Fearcapefeed (FCF)

Project Document

1. INTRODUCTION

Feedcapefear is a social blogging and networking solution for company based employees to post and discuss community services to aid CSR of the participating companies. Employees from different organizations can collaborate, share, discuss and participate in social events and also communicate emergencies to everyone on the platform for immediate attention and help. This is a web based application with responsive design enabling users to interact with application from anywhere.

2. APPLICATION FEATURES

2.1. User Signup

Users can register/sign up on the application with User Image, Firstname, Lastname, Organization and Password of their choice. An Auto generated user-id is presented to the user in the following format.Ex: User with following credentials
Firstname: **J**ohn
Lastname **D**oe
Organization: **W**ipro
will have user-id as FCFWJDxxxx the last four characters are random digits with W from Wipro, J from John and D from Doe. The user can upload image files in jpg, jpeg, or png formats only not exceeding the size of 2 Mb. After signup the user account is submitted for approval to the admin.

2.2. Access control (Admin).

Admin can approve or reject users after verifying their credentials offline. Admin approval will provide access to the user account to all features of the application. Users can then login to the application with auto generated user-id and password set during registration/sign-up.

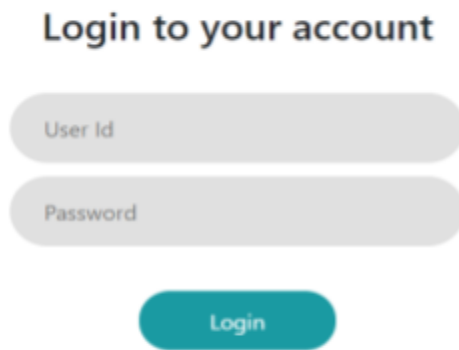## 2.3. User Login

User can login with the auto generated user-id as mentioned in Section 2.1 and password provided during the registration process. In case a user forgets his/her password he/she can then create a new account or contact admin for access.
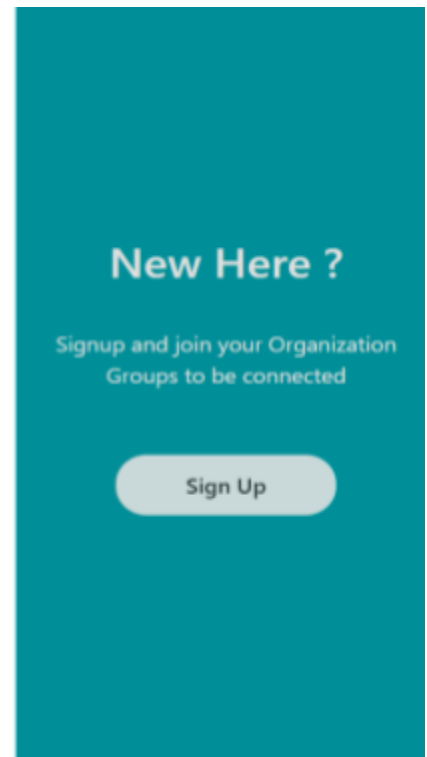
2.4. User Dashboard

