PhD Conference Grant Management Portal

CP301 - Development Engineering Project (Sem-2 AY 22-23)

Yadwinder Singh - 2020CSB1143
Adish Lodha - 2020CSB1063
Vinay Kumar - 2020CSB1141
Tanuj Kumar - 2020CSB1134

Consultant and Mentor - Dr. Puneet Goyal (IIT Ropar)

PPT3

Motivation and Idea

- The project stems from the need to simplify and streamline the grant application process. Currently, the offline process is time-consuming and cumbersome, making it difficult for both students and authorities to maintain and track the application data.
- With the use of our web portal, we aim to reduce the time it takes to complete the process and improve its efficiency.
- The PHD Conference Grant Management Portal aims to simplify the cumbersome and time-consuming process of applying for PHD Conference grants.
- Through the portal, students can easily apply for domestic or international PHD conferences and track the status of their application.
- The portal is designed to streamline the process and provide faster approvals by routing applications to the appropriate authorities.
- The project is driven by the desire to make the application process simpler, faster, and more efficient for all stakeholders involved.

How to run our application?

Client Side

First Navigate to our client directory using **cd ./client** from the root of our directory.

Run command **npm install** to install all the necessary dependencies.

Now to Start our client Side run the command **npm start**. this would start the front end of our application.

Server Side

First Navigate to our server directory using **cd ./server** from the root of our directory.

Run command **npm install** to install all the necessary dependencies.

Now to Start our server Side run the command **npm start**. this would start the back-end server of our application.

Email for sending OTPs/notification

ID: deptestbott12@gmail.com

Password: dep@team12

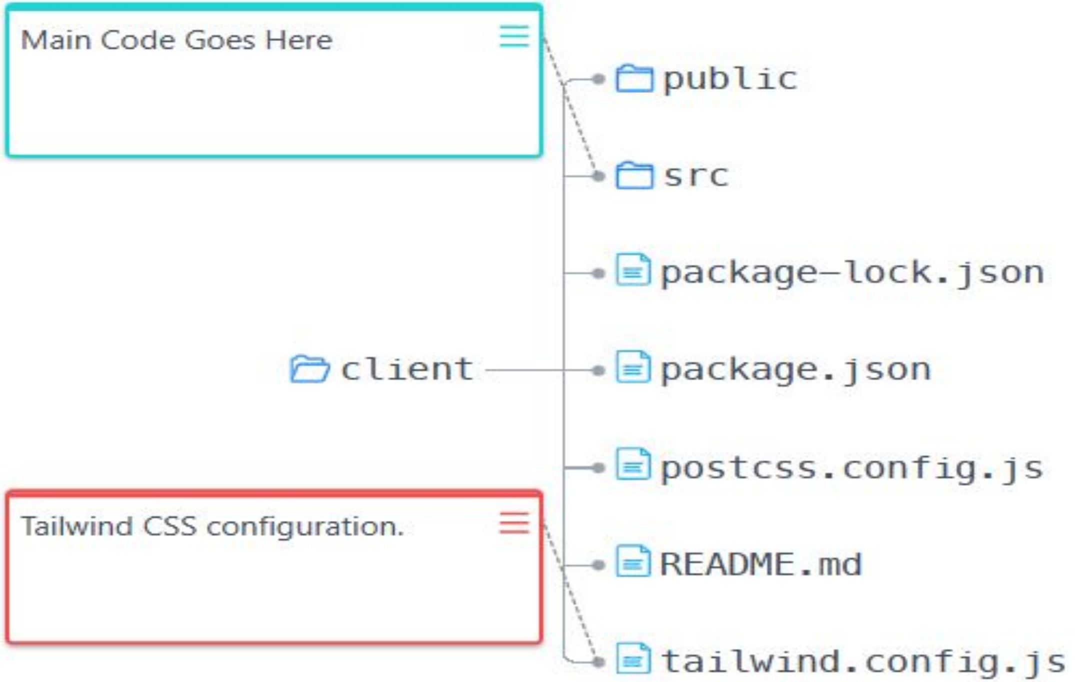
User Guide

- **User can login by using the OPT sent to their institute mail id.**
- **Student can apply for new application for across India and over the globe.**
- **Student can keep track of the application status.**
- **Student can also fill settlement form after coming from conference.**
- **Supervisor, HOD and Dean PG can view details of the submitted applications and can approve by uploading sign.**
- **Research Section and Account Section can can view details of the submitted applications and can approve by uploading sign.**
- **Research Section can also upload data of students and faculty using excel file.**
- **Research Section can attach details of granted sanctions.**
- **Account Section can check available balance and make payment.**

