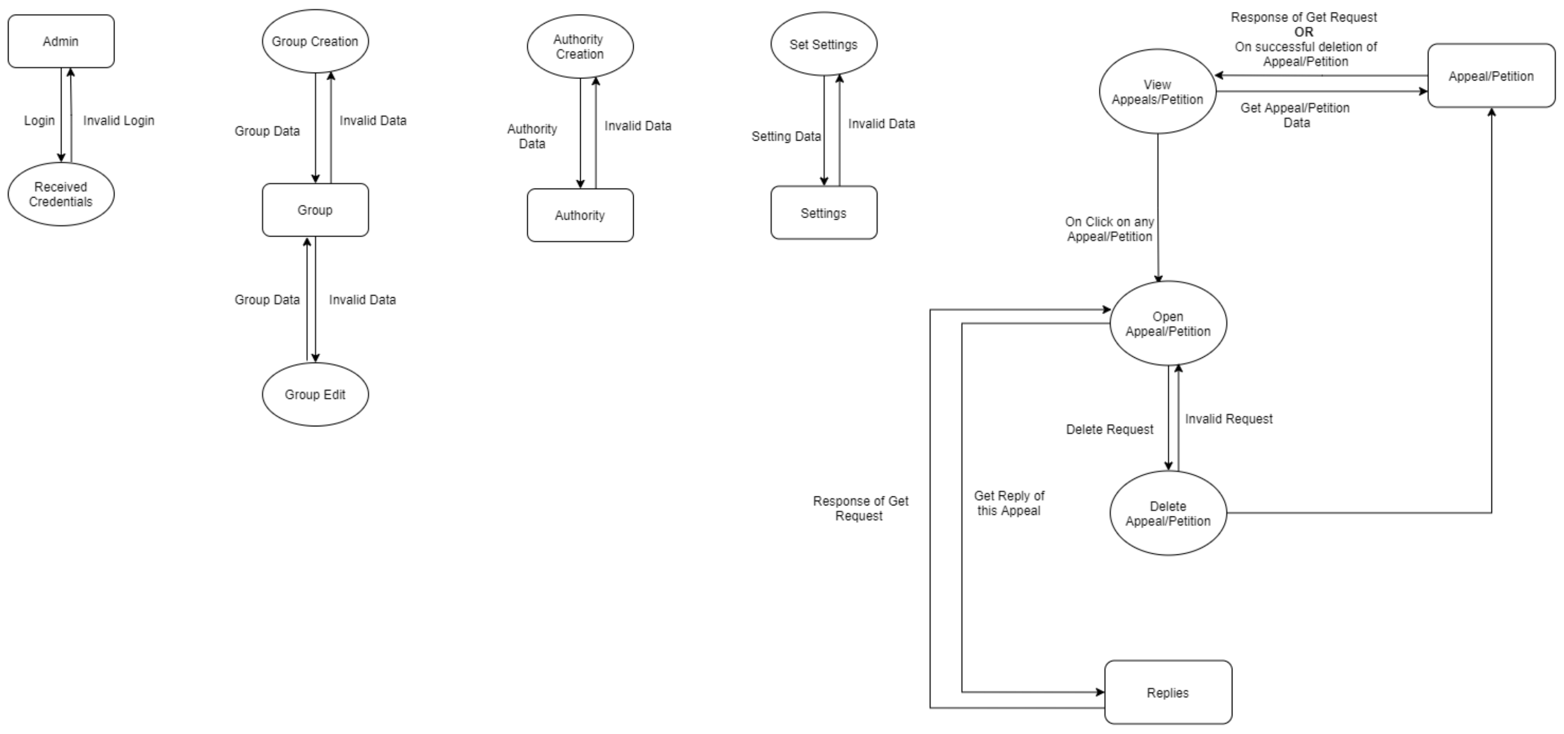


Student authentication sequence diagram:

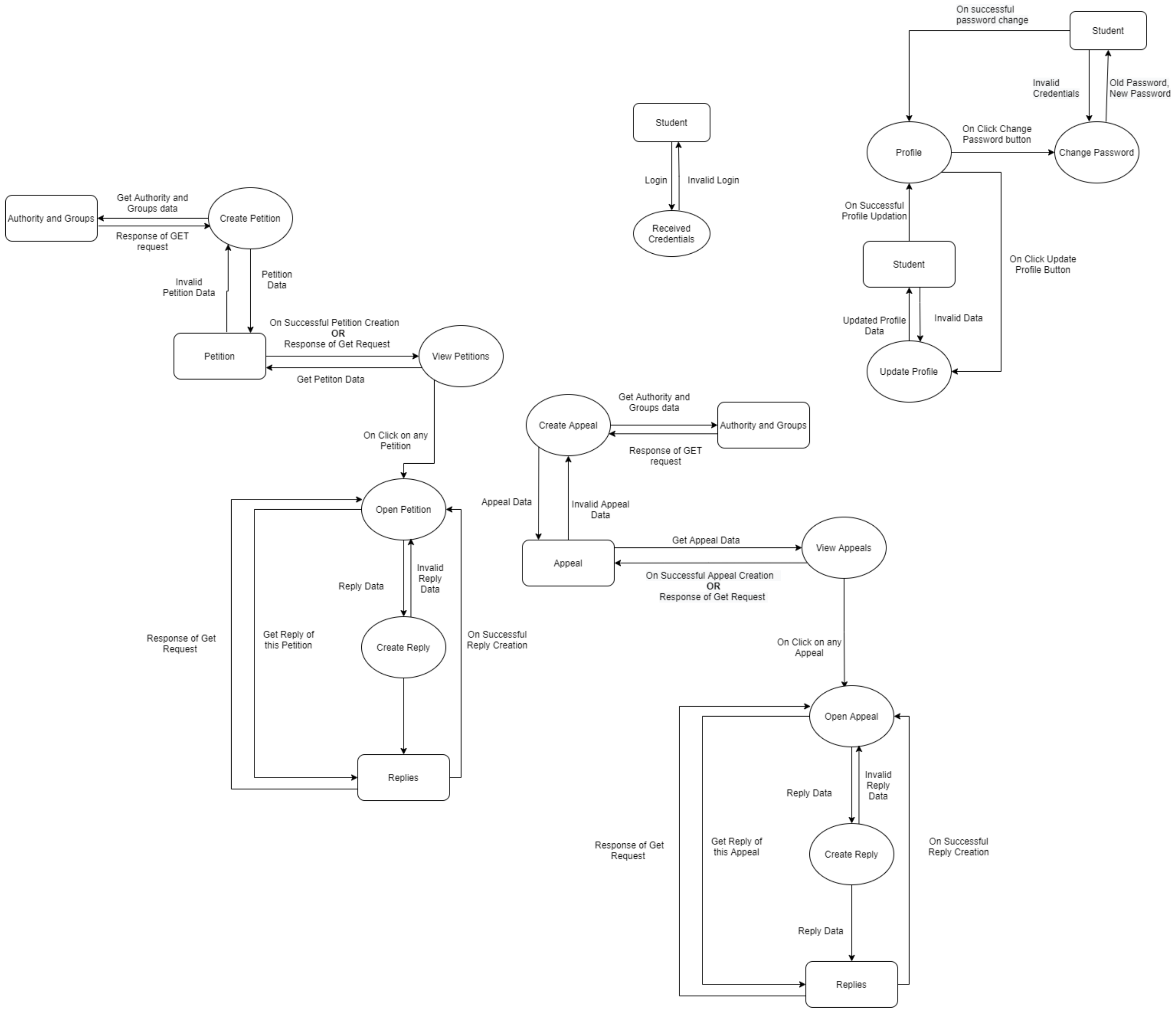


Data Flow Diagrams:

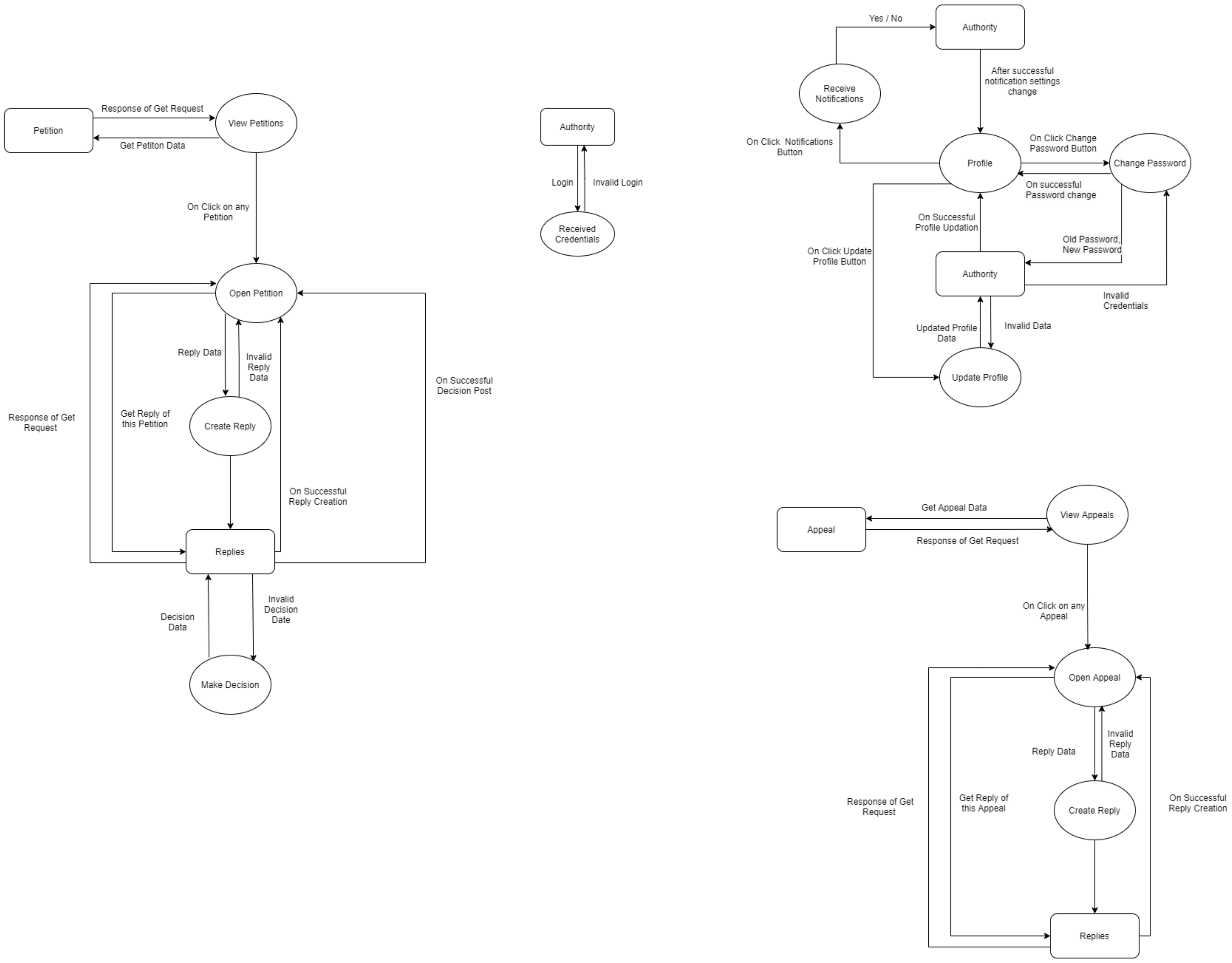
DFD-1: Admin



DFD-2: Student



DFD-3: Authority



Logical Architecture Description

State Diagram

- Admin:
    - **ST-1:** Login: Admin can login using a form on the login page which takes them to the home page.
    - **ST-2:** Add authorities: Admin can add authority emails to the application using a form given in the app.
    - **ST-3:** Set Domain for Students: Set Email Domain for Students through which they can register on app.
    - **ST-4:** View Appeals/Petitions: Can access all the Appeals and Petitions and all the conversations between authorities and students on any specific Appeal/Petition.
    - **ST-5:** Create and Add to groups: Can add emails to specific groups on the create group page, and later add any email to existing group.

- Authorities:
    - **ST-6:** Signup: Authorities enter their details into a form and signup using the emails added by admin.
    - **ST-7:** Login: Authorities login using a form with their email and password, and are taken to the home screen in case of a successful login, or are shown an error message on the login page itself in case of some issue.
    - **ST-8:** Appeals/Petitions: Can view appeals/petitions made by students on the appeal's/petition's view page and reply to it.
    - **ST-9:** Edit profile: Can visit edit section on their profiles to modify their data, change notification permissions and change password.

- Students:
    - **ST-10:** Signup: Students enter their details into a form on signup page.
    - **ST-11:** Login: Students login using the login page with their email and password, and are taken to the home screen in case of a successful login, or are shown an error message in case of some issue.

    - **ST-12:** Make Appeal: Can create new appeal on the Create page with the type appeal and address it to a particular authority using the form.
    - **ST-13:** Make Petition: Can create new petition on the Create page with the type petition and address it to a particular authority using the form.
    - **ST-14:** Appeals/Petitions: Can view appeals/petitions made by other students on the appeal's/petition's view page. Can also use the sign button to support a petition.
    - **ST-15:** Edit profile: Can visit edit section on their profiles to modify their data or change password.