Upon successful login the user is redirected to the user dashboard page and his details are stored in browser's local cache to allow auto login of the application across the browser sessions. Dashboard has a navigation bar at the top of it which contains Logo of application , Link to groups, a Home button and a Logout button, Clicking on the button allows the user to log out of the application which will also clear the data in the browser's local cache Dashboard has three Main panels. Left panel lists the user details such as User Image, user-id, organization and user details provided during sign-up
Central Panel contains a accordion model form to create posts with message, post type, image in jpg/jpeg/png format, location fields followed by list of posts each containing image of the author of the post(user), Firstname and Lastname and time lapsed from post creation along with link to the location tagged in the post and image. Each post allows logged in users to like/unlike the post, view comments for the post and comment on the post.
Right panel consists of a list of users in the top section followed by a chat box in the bottom section. Upon clicking on user details in the user list, the chat box is populated with the appropriate chat data with message and time lapsed from message creation. There is a provision for messaging the user at the bottom of the chat box.
This concludes the dashboards section.
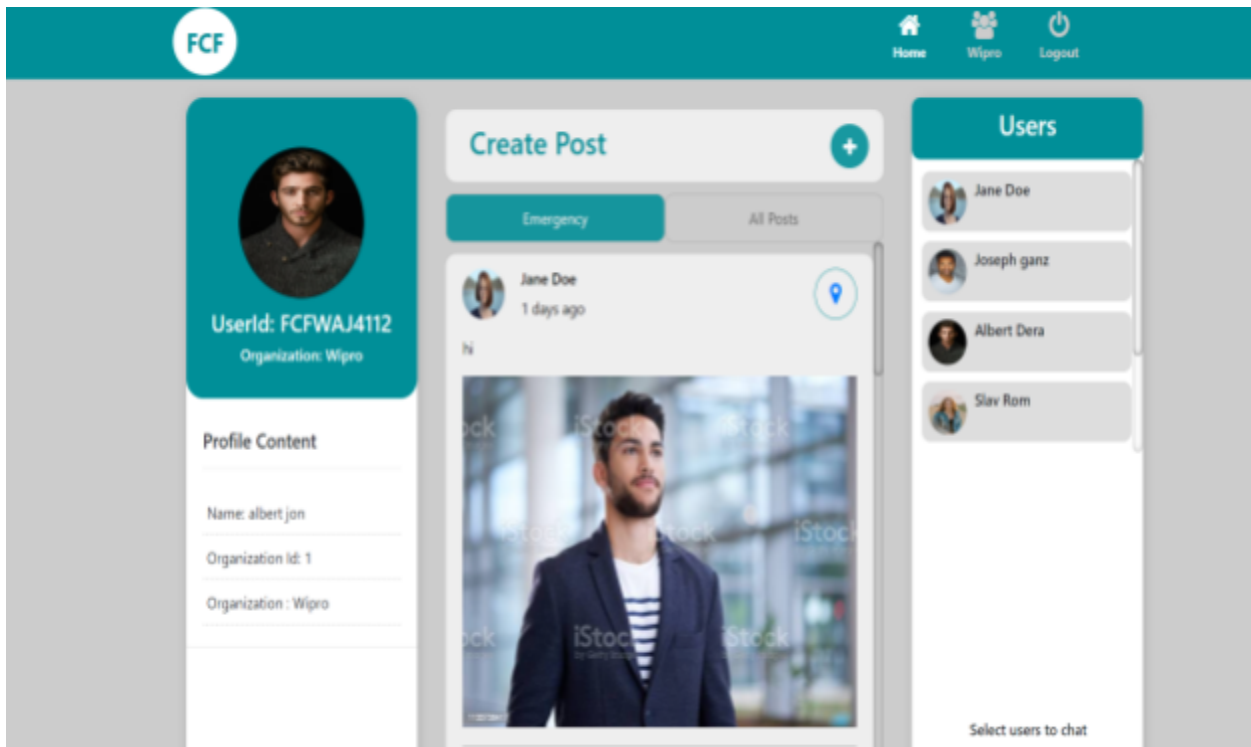
2.5. Emergency Post

In the central panel of the dashboard as mentioned in the section 2.4 , It has an accordion model for creating posts. If the user wants immediate attention to the post then change the post type to emergency and Post. So the post will be displayed in the emergency panel as the latest post.

2.6. Post

The users can send a normal post or emergency post . After posting, There will be a list of posts from different users according to the time, and the latest emergency post will be displayed first. And to see all the posts in the panel. Click on All posts tab, user can see a list of all the posts including emergency posts.

2.7. Groups

In the Navigation bar of the dashboard, there is a link to groups. Clicking on the link redirects user to the groups panel. The Group Panel is the same as the dashboard panel as mentioned in section 2.4. In the groups panel, only the users who are in the same organization can see the data posted by other users. . Left panel lists the user-details such as User Image, user-id, organization.
Central Panel contains a accordion model form to create group-posts with message, post type, image in jpg/jpeg/png format, location fields followed by list of group-posts each containing image of the sender of the post(user), Firstname and Lastname and time lapsed from post creation along with link to the location tagged in the post and image. Each post allows logged in users to like/unlike the post, view comments for the post and comment on the post.

