PORTAC

DESIGN DOCUMENT

10/13/2012

Group16

Table of Contents

1. INTRODUCTION

1.1 Purpose

The purpose of this document is to give detailed design description of our web portal to providing a single platform to handle all academic business in the institution. The primary purpose of the portal would be:

- course registration
- lecture room allotment
- academic calendar
- college wiki
- centralised admission and counselling system

1.2 Scope

Our web portal is aimed at academic institutes and any academic institute can use our portal for their academic requirements. Other than course registration system it includes a unique feature of an efficient lecture room allotment system, which can be used by institutions with scarcity of rooms. It also empowers the institution by enabling them to handle admission and counselling procedure.

1.3 Definitions, Acronyms and Abbreviations (to be edited)

- *PortAc* : Short for 'Academic portal' and the name of product.
- HTML : Hyper Text Markup Language
- HTTP : Hyper Text Transfer Protocol
- LAN : Local Area Connection
- ID : Identity
- E-R Diagram : Entity-Relationship Diagram

1.4 References (to be edited)

- [IEEE] SRS Template: The applicable IEEE standards are published in "IEEE Standards Collection," 2001 edition.
- The principal source of textbook material is: "Software Engineering" by Ian Somerville (8th Edition, Pearson Education)
- www.stanford.edu : Academic portal of Stanford University, to gain idea about academic portal.
- https://wiki.iitd.ernet.in/mediawiki : IIT Delhi's wiki page.

1.5 Overview

Our web portal will be an online login based system where each student and professor will get a unique id and password. There will be some features open to guest users too. Any institute can use the portal by customizing the features that best suits their needs.

2. SYSTEM ARCHITECTURE OVERVIEW

2.1 Architecture Design Diagram

2.2 Overview of subsystems and modules

The architecture has been identified under five major subsystem and their modules:

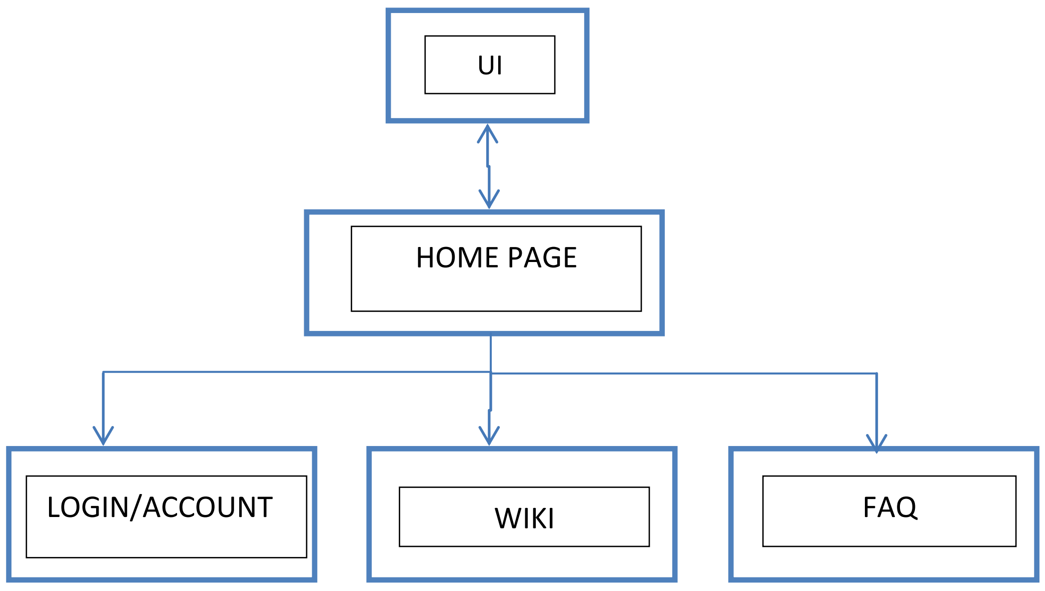
- Home page
- Login/Account subsystem **add sub modules
- Course Registration
- Room booking
- e-Counselling
  - Fill choices
  - Allot Courses
- College wiki management sys
- FAQ

The detailed description of each of these components is provided in section 3.

2.3 Subsystem/Module Communication and Dependencies

3. DETAILED COMPONENT DESCRIPTION

3.1 HOME PAGE



3.1.1 Purpose: This is the top most sub-system of our design. It will provide gateway for different users to get through the portal. It will link three other sub-system namely *login/account sub-system, college wiki sub-system* and the *FAQ sub-system.* Once a user has given an input it will transfer control to appropriate sub-system sitting below. It will contain introductory information of the web portal that will appear to a user when he first visits the portal.

3.1.2 Interface Specification: Following are the parameters and calls:

3.1.2.1 Parameter: i. *user_click*;

3.1.2.2 Function Signature: *transfer_control(user_click);*

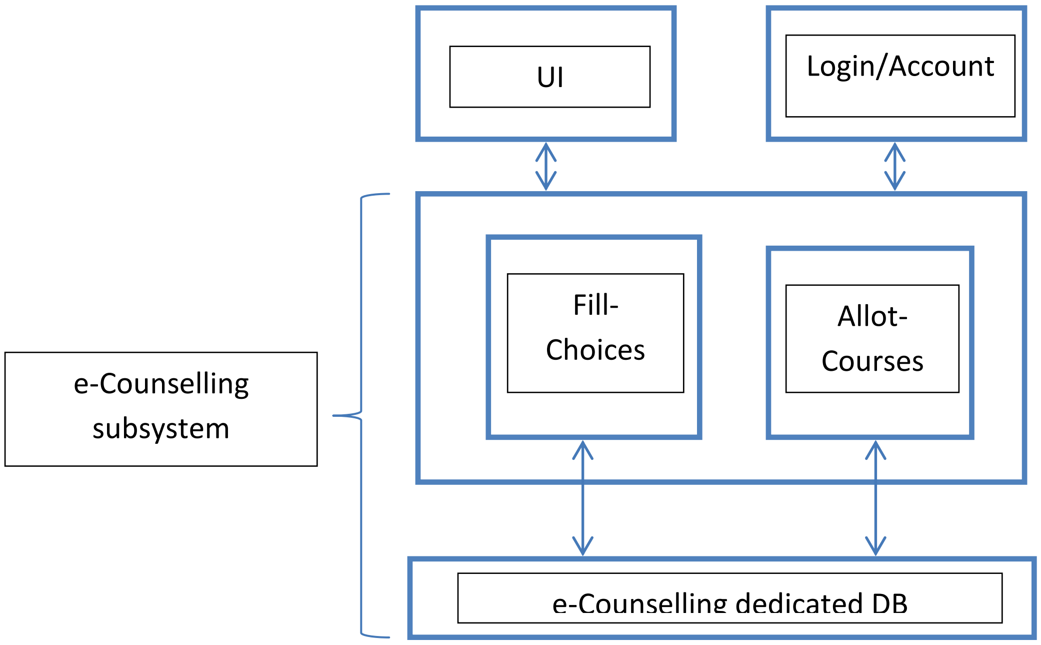
3.1.2.3 Input/Output: *transfer_control()* will take *user_click* as the input and its output will be transfer of control to appropriate subsystem namely – login/account, college wiki or the FAQ subsystem. Output will be a call to any of following functions:

i. *call_login();*
ii. *call_wiki();*
iii. *call_FAQ();*

3.1.3 Data structures and Algorithms: Depending on where the *user_click* points (to login/account or college wiki or FAQ), it will transfer control to that particular subsystem by calling representative function of each subsystem. It can use a simple *integer* like *user_click* for each subsystem depending on whether the click is on login, or wiki or FAQ.

3.1.4 Dependencies: It will depend on the UI subsystem for getting input from the user, specifically the value of *user_click*.

3.2 e-COUNSELLING



e-Counselling subsystem will allow users to fill in their choices of courses, and upon validation by administrator it will allot (or regret) course programmes to each user. Following two modules have been identified for this subsystem:

- Allow filling of choices (module 1.2.1)
- Allot course programmes (module 1.2.2)

3.2.1 Filling of courses

3.2.1.1 Purpose: It will allow users to fill in their choice of course programmes offered at the institute. Users can select from list of available courses (facilitated by the UI subsystem). Specifically for admission purpose.

3.2.1.2 Interface Specification: Following are the parameters and the calls:

3.2.1.2.1 Parameter: i. *userID*
ii. *choice_list*
iii. *priority_list*
iv. *deadline_date*
v. *choice_to_delete*
vi. *choice_add* (new choice to be added)
vii. *priority_new* (priority of new choice added)

3.2.1.2.2 Function Signature:

i. *fill_choice(userID,choice_list,priority_list,deadline_date);*

ii. *edit_choice(usrID,choice_add,priority_new,deadline_date);*

iii. *delete_choice(userID,choice_to_delete,deadline_date);*

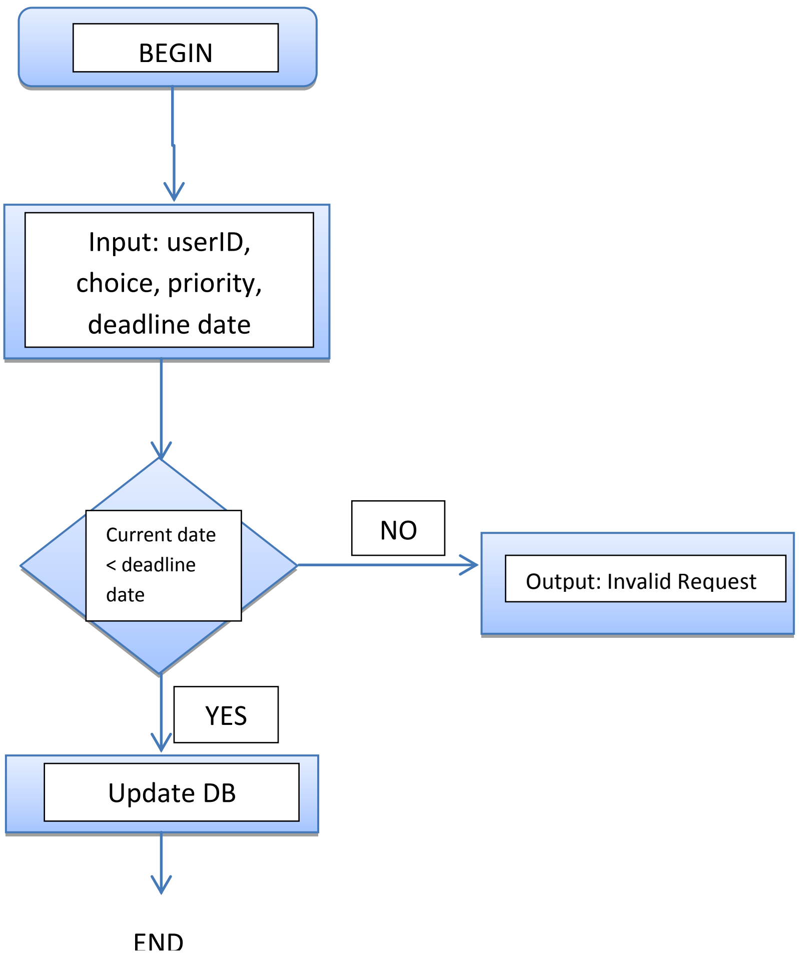
3.2.1.2.3 Input/Output:

i. *fill_choice();* = As input, it will take userID (user's identification), the list of his/her course choices, user's priority amongst the course choices and the deadline date. Its output will be in the form of update of the DB entry corresponding to the user (create the entry and fill it).

ii. *edit_choice();* = As input, it will take userID (user's identification), the new choice to be added, user's priority of the new choice and the deadline date. Its output will be in the form of update of the DB entry corresponding to the user ,i.e add the new choice and update the priority of existing choices.

iii. *Delete_choice();* = As input, it will take userID (user's identification), the choice to be deleted, and the deadline date. Its output will be in the form of update of the DB entry corresponding to the user i.e, delete the choice and update the priority of existing choices

3.2.1.3 Data Structure/DB and Algorithms:


BEGIN
Input: userID, choice, priority, deadline date
Current date < deadline date
NO
Output: Invalid Request
YES
Update DB
END

Use *hash-map* and robust *hash-table function* that minimise collision using userID as the key. The hash-map is used for updating and creating entry into the database. An index will be kept by the name of the courses in the database and it will point to the list of students applied for the course. Use *unique identifier* for each course programme such as CS for computer science. Use a proper *structure* based data structure for storing different choices of a user. It will allow users to add/edit choices till the deadline date only. This module will use dedicated DB of the e-Counselling subsystem. It will update and retrieve data through *SQL queries.*

3.2.1.4 Dependencies: It will depend on the UI subsystem for getting input from the user (list of choices etc.) and the Login/Account subsystem for getting the right userID. It will also need e-Counselling dedicated DB for adding/editing/deleting of choices.

3.2.2 Allot Course programmes

3.2.2.1 Purpose: Once the deadline date has been reached and after the administrator has validated for the allotment of courses, it will run its algorithm to either allot courses to users or signal them regret, and display the same through UI.

