Prompt Engineering Handbook

Project: ICECD Global Community Platform

Event: JPMC Code for Good Hackathon
Organization: ICECD (International Centre for Entrepreneurship and Career Development)
Team: 24

Table of Contents

1. **Use Cases**
    - The Problem Statement
    - Key Pain Points
    - Stakeholder Needs
2. **Tools Used**
    - Tools Utilized
    - Our Methodology: AI-as-a-Co-Developer
3. **The Prompting Process: Prompts Used**

---

1. Use Cases

The Problem Statement

ICECD, an NGO active in 55 countries, empowers individuals through entrepreneurship training. Their primary communication and resource-sharing tool is WhatsApp, which has significant limitations. They require a scalable, centralized platform to foster a global community, track beneficiary progress post-training, and provide continuous learning.

Key Pain Points

- **Scalability:** WhatsApp groups are difficult to manage and scale across a global user base.
- **Tracking:** Inability to systematically track the long-term progress and success of graduate entrepreneurs.
- **Engagement:** Lack of a structured platform for trainees to engage with curriculum, ask questions, and connect with mentors.
- **Community:** Difficulty in connecting entrepreneurs separated by distance but united by profession or shared challenges.

Stakeholder Needs

1. **ICECD Team (Admin):** Needs to oversee the ecosystem, manage users, track progress visually, and facilitate mentorship connections.

2. **Trainees:** Need access to learning materials, a way to track their progress, and a forum to ask for help.
3. **Graduates (Entrepreneurs):** Need access to refresher courses, a platform to mentor new trainees, and a way to share their success stories.

2. Tools Used

Vision

To create a unified, role-based digital ecosystem that serves as the central hub for ICECD's global network. The platform is designed to be intuitive and accessible for users from diverse backgrounds (rural, urban, slum) while providing powerful administrative and analytical tools for the ICECD team.

Core Features

- **Role-Based Dashboards:** Tailored interfaces for Admins, Trainees, and Graduates, providing relevant tools and information to each user.
- **Interactive Learning & Progress Tracking:** A simple system for trainees to consume course content and mark their progress, with visual feedback.
- **Community Forum & Mentorship Matching:** An open forum for queries, which Admins can use to create dedicated, private "WhatsApp-like" group chats, connecting trainees with experienced graduate mentors.
- **Automated Success Tracking:** A system to simulate the 1000-day follow-up process, culminating in a "Success" tag and the ability for graduates to share their inspiring stories, creating a cycle of motivation.

3. The Prompting Process: Prompts Used

I have a project idea:
Currently we are in JPMC CFG hackathon, we have chosen the problem statement:
"Organization: ICECD
Website: https://icecd.org/
Mission
ICECD's mission is to create an "Entrepreneurial Ecosystem", through increasing access and control over productive resources like study materials for the needy to facilitate their economic, social and political empowerment through Human Resource Development by upskilling and training their beneficiaries in vital life and career skills, keeping gender issues as a key focus.
Context
ICECD is currently active in 55 countries with over 38+ years of experience, having impacted 10 million people including slum, rural and urban populations. Having WhatsApp groups for communications and sharing entrepreneurship-based resources on a common platform. Currently, the organization needs a suitable alternative for WhatsApp, which is their main communication channel. The key methodology that enhances ICECD's success rate is following up and tracking the progress of their beneficiaries after their training is completed to ensure the business lasts beyond its critical time ceilings. The organization measures the success of their ventures by tracking the number of people reaching out to their organization, people's stories/testimonies, and the profit made by the organization.
Strategy:
The organization is in an intermediate stage of its tech roadmap, having identified key areas for improvement in networking, two-way community-based communication, and entrepreneurship curriculum