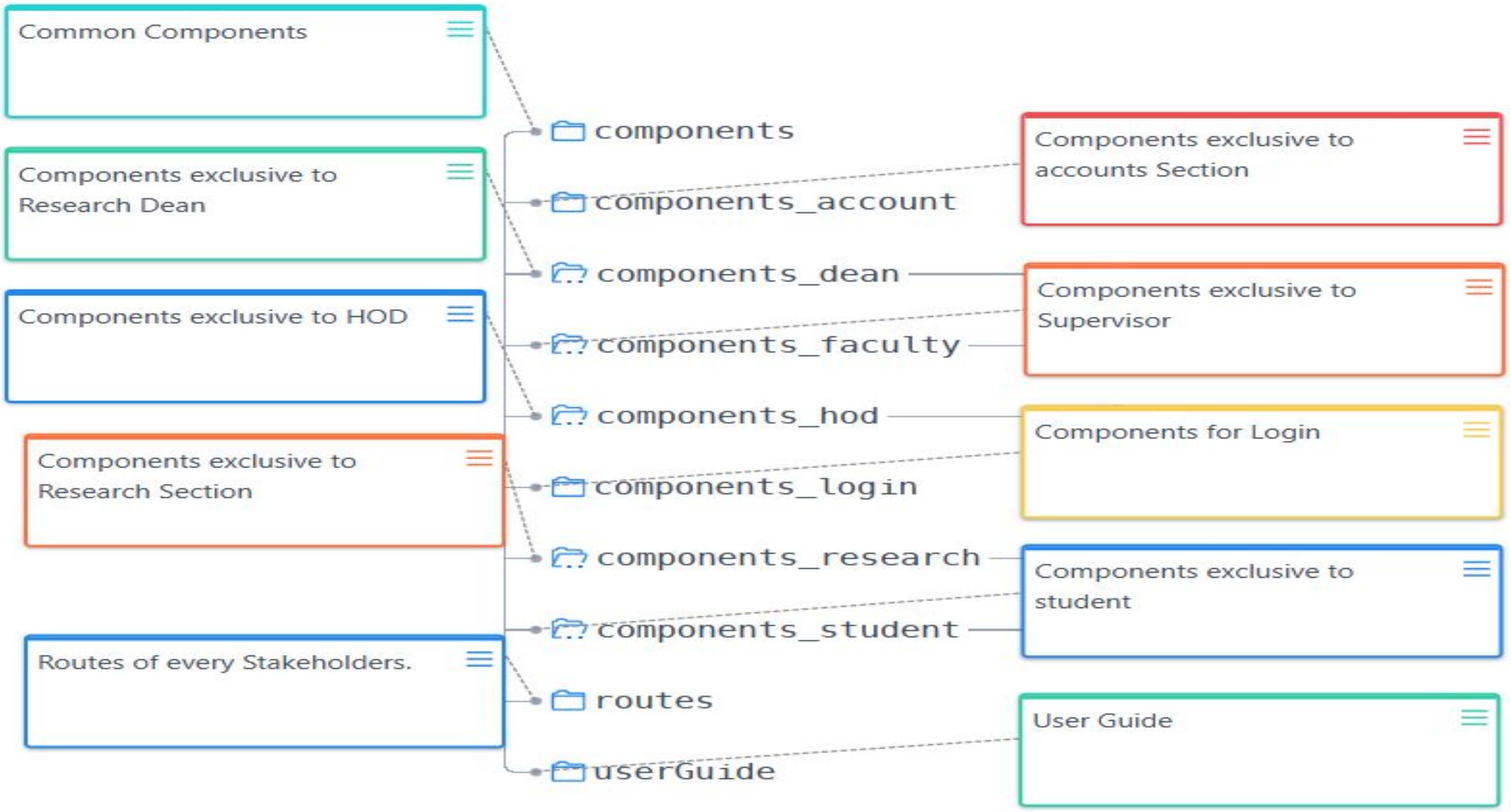
Front End

Code Crumbs

/client

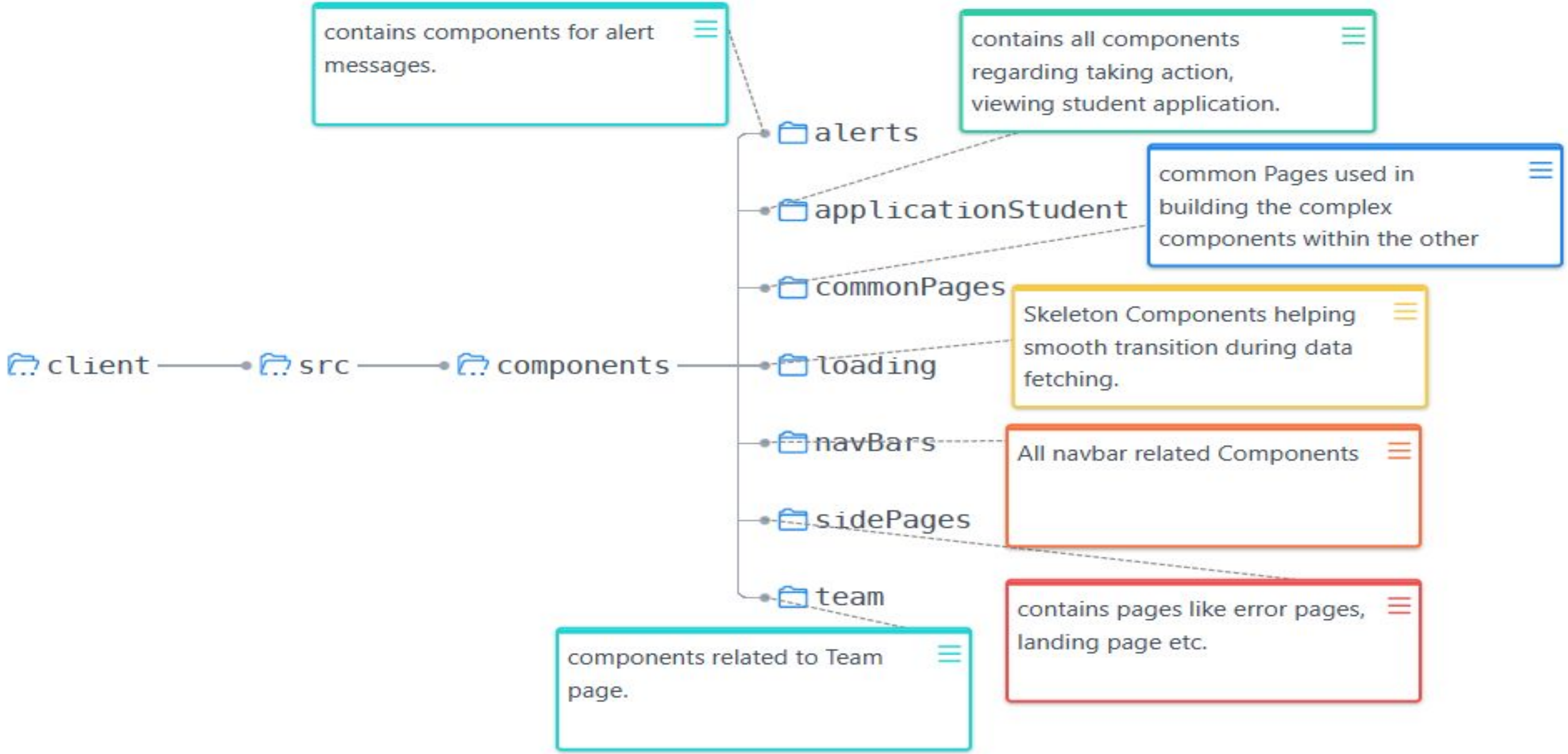

Main Code Goes Here
public
src
package-lock.json
client
package.json
postcss.config.js
Tailwind CSS configuration.
README.md
tailwind.config.js