3.2.2.2 Interface Specification:

3.2.2.2.1 Parameter: : i. *userID*
ii. *deadline_date*
iii. *admin_signal*

3.2.2.2.2 Function Signature:

i. *allot_courses(admin_signal, deadline_date);*

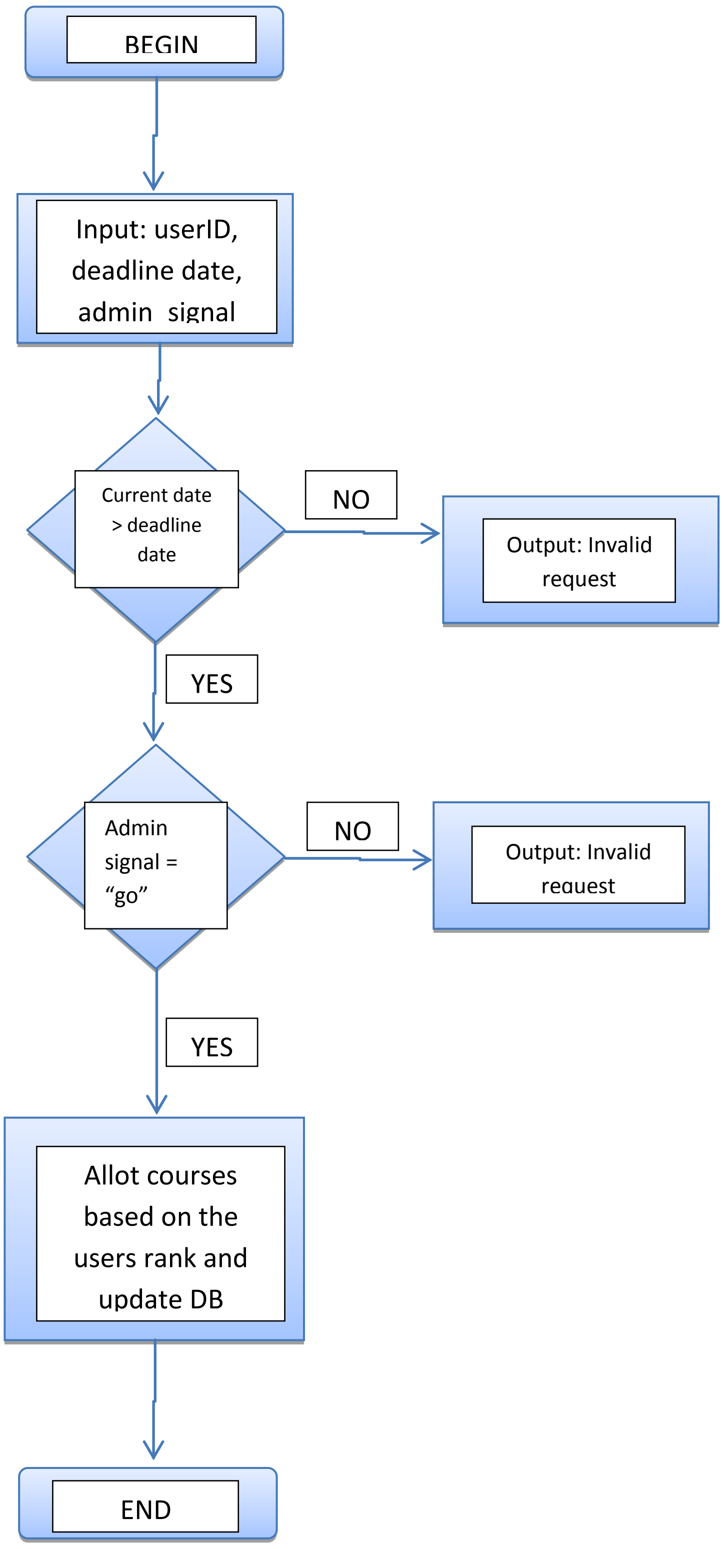
ii. *user_status(userID, deadline_date);*

3.2.2.2.3 Input/Output:

i. *allot_courses();* = As input, it will take the administrator signal (whether or not to allot courses) and deadline date. Its output will be in the form of update of the DB entry corresponding to the user i.e. against each user's entry it will either fill in *identifier* of courses if the course is allotted or signal *regret* if its not.

ii. *user_status();* = As input, it will take the userID (user's identification) and the deadline date. Its output will be either course *identifier* or *regret,* or *invalid request* if the deadline date has not been reached.

3.2.2.3 Data Structures/DB and Algorithms: First the function should check whether deadline date has been reached or not. If its reached then it will check whether the signal is *go* or *not go*. If its 'go', then it map the preferences of users based on their *ranks* as priority. Ranks are provided beforehand, and it exists in the database. It will keep the entry of users in the database in sorted order (on basis of ranks), so that implementing *higher-rank basis* becomes easier. This module will also use dedicated DB of the e-Counselling subsystem. It will update and retrieve data through *SQL queries.*


BEGIN
Input: userID, deadline date, admin signal
Current date > deadline date
NO
Output: Invalid request
YES
Admin signal = "go"
NO
Output: Invalid request
YES
Allot courses based on the users rank and update DB
END

3.2.2.4 Dependencies: It will depend on the UI subsystem for getting input as well displaying the output to the user, and the Login/Account subsystem for getting the right userID. It will also need e-Counselling dedicated DB for updating user's status.

3.3 FAQ

3.3.1 Purpose: It will provide users with the information about "how to use" web portal and also the information relating to the institute. It will be a static webpage with the editing rights lying with the administrator.

3.3.2 Interface Specification

3.3.2.1 Parameter: i. *validID*
ii. *update_data*

3.3.2.2 Function Signature: i. *edit_FAQ(validID, update_data)*

3.3.2.3 Input/Output: i. *edit_FAQ();* = As input it will take *validID* (which tells whether or not the user trying to edit is admin) and the data to be updated in the webpage. Its output will be update of the DB form where this webpage derives it data.

3.3.3 Data Structures/DB and Algorithms: During the edit of FAQ webpage, the module will check if the user trying to edit corresponds to admin or not. If yes then it will simply update the FAQ dedicated DB. This webpage being static does not need any rigorous algorithm and processing.

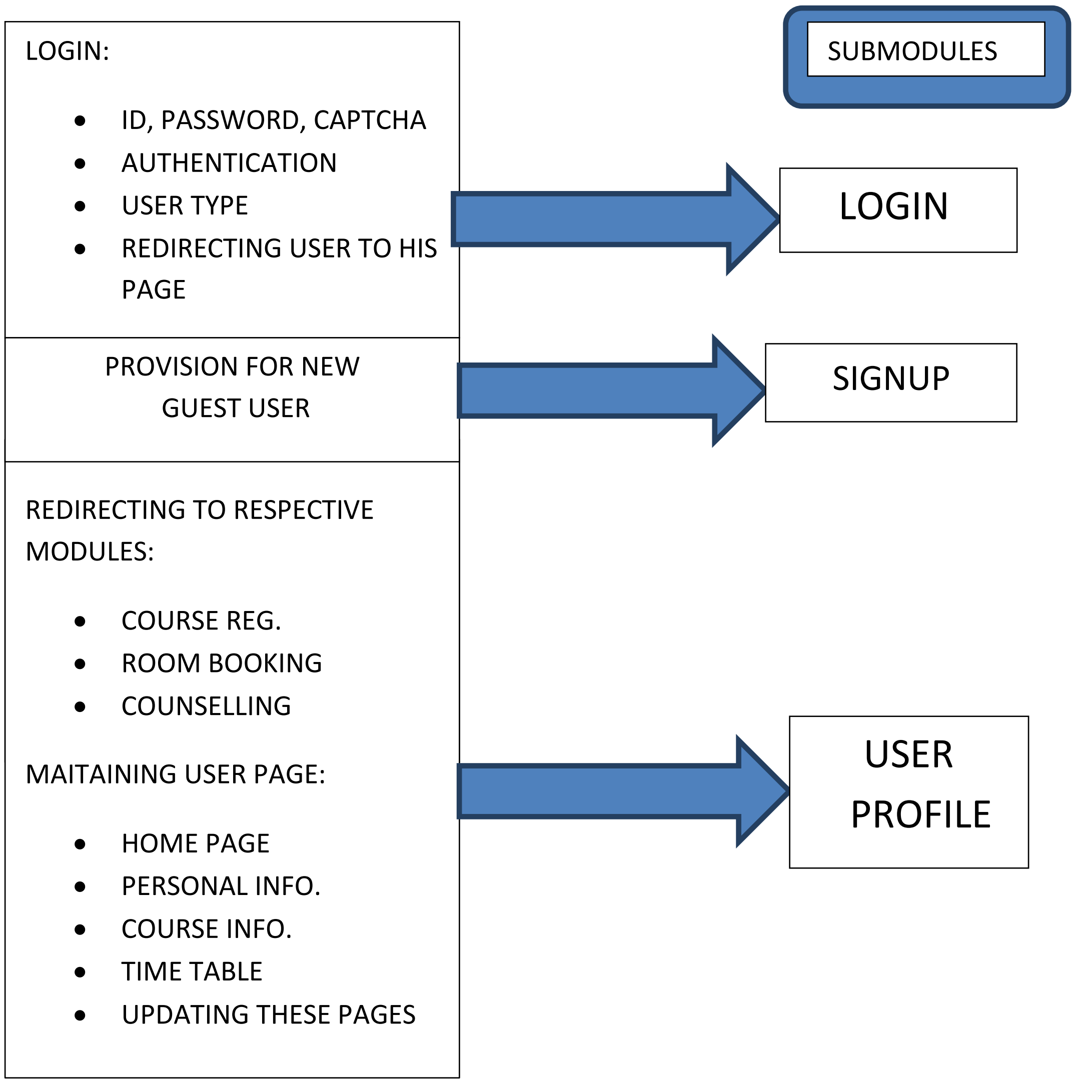
3.3.4 Dependencies: It will depend on the UI and Login/Account subsystem for getting input from the user and for checking access rights to edit privilege respectively.

3.3 LOGIN

MODULE INTRODUCTION: It will be the main module dealing with user interaction. This will be the module which will deal with user login (for existing users); authentication of user ID, password and Captcha (after filling in the details); provision for new user to register and make ID; redirecting the user to their page depending on user's type (e.g. prof., student, etc.); maintaining user interface (page); redirecting user to requested modules by making call to the same. It will basically have three sub modules namely:

- LOGIN (3.3.1)
- USER PROFILE (3.3.2)
- CALL OTHER MODULE (3.3.3)

This module is having many sub-modules and thus the interface will be explained in each of the separately. Moreover this module will be the overall user interacting module and thus will be described in detail in each sub modules.



3.3.1 LOGIN

3.3.1.1 PURPOSE

It will allow user to fill in the required details and login into the system. It will deal with the authentication check and redirection to designated user account. It will maintain its own database and will be independent of afterward modules.

3.3.1.2 INTERFACE SPECIFICATION

Following are the parameters and the calls:

3.3.1.2.1 PARAMETER:

i. User_ID (not case sensitive).
ii. User_password (case sensitive).
iii. Captcha text.
iv. User_click.

v. User_information (for new users).

3.3.1.2.2 FUNCTION SIGNATURE:

i. Login(User_ID, User_password, Captcha).
    a. Check_captcha(Captcha_text, Captch_picture).
    b. Check_authentication(User_ID, User_password).
    c. Display_error (error type).
    d. Redirect_user_to_profile (User_ID).

3.3.1.2.3 INPUT/OUTPUT

i. Check_Captcha(): As an input it will take text from the user and a Captcha picture from the system. It will then match them. The output will be a call to Display_error (Captcha_error) or call check_authentication().
ii. Check_authentication(): As an input it will take User_ID and User_password from the user. User_ID and user_password may be anything e.g. text, symbols, numbers, etc. It will check he database for those. The output will be a call to Display_error(User does not exist) or Display_error (wrong password) or successful login and call to redirect_user_to_profile.

iii. Display_error(): This function will be an internal function of the system and it will be called by above functions. As an input it will take the error type. As an output it will display the required error message and redirect the user to the login page.
iv. Redirect_user_to_profile(): It will take as an input the result of Check_captch() and Check_authentication(). If they return true it will redirect user to his/her profile else return them to loin page.

3.3.1.3 DATA STRUCTURE/DB AND ALGORITHM

After the user presses the Login button a function will be called which will make an object of all the information given by user. Data will be stored in the database in tables having rows for different users. In each row first column will be user ID followed by password and the last column will tell whether the user is student or faculty or guest. There will be a provision for new entries to be added.

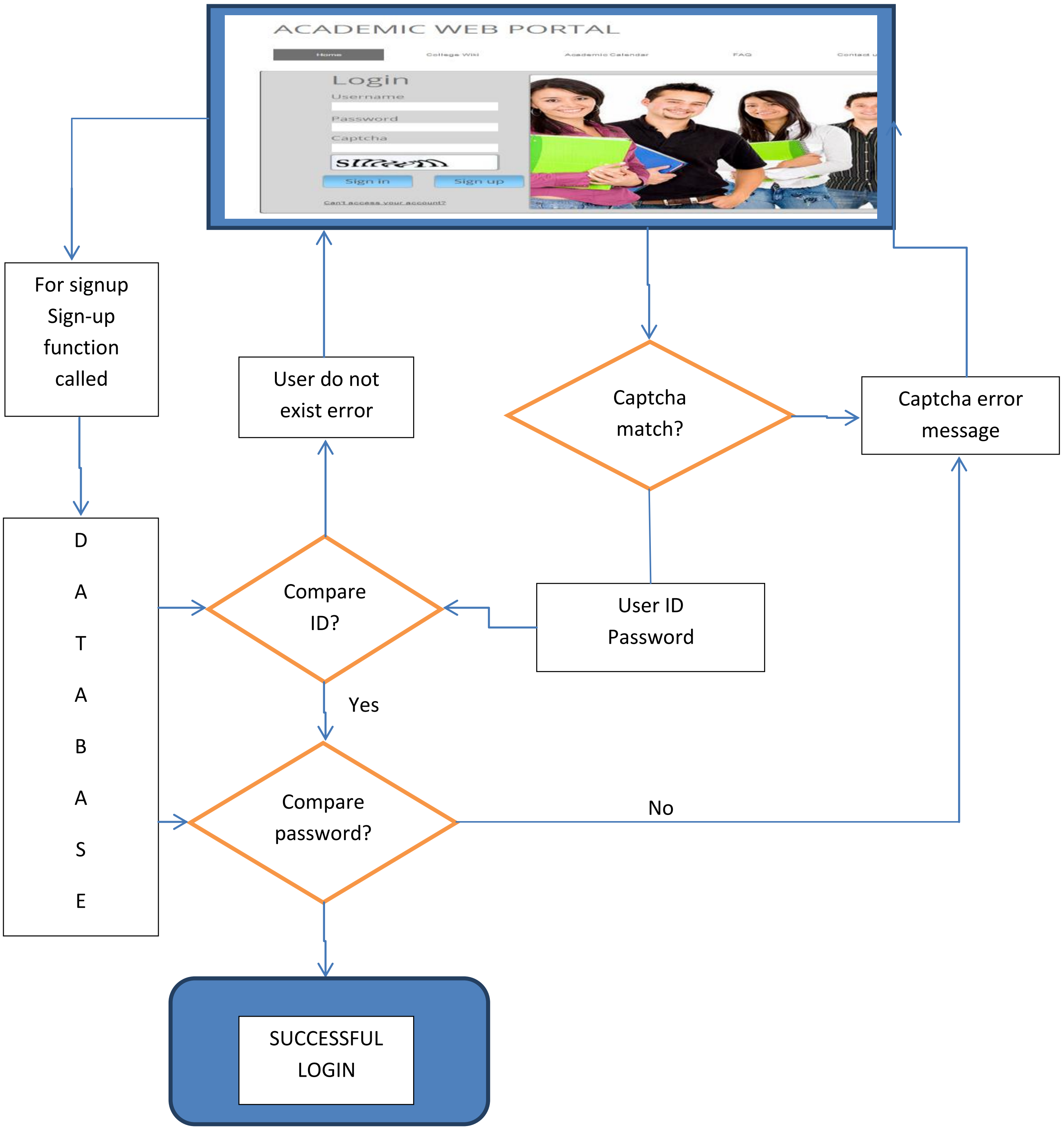
Database example:

USER ID	PASSWORD	USER TYPE
2010cs10209	87rtgrd@445	student
Cs192836	9dhu%#12	faculty
Sh20390	Jkxfklvn008	guest

The first column will be checked for matching with the user ID:

- If a match exists, then password column will be checked and if it matches then depending upon the Captcha match the user will be granted access.
- If the user ID does not match then the user will be asked to re-enter user ID or signup.
- If the user ID matches but password does not then he/she will be redirected to login page with an error message.

