SRS Document
CS 258

Group website development

Team-

Anmol Gomra(180001007)
Shubham Nimesh(180001054)
Roopraj B S(180001043)
Durga Mahesh(180001014)
Prashant(180001037)

Purpose

To develop a fully functional and user interactive online personal website for Dr Rupesh S Dewan(MEMS Dept.) as CS 258 Lab project.

Scope

- ✓ To create a responsive website with multiple device support.
- ✓ To provide Content management system on low level.
- ✓ To provide form validation on website for the user only.
- ✓ Provide a good and easily interactable GUI.

1.1 Definitions, Acronyms and Abbreviation

Admin – Administrator.

PM – Project Manager.

HTML – Hyper Text Markup Language.

HTTP – Hypertext Transfer Protocol.

ES6 – Javascript Lang.

BS4- Bootstrap 4(Front end lib.)

CSS-Cascading Style sheet.

1.2 References

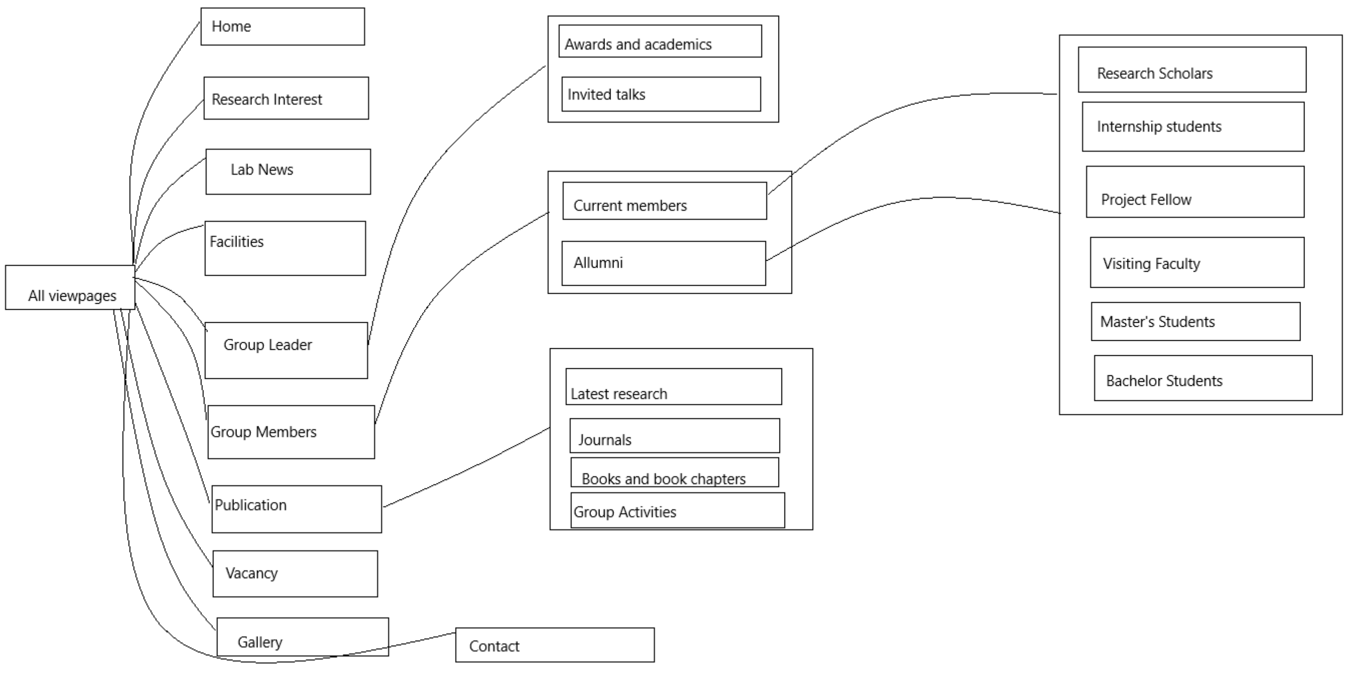
W3S, MDN ,Stack Overflow, r/webDev.

1.3 Overview

This project aims to build a personal website having interactable GUI. The website will have suitable elements such as nav bars, a body section , footer section having multiple sections/div which will hold all info required by the user.
Suitable links will be provided so as to direct to required webpages. Form validation will also be provided so the user can login and update the text and images inside certain sections for future maintainability purposes.