/client/src



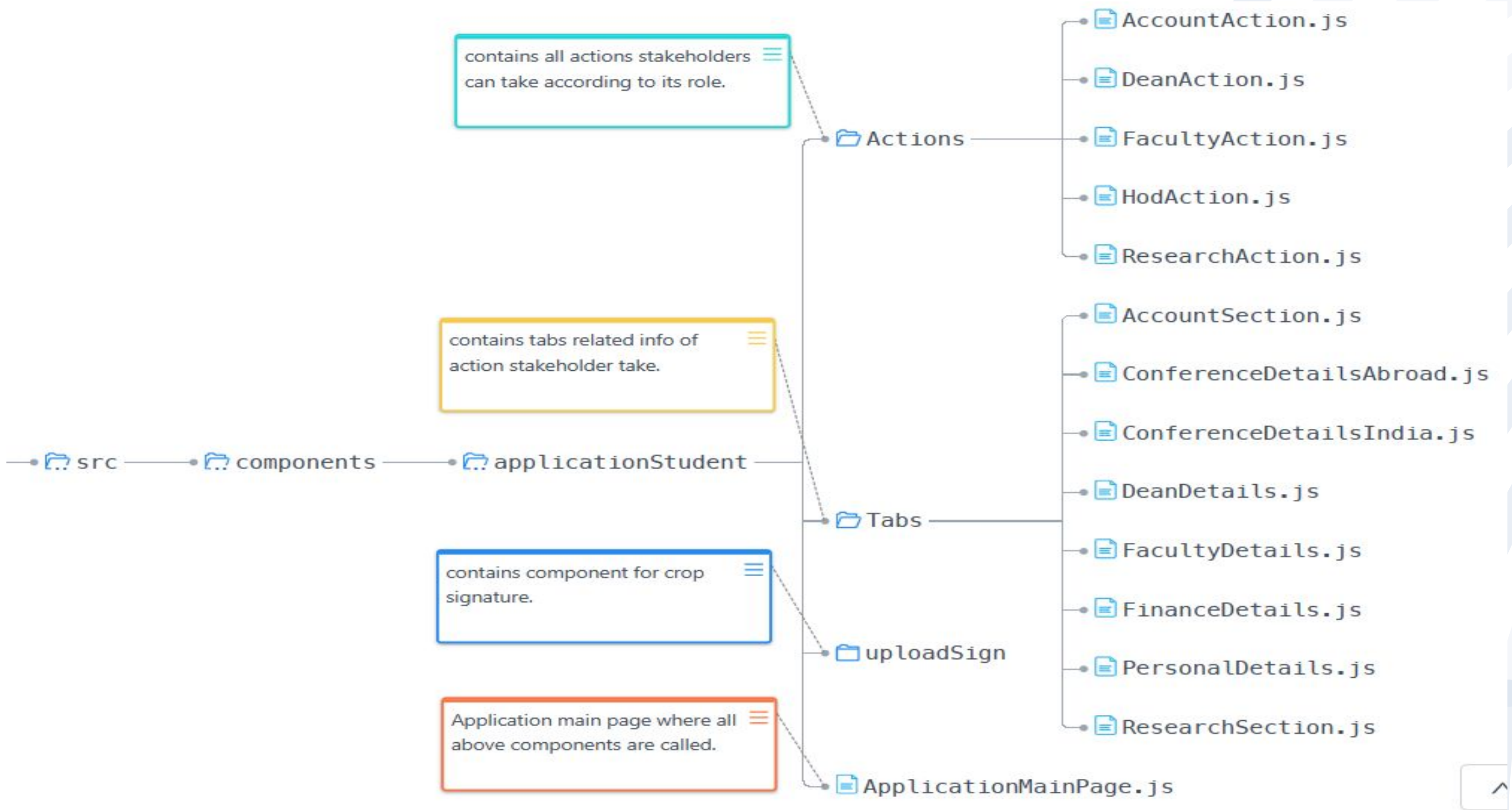
Common Components
Components exclusive to Research Dean
Components exclusive to HOD
Components exclusive to Research Section
Routes of every Stakeholders.
components
components_account
components_dean
components_faculty
components_hod
components_login
components_research
components_student
routes
userGuide
Components exclusive to accounts Section
Components exclusive to Supervisor
Components for Login
Components exclusive to student
User Guide