Right panel consists of a list of users in the top section followed by a group chat box in the bottom section. The group chat box is populated with the appropriate chat data with Name of the user ,message and time lapsed from message creation. All the members in the group can chat in this Group chat box .

2.8. Group Posting

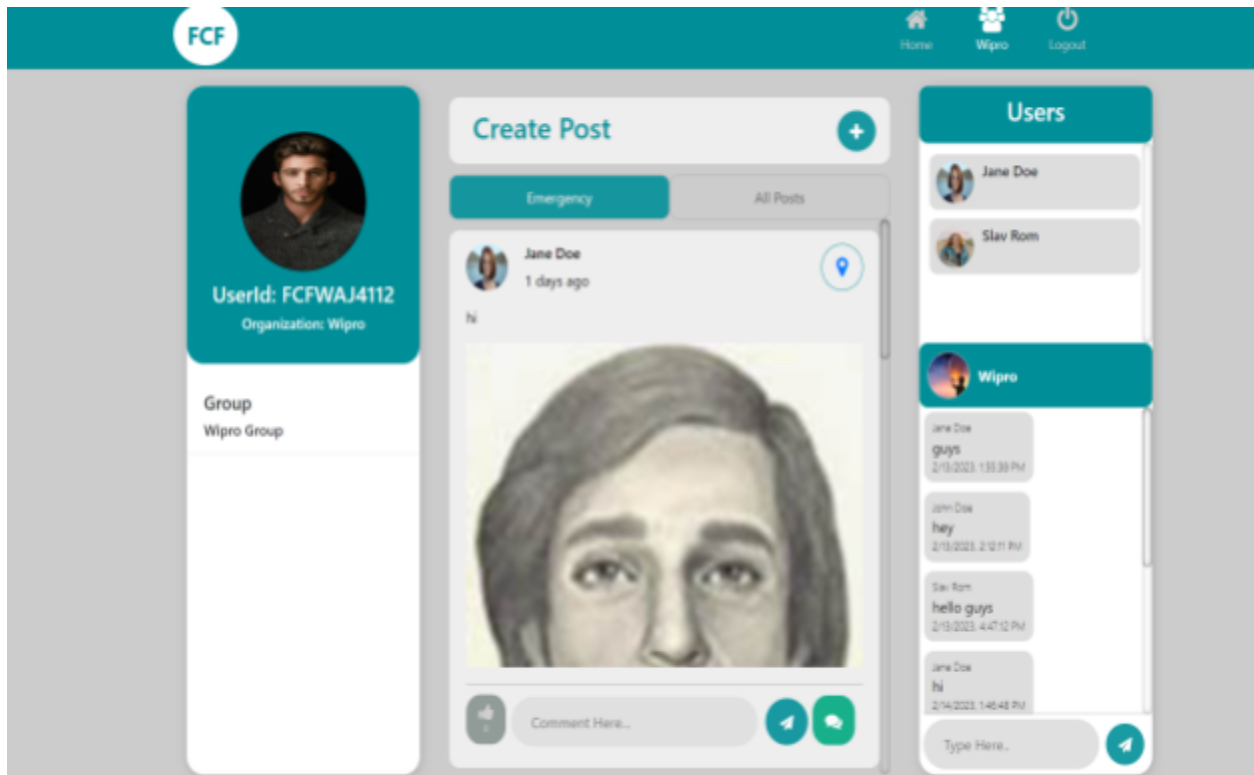In the central panel of the groups as mentioned in the section 2.7 ,The users in the group can send a normal post or emergency post . After posting, There will be a list of posts from different users according to the time, and the latest emergency post will be displayed first. And to see all the posts in the panel. Click on All posts button, user can see a list of all the group-posts including emergency posts.

2.9. Comments

In the central panel of dashboard and groups as mentioned in section 2.4 and 2.7 respectively, The Logged in user can comment on any posts posted by different users and the user can view the comments which are posted by different users at the bottom of the Post.