The administrator will have the right to delete any entry from the database.


ACADEMIC WEB PORTAL
Home
College Wiki
Academic Calendar
FAQ
Login
Username
Password
Captcha
Sign in
Sign up
Can't access your account?
For signup Sign-up function called
User do not exist error
Captcha match?
Captcha error message
D A T A B A S E
Compare ID?
User ID Password
Yes
Compare password?
No
SUCCESSFUL LOGIN

3.3.1.4 DEPENDENCIES

This sub-module will depend upon main page module (as it is called from that module). It also depends upon the proper functioning of other modules like e-counselling, student course registration, etc.

3.3.2 SIGNUP

3.3.2.1 PURPOSE

This sub-module will deal with the registration of new user into the system. A new user can only form a guest account for e-counselling purpose. Only an administrator can add a new user (faculty or student). This module will handle addition of new user information into database and management of new user account.

3.3.2.2 INTERFACE SPECIFICATION

3.3.2.2.1 PARAMETER

i. User_type.
ii. User_name.
iii. User_information(including all the information system needs).
iv. User_click.

3.3.2.2.2 FUNCTION SIGNATURE

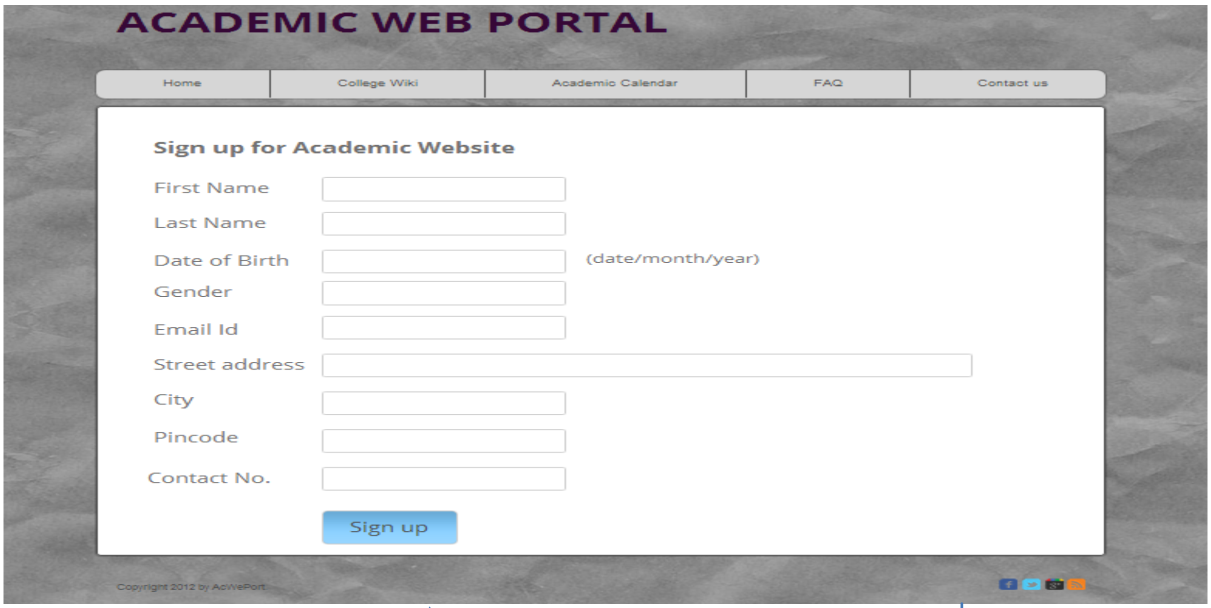
i. Fill_data(user information).
ii. Check_database(user ID).
iii. Update_database(user information).

3.3.2.2.3 INPUT/OUTPUT

i. Fill_data(): As an output it will show fields to user to enter data and as an input it will take data from user.
ii. Check_database(): it will take the user information and check the database for the User_ID the user has provided and and will return error message if it already exist.
iii. Update_database(): It will take user information and update the Database with these entries.

3.3.2.2.4 DATA STRUCTURE/DB AND ALGORITHM

First the function will prompt the user to fill in the personal information. Then it will ask for user ID choice and password. After the user has filled all the information an object of these data will be formed (compatible with the database entry). The user ID provided will be matched with ID entries in the database and if a match is found error message will be shown and user will be redirected to fill in the details again otherwise the data will be stored in the database and the user will be shown the message that the registration is complete.



ACADEMIC WEB PORTAL

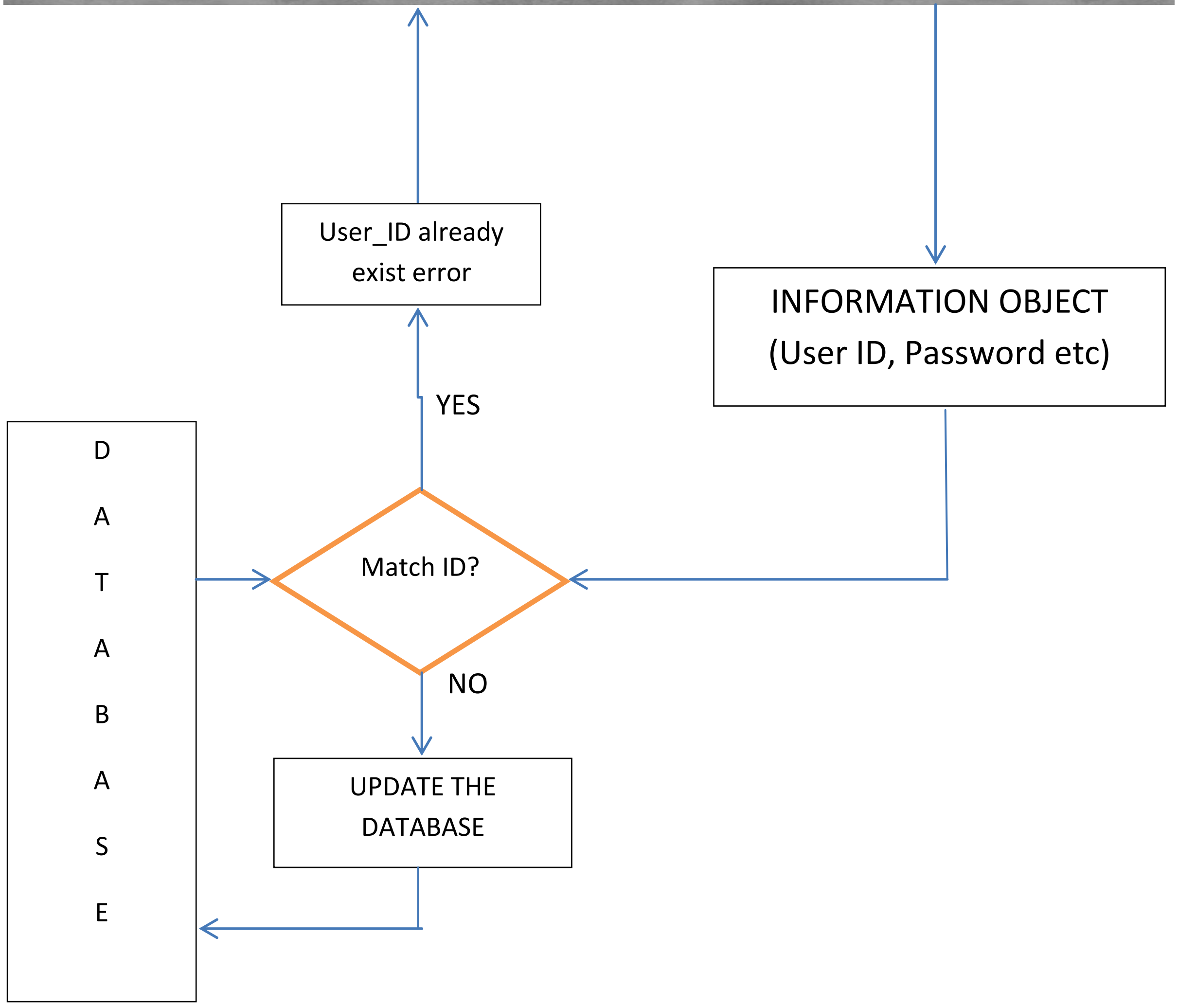
Home | College Wiki | Academic Calendar | FAQ | Contact us

Sign up for Academic Website

First Name	
Last Name	
Date of Birth	(date/month/year)
Gender	
Email Id	
Street address	
City	
Pincode	
Contact No.	

Sign up

Copyright 2012 by AcWePort


User_ID already exist error
INFORMATION OBJECT (User ID, Password etc)
YES
DATABASE
Match ID?
NO
UPDATE THE DATABASE

3.3.2.2.5 DEPENDENCIES

This sub-module will depend on database management system and the login sub-module as its call and return are dependent on the respectively.

3.3.2 USER PROFILE

3.3.2.1 PURPOSE

After the user has successfully logged in he/she will be redirected towards his/her designated profile. This sub-module is concerned with the management of different user profile pages. Different pages must be updatable and at the same time the information in them must be reliable. They should be secure and easily maintainable for the user.
This sub-module will be called upon after the user has successfully logged in. It will also redirect the user to other modules also, on request.

3.3.2.2 INTERFACE SPECIFICATION

3.3.2.2.1 PARAMETER

i. User_ID.
ii. User_type.
iii. User_response(the user will click on the links according to his/her need).

3.3.2.2.2 FUNCTION SIGNATURE

i. Update(user ID, Field to be updated).
ii. Page_display (user type, user click).
iii. Retrieve_information(User ID, page name).
iv. Call_module (module name).

3.3.2.2.3 INPUT/OUTPUT

i. Update(): This function will take user ID and user entered data given by the user. As an output it will change the database corresponding to the user.
ii. Page_display(): This will take user ID, User type and the corresponding user choice of page which the user want to be displayed.
iii. Retrieve_information(): this will take user’s choice and user ID to retrieve information from the database and will show those data to the user on the screen.
iv. Call_module(): This will take user’s response as an input e.g. if the user wants to go to course registration or room booking (depending upon the user) etc. As an output it will redirect user to the required module by calling the same.

3.3.2.2.4 DATA STRUCTURE/DB AND ALGORITHM

Once the user login he/she will be redirected to his/her home page giving the general information. Depending upon the user's choice pages will be shown to him and if he/she wishes to change some or any information same will be provided to him/her. The user may call upon the links e.g.

- Course information page: Containing the links of courses the user is currently enrolled in.
- Personal information page: Containing personal information of the user.
- Time table: weekly and monthly format. Etc.

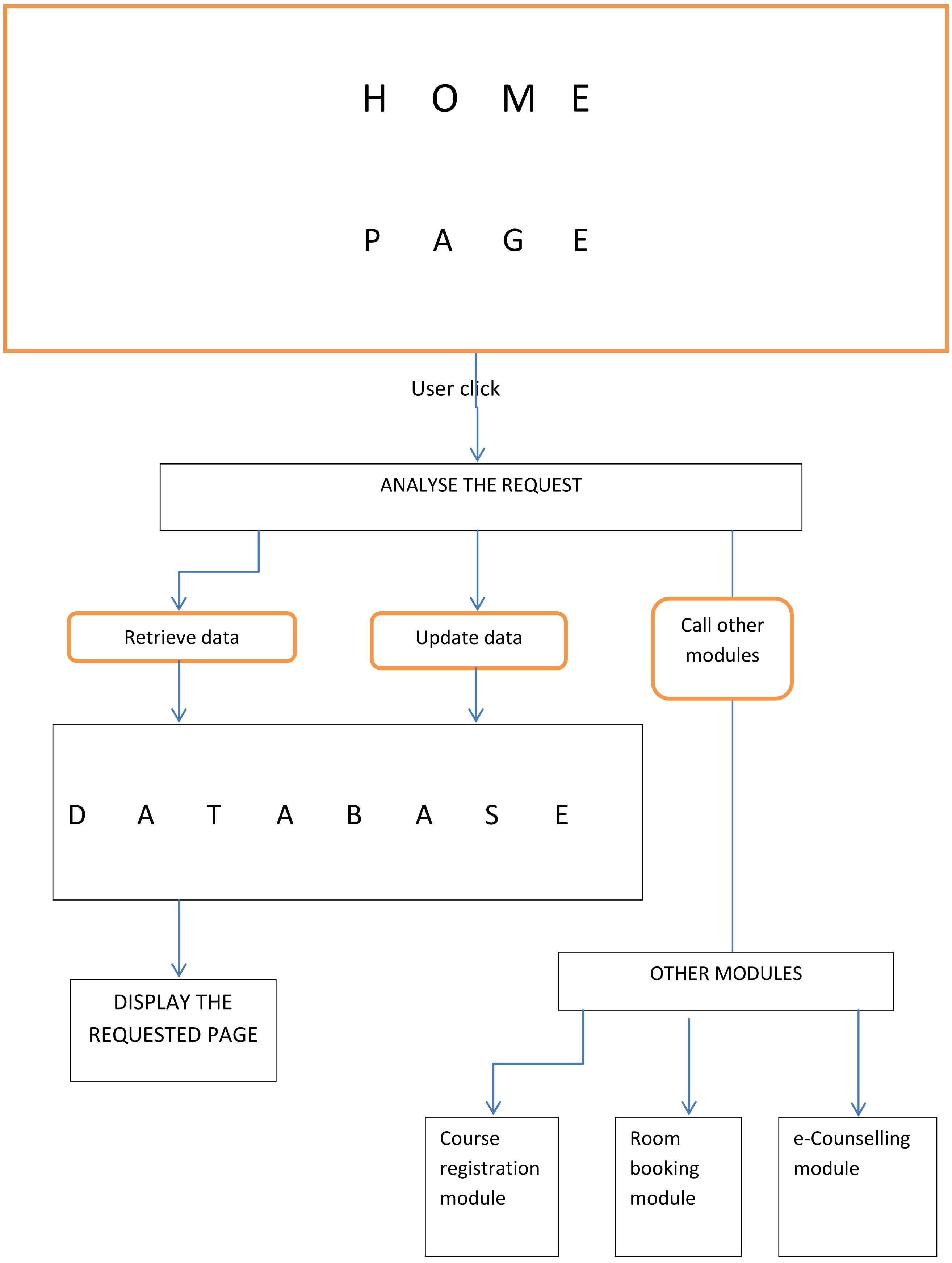
The processing of this module is basically web based and it will respond to user stimulus. It will deal with the maintaining and managing more related to the web part i.e. the interface with the user.
The Database will be simple tables containing the user information and any time a function call is made the information will be displayed on the screen. This part basically deals with retrieval and management of information/data.