2. Overall Description

2.1 Website Schematic

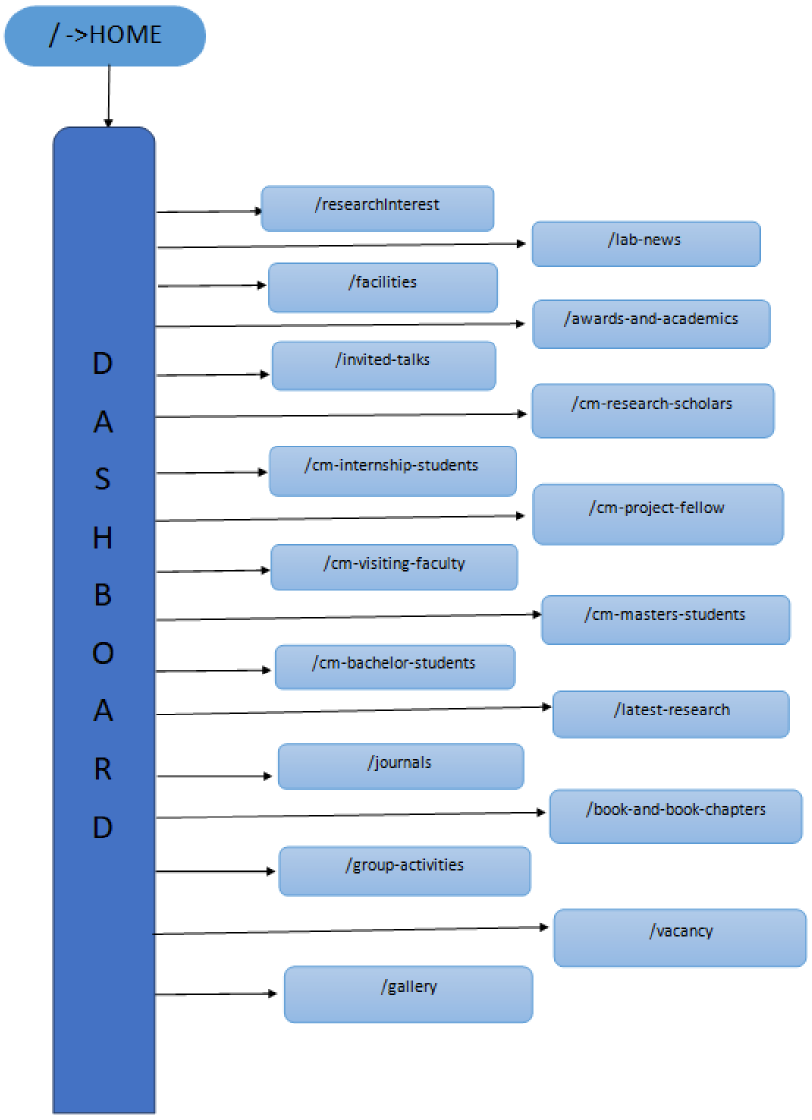


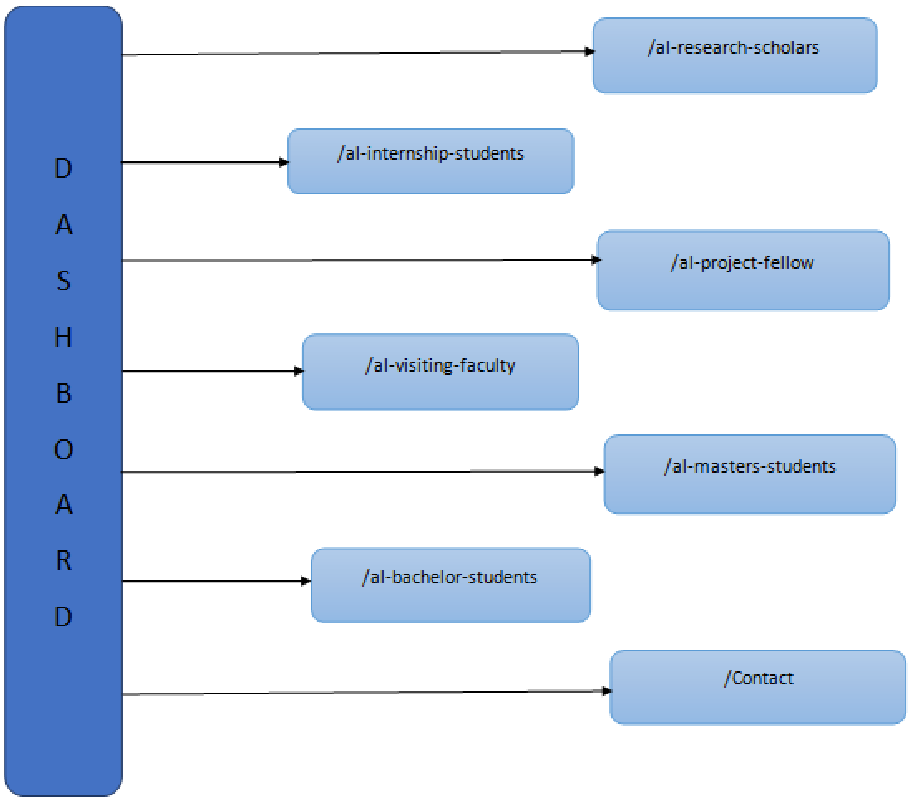
DESCRIPTION OF WEBPAGES:

Note: Please view all different types of cards used to display data on different viewPages by goin through the diagrams.
There are 4 types:
1> Text Cards
2> Info Cards
3> Carausal
4> Gallery Collage

Note: All Textual data which will be given as input by the user will be updatable without touching the code. Content Management System will be applied appropriately.
Images can be added by user without touching the code.

At the top of all the webpages, there will be a fixed navigation bar with list items and under some list items there will be a drop down menu where some sub-links will be present as shown in the following manner:


/ ->HOME
DASHBOARD
/researchInterest
/lab-news
/facilities
/awards-and-academics
/invited-talks
/cm-research-scholars
/cm-internship-students
/cm-project-fellow
/cm-visiting-faculty
/cm-masters-students
/cm-bachelor-students
/latest-research
/journals
/book-and-book-chapters
/group-activities
/vacancy
/gallery


DASHBOARD
/al-research-scholars
/al-internship-students
/al-project-fellow
/al-visiting-faculty
/al-masters-students
/al-bachelor-students
/Contact

Detailed Introduction to all webpages

1. **Main Homepage(Profile Of User)-**

   The top-left of page will consist of Text Card containing all the information related to the user(ex: Education etc).

   The top-right of page will consist of info-card with image and all details related to that of image.If no image is present, the user can add the image as input option will appear.
   Note that these text cards will be draggable and data between any 2 cards could be swapped. Info Cards are not draggable and data between them is not swappable.