Sequence Diagram

- Admin:
    - **SE-1:** Login: Logs in using the credentials provided and can access the app with admin privileges.
    - **SE-2:** Add authorities: Admin can add authority emails to the application using which authorities can log into the application.
    - **SE-3:** Set Domain for Students: Set Email Domain for Students through which they can register on app.
    - **SE-4:** View Appeals/Petitions: Can access all the Appeals and Petitions and all the conversations between authorities and students on any specific Appeal/Petition.
    - **SE-5:** Create groups: Can add emails to specific groups on the create group page using which the students can direct an appeal or a petition to a specific group of authorities.
- Authorities:
    - **SE-6:** Signup: Authorities enter their details and then can use their credentials to login after verifying their email.
    - **SE-7:** Login: Authorities login and then access the app with the authority role.
    - **SE-8:** Appeals/Petitions: Can see and reply to the appeals/petitions which the students can then view.
    - **SE-9:** Edit profile: Can edit their profiles and modify the data which will be visible to the other users.
- Students:

- **SE-10:** Signup: Students enter their details and then can use their credentials to login after verifying their email.
- **SE-11:** Login: Students login and then access the app with the student role.
- **SE-12:** Make Appeal: Can create new appeal on the Create page for any authority, who will be notified and can then check out and reply to it.
- **SE-13:** Make Petition: Can create a new petition which will notify the authorities and who can then check out and decide on it and the students can view it and sign it.
- **SE-14:** Appeals/Petitions: Can view appeals/petitions made by other students on the appeal's/petition's view page. Can also use the sign button to support a petition.
- **SE-15:** Edit profile: Can edit their profiles and modify the data which will be visible to the other users.

Section 4: Execution Architecture

Runtime Environment required is any device with the latest stable version of any Web Browser (Chrome recommended) with JavaScript and Cookies enabled for the site.
Deployment Requirement:
Backend: NPM/Yarn Package manager, Node.js Runtime, MongoDB Community Server for Development Purposes.
Frontend: React Library

Reuse and Relationships to Other Products

NIL

Section 5: Design Decisions and Trade-offs

The design decision to use three screens separately for Admin, Authority and Student is to provide encapsulation. It may have been possible to get all the information on one screen. However, using three screens will keep the data of Admin/Authority separate from Students.

A possible trade-off when considering navigating to a particular Petition/Appeal is to have Cards with brief information linking to the original Appeal/Petition. This design decision – to contain Links in cards instead of having a separate button at a corner to View Appeal/Petition is to enhance UX and improve UI, as buttons would take up space and might take attention away from primary content which would make navigating through a number of Appeals/Petitions difficult. Also, if the web app is opened on a small screen such as a mobile screen, the button might mess up the design. Having Cards linking to complete content is more intuitive for the user and enhances UX. Thus, navigation to specific content becomes easier.

Another possible trade-off while considering Send/Add/Post actions is to include icons as buttons rather than having text, as it makes it more intuitive that what clicking on the button would actually do, thus improving the User Experience.

Section 6: Pseudocode for components

Admin

```
admin login(){
     get login details(email,password)
     If email matches organisation domain() then:

          If email in admin email list then:
               user = authentication(email,password)
               If user then:
                    Login(user)
                    Return Response(“Success”)

               Else then:
                    Return Response(“Password Incorrect”)

          Else then:
               Return Response(“You are not allowed to login as
admin”)

     Else then:
          Return Response(“Only Organisation Domain Email
Allowed”)
}



set organisation domain(){
     If user is authenticated then:

          If user is admin then:
               get organisation domain(name)
               update organisation domain in database()

               Return response(“Success”)

          Else then:
               Return response(“Only Admin can set organisation
          email”)

     Else then:
          Return redirect(admin login())
}
```

```
create authorities group(group name, authority ids) {
     If user is authenticated then:

          If user is admin then:

               authority group = create authority group with
          name = name
               authority id = authority group[id]

               for authority id in authority ids {
                    if authority id exists() {
                         add authority id as group id member()
                         update details in database()
                         add authority id to success list
                    }
                    else {
                         add authority id to failed list
                    }
               }


               Return {"fail": failed list ,"success":success
          list,"group id":group id}

          Else then:
               Return response("Only Admin can add authorities
          to group")

     Else then:
          Return redirect(admin login())

}



add authorities to group(group id, authority ids) {
     If user is authenticated then:

          If user is admin then:
               for authority id in authority ids {
                    if authority id exists() {
                         add authority id as group id member()
                         update details in database()
                         add authority id to success list
                    }
                    else {
                        add authority id to failed list
                    }
                }


                Return {“fail”: failed list ,”success”:success
            list,”group id”:group id}

            Else then:
                Return response(“Only Admin can add authorities
            to group”)

        Else then:
            Return redirect(admin login())

}



delete petition(petition id){
        If user is authenticated then:

            If user is admin then:
                petition = get petition where id = petition id
                If petition then:
                    delete petition in database()
                    Return response(“Petition deleted
            successfully”)

                Else then:
                    Return response(“Petition does not exists”)


            Else then:
                Return response(“Only Admin can delete petition”)

        Else then:
            Return redirect(admin login())

}
```