/client/src/components

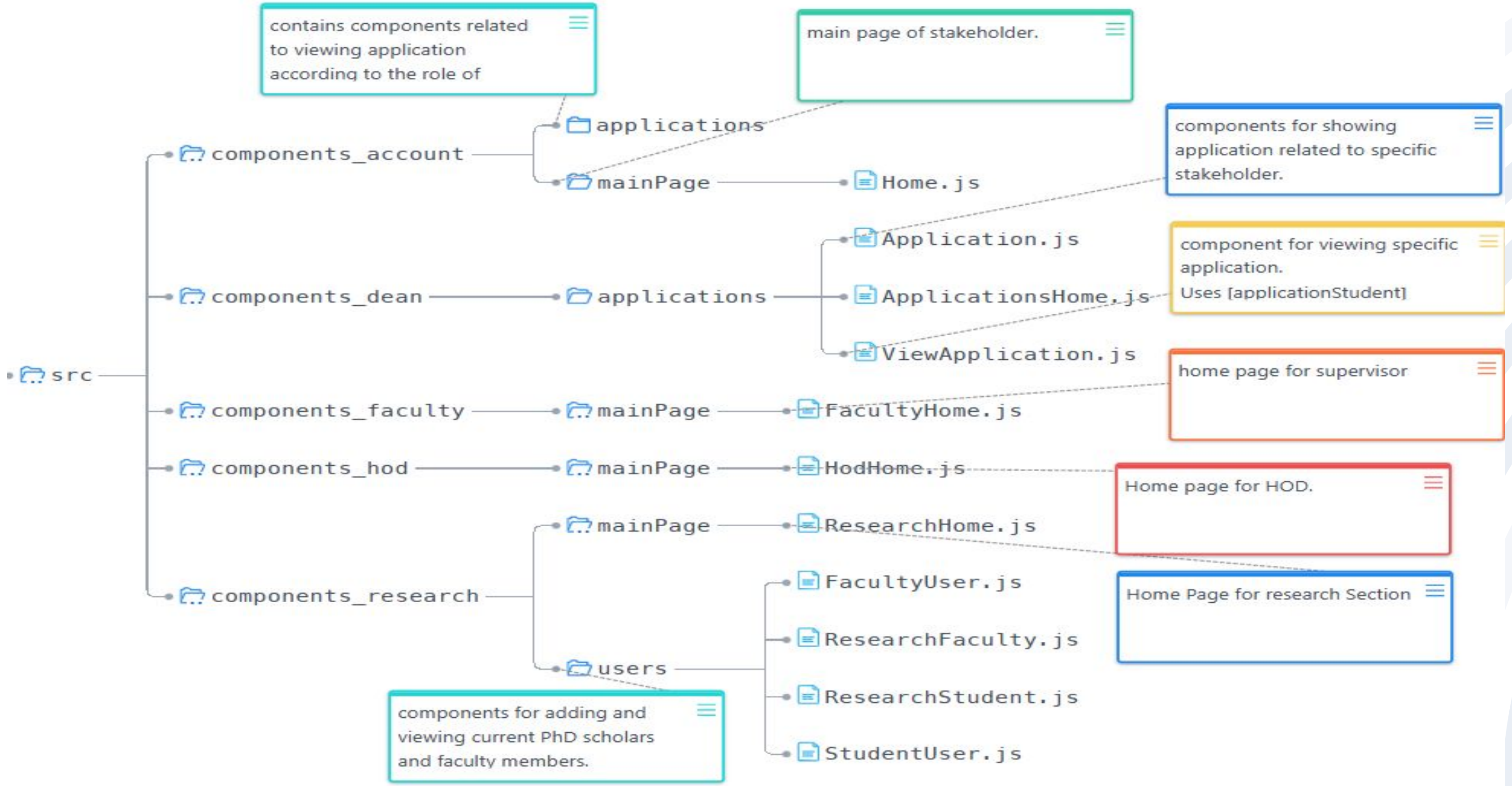


client
src
components
alerts
applicationStudent
commonPages
loading
navBars
sidePages
team
contains components for alert messages.
contains all components regarding taking action, viewing student application.
common Pages used in building the complex components within the other
Skeleton Components helping smooth transition during data fetching.
All navbar related Components
contains pages like error pages, landing page etc.
components related to Team page.