3.3.2.2.5 DEPENDENCIES

This sub module will depends upon basically the database and the interface management sub system. Its processing is basically the web based management and retrieval of information. It will be the main user interaction interface. It will provide the main platform for the user to interact with other modules. Its processing will also depends upon the database of previous sub-module i.e. login database to make the decision whether the user is a faculty member or a student or a guest user because:

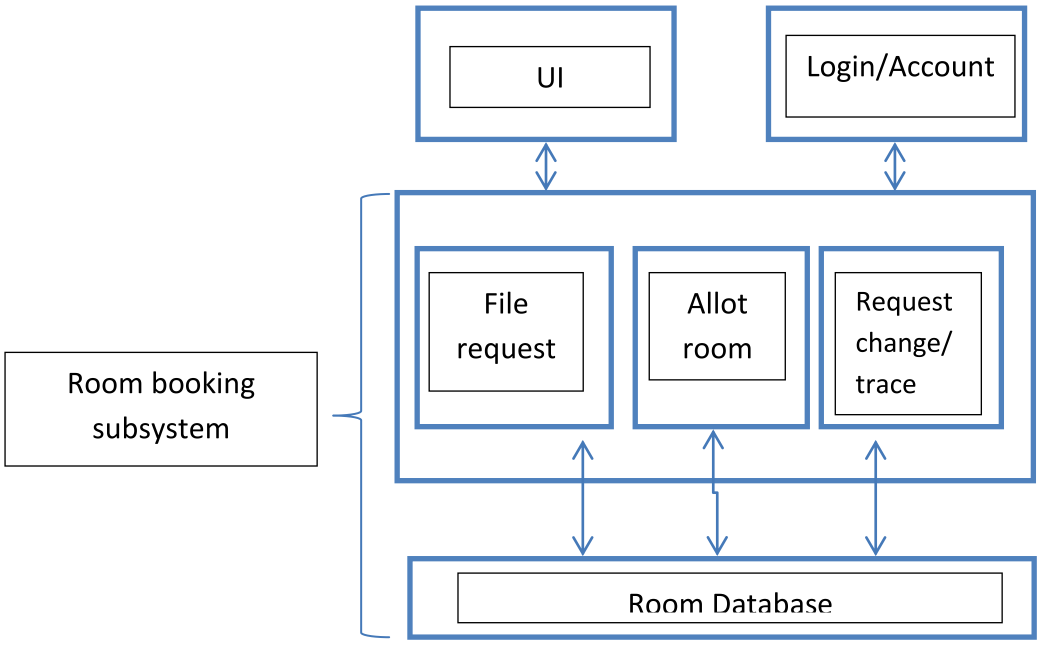
- A student cannot call the room booking module and e-Counselling module.
- A faculty cannot call the course registration module and e-Counselling module.
- A guest user does not have rights to maintain a permanent home page and could not either call a room booking module and course registration module.

Locks and other safety and security measures will also be considered so as to avoid collision and maintain the integrity of the system.


H O M E
P A G E
User click
ANALYSE THE REQUEST
Retrieve data
Update data
Call other modules
D A T A B A S E
DISPLAY THE REQUESTED PAGE
OTHER MODULES
Course registration module
Room booking module
e-Counselling module

3.4 ROOM BOOKING

This module will help the user (faculty) to make request for booking room. This module will provide the user facilities like room booking with specified requirements; trace his/her request status; change request or cancel request. It will also provide facilities like daemon server to operate 24*7 and mutex lock to ensure integrity.



This module will comprise of two sub modules to ease work load and do the same more efficiently. Both the module will work together and will depend upon each other for proper functioning. Sub modules are:

- File request (3.4.1)
- Allot room(3.4.2)
- Request change/trace (3.4.3)

3.4.1 FILE REQUEST

3.4.1.1 PURPOSE

It will allow user to fill in his/her requirement in required fields and send the request to administrator for approval. The user can at any time change or modify the request and can even delete the request. He/she can also track the request status and can even place a priority request.

3.4.1.2 INTERFACE SPECIFICATION

Following are the parameter and call:

3.4.1.2.1 PARAMETER

i. Student capacity
ii. Non-technical requirement (e.g. projector, black board, etc.)
iii. Technical requirement (e.g. microphone, technician, etc.)
iv. Date/Dates
v. Time

3.4.1.2.3 FUNCTION SIGNATURE

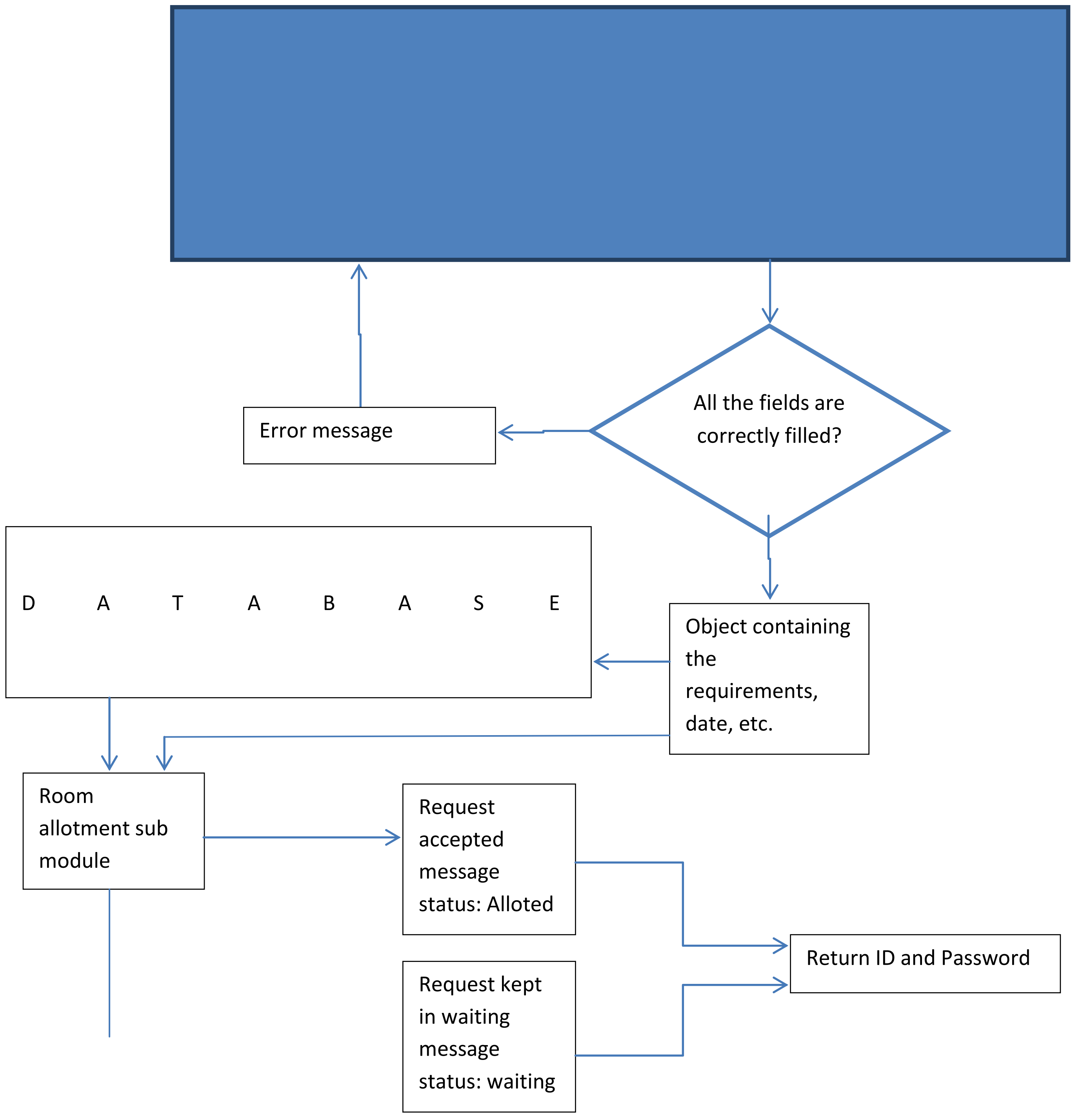
i. Fill_choice(student number, NT requirement, T requirement, date/dates, time).
ii. Store_in_db(student number, NT requirement, T requirement, date/dates, time).
iii. Generate_userid(student number, NT requirement, T requirement, date/dates, time).
iv. Return_userid().

3.4.1.2.4 INPUT/OUTPUT

i. Fill_choice(): As an input this function will take the user's requirement. All the requirements filled by the users will be recorded and an object will be formed of the same. As an output this function will either call the next function i.e. store_in _db() or may return an error message that the user's has not filled/or wrongly filled some choices.
ii. Store_in_db(): This function will take as an input the above mentioned requirements and will write it in the database. It will form a file for each date and enter all the request corresponding to that date in that file. The output of this function will be the corresponding entry in the file.
iii. Generate_userid(): this will be called by above function and will generate an automated ID for the user for further references. It will output a user ID.
iv. Return_userid(): This will return a userid and will also tell the user that his/her choice has been filed. This will interact with next sub-module (room allot) and will also tell the user whether his/her request is accepted or is kept in waiting.

3.4.1.3 DATASTRUCTURE/DB AND ALGORITHM

File system will be used to maintain the database. Corresponding to each day a folder will be maintained and in each of them the request for room booking and availability of room properties will be listed. Each time a user enter some request it will be written in there generating a userid. After the user place a request an object of this information will be formed and stored in the database. Efficient use of resources will be ensured and lock mechanism will be used to ensure safety and security. Use a proper *structure* based data structure for storing different choices of a user. It will allow users to add/edit choices till the deadline date only. This module will use dedicated DB of the room booking subsystem. It will update and retrieve data through *SQL queries.*



All the fields are correctly filled?
Error message
D A T A B A S E
Object containing the requirements, date, etc.
Room allotment sub module
Request accepted message status: Alloted
Request kept in waiting message status: waiting
Return ID and Password

3.4.1.4 DEPENDENCIES

This sub-module will depend upon the Room allotment sub-module as its output will heavily rely on that. Both of them will also maintain the same DATABASE. **:** It will depend on the UI and Login/Account subsystem for getting input from the user and for checking access rights to edit privilege respectively.

3.4.2 ALLOT ROOM

3.4.2.1 PURPOSE

This sub-module will deal with allotment of rooms to the registered user. It will automatically check for the most efficient allotment and will notify the user for the same. This sub-module will be called upon by other sub-modules (filling choice and changing/deleting request) as it in both of them this sub-module will be applied again. An extra provision will be made for administrator to check/change/delete any request.

3.4.2.2 INTERFACE SPECIFICATION

3.4.2.2.1 PARAMETER

i. Student capacity
ii. Non-technical requirement (e.g. projector, black board, etc.)
iii. Technical requirement (e.g. microphone, technician, etc.)
iv. Date/Dates
v. Time
vi. Database
vii. userid

3.4.2.2.2 FUNCTION SIGNATURE

i. Check_availability(student number, NT requirement, T requirement, date/dates, time,database)
ii. Update_databse (student number, NT requirement, T requirement, date/dates, time,database,status).
iii. Return_status(student number, NT requirement, T requirement, date/dates, time,database).

3.4.2.2.3 INPUT/OUTPUT

i. Check_availability():As an input this function will take the user's requirement and database and will check whether the request could be honoured or not. If not then it will output true and will call update_database else it will write the request and will return waiting.

ii. Update_databsase():As an input this function will take the user's requirement and database and will change the database as output. This will in turn call the return status function and will return the request status.

iii. Return_status():As an input this function will take the user's requirement, user ID (for authentication) and database and will return the status of the user.