Student

```
student login(){
     get login details(email)
     If email matches organisation domain() then:
          Send verification email
          If email verified() then:
               Set password()
               get password
               Save credentials in database()

               Return Response(“Success”)
          Else then:
               Return Response(“Email Not Verified”)

     Else then:
          Return Response(“Only Organisation Domain Email
Allowed”)

}



create appeal() {
     If user is authenticated then():
          If user is Student then():
               get appeal details(title, description, category)
               suggest authority or authority group(category)
               get chosen authority(choice)
               update details in database()
               notify authorities()
               Return Response(“Success”)

          Else then():
               Return Response(“Only Student Can Create Appeal”)

     Else then():
          Return redirect(student login())


}
```

```
create petition() {
     If user is authenticated then():
          If user is Student then():
               get petition details(title, description,
          category)
               suggest authority or authority group(category)
               get chosen authority(choice)
               update details in database()
               notify authorities() and notify students()
               Return Response("Success")

          Else then():
               Return Response("Only Student Can Create
Petition")

     Else then():
          Return redirect(student login())

}



sign petition {
     If user is authenticated then():
          If user is Student then():
               add to petition signees()
               update details in database()
               Return Response("Success")

          Else then():
               Return Response("Only Student Can Sign Petition")

     Else then():
          Return redirect(student login())
}



student reply{
     If user is authenticated then():
          If user is Student then():
               get reply details(title, message)

               If reply is on petition then:
                    reply.is_petition = True
               Else then:
                    reply.is_petition = False
```

```
                add reply to database()
                notify authority()

                Return Response("Success)

           Else then():
                Return Response("Only Student Can Access this")

      Else then():
           Return redirect(student login())

}


change password student{
      If user is authenticated then():
           If user is Student then():
                get password details(new password)

                update password to database()

                Return Response("Success)

           Else then():
                Return Response("Only Student Can Access this")

      Else then():
           Return redirect(student login())

}


update profile student{
      If user is authenticated then():
           If user is Student then():
                get details()

                update details to database()

                Return Response("Success)

           Else then():
                Return Response("Only Student Can Access this")

      Else then():
           Return redirect(student login())
```

```
}
```

Authority

```
authority login(){
     get login details(email,password)
     If email matches organisation domain() then:

          If email in authority email list then:
               user = authentication(email,password)
               If user then:
                    Login(user)
                    Return Response(“Success”)

               Else then:
                    Return Response(“Password Incorrect”)

          Else then:
               Return Response(“Inform Admin to add your email”)

     Else then:
          Return Response(“Only Organisation Domain Email 
Allowed”)
}

authority reply {
     If user is authenticated then():
          If user is Authority then():
               get reply details(title, message)

               If reply is on petition then:
                    reply.is_petition = True
               Else then:
                    reply.is_petition = False

               reply.is_authority = True

               add reply to database()
               Return Response(“Success)

          Else then():
               Return Response(“Only Authority can Access this”)

     Else then():
          Return redirect(authority login())
}
```

```
change password authorities{
     If user is authenticated then():
          If user is authority then():
               get password details(new password)

               update password to database()

               Return Response(“Success)

          Else then():
               Return Response(“Only Authority Can Access this”)

     Else then():
          Return redirect(authority login())

}

update profile authorities{
     If user is authenticated then():
          If user is authority then():
               get details()

               update details to database()

               Return Response(“Success)

          Else then():
               Return Response(“Only Authority Can Access this”)

     Else then():
          Return redirect(authority login())

}
```