refreshers via a common platform for all. The next stage involves tracking growth for the recipients of their training program beyond their graduation and developing a digital solution that can bring entrepreneurs separated by distance but united by profession together to build a global community of shared interests and mutual struggles. The solution should also have the capacity to allow users to engage creatively with the curriculum and content covered in the training program as a refresher or continuous learning journey.
Stakeholder Universe:
Stakeholders include the nonprofit team, trainees, and graduates of the entrepreneur program. The nonprofit team should be able to facilitate and monitor the interactions of the platform while also tracking and managing existing entrepreneurs within their ecosystem. They should also have the ability to onboard, collate, and prescribe the ecosystem to budding entrepreneurs who will be recipients of their training. The trainees must be able to leverage the existing ecosystem to advance their learning journey and effectively track their engagement and performance. Graduate entrepreneurs must be able to interact with their peers, collaborate across borders effectively, and upskill themselves with the latest developments in the curriculum.
Challenge
Develop a comprehensive digital solution that addresses the ecosystem tracking challenges faced by ICECD organization. The solution should creatively engage the ecosystem through on-platform elements while also allowing for easy access to graduate entrepreneurs within the ecosystem to help their peers and the new recruits in their journey. This solution should help the organization effectively collate, visualize, and derive insights and bring the beneficiaries of their programs under one roof.
Technology Baseline
ICECD are currently using WhatsApp for their communication needs."

Our team discussed over this and we brainstormed to come up with an idea.
The idea includes a flow that I will be putting in:
A page where there would be 3 sections, which correspond to:

1. ICECD Team (Admin)
2. Trainees
3. Graduates (Entrepreneurs)
   Upon clicking on any of these sections, it should:
   Open up a login page, there should be a phone number and OTP option but for now, it should be able to accept any number in OTP as it is a prototype.

Upon login being done, for Trainees:

1. Show some courses, videos and resources to the trainee (for now, keep placeholders blocks) that they can choose from through tabs/sections and it should be able to track them upon their completion which they can provide through checkboxes/etc.
2. Have an open community button which upon clicking should take them to the open community page (which I will provide details later)
3. Show the group chats (which I will provide details later) that they are in, which are added by the Admin.

Upon login being done, for Graduates (Entrepreneurs):

1. It should show them refreshers for the courses, that they can choose out of, in case they want to and again, similar to (1) from Trainees, keep placeholders for the videos/resources etc. that are involved and ensure to have a progress check similar to that.

2. Have the same community button which upon clicking should take them to the open community page (which I will provide details later)
3. Have access to the group chats that they are added to, which would basically mean the trainee that they are assigned to.

Upon login being done, for ICECD Team (Admin):

1. It should show all the trainees and their assigned mentors and their progress.
2. It should also show the Graduates (Entrepreneurs) and their success stories (if any, I will provide details about what a success story is, later) and what is the follow up status (I will give information about this, later)
3. Have the same community button which upon clicking should take them to the open community page (which I will provide details later)
4. Also have access to all of the group chats and create the group chats and add the corresponding trainee and graduates.

Now, let us move into the tags involved.

1. Trainee should have a tag, talking about the business sector that they're part of (for now, choose 4-5 for demo purposes like beauty, groceries, furniture etc.) and this is pre-assigned.
2. Graduates can have the business sector tag and also, a success tag which I will talk about later.
   The tags here play a vital role in being able to filter out people by tag-based searching on a specific sector that they are in.

Now let's move on to the 'discuss later' that I mentioned:

1. Open community: This is a page where the trainees can post their queries and then the admin can form these "group chats".
2. Group chats: These include the admin and the corresponding members added by the admin.
3. Success stories and success tag: There should be a monthly-based SMS that is sent to the graduates pursuing a business, which would include a form that has questions based on: "Indicators include investment and loans secured, number of jobs generated, profitability, startup sustainability, and impact on someone's life. Intangible outcomes include improved quality of life, increased societal status, better healthcare, and overcoming social taboos." and then once the progress looks good according to the Admin and they receive a positive response for 1000 days then that graduate is qualified to give their success story which would mean that they would have a final SMS that sends them the link to the Graduate portal with the "success" button now in their page, where they can write their stories etc. Additionally, they should also be given a "success" tag now for better monitoring.