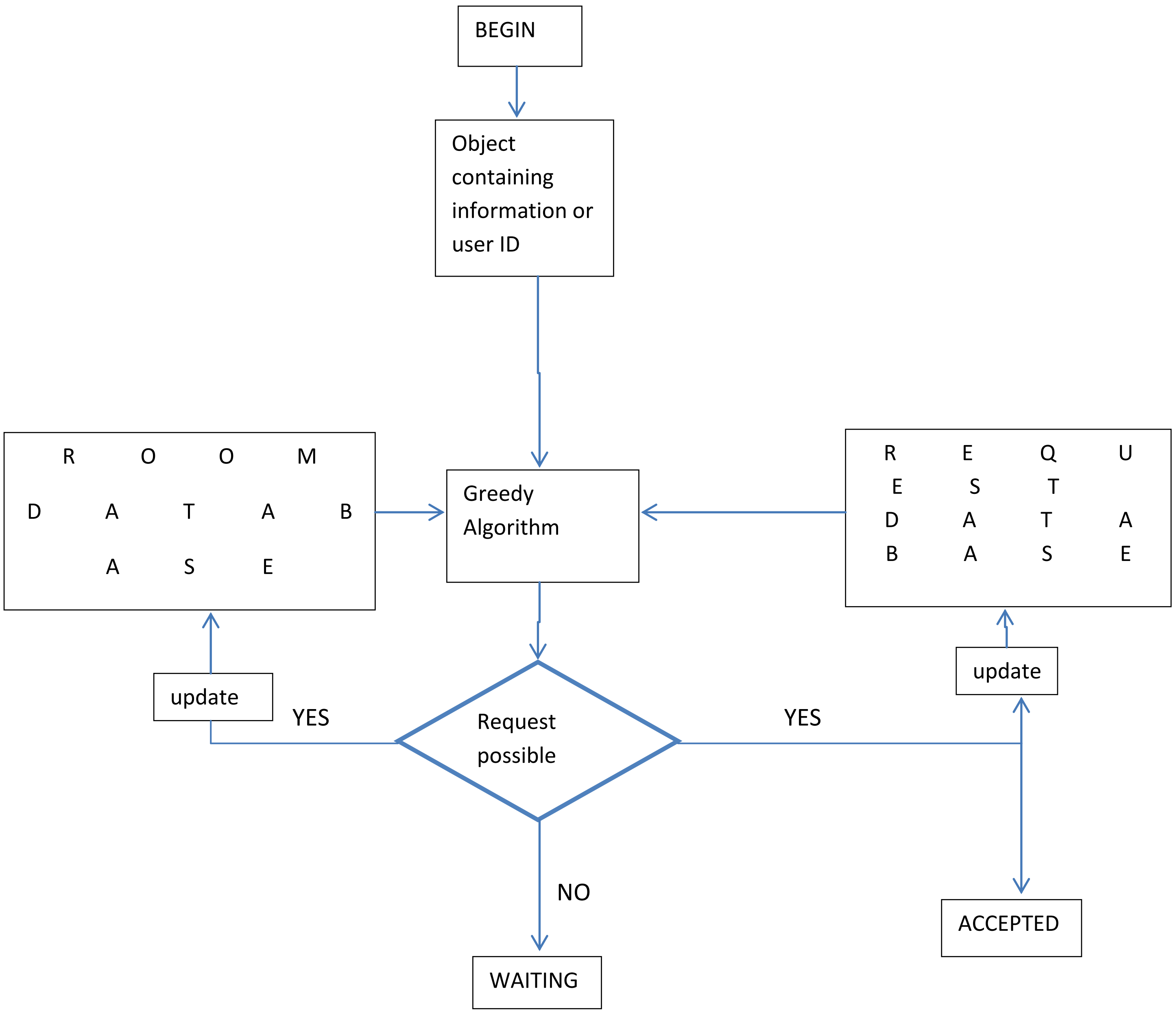
3.4.2.3 DATASTRUCTURE/DB AND ALGORITHM

This module deals with the processing of information and thus does not specifically require any data structure but it will deal with objects, database and will have its own UI. The database will be same as above and it will make use of it without changing its format. At times it will also create some of the new folders if the user change the date and a folder of that date does not exist.

The algorithm used will be "*GREEDY ALGORITHM*". At each time this sub-module is called it will apply greedy algorithm on the database and will try to fulfil as much request as possible giving preference to first come first serve. In the algorithm the user entry will be matched for availability in the database:

- If a slot is available with fulfilling all the requirements the database will be changed and the user id along with status will be returned. Database will also be changed accordingly. The user can at any time in the future see his/her request status as well as change it.
- If a slot is not possible the request will be stored and a unique user id will be provided. Each time this module will be called it will run the algorithm and will try to accommodate the request. It will then return the status. If a user changes any request or any request get deleted it will run the algorithm with remaining waiting requests and will try to fulfil them.

It will have special provision for administrator as he/she can change/delete/allot request out of order.



3.4.2.3 DEPENDECIES

This module will depend upon both of the other two modules as it will take input from both of them. And it will depend upon the UI of them.

3.4.3 REQUEST CHANGE/TRACE

3.4.3.1 PURPOSE

This sub-module aimed to provide user extra features like change the request or delete the request or even can trace the request (whether granted or waiting). This will deal with changing the database accordingly and again calling the allot room sub module to again apply the greedy algorithm on the changed database and user requirement.

3.4.3.2 INTERFACE SPECIFICATION

3.4.3.2.1 PARAMETER

i. Student capacity
ii. Non-technical requirement (e.g. projector, black board, etc.)
iii. Technical requirement (e.g. microphone, technician, etc.)
iv. Date/Dates
v. Time
vi. Database
vii. Userid

3.4.3.2.2 FUNCTION SIGNATURE

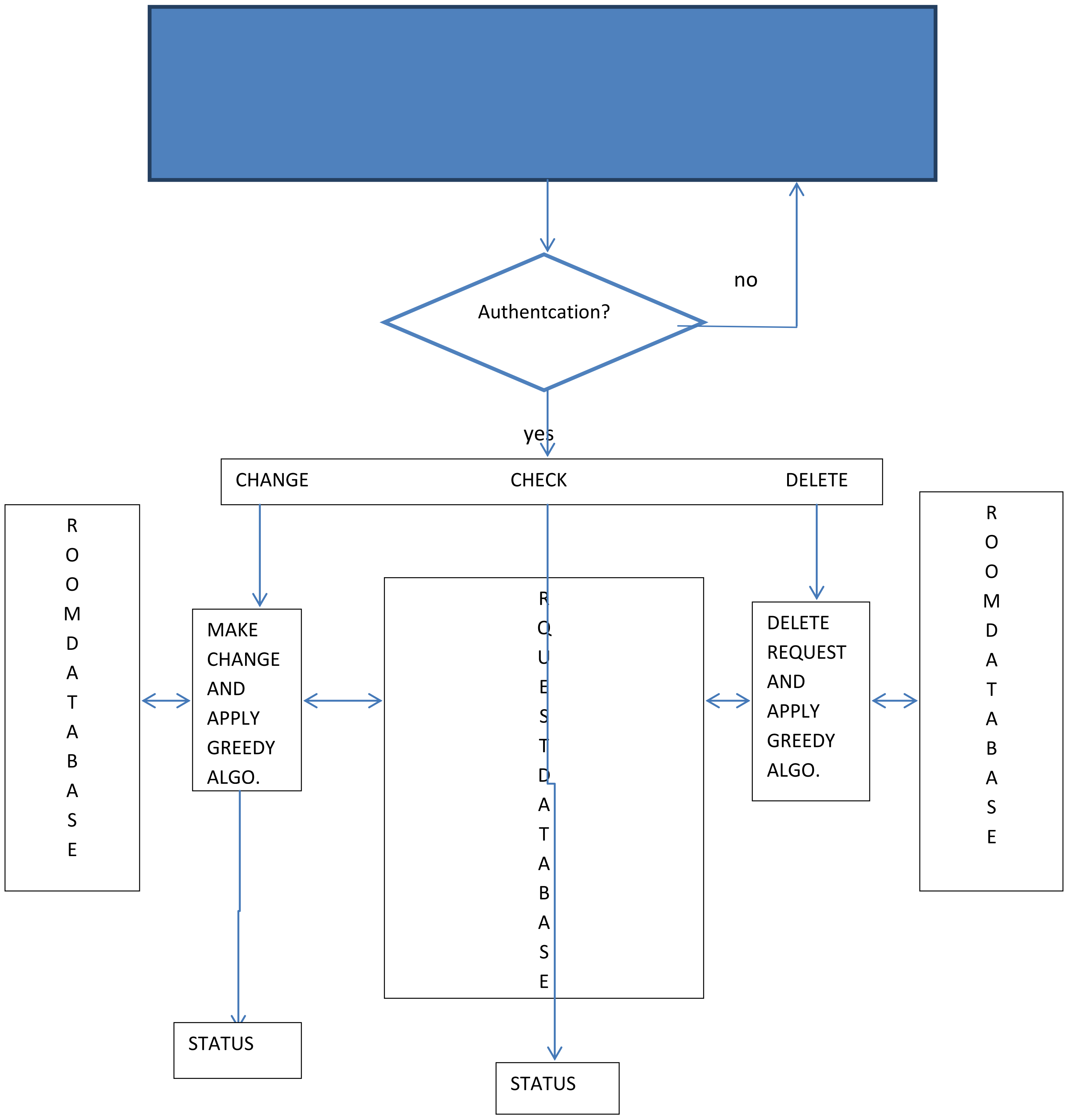
i. Check_status(userID,password).
ii. Change_request(user_ID,password,student number, NT requirement, T requirement, date/dates, time,database).
iii. Delete_request(user_id,password)

3.4.3.2.3 INPUT/OUTPUT

i. Check_status(): This will take user ID as input and will check the status of that ID from the request database. It will output the status of that user.
ii. Change_request(): This will take the user ID and the new requirements from the user. It will then change both the database and update the request status of corresponding users and will notify the user about the same.
iii. Delete_request(): It will take the user ID and password and will delete the user request from request database and change the room database as well and will again run the greedy algorithm.

3.4.3.3 DATABASE/DB AND ALGORITHM

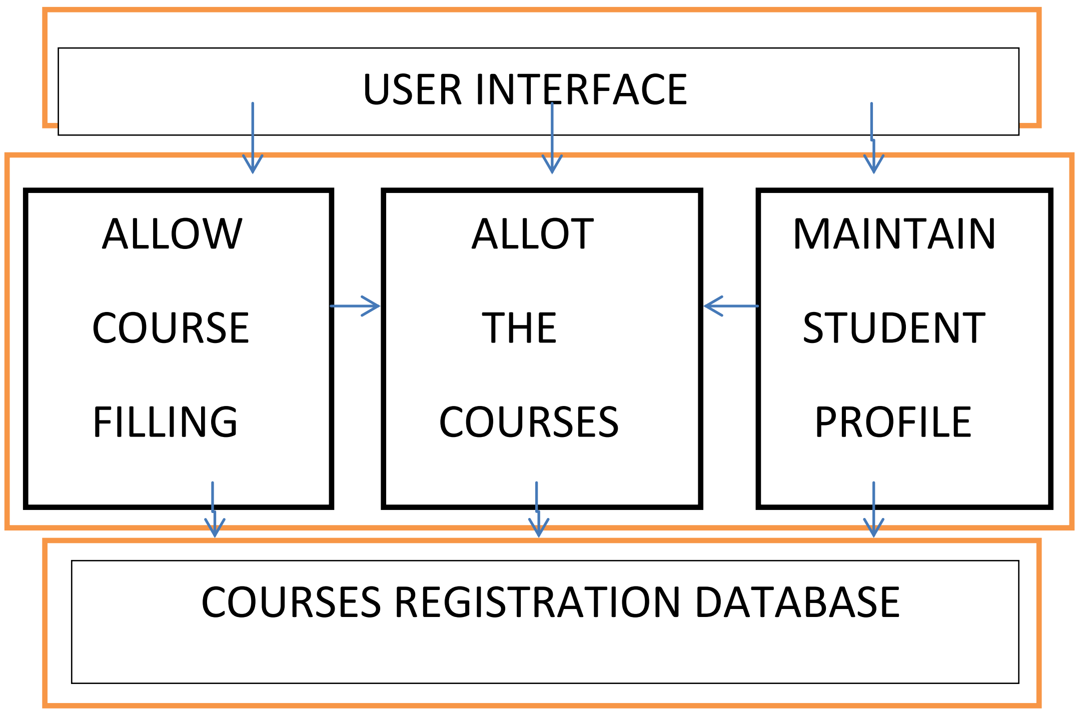
The database used will be the same as above and the datastrures will also be the same. It will make object from the information given by the user and then will match it with the database and change the database accordingly.



3.5 COURSE REGISTRATION

The course registration will allow the students to register the courses which they want to do in coming semester. After a fixed deadline it will distribute the courses to the students. It will also maintain a database of the profile of each student which keeps the performance of the student in their previous semesters. Following three modules have been identified for this subsystem.

- Allow choice filling
- Allot the courses
- Maintain the student profile



Allot the courses module has to information from both Allow course filling and student profile. Following is the detailed description of all three sub modules.

3.5.1 Allow filling courses

3.5.1.1 Purpose

It will allow users to fill the courses that they want to do in their next semester. The choices can be made either by selecting from a list or by writing the course name (depending upon the UI subsystem).

3.5.1.2 Interface specification

3.5.1.2.1 Parameters

1. USER ID
2. Choice list
3. Deadline date
4. Date of submission

3.5.1.2.2 Function signature

1. fill_choice (user id, choice list, deadline date, date of submission)
2. edit_choice (user id, choice list, deadline date, date of submission)

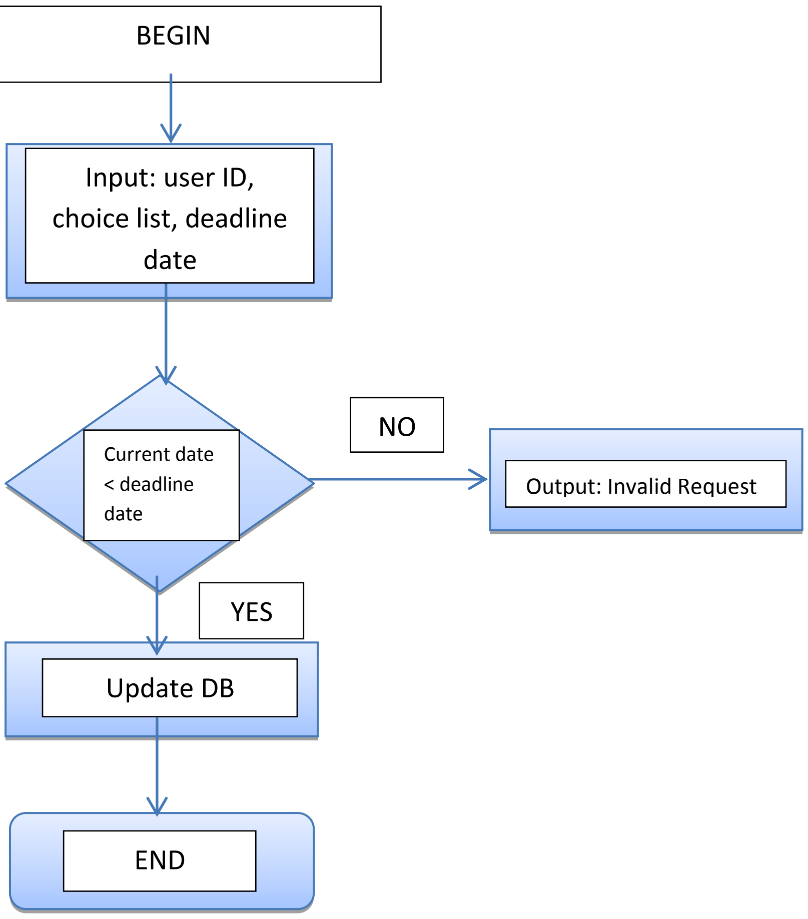
3.5.1.2.3 Call specification

1. fill_choice- it will take user id, the list of courses filled by the students, date of submission and deadline date and the output will the new entry in the database corresponding to that user id.