2.10. Like/Unlike Posts

In the central panel of dashboard and groups as mentioned in section 2.4 and 2.7 respectively, The Logged in user can like/unlike the post switch are posted by different users. If the post is liked by the user, the color of the like button changes to blue and displays how many users like that post.

2.11. Personal Message (Chat)

In the Right panel of the dashboard as mentioned in the section 2.4, It displays a list of users, Upon clicking on user details in the user list, the chat box is populated with the appropriate chat data with message and time lapsed from message creation.

2.12. Group Message

At the bottom of the Right panel of the Groups as mentioned in the section 2.7, There is a fixed group-chatbox. Multiple users from the group can chat at the same time. The chat box is populated with the appropriate chat data with Name of the sender(user), message and time lapsed from message creation.

2.13. Location

User can tag his/her location when he/she clicks on the Tag location icon. It actually takes the user's system latitude and longitude coordinates to locate the user's location.

2.14. Logout

Logout button is given to the right side of the nav-bar. Clicking on the button allows the user to log out of the application which will also clear the data in the browser's local cache.

3. TECHNOLOGIES EMPLOYED

3.1. React

- React is a free and open-source front-end JavaScript library for building fast and interactive user interfaces for web or Mobile Applications. Any user can access the source code, modify it and can use it. In Model, View, Controller architecture , React is the View which is responsible for how the app looks like and feels.
- React divides User Interface(UI) into different Components which makes the code easier to debug and each Component has its own Property and Function. Components also make the development and maintenance of web applications faster as multiple developers can work on different components of the same application simultaneously.
- However,React is only concerned with the user interface and rendering components to the DOM(Document Object Model), so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

3.2. Express

- Express.js or Express is a back-end web application framework for building APIs with Node.js. Express is released as free and open-source Software under the MIT License.
- It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js. Express provides mechanisms to write handlers for requests with different HTTP verbs at different URL paths (routes).

- With Express, developers can create an API very easily for the server. Express includes significant features that let you dynamically create a back-end API. It doesn't require any third-party dependencies and doesn't come with any frameworks or plugins.

3.3. NPM

- NPM is a package manager for the JavaScript programming language maintained by npm, Inc. npm is the default package manager for the JavaScript runtime environment Node.js . npm is the world's largest software registry. Open source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well.
- Node uses a node package manager or an NPM to install all new updates and packages. Node bundles single applications built through React to simplify the compilation process. This is done through the use of appropriate Node modules and by using a web pack.
- npm is a lean, modular web server for rapid full-stack development. Supports HTTP, HTTPS and HTTP2. Small and 100% personalisable. Load and use only the behavior required by your project.

3.4. MySQL

- MySQL is a Database Management System. It is the World's most popular open source database.
- MySQL Database is a client/server system that consists of a multi-threaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of Application Programming Interfaces (APIs).
- MySQL allows you to handle, store, modify and delete data and store data in an organized way. SQL does not support any connector. MySQL comes with an in-built tool known as MySQL Workbench that facilitates creating, designing, and building databases.

4. APPLICATION ROUTES
4.1. Back-end Routes
4.1.1.   Signup

Url : http://localhost:2000/signup
Input parameters(query parameters): user_id, image, firstname, lastname, organization_id, organization_name, role, status .
Process: After the user signup, All the details entered by the user in the form are sent to the backend using axios as formdata and unique user Id is created to the user as mentioned in session 2.1, role is set as "USER" and status as 0 for all the new users who registered in this application. Then the user details are inserted into the user_details and users table using database connection.

Response: user object as JSON string with following parameters
```
user = {
        userid:",
        firstname:"",
        lastname: "",
        organizationid:"",
        organizationname:"",
        password:"",
        role:",
        status:""
    };
```

4.1.2.   Admin

Url : http://localhost:2000/getadmin?status=0
Input parameters(query parameters) : status
Response: The admin gets the details of users from the url whose status is 0.

4.1.3.   Admin Approve

Url : http://localhost:2000/adminapprove
Input parameters: approve
Response: Once the admin approves the user ,using axios the call goes to the url and using UPDATE command ,the status of the user changes to 1 in the user_details table.