   The bottom of the Page will contain the input for text card if the user wants to add more skills.

2. **Research Interest Page** –

   The top of the page will contain a carousal containing different pictures related to the research department.

   The remaining bottom of the page will contain specific topics and information about those topics(specifically in <p> tags). These topics will be all about different research fields related to the branch in which the individual owner of the website works in.

   Specific <h1> ,<h2> or <h3> tags will be provided for main headings.

3. **Facilities Page**-

This page contains different cards inside <divs> of bootstrap 4 .

These cards will be used to store images of same width and height . The images will portray the facilities inside the lab and also those of which represent the subject branch. Suitable Title an details related to each image could also be added.

The owner will be able to add and delete the pictures as and when required and will also be able to update image Title and Details.

4. a Group Leader – Awards and Academics

This page will contain information about the Awards, memberships, Academic and corporate Activities of the website owner.

4. b Group Leader – Invited Talks

This page is all about the research talks held by the user at different events in the past or the ones that are going to be held in the future.

5. a Group Members – Current Members

This page contains different cards inside <divs> of bootstrap 4

These cards will be used to store images of same width and height .The images will those of all the Research Scholars, Internship Students, Project Fellow, Visiting Faculty, Master Students, Bachelor Students.

The owner will be able to add and delete the pictures as and when required and will also be able to update image Title and Details.

5.b <u>Group Members – Allumni</u>

This page will be similar to that of Current Members but will contain the images of all the alumni students.

6.a <u>Publications – Latest Research</u>

The top of the page will contain few images related to the latest research in which the faculty is employed in.

The remaining bottom of the page will be all about the affiliations and authors of the research papers, the research of whom is important in the latest research of the user itself.

6.b <u>Publications – Journals</u>

The page will contain multiple list items where each of the list items will be all about the different journals of which the faculty individual is involved in. Links to download the manuscripts will also be provided.

6.c <u>Publications – Book and Book Chapters</u>

This page will be similar to that of journals excepts the name of books which the faculty individual has written will be listed.

6.d <u>Publications – Group Activities</u>

This page will be all about the group activities the the user is involved in with his/her research group. All those research projects and research activities will be listed below.

7 . <u>Vacancy-</u>

The page will list all the vacancy positions available in the office
Suitable Links will be provided for filling up the form.

8. <u>Gallery-</u>

Multiple pictures of the faculty and his/her research group will be made available here. Pictures that belong to multiple genre can also be put here.

9. <u>Contacts-</u>

The contact info of the faculty(office phone no, lab details etc) will be made available here using suitable HTML elements such as <section> or <div>.

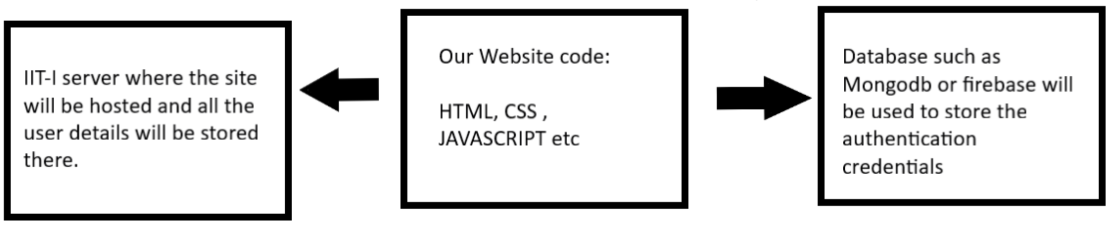
<u>The following will be for: Public website(used by public)</u>

A form will also be made using the <form> which can be used if any visitor on the website has some query. This will be placed inside a div.

<u>The following will be for: Personal website(used by user)</u>

All details entered by users on contact form of public website will be displayed over here.

2.2 Web Architecture Diagram


IIT-I server where the site will be hosted and all the user details will be stored there.
Our Website code:
HTML, CSS , JAVASCRIPT etc
Database such as Mongodb or firebase will be used to store the authentication credentials

What is a ViewPage?

ViewPage is the page on which data is being shown on secondary website(The website with developer options and content management system). Example: researchInterest, lab- news, facilities etc. Starting with GET routes we take data from database through request/ response.

REST API IMPLEMENTATION DETAILS

GET REQUEST:

1. ViewPage/getCarausalImages/images/:id :-

➔ **route for retreiving specific image with a given id which can be viewed in the browser and deleted selectively. The above route is used in routes/get/deleteCarausalImages.js.**

2. ViewPage/getCarausalImages :-

➔ **route for retreiving all carausal images based on the viewPage being viewed and if that viewPage has carausal. The above route is used in routes/get/getCarausalImages.js.**

3. **ViewPage/ :generic :-**

➔ route for retreiving subject and details (not title and details of cards with image upload) and rendering them onto viewpage whenever specific viewPage is loaded. The above route is used in routes/get/generic.js file which has code for retreiving only textual card data(i.e subject and details).

4. **ViewPage/ genericWithCarausal :**

   ➔ route for retreiving carausal images and rendering them onto viewpage whenever specific viewPage is loaded and that viewpage has carausal. The above route is used in routes/get/ genericWithCarausal.js file which has code for retreiving only carausal data(i.e which are images).

5. **ViewPage/getCardImages :-**

   ➔ route for retreiving all cards(with title and details) images based on the viewPage being viewed and if that viewPage has a info card such as facilities viewPage. The above route is used in routes/get/getCardImages.js.