2. edit_choice-it will also take the same parameters as fill_choice() but the output will be the update in the existing entry in the database corresponding to that user id.

3.5.1.3 Data Structure/DB and Algorithms

Use *hash-map* and robust *hash-table function* that minimise collision using user ID as the key. The hash-map will be used for updating and creating entry into the database dedicated to the course registration. An index will be kept by the name of the courses in the database and it will point to the list of students applied for the course. Use *unique identifier* for each course such as CSL101 for introduction to computer science. Use a proper *structure* based data structure for storing different choices of a user. It will allow users to add/edit choices till the deadline date only. It will update and retrieve data through *SQL queries.*


BEGIN
Input: user ID, choice list, deadline date
Current date < deadline date
NO
Output: Invalid Request
YES
Update DB
END

3.5.1.4 Dependencies:

It will depend on the UI subsystem for getting input from the user (list of choices etc.) and the Login/Account subsystem for getting the right user ID. It will also need course registration dedicated DB for adding/editing/deleting of choices.

3.5.2 Allot the courses

3.5.2.1 Purpose:

After getting the requests and the deadline date this module of the subsystem run an algorithm and distribute the courses to the students after validating it from authorities.

3.5.2.2 Interface specification

3.5.2.2.1 Parameter:

1. *User ID*

2. *Deadline date*

3. *Admin signal*

4. Mode of allotment

3.5.2.2.2 **Function Signature:**

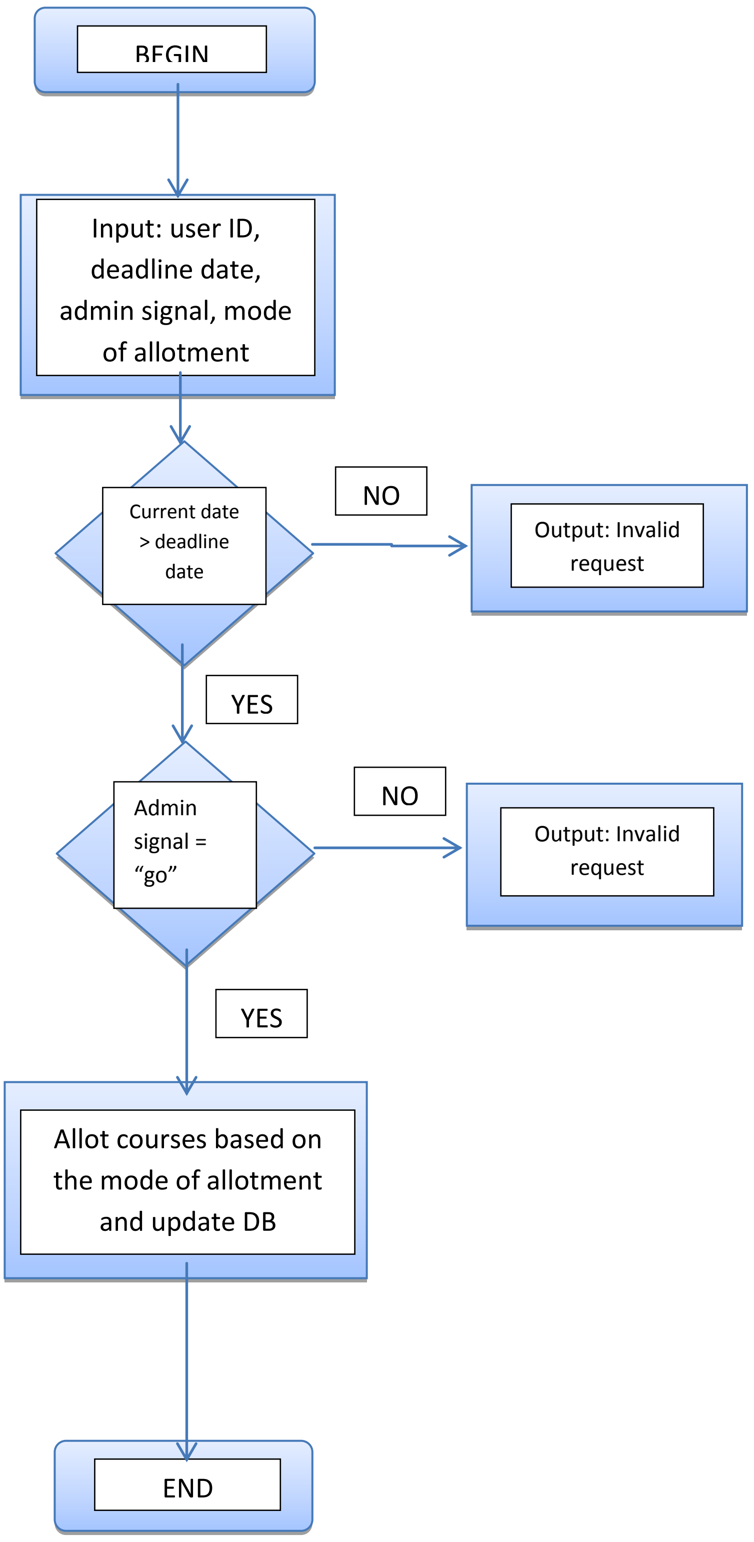
1. allot_courses (admin signal, deadline date, mode of allotment)
2. user_status (user ID, deadline date)

3.5.2.2.3 **Call Specification:**

3.5.2.2.3.1 allot_courses - As input, it will take the administrator signal (whether or not to allot courses), mode of allotment and deadline date. Its output will be in the form of update of the DB entry corresponding to the user i.e. against each user's entry it will either fill in *identifier* of courses if the course is allotted or signal *regret* if it's not.

3.5.2.2.3.2 *User_status* - As input, it will take the user ID (user's identification) and the deadline date. Its output will be either course *identifier* or *regret,* or *invalid request* if the deadline date has not been reached.

3.5.2.3 Data Structures/DB and Algorithms:


BEGIN
Input: user ID, deadline date, admin signal, mode of allotment
Current date > deadline date
NO
Output: Invalid request
YES
Admin signal = "go"
NO
Output: Invalid request
YES
Allot courses based on the mode of allotment and update DB
END

First the function should check whether deadline date has been reached or not. If it's reached then it will check whether the signal is go or not go. If it's 'go', then it will check the mode of allotment. If it is on the basis of performances then for each course it will sort on the basis of their academic performances which are maintained in student profile sub module. If it is first come first serve then it will sort on the basis of date of submission. After allotting each course it will update the student profile weather he is getting the course or not. This module will also use dedicated DB of the course registration subsystem. It will update and retrieve data through SQL queries.

For the user status function first it will check the deadline date and if deadline has not been reached then it will show an invalid request. If deadline has been reached then it will go to the index in database according to that user ID and show the details to the user.

3.5.2.4 Dependencies:

It will depend on the UI subsystem for getting input from both students and admin as well displaying the output to the user, and the Login/Account subsystem for getting the right user ID. It will also need course registration dedicated DB for updating user's status.

3.5.3 Maintain the student profiles

3.5.3.1 PURPOSE:

The purpose of this module is to keep a systematic record of performance of each student in their previous academic years. Further the record can be used in various ways such as to distribute the courses for the current year as described above, to distribute the scholarships etc.

3.5.3.2 Interface specification

3.5.3.2.1 Parameters:

1. User ID

2. Course name

3. Grade in that course

3.5.3.2.2 Function signature:

1. update_performance(user id, course, grade)

2. view_performance (user id, course)

3. performance_overall (user id)

3.5.3.2.3 Call specification:

1. Update_performance- as input it will take user id, course name and grade and as output it will update the database according to that user id i.e. it will make a new entry for that course in the courses column and mark the grade in that course in the grades column.

2. view_performance- input of this function will be user id of the student whose performance we want to see and the name of the course in which we want to see his performance. Output of the function is the grade in that course.

3. performance_overall- input of this function is user id of student and output will be the average of grades of all the course of that student.

3.5.3.3 Data Structures/DB and Algorithms:

Use *hash-map* and robust *hash-table function* that minimise collision using user ID as the key. The hash-map will be used for updating and creating entry into the database dedicated to the student profile. It will allow only professors to add/edit the grade in each course. It will update and retrieve data through *SQL queries.*

3.5.3.4 Dependencies:

It will depend upon the UI subsystem to get the inputs from both students and professors and to display the outputs to them. It will also depend upon the Login/Account subsystem for getting the right user ID in the view_performance () and performance_overall () functions. It will also need a dedicated database to maintain and update the student profiles.

3.6 COLLEGE WIKI

3.6.1 Purpose:

The college wiki will allow the students to go through all the information about the college which is available on wiki. It will have information pages on content like college lingo (local language of students of the college), events of college, academic information.

3.6.2 Interface specification:

3.6.2.1 Parameters:

1. User id
2. Keyword of the desired information
3. Information or stuff that user want to share
4. Admin approval

3.6.2.2 Function signature:

1. Read_wiki(keyword of the desired information)
2. Edit_wiki(user id , information that user want to share, Admin approval)

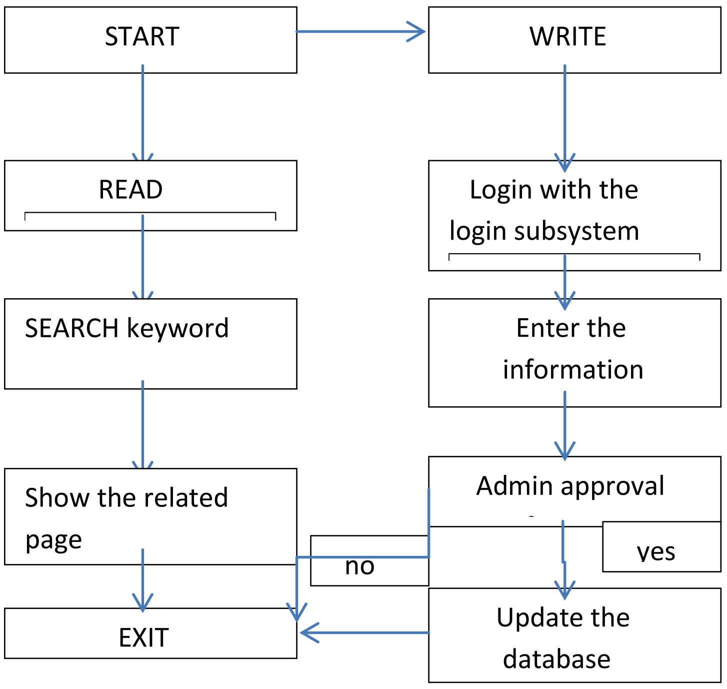
3.6.2.3 Call specifications:

1. Read_wiki()- this function requires the keyword of the information that user want to read and the output will be the page related to that keyword. Note that reading the stuff does not require any user id.
2. Edit_wiki() – the input of this function is user id of the editor, the stuff that the user wants to share and admins approval. The output will be either and editing in some existing page or creation of a new page.

3.6.3 Data Structures/DB and Algorithms:

1. Read: If the user wants to read some data then he needs not to login and he can directly search from the keyword. It will then query the database with the keyword and returned data will be shown.

2. Edit: To edit the available information the user shall first log in to the system via the login subsystem. After logging in, to edit some information the user will first go to the page he want to edit by search and then his user id and data he wants to add/remove will be processed by the administrator. If there is no page corresponding to that information then the function will create a new page. The database will have an index of keywords and ids of each page. Queries will ask with keywords and the keyword will be searched in index which is pointed to the id of the page. ID will be returned and the corresponding page will be available to the user.



3.6.4 Dependencies:

The college wiki module will depend upon the UI subsystem to show and take the information to the users. It will also depend on the login subsystem to take the correct user ID for the function Edit_wiki(). It will have a dedicated database to store the information that the students want to share.