Now, here is some information about the linking part:

1. The trainee will basically have the phone number, password and tags in the database (or whatever) and the assigned mentor stored.
2. The graduate will be having the phone number, password and tags stored too but also the trainee that they are assigned to.
3. The admin will be having the access to everything that was mentioned from the points under admin at the top.

The UI/UX caters to rural, urban and slum area part of
the population, so keep that in your mind but at the same time it should be appealing and make the NGO like it. Just mention the tech stack once

My flow in the head:
The person opens the page, chooses the corresponding section and then based on that they proceed with the features linked as already mentioned but the most important part is the open community being a place for query pitch in, the group chat being a place for commonality (with respect to tags) and additionally the success story and post-process linking to be done as the problem statement mentions already.

Now, before you go ahead with the code generation or any of that, please give me your current process flow, from a user point of view so that I can verify if it matches with my vision too and if no, I could give feedback and if yes, I will give you a headsup about code generation.

No issues about time, so don't compromise on it.

This is the Project flow:
Of course. This is a critical first step. It's essential to ensure the user journey is perfectly mapped out before any code is written. A strong, clear user flow is the foundation of a successful application, especially in a hackathon setting.

Based on your detailed brief, I've translated your ideas into a step-by-step process flow from the perspective of each user. This narrative will help us visualize the platform in action.

Proposed User Process Flow
Phase 1: The Universal Entry Point

The Landing Page: A user opens the application URL. They are greeted with a clean, simple, and welcoming page. The ICECD logo is prominently displayed. The page has a clear heading like "Welcome to the ICECD Global Community."

Role Selection: Below the heading, there are three distinct, easily clickable cards or sections, each with a simple icon and clear text:

"For Our Team" (Admin)

"For Trainees" (Current Students)

"For Graduates" (Entrepreneurs)

Login Action: The user clicks on the section that represents their role (e.g., a new student clicks "For Trainees").

Authentication: They are taken to a login screen.

It prompts for a Phone Number.

After entering the number, they click "Send OTP".

An OTP input field appears.

Prototype Rule: For the hackathon, entering any 4-digit number (e.g., "1234") will successfully log them in. The system fetches the user profile associated with the phone number.

Phase 2: The Role-Based Experience
A) The Trainee's Journey (Let's call her Anika)

Login & Dashboard: Anika, a trainee in the "Beauty" sector, logs in. She lands on her personal dashboard. It's designed to be encouraging and easy to navigate.

Dashboard View: Her dashboard contains three main parts:

My Learning Path: A section with cards representing her assigned courses ("Module 1: Sourcing Materials," "Module 2: Basic Accounting"). Each card has a checkbox. When she completes a module, she checks the box, and a progress bar on her dashboard updates visually (e.g., "2 of 5 Modules Complete - 40%").

My Mentor Groups: A list of the private group chats she is in. For example, "Beauty Sector Cohort" and "Query: Skincare Business Plan" (a group created specifically for her).

Open Community Access: A large, friendly button labeled "Ask the Community".

Seeking Help: Anika is struggling with creating a business plan.

She clicks the "Ask the Community" button.

She is taken to the Open Community page. This looks like a simple forum or feed. She can see questions from other trainees. She can filter posts by a business sector tag (e.g., she can filter to see only "Beauty" related questions).

She clicks "Post a New Query," writes her question: "I need help structuring the financial projections for my new skincare line. Any advice or templates?", and submits it. Her post is now live in the Open Community feed.

B) The Graduate's Journey (Let's call him Raj)

Login & Dashboard: Raj, a successful graduate in the furniture business, logs in. His dashboard is focused on mentorship and continuous growth.