6. **ViewPage/getCarausalImages/files/:filename :-**

➔ Although the route is not specifically used to retreive data into project but is usefull for getting info about any single carausal image depending upon the viewPage and if that viewPage has carausal or not . The info can be retreived onto the browser. The above route is used in routes/get/ getfilename.js.

7. **ViewPage/getCarausalImages/images/:filename :-**

➔ **This route is usefull for displaying any specific single carausal image onto browser using filename property and again it depends upon the viewPage and if that viewPage has carausal or not . The above route is used in routes/get/ getimagename.js.**

8. **ViewPage/getCardImages/files/:filename :-**

➔ **Similar to {viewPage}/getCarausalImages/files/:filename the route is not specifically used to retreive data into project but is usefull for getting info about any single card image depending upon the viewPage and if that viewPage has info card or not. The info can be retreived onto the browser. The above route is used in routes/get/getsecondimagename.js.**

9. **ViewPage/getCardImages/images/:filename :-**

➔ **This route is usefull for displaying any specific single card image onto browser using filename property and again it depends upon the viewPage and if that viewPage has info card or not. The above route is used in routes/ get / getsecondimagename.js.**

10. **ViewPage/ seondgeneric :-**

➔ **route for retreiving title and details (not subject and details present at the bottom of viewpage) and rendering them onto viewpage whenever specific viewPage is loaded. The above route is used in routes/get/seondgeneric. js file has code for retreiving only info card data(i.e title and details).**

Note ： Routes in 3. , 4. and 10. are capable of rendering simultaniously when page reloads and that is because they have same routes. In the website there are cases where either routes in 3. and 4.(example /researchInterest or Research Interest page) are rendered simultaniously and there are also cases where 3., 4., and 10. are renderedPage simultaniously(ex: /dashboard or home page).

GETTING ON WITH POST ROUTES

We are putting in data from client side through request/responses using form or web sockets. ViewPage is the page on which data is being shown on secondary website(The website with developer options and content management system). Like researchInterest, lab-news, facilities etc.

In order to delete and edit info cards on viewPages like facilities etc Forms have not been used so POST requests on those parts have not been applied.

Over there Web Sockets have been used where there is bi-directional data transfer between client and server side. This has been done so as to avoid complication which occurs when relationship between info card image and its details has to be made in the database so as when we delete a certain info card all of its details and the corrosponding image is deleted and not either of those.

The main problem was info-card images are stored in 2 seperate collections and info-card details are stored in another collection insted of all of these being stored in one single collection. So as to avoid complexities web sockets are implied.

The client side functioning for info cards editing and deleting is controlled by the facilities-draggable.js file in public/scripts/facilities-draggable.js. Also note web sockets have been implied in other areas also where there use will be specified wherever necessary.

API REQUEST :

1. viewPage/delete :-

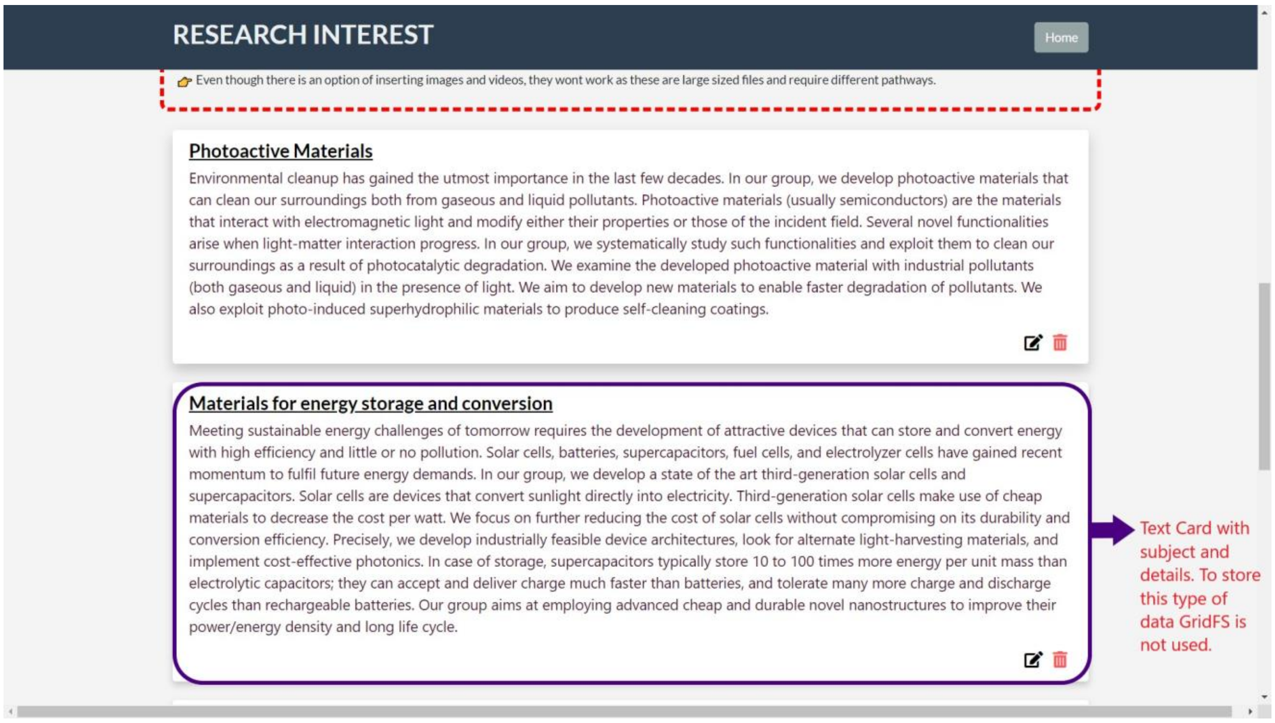
→ This route is used for deleting the subject and details of text-cards (not info card that have images as well in them). These are the cards present on view pages like researchInterest below at the page and in this case showing the research interest of the user. This post route has been implemented in routes/post/delete.js.