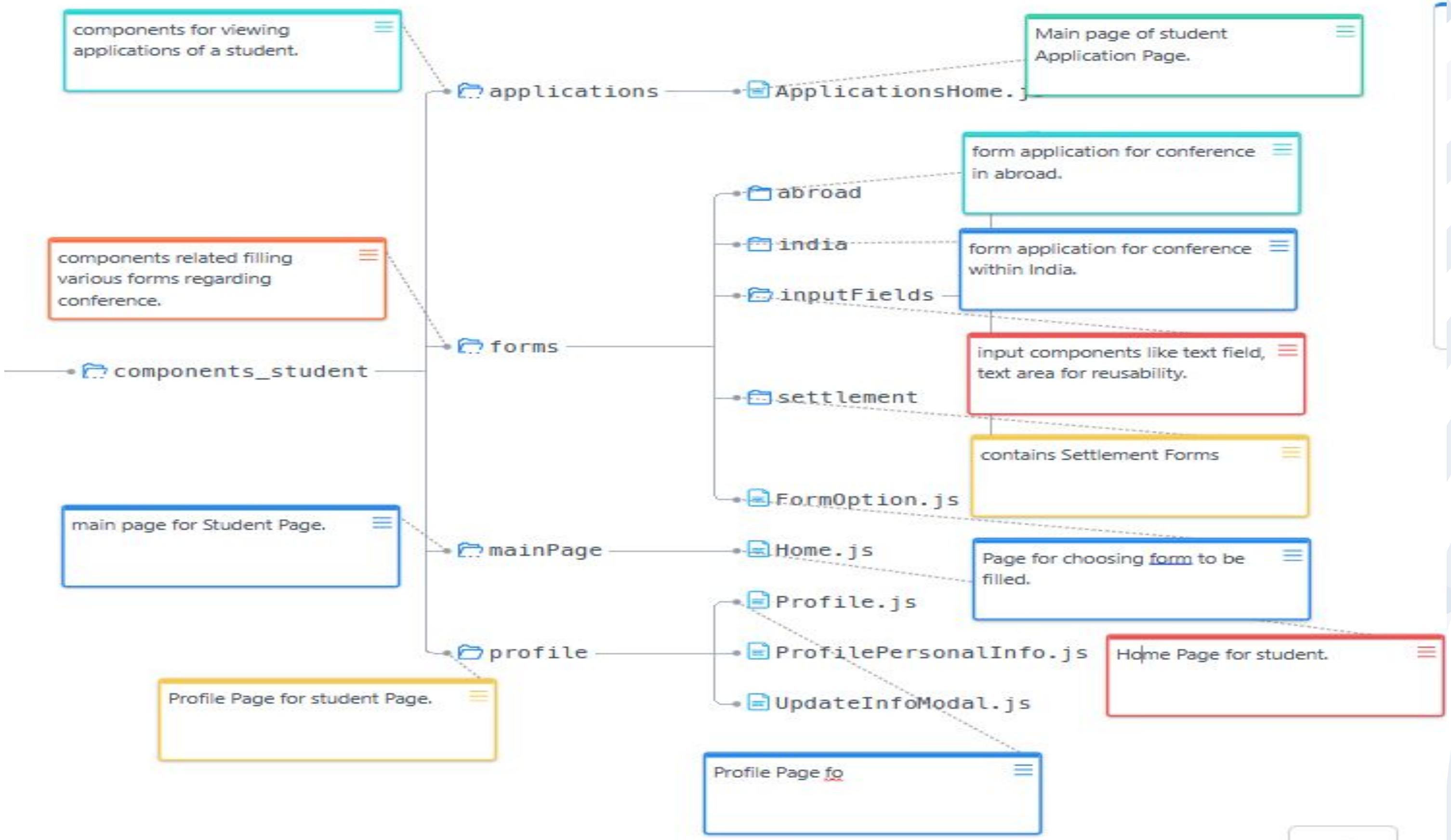
/client/src/components/applicationStudent


contains all actions stakeholders can take according to its role.
contains tabs related info of action stakeholder take.
contains component for crop signature.
Application main page where all above components are called.
src
components
applicationStudent
Actions
AccountAction.js
DeanAction.js
FacultyAction.js
HodAction.js
ResearchAction.js
Tabs
AccountSection.js
ConferenceDetailsAbroad.js
ConferenceDetailsIndia.js
DeanDetails.js
FacultyDetails.js
FinanceDetails.js
PersonalDetails.js
ResearchSection.js
uploadSign
ApplicationMainPage.js

/client/src


contains components related to viewing application according to the role of
main page of stakeholder.
components for showing application related to specific stakeholder.
component for viewing specific application. Uses [applicationStudent]
home page for supervisor
Home page for HOD.
Home Page for research Section
components for adding and viewing current PhD scholars and faculty members.
src
components_account
applications
mainPage
Home.js
components_dean
applications
Application.js
ApplicationsHome.js
ViewApplication.js
components_faculty
mainPage
FacultyHome.js
components_hod
mainPage
HodHome.js
components_research
mainPage
ResearchHome.js
users
FacultyUser.js
ResearchFaculty.js
ResearchStudent.js
StudentUser.js

/client/src/components_student


components for viewing applications of a student.
applications
ApplicationsHome.js
Main page of student Application Page.
form application for conference in abroad.
abroad
india
form application for conference within India.
inputFields
components related filling various forms regarding conference.
forms
input components like text field, text area for reusability.
components_student
settlement
contains Settlement Forms
FormOption.js
main page for Student Page.
mainPage
Home.js
Page for choosing form to be filled.
Profile.js
profile
ProfilePersonalInfo.js
Home Page for student.
UpdateInfoModal.js
Profile Page for student Page.
Profile Page fo

Back End

APIs

Student Side API’s

studentInfoLoading

- This is a Node.js route that handles a POST request to the "/studentInfoLoading" endpoint.
- The first thing the route does is extract the JWT token from the "Authorization" header of the request. It checks to make sure that the header exists and that a token was provided. If either of these checks fail, it sends a response with a 422 error status and an error message.
- After extracting the token, the route attempts to verify it using the jwt.verify() method, passing in the token and a JWT_SECRET environment variable. If the verification fails (i.e. the token is invalid or has expired), it sends a response with a 422 error status and an error message.
- If the token is successfully verified, the route extracts the user's email and role from the token's decoded payload. It then uses the findOne() method of a User model to query the database for a user with that email. If the query succeeds, the route sends a response with a 200 status and the user's data in JSON format. If the query fails, it sends a response with a 420 error status and an error message.
- Overall, this route is responsible for retrieving student information from a MongoDB database and ensuring that the request is authorized using a JWT token.