Dashboard View:

Mentor Center / My Groups: He sees the group chats where he is a mentor. He'll see "Furniture Q&A Group" and any specific groups he's been added to, to help a trainee.

Skill Refreshers: A library of content (videos, articles) he can access. These are advanced or refresher topics like "Advanced Digital Marketing" or "Scaling Your Production." These also have progress tracking if he chooses to engage.

Open Community Access: The same "Ask the Community" button, which for him, is a place to share expertise.

(Conditional) Success Story Button: If Raj has met the 1000-day criteria, a new, prominent button appears: "Share Your Success Story". His profile now also shows a "Success" tag/badge visible to the Admin and others in his groups.

The Follow-up & Success Path (Happening in the background):

Monthly SMS: Once a month, Raj receives an automated SMS: "Hi Raj, from ICECD! Time for your monthly check-in. Please tell us about your progress: [link_to_form]".

The Form: The link opens a simple mobile-friendly form asking about profitability, jobs created, investment secured, and a question about his personal quality of life.

Achieving "Success": After 1000 days of positive responses (tracked by the Admin), the system automatically grants him the "Success" tag and enables the "Share Your Success Story" button on his portal.

C) The ICECD Admin's Journey (The Facilitator)

Login & Master Dashboard: The Admin logs in to a powerful but clean dashboard, the mission control for the entire ecosystem. The dashboard has clear tabs or sections.

Dashboard Tabs:

Trainee Overview: A searchable and filterable list of all trainees. The columns show: Name, Business Sector (Tag), Course Progress, and Assigned Mentor (Graduate). The Admin can immediately see who is on track and who might need help.

Graduate Overview: A list of all graduates. The columns show: Name, Business Sector (Tag), Follow-up Status (e.g., "Last response: 5 days ago"), and the crucial Success Tag (Yes/No). The Admin can click on a graduate to view their historical monthly form submissions.

Community Feed: This is the live Open Community feed. The Admin sees Anika's new post about financial projections.

Facilitating a Connection: The Admin decides to act on Anika's post.

They use the Graduate list to find a suitable mentor. They filter by Business Sector: Beauty and Success Tag: Yes. They find a perfect match.

The Admin clicks a "Create Group Chat" button.

A simple form appears: Group Name, Add Trainee(s), Add Graduate(s).

They name the group "Query: Skincare Business Plan".

They add Anika (the trainee) and the selected successful Beauty graduate. The Admin is automatically included.

They click "Create". Now, this new group appears in the "My Mentor Groups" section for both Anika and the graduate. The connection is made.

This flow ensures that every feature you mentioned is integrated into a logical and purposeful user journey. The platform moves from a passive content repository to an active, admin-facilitated community hub that directly addresses the core challenges in the problem statement.

This is my current project that I have:

Part 1: Full Directory Structure

Create a new project folder (e.g., `icecd-platform`) and replicate this structure inside it.

```
icecd-platform/
├── public/
│   ├── index.html
│   └── icecd-logo.png  (You'll need to add a logo image here)
├── src/
│   ├── assets/
│   │   └── (empty for now, can hold other images/icons)
│   ├── components/
│   │   ├── Header.js
│   │   └── ProtectedRoute.js
│   ├── context/
│   │   └── AuthContext.js
│   ├── data/
│   │   ├── mockChats.js
│   │   ├── mockCommunity.js
│   │   ├── mockCourses.js
│   │   └── mockUsers.js
│   ├── pages/
│   │   ├── AdminDashboard.js
│   │   ├── CommunityPage.js
│   │   ├── GraduateDashboard.js
│   │   ├── LandingPage.js
│   │   ├── LoginPage.js
│   │   └── TraineeDashboard.js
│   ├── App.js
│   ├── App.css
│   ├── index.js
│   └── index.css
├── .gitignore
├── package.json
└── README.md
```

Part 2: Complete Code for All Files

Here is the complete code for every file. Copy and paste the contents into the corresponding files you created in the structure above.

`package