2. ViewPage/edit :-

→ This route is used for editing the subject and details of text-cards (not info card that have images as well in them). These are the cards present on view pages like researchInterest below at the page and in this case showing the research interest of the user. Whenever such post requests are made a modal opens up and with previous subjects and details present over there which can then be edited or completely changed. Now over here web sockets are used so as to retreive previous subject and details and this required communication between frontend and server without forms. Now when changes are made then this post request is made so as to save the new data onto database. This route has been implemented in routes/post/edit.js.

3. ViewPage/generic :-

→ This route is used for adding entirely new data i.e the subject and details of text-cards (not info card that have images as well in them). These are the cards present on view pages like researchInterest below at the page and in this case showing the research interest of the user. This post route has been implemented in routes/post/generic.js.


RESEARCH INTEREST

Home

Even though there is an option of inserting images and videos, they wont work as these are large sized files and require different pathways.

Photoactive Materials

Environmental cleanup has gained the utmost importance in the last few decades. In our group, we develop photoactive materials that can clean our surroundings both from gaseous and liquid pollutants. Photoactive materials (usually semiconductors) are the materials that interact with electromagnetic light and modify either their properties or those of the incident field. Several novel functionalities arise when light-matter interaction progress. In our group, we systematically study such functionalities and exploit them to clean our surroundings as a result of photocatalytic degradation. We examine the developed photoactive material with industrial pollutants (both gaseous and liquid) in the presence of light. We aim to develop new materials to enable faster degradation of pollutants. We also exploit photo-induced superhydrophilic materials to produce self-cleaning coatings.

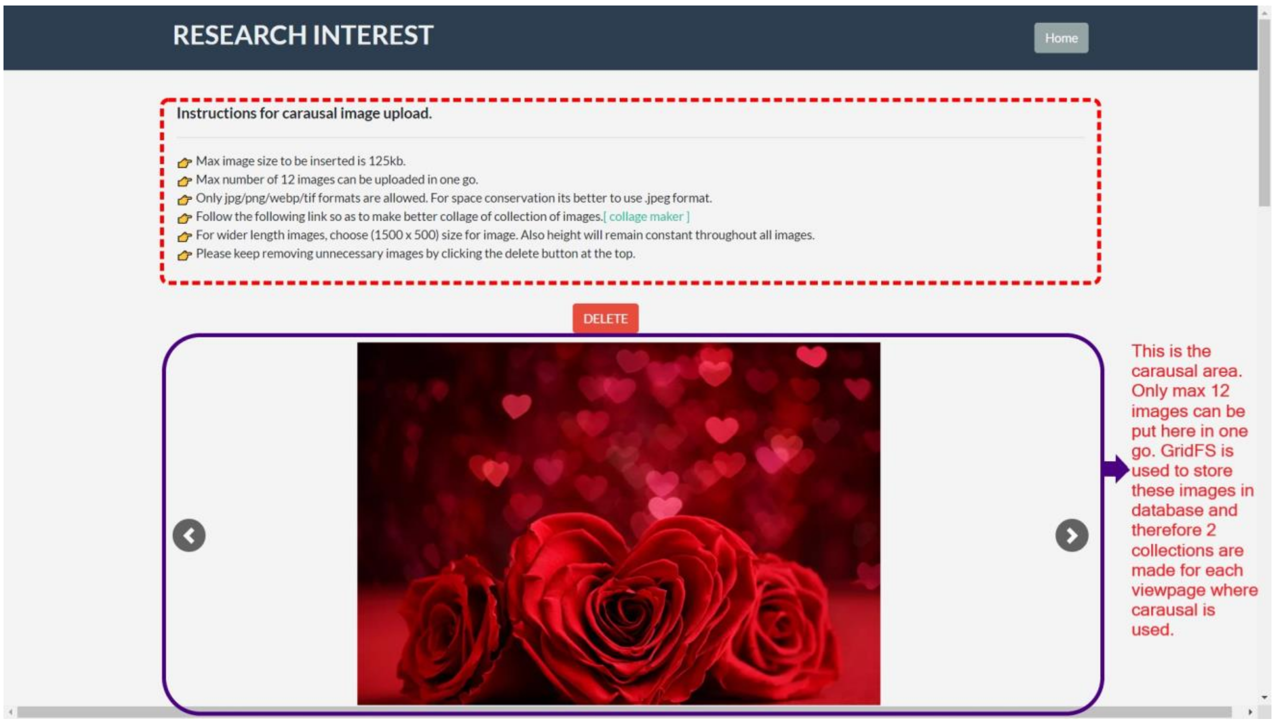
Materials for energy storage and conversion

Meeting sustainable energy challenges of tomorrow requires the development of attractive devices that can store and convert energy with high efficiency and little or no pollution. Solar cells, batteries, supercapacitors, fuel cells, and electrolyzer cells have gained recent momentum to fulfil future energy demands. In our group, we develop a state of the art third-generation solar cells and supercapacitors. Solar cells are devices that convert sunlight directly into electricity. Third-generation solar cells make use of cheap materials to decrease the cost per watt. We focus on further reducing the cost of solar cells without compromising on its durability and conversion efficiency. Precisely, we develop industrially feasible device architectures, look for alternate light-harvesting materials, and implement cost-effective photonics. In case of storage, supercapacitors typically store 10 to 100 times more energy per unit mass than electrolytic capacitors; they can accept and deliver charge much faster than batteries, and tolerate many more charge and discharge cycles than rechargeable batteries. Our group aims at employing advanced cheap and durable novel nanostructures to improve their power/energy density and long life cycle.