Student Side API’s

updateInfo

- This is another Node.js route that handles a POST request to the "/updateInfo" endpoint.
- Like the previous route, it extracts the JWT token from the "Authorization" header of the request and verifies it. If the header or token are missing, it sends an error response.
- After verifying the token, the route extracts the user's email from the decoded payload of the token, and also extracts the mobileNo field from the request's body.
- It then uses the updateOne() method of a User model to update the user's information in the MongoDB database. Specifically, it sets the mobileNo field of the user with the given email to the value passed in the request body.
- If the update is successful, the route sends a response with a 200 status and a message indicating that the update was successful. If the update fails, it sends a response with a 422 error status and an error message.
- Overall, this route allows an authenticated user to update their mobile number in the database by sending a POST request with a valid JWT token and a mobileNo field in the request body.

Student Side API’s

studentApplicationSubmit

- This is a POST request to the '/studentApplicationSubmit' endpoint of a router in a Node.js app using Express.js. The route is used to handle the submission of a student's application data for a conference. The request contains data about the student's personal details, the conference details, and files such as abstract, brochure, and acceptance.
- The route first extracts the data from the request and then uploads the files to a Google Drive folder using the Google Drive API. Then it saves the data to a MongoDB database using Mongoose.
- Finally, it sends a response to the client with a message indicating whether the application has been successfully submitted or not.
- This API is for applications within India.

Student Side API's

studentApplicationSubmitAbroad

- This is a POST request to the '/studentApplicationSubmitAbroad' endpoint of a router in a Node.js app using Express.js. The route is used to handle the submission of a student's application data for a conference. The request contains data about the student's personal details, the conference details, and files such as abstract, brochure, and acceptance.
- The route first extracts the data from the request and then uploads the files to a Google Drive folder using the Google Drive API. Then it saves the data to a MongoDB database using Mongoose.
- Finally, it sends a response to the client with a message indicating whether the application has been successfully submitted or not.
- This API is for abroad applications.

Student Side API’s

<u>studentApplicationView</u>

- This is an API endpoint for the '/studentApplicationView' route. It expects a POST request with a JWT token in the authorization header for authentication.
- The endpoint retrieves the email from the JWT payload, and queries the AppData collection in the database for all documents with that email and a status of "0".
- The retrieved documents are sorted in descending order of updatedAt field and returned as a JSON response with status code 200. If there is an error, it is logged to the console, but no response is sent back to the client.
- Overall, this endpoint allows a student to view all their submitted applications with a status of "0".

Student Side API’s

createApplicationToken

- This is an Express.js route that creates a token for a particular student application.
- It receives a POST request with the application ID in the request body, generates a token using the genAppToken function (which is not shown here), and returns the token as a response in a JSON object with the key appToken.
- The purpose of this token is likely to authenticate the student when they later access the application, possibly for editing or updating information.

Student Side API’s

infoLoading

- This is an Express.js API route handler that listens to the HTTP POST requests on the '/infoLoading' endpoint
- The handler first reads the authorization header from the request to ensure that the client is authenticated to access this route. The handler then verifies the JSON web token (JWT) contained in the authorization header to ensure that it is valid and has not been tampered with.
- The handler extracts the user's role and email from the decoded JWT. The handler uses the user's email to fetch the corresponding user record from the Mongo database.
- The handler then returns the fetched user record in the response to the client.Overall, this handler is used to load the user information for a given authenticated user.

Student Side API’s

applicationPending

- **This is a route handler function that handles a POST request to retrieve all pending applications for faculty members to review. Here's a breakdown of the code**
- **The function starts by using a try-catch block to handle any errors that might occur during the execution of the function. The variable status is set to 0, indicating that the application is pending.**
- **The await keyword is used to wait for the results of the AppData.find() method, which is used to query the MongoDB database for all application data where the status is 0.**
- **The results are stored in the appData variable.**

Faculty Side API's

studentInfoFaculty

- **This is a route for handling a POST request to retrieve student information for a particular faculty.**
- **The route expects two parameters in the request body: semail and femail, which correspond to the email of the student and the email of the faculty, respectively.**
- **The route first attempts to find the user with the given semail using the User.findOne() method. If the user's nameOfSupervisor matches the given femail, the route proceeds to retrieve the student's application data using the AppData.find() method with the semail parameter. The retrieved data is then returned as a JSON response with a 200 status code.**