4.1.4.   Admin Decline

Url : http://localhost:2000/admindecline
Input parameters : decline
Response: If the admin declines the user, using axios the call goes to the url and using DELETE command that user information is deleted from users table and user_details table.

4.1.5.   Login

Url : http://localhost:2000/login
Input parameters : userid, password
Process: When the user login, the call goes to the url with the data user id and password, then using a SELECT command, user details are selected based on the user id. Once the user logged in then all the user details are stored in the local storage.
Response: It takes the user to the dashboard once the user login with valid credentials, then all the user details are stored in the local storage.

4.1.6.   Create Post

Url : http://localhost:2000/createpost

Input parameters : Image, posttype, posttext, userid, organizationid, organization name, latitude,longitude.

Process: Using the details entered by the user from session 4.1.5 , Post is created and during Post creation, All the details entered by the user in the form are sent to the backend using axios as formdata. Using the INSERT command all the Post details are stored in the post table.

Response: user object as JSON string with following parameters.

```
user = {
        posttext : "",
        posttype : "",
        lat: "",
        long: "",
        userid: "",
    };
```

4.1.7.   Upload Post

Url : http://localhost:2000/uploadpost?userid=""

Input parameters : userid

Response: After creating the Post , the call goes to the url for displaying the created posts. using userid the SELECT command is used to retrieve the post details from the post table and user_details table.

4.1.8.   Like Post

Url : http://localhost:2000/likepost

Input parameters : postid,userid

Process: When the user likes the post, the call goes to the url using axios, and using the INSERT command the postid and userid are stored in the user_likes table.

Response: Once the user likes the post, It will be shown in blue color.

4.1.9.   Unlike Post

Url : http://localhost:2000/unlikepost

Input parameters : postid,userid

Process: When the user unlikes the post, the call goes to the url using axios, and using DELETE command the postid and userid are deleted from user_likes table.

Response: Once the user unlikes the post, it will be shown in gray color.

4.1.10.   Comments

Url : http://localhost:2000/commentdata

Input parameters : comment, userid , postid, timestamp.

Process: comment is created by the user and all the details entered by the user in the form are sent to the backend using axios as formdata. Using the INSERT command all the comment details are stored in the comments table.

Response: user object as JSON string with following parameters.

user = {

```
      comment:"",
      userid:"",
      postid:"",
      timestamp:""
};
```

4.1.11.   Get Comment Data

Url : http://localhost:2000/getcommentdata?userid=""&postid=""
Input parameters : userid,postid
Response: user object as JSON string with following parameters.
```
user = {
       user_id: "",
       post_id : "",
       comments: "",
       time_stamp: "",
       image: "",
       firstname:"",
       lastname: "",
       organizationid:"",
       organizationname:"",
       role:",
       status:""
};
```

4.1.12.   Select Users

Url : http://localhost:2000/selectusers?userid=
Input parameters : userid , orgid
Response: The user gets the details of other users from this url, the Select command is
used to select the users other than the current user where status is 1 and orgid is 0.

4.1.13.   Chat Data

Url : http://localhost:2000/chatdata
Input parameters : chat ,sender , receiver , timestamp
Process: chat is sent by the user and all the details entered by the user in the form are
sent to the backend using axios as formdata. Using the INSERT command all the chat
details are stored in the chat table.
Response: user object as JSON string with following parameters.
```
user = {
       message:"",
       to_user:"",
       from_user:"",
       timestamp:""
};
```

4.1.14.   Get Chat Data

Url : http://localhost:2000/getchatdata?userid=""&recieverid=""
Input parameters : userid,recieverid
Response: user and receiver can see the chat data in user chatbox.