Text Card with subject and details. To store this type of data GridFS is not used.

4. ViewPage/uploadCarausalImages :-

→ **This route is used for adding entirely new carausal images onto the database. This route is present on whatever viewpages where carausal is present in them. Something called storage engine has been implemented during implementation of this route so as to store image data onto the database about which more will be told in some other section of documentation.**



Instructions for carausal image upload.

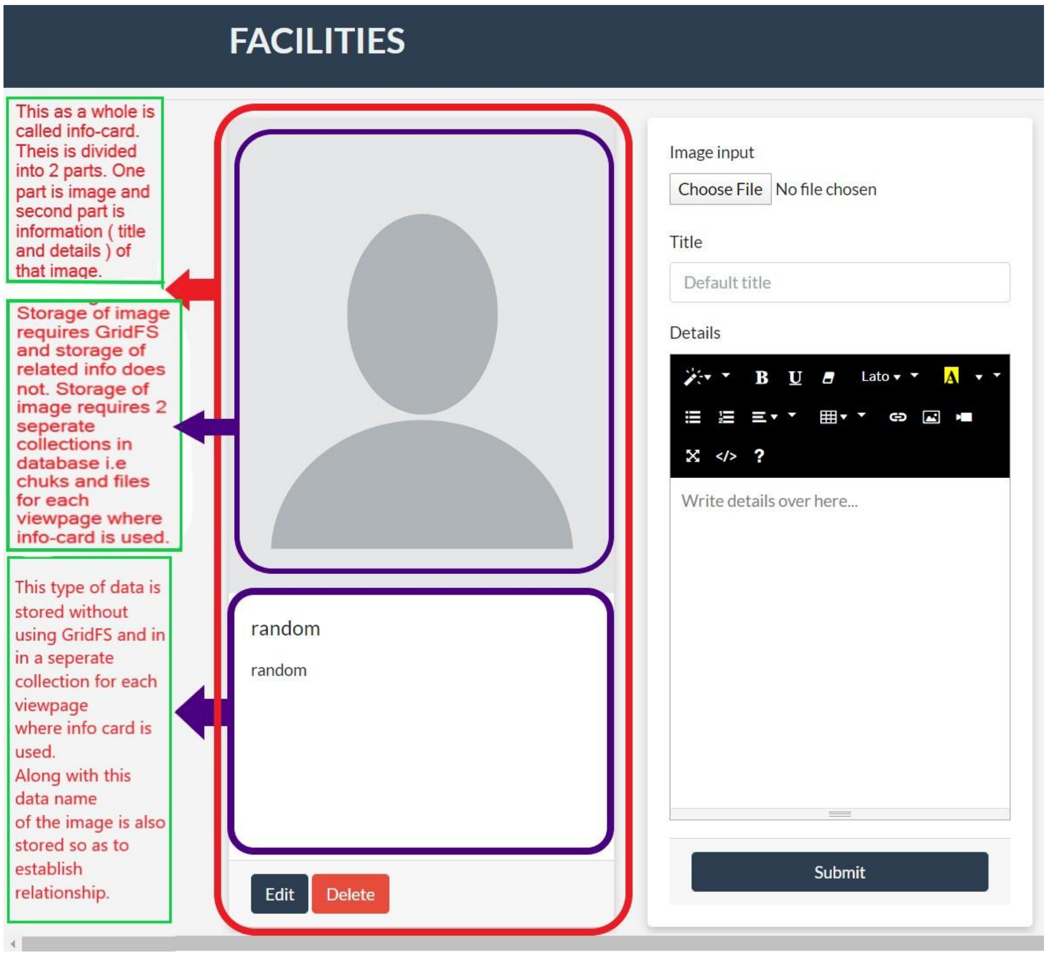
- Max image size to be inserted is 125kb.
- Max number of 12 images can be uploaded in one go.
- Only jpg/png/webp/tif formats are allowed. For space conservation its better to use .jpeg format.
- Follow the following link so as to make better collage of collection of images.[collage maker]
- For wider length images, choose (1500 x 500) size for image. Also height will remain constant throughout all images.
- Please keep removing unnecessary images by clicking the delete button at the top.

This is the carausal area. Only max 12 images can be put here in one go. GridFS is used to store these images in database and therefore 2 collections are made for each viewpage where carausal is used.

Storing of carausal images also uses 2 seperate collections in the database but since there are no details to be stored in relationship with it, No use of web sockets occur over merthodOverride and simple post requests through form does the job Deleting of carausal images uses a special technique of method override which helps to prevent AJAX request to be made easily image is deleted. However this method cannot be implemented everywhere. Over here web sockets could have also done the job but we had a better alternative available so we used it. This post route has been implemented in routes/ post/uploadCarausalImages.js.

5. ViewPage/uploadCardDetails :-

→ This route is used for adding only details(title and details) of info card onto the database. This route is present on whatever viewpages where info card is present in them(ex facilities page). Also a little use of web sockets has been made so that we could get the name of the image to uploaded to seperate 2 collections as explained earlier.