4. Design Decisions

4.1 Use of PHP

We decided to use PHP for our website because it provides for more flexibility in comparison to JSP in terms of design and background processing. We could also have used ASP.NET but decided instead to use PHP, because our team-members are more familiar with PHP than .NET

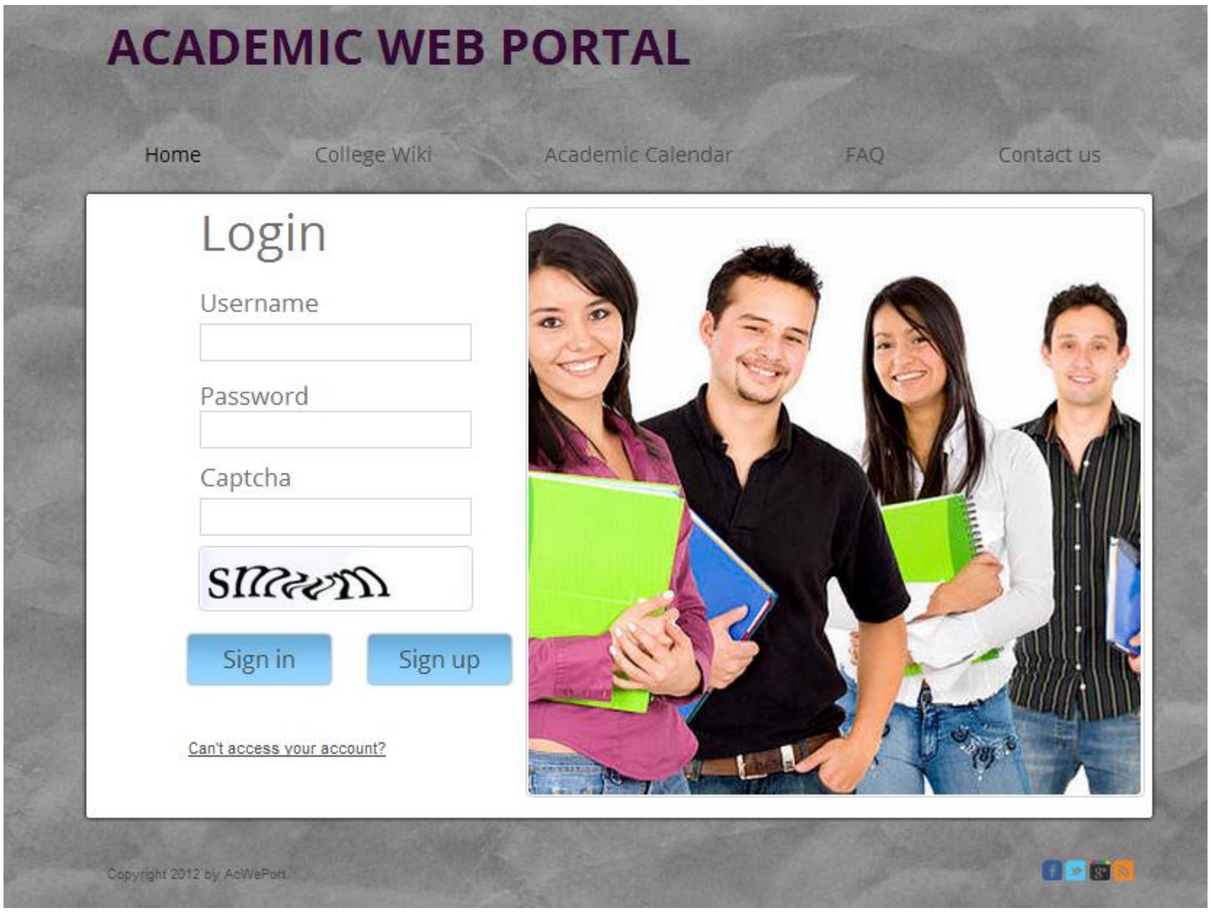
4.2 Use of MySQL

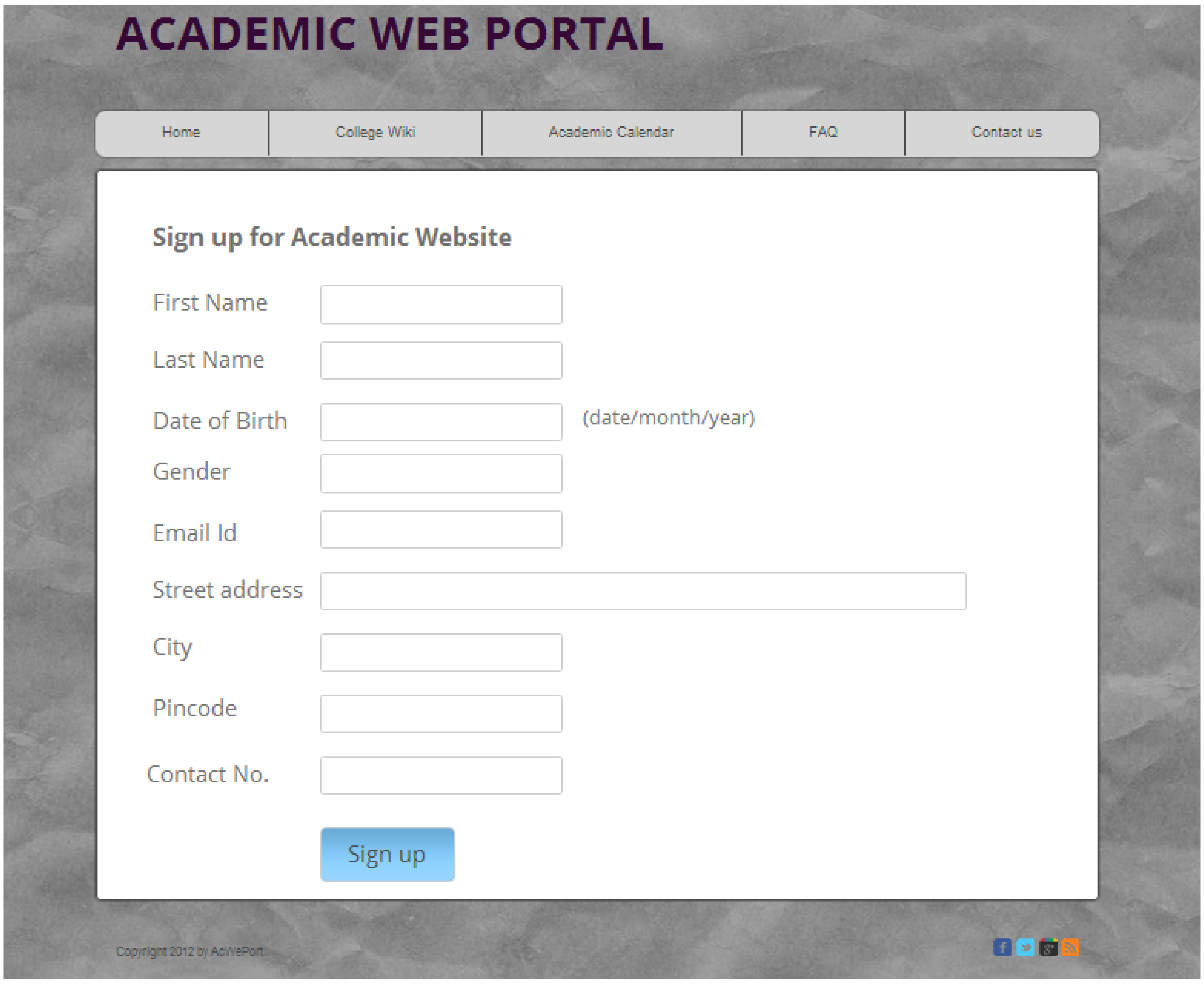
We decided to use MySQL for our website because it provides more speed and is lightweight in comparison to other database engines. It is also free and open-source

4.3 Use of Apache

We decided to use Apache for our website because it's free, open-source and extremely stable web server. Several modules are already available for apache. It is also used by most large websites

5. User Interface





ACADEMIC WEB PORTAL

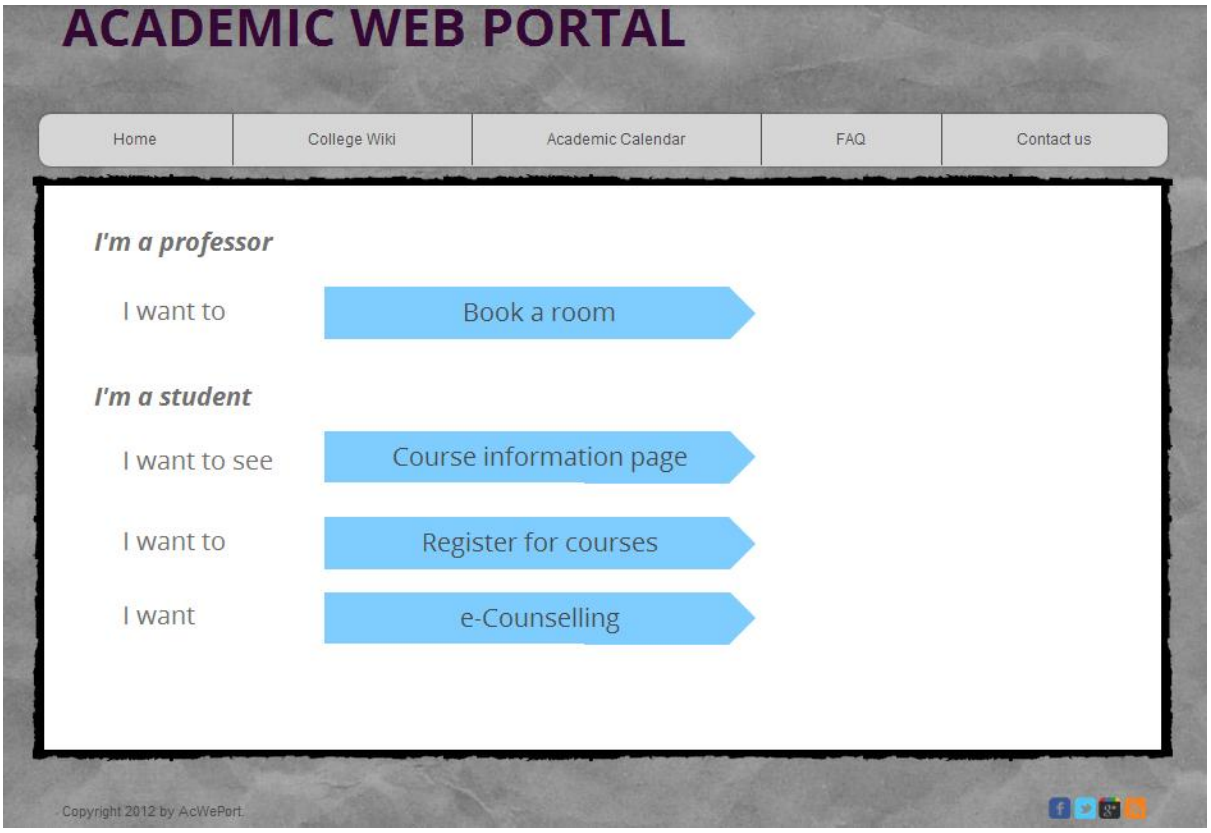
Home | College Wiki | Academic Calendar | FAQ | Contact us

Sign up for Academic Website

First Name		
Last Name		
Date of Birth		(date/month/year)
Gender		
Email Id		
Street address		
City		
Pincode		
Contact No.		

Sign up

Copyright 2012 by AcWePort



ACADEMIC WEB PORTAL

Home | College Wiki | Academic Calendar | FAQ | Contact us

I'm a professor

I want to — Book a room

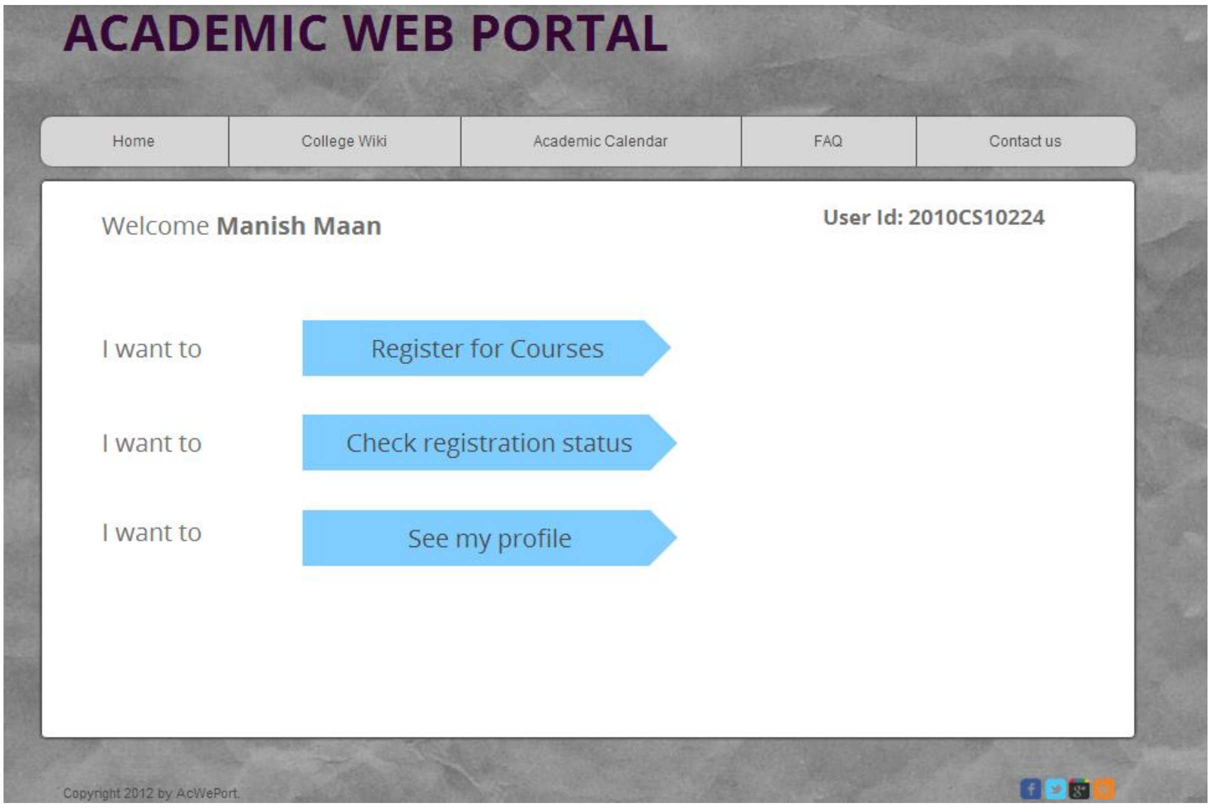
I'm a student

I want to see — Course information page

I want to — Register for courses

I want — e-Counselling

Copyright 2012 by AcWePort.