4.1.15.   Group Post

Url : http://localhost:2000/grouppost
Input parameters : Image, posttype, posttext, userid, organizationid, organization name, latitude,longitude.
Process: Using the details entered by the user from session 4.1.5 , Group Post is created and during Post creation, All the details entered by the user in the form are sent to the backend using axios as formdata. Using the INSERT command all the Group Post details are stored in the post table using organization id.
Response: user object as JSON string with following parameters.
user = {
        posttext : "",
        posttype : "",
        lat: "",
        long: "",
        userid: "",
        orgid:""
    };

4.1.16.   Upload Group Post

Url : http://localhost:2000/uploadpost?userid=""&orgid=""
Input parameters : userid ,orgid
Response: After creating the Grouppost , the call goes to the url for displaying the created posts. using userid and orgid the SELECT command is used to retrieve the post details from the post table and user_details table.

4.1.17.   Group Users

Url : http://localhost:2000/groupusers?orgid=""&userid=""
Input parameters : userid , orgid ,status
Response: The user gets the details of other users from this url, the Select command is used to select the users other than the current user where status is 1 and orgid is not equal to 0.

4.1.18.   Group Chat Data

Url : http://localhost:2000/groupchatdata

Input parameters : group_id ,userid , message, timestamp
Process: In Group chat box all the users in the group can chat with other members and all the details entered by the user in the form are sent to the backend using axios as formdata. Using the INSERT command all the chat details are stored in the group_chat table.
Response: user object as JSON string with following parameters.
user = {
    group_id:"",
    userid:"",
    message:"",
    timestamp:""
    };

4.1.19. Get Group Chat Data

Url : http://localhost:2000/getgroupchatdata?userid=""&orgid=""
Input parameters : userid,orgid
Response: users with the same orgid can see the chat data in the group chatbox.

4.2. Front-end Routes

4.2.1. Signup

This lets users to signup as per section 2.1
Url: http://localhost:4000/signup .
Input parameters: Image, firstname, lastname, organization, password
Response: This app starts by signing up users. The user needs to enter his details like Image, firstname,lastname, Name of Organization and password to sign up, then the user can see an alert box which has user id. The user needs to copy that "user id" for login. As per session 4.1.1 , The user details are stored in the database.

4.2.2. Admin

This is admin page as per section 2.2
Url: http://localhost:4000/dashboard .
Input parameters: Approve, Decline
Response: Once the user is registered, The admin can see the registered users image, name and organization as mentioned in session 4.1.2. The admin can Approve or Decline the user. Once the admin approves the user as mentioned in session 4.1.3 , he/she can login using user-id and password. If the admin Declines the user as mentioned in session 4.1.4, the declined user cannot login with his details as it will show invalid credentials.

4.2.3.   Login

This lets users to login as per section 2.3
Url: http://localhost:4000/ .
Input parameters: userid,password
Response: After logging in with valid credentials as per session 4.1.5 ,then the local storage will be set with user data and It will redirect to the dashboard page.

4.2.4.   Dashboard

This is Dashboard page as per section 2.5
Url: http://localhost:4000/dashboard .
Response: Renders a page with three panels as mentioned in 2.5. In main panel, User can create a post as per session 4.1.6, and the created post is displayed in the main panel as mentioned in 4.1.7 and and other users can like and unlike the post as mentioned in 4.1.8 and 4.1.9 , the users can comment and view the comments as per 4.1.10 and 4.1.11 and in right panel there are different users for the current user to select and chat with the specific user as per 4.1.12 and on clicking the specific user, the chat box is populated with the appropriate chat data with message and time lapsed as mentioned in 4.1.14 and user can send message to the selected user as per 4.1.13.

4.2.5.   Groups

This is Dashboard page as per section 2.8
Url: http://localhost:4000/groups .
Response: Renders a page with three panels as mentioned in 2.8. In main panel, User can create a post as per session 4.1.15, and the created post is displayed in the main panel as mentioned in 4.1.16 and and other users can like and unlike the post as mentioned in 4.1.8 and 4.1.9 , the users can comment and view the comments as per 4.1.10 and 4.1.11 and in right panel the user who are in the same group as current user are displayed as per 4.1.17 and The current user can send chat with all the other users in the group as per 4.1.18. The chat box at bottom of the right panel is populated with the appropriate chat data with name , message and time lapsed as mentioned in 4.1.19.