FACILITIES
This as a whole is called info-card. Theis is divided into 2 parts. One part is image and second part is information (title and details) of that image.
Storage of image requires GridFS and storage of related info does not. Storage of image requires 2 seperate collections in database i.e chuks and files for each viewpage where info-card is used.
This type of data is stored without using GridFS and in in a seperate collection for each viewpage where info card is used. Along with this data name of the image is also stored so as to establish relationship.
random
random
Edit
Delete
Image input
Choose File
No file chosen
Title
Default title
Details
Lato
Write details over here...
Submit

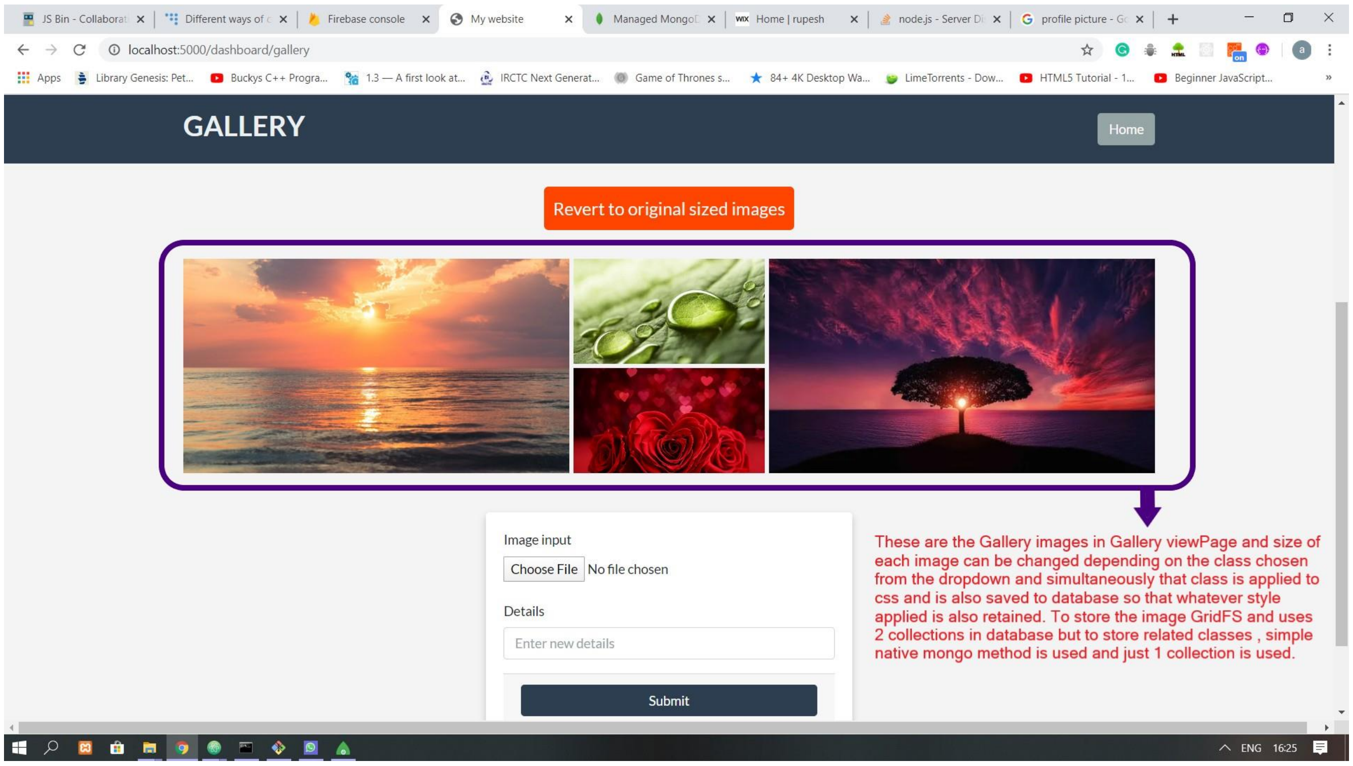
This name of image is necessary so as make a relationship between those 2 collections(chunks and files) where the image will be stored through a seperate route (talked about next) and the 3rd collection where the image details will be stored using this route. This image name is also used whenever we want to delete a specific info card. Now you can see why web sockets are implied. They somewhat help to reduce compexity. The main problem was 3 diferent collections being formed in database. We will also talk why 3 different databases are formed and why not one in another specific section when we talk about storage engine. Essentially the point to be noted is this route is only responsible for storing info card details only onto the database. This post route has been implemented in routes/post/uploadCardDetails.js.

6. ViewPage/uploadCardDetails :

→ This route is responsible for storing images of info card on to the 2 seperate collections in database. A little use of web sockets have also been made here so as to get image name. This post route has been implemented in routes/post/ uploadCardImages.js.