Faculty Side API’s

<u>facultyApproveOrDisapprove</u>

- **The route is defined with the endpoint '/facultyApproveOrDisapprove' and a callback function that takes in the request (req) and response (res) objects.**
- **The function begins by checking for the presence of a bearer token in the request headers. If there is no token, it returns an error response with status code 422.If a token is present, it is verified using the JWT_SECRET key. If verification fails, it returns an error response with status code 422.**
- **The function then creates a new folder in Google Drive with a name based on the conference dates and name, and uploads the faculty member's signature image to this folder.If the upload is successful, it creates a public URL for the image and updates the application data in the database with the new status, the email address of the user who made the update, and the URL of the faculty member's signature image.**

Faculty Side API's

<u>viewFacultyApplications</u>

- **Extracts the JWT token from the request headers.Verifies the token to extract the email of the faculty user.Finds all the applications in the AppData collection, sorted by updatedAt timestamp in descending order.**
- **Finds all users in the User collection.**
- **Filters the applications to only include those that belong to the faculty user's students.Returns the filtered applications as a JSON response with a 200 status code.**

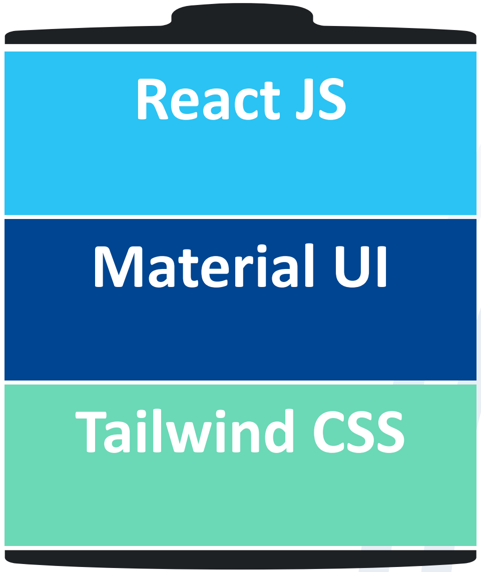
Other API’s

There are other similar API’s similar to the one explained

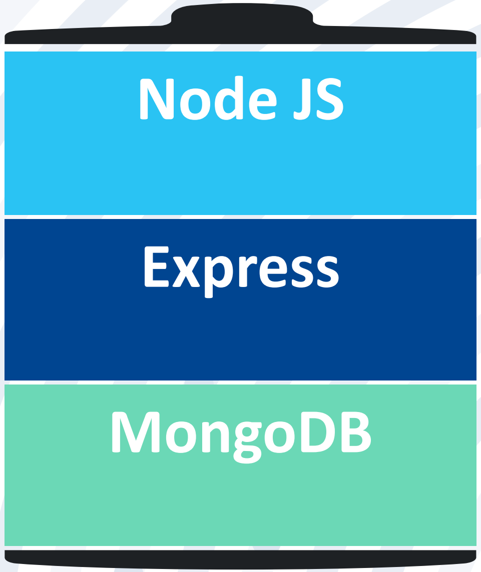
- viewAllApplications
- addStudent
- addFaculty
- researchApproveOrDisapprove
- hodApproveOrDisapprove
- accountsApproveOrDisapprove
- deanApproveOrDisapprove
- studentInfoFaculty

Technology Stack

FrontEnd



BackEnd



Deployment

We are pleased to announce that the PhD Conference Grant Management site has been successfully deployed and is now accessible on the internet via an IP address. The site is now live and ready to accept applications from prospective students who are interested to attend conferences across India or over the globe.

PhDCGM: http://172.30.2.244/

Acknowledgement

We would like to take this opportunity to express our heartfelt gratitude to everyone who has contributed to the successful completion of our Development Engineering project. Without the guidance and support of several individuals, this project would not have been possible.

First and foremost, we extend our sincere thanks to Dr. Puneet Goyal , our project guide and mentor, for his constant support, guidance, and motivation throughout the project. His valuable insights and feedback have been instrumental in shaping our project and helping us achieve our objectives.

We would also like to express our gratitude to the Research Section and Accounts Section (IIT Ropar) and Teaching Assistants (TAs) for their valuable feedback and suggestions during the course of the project. Their timely inputs and suggestions have been invaluable in improving the quality of our work.

Once again, we extend our heartfelt thanks to everyone who has contributed to the successful completion of this project. Your support and guidance have been crucial in making this project a success.

Thank You!