ACADEMIC WEB PORTAL

Home | College Wiki | Academic Calendar | FAQ | Contact us

Welcome **Manish Maan**

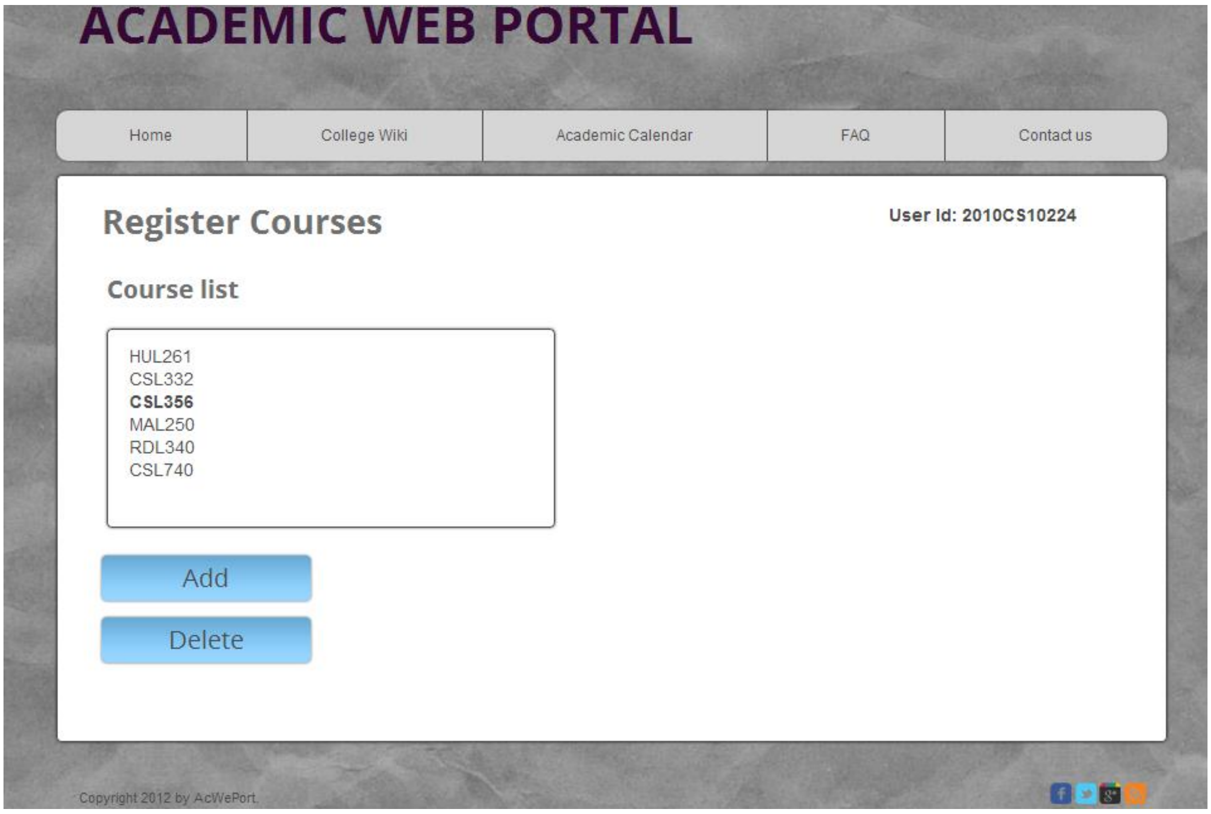
User Id: 2010CS10224

I want to — Register for Courses

I want to — Check registration status

I want to — See my profile

Copyright 2012 by AcWePort.



ACADEMIC WEB PORTAL

Home | College Wiki | Academic Calendar | FAQ | Contact us

Register Courses

User Id: 2010CS10224

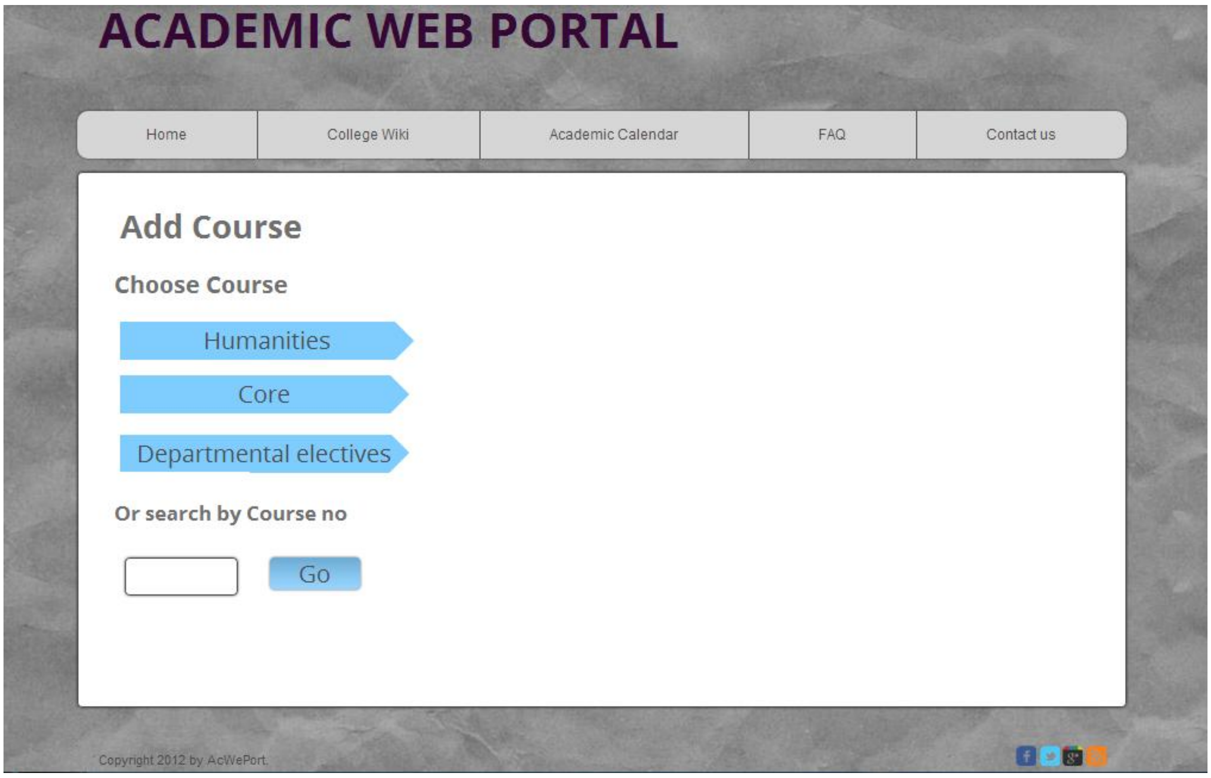
Course list

HUL261
CSL332
CSL356
MAL250
RDL340
CSL740

Add

Delete

Copyright 2012 by AcWePort.



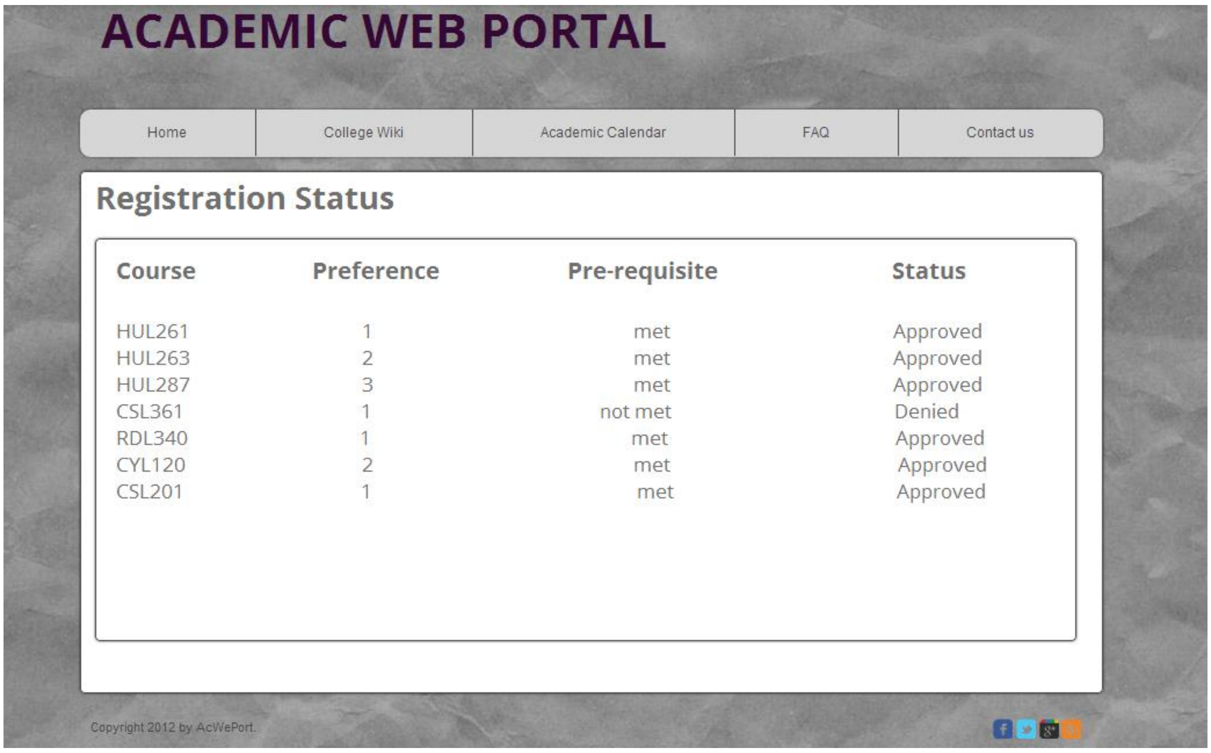
ACADEMIC WEB PORTAL

Add Course

Choose Course

Or search by Course no

Copyright 2012 by AcWePort.

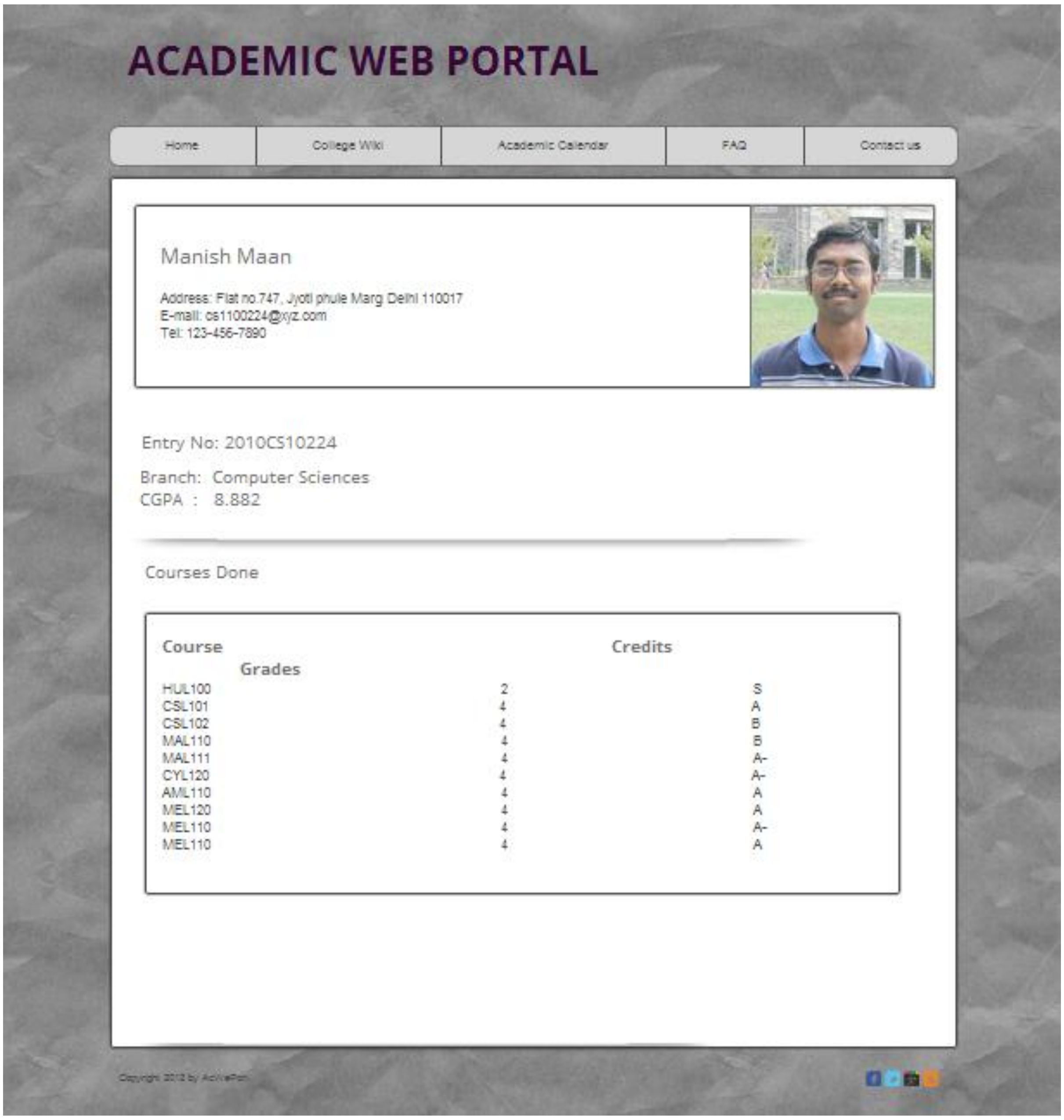


ACADEMIC WEB PORTAL

Registration Status

Course	Preference	Pre-requisite	Status
HUL261	1	met	Approved
HUL263	2	met	Approved
HUL287	3	met	Approved
CSL361	1	not met	Denied
RDL340	1	met	Approved
CYL120	2	met	Approved
CSL201	1	met	Approved

Copyright 2012 by AcWePort.



ACADEMIC WEB PORTAL

Home | College Wiki | Academic Calendar | FAQ | Contact us

Manish Maan

Address: Flat no.747, Jyoti phule Marg Delhi 110017
E-mail: cs1100224@xyz.com
Tel: 123-456-7890

Entry No: 2010CS10224

Branch: Computer Sciences
CGPA : 8.882

Courses Done

Course	Credits	Grades
HUL100	2	S
CSL101	4	A
CSL102	4	B
MAL110	4	B
MAL111	4	A-
CYL120	4	A-
AML110	4	A
MEL120	4	A
MEL110	4	A-
MEL110	4	A