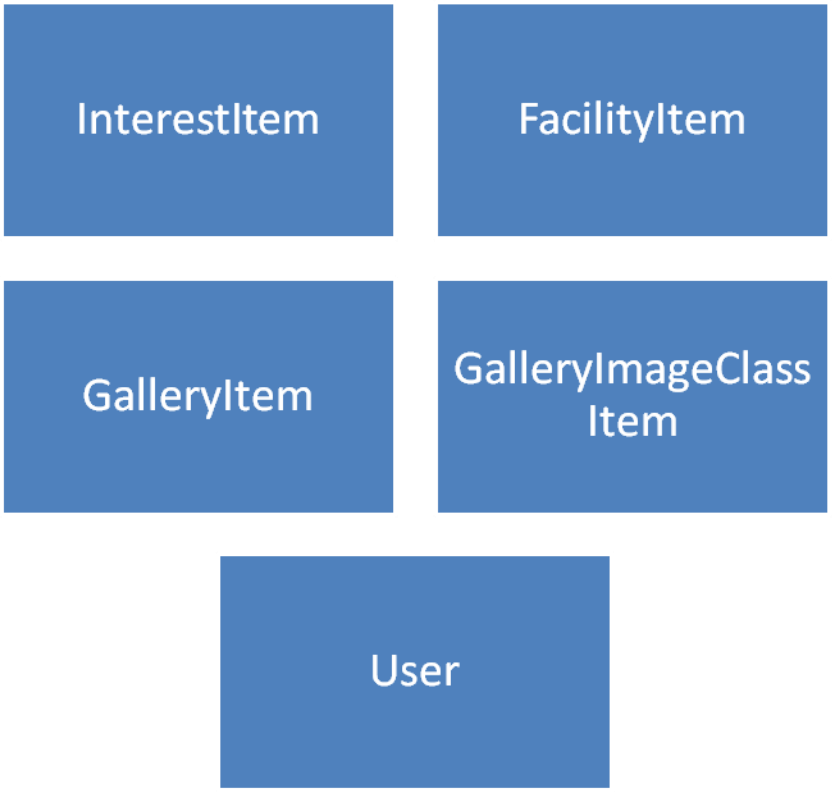
7. ViewPage/uploadGalleryCardDetails :-

→ This route is responsible for storing images of gallery card present in gallery viewpage on to the 2 seperate collections in database. A little use of web sockets have also been made here so as to get image name. This post route has been implemented in routes/post/uploadGalleryCardImages.js.



These are the Gallery images in Gallery viewPage and size of each image can be changed depending on the class chosen from the dropdown and simultaneously that class is applied to css and is also saved to database so that whatever style applied is also retained. To store the image GridFS and uses 2 collections in database but to store related classes , simple native mongo method is used and just 1 collection is used.

DIFFERENT MODELS USED IN MONGO DB



- ➔ InterestItem: This model is responsible for providing structure to all the text data (subject and details , and not the info-card data) in the database and is used by multiple viewPages.

- ➔ FacilityItem: This model is responsible for providing structure to all the info data (title, details, image name, and not the text data) in the database and is used by viewPages such as facilities , Home etc.

- ➔ GalleryItem: This model is responsible for providing structure to the information about each of the collage image present in the Gallery viewPage of the website.

- ➔ GalleryImageClassItem: This model is responsible for keeping track of all style classes(which are responsible for different size of images in collage in Gallery PageView) as whatever size chosen by the user must be retained so as the shape of collage is mantained. The collage uses a css concept called Grid Box which in-turn uses dense packing algorithm so that there is no white space   present between different gallery images whenever they are resized and even if there is small space, it should be best minimized.

- ➔ User: This model is responsible for the login info of all the users of the website which in this case is only one but more users can be introduced.

STORAGE FUNCTIONALITY IN DATABASE

It is not possible to store data in mongodb more than 16mb using simple post request simple mongodb storage operation. (ref: https://docs.mongodb.com/manual/core/gridfs/). So the solution to the problem is GridFS which has some similar operations to that of mongoDB but it lets you store data in chunks and using 2 collections in database.

The storage functionality for carousal images is implemented using these 3 files.

1. routes\storage\gfs.js
2. routes\storage\multerFunction.js
3. routes\storage\storageEngine.js

The storage functionality for info card images is implemented using these 3 files

1. routes\storage\secondgfs.js
2. routes\storage\secondMulterFunction.js
3. routes\storage\secondStorageEngine.js

STORAGE LIMITS OF WEBSITE

1> Each carousal image should not be more than 125 kb and these limits are described in {routes\storage\multerFunction.js}.
2> Each info-card image should not be more than 150 kb and these limits are described in {routes\storage\secondMulterFunction.js}.

Note that these sizes can be altered but should be kept low as much as possible so that database limits are not crossed.
The storage database used will mongdb Atlas which is cloud storage and it has only 500 mb storage under free plan.

HOW AUTHENTICATION WORKS

The registration , login and forgot activity cycles mostly clears up all the details on how everything is connected and what goes in and out.

Some terms are explained in more detail below:

Note: MongoURI and registeredEmail is present in config/ keys.js. To keep track if which routes are accessible in website when user logged out or logged in is present in config/auth.js Note: config folder will not be added to github since it consists of sensitive data which can be made to public.

- **Passport Js:**

This is a middleware module that makes it a little bit easier to provide authentication in a website. More about it can be known here: http://www.passportjs.org/docs/.

- **Hashing:**

This is the technique of turning password into a combination of random digits, characters etc so that password could be saved in database. This process is irreversable. Whenever a somone logins with a password. The password must also be hashed and same rounds of salting must be done and then , it should be compared with password in database. Then only person get logged in. This process applied in config/passport.js file (This mostly handles error handling during login session) and routes/users.js file in project folder.

- **bcrypt:**
This module is used to convert passwords to hashed passwords so as to store them in database safely.

- **Salting:**
Salting is basically the technique of hashing an already hashed password more number of times so as it becomes more stronger. Salting requires computing power and thus limited round of it should be done. 10 rounds are more than enough.

- **Crypto:**
This is the module which is reponsible for providing us any 20 random bytes of hexadecimal string which becomes a token that we send as a unique number along with the password reset mail.

- **nodemailer:**
This module allows us to send email to a registered email address. We send a link to a route through which we reset the password.