Modules

# Create User

Inputs: username, String

password, String

confirmPass, String

email, String

Name, String

Student Id, Number/Integer

Gender, String

Campus, String

Output: These inputs are stored in the database. These inputs then become accessible to the user.

Detail Processes: doesConfirmEmailMatchPass()

isPlaceholderString()

isPlaceholderInteger()

doesEmailAlreadyExists()

doesEmailEndInMun()

isPlaceholderEmpty()

# Post

Inputs: id, number

Text, String

Image

userId, number

visible, Boolean

Detail Processes: changeVisiblilty(setting)

Input->setting(boolean), the intended setting to change to to. If true, the post is visible to all. If false, the post is visible to just the user

Output: Post.visible->setting

getComments()

Input->id, this is used to match the post id with the post id of the comment model

Output: Comment

editText(newText)

Input->newText(String), the new text to replace what is current in the post

Output post.text->newText

removeText()

Output: removes the text from the post

removeImage()

Output: removes image from post

getId()

Output:post.id

Delete(id)

Input->post.id(number), the id of the post to be deleted from the database

Output: post deleted from the database

# Comment

Inputs: id, number

userId, number

postId, number

text

Detail Processes: addComment(userId, postId, text)

Input->userId(number), the id of the associated user, or the author of the comment

->postId(number), the id of the post from the comment

->text(String), the text of the comment

Output: new comment created in the database

editComment(id, newText)

Input->id(number), the id of the comment to be edited

->newText(String), the text to be replace the existing comment text

Output: comment.Text->newText

deleteComment(id)

Input->id(number), the id of the comment to be deleted

Output: the comment is deleted from the database

# Group

Inputs: name, String

Admin, User

Member(s), User

Description, String

Detail Processes:

createGroup(user, name, description)

Input-> user, this user that creates the group will be set to admin by default

->name(String), the name of the group

->description(String), the description for what the group is for

Output: new group created in the database

addUser(id, user.id)

Input: id(number), the id of the group

user.id, the intended user and the id to be added

Output: user added to group

Join(id, user.id)

Input: id(number), the id of the group

user.id, the intended user and the id requesting to join

Output: request is added to group

handleRequest(id, user.id, option, request)

Input: id(number), the id of the group

user.id, the intended user and the id requesting to join

option(boolean), this will be determined by the admin. If true, the request is accepted, else the request is rejected

request, the pending request

Output: if option is true, user is added to group, else, request is removed

deleteGroup(id)

Inputs: id(number), the id of the group

Output: the group is deleted from the database

editName(newName)

Inputs: newName(String), the name to replace the existing name of the group

Output: group.name->newName

editDescription(newDesc)

Inputs: newDesc(String), the new description to replace the existing description of the group

Output: Group.Description->newDesc

# Schedule

Inputs: slot, number

Days, String(array)

Time, String(array)

courseName, String

scheduleName, String

Detail Processes:

addSchedule(slot, Days, Time, courseName, scheduleName)

Inputs: ->slot(number), the slot number of the course

->Days (String array), the days for each course

->Time (String, array), the times for each course (Array has to correspond with Days)

->courseName(String), the name of the course

->scheduleName(String), the name of the schedule.

Output: new schedule created in the database

editCourseName(courseName, newCourseName)

Input: ->courseName(String), the current course Name

-> newCourseName(String), the new course name to replace the current one

Output: Schedule.courseName->newCourseName

editScheduleName(scheduleName, newScheduleName)

Input: ->scheduleName(String), the current schedule Name

-> newscheduleName(String), the new schedule name to replace the current one

Output: Schedule. scheduleName->newScheduleName

deleteSchedule(id)

Input: id(number), the id of the schedule to be deleted

Output: schedule is deleted from the database

# Resume

Inputs: user.id, number

File, file

Name, String

Detail Processes

addResume(user.id, file, name)

Inputs: ->user.id(number), id of the user uploading the resume

->file(file), the resume file

->name(String), the name of the resume

Output: new resume is created in the database

deleteResume(id)

Inputs: id(number), the id of the resume

Output: the resume is deleted from the database

# Lost and Found

Inputs: user.id, number

Location, String

Description, String

Detail Processes:

addObject(user.id, location, description)

Inputs: ->user.id(number), the id of the user adding to the list of lost and found objects

->location(String), the location of the object  
->description(String), the description of the object

Output: new lost and found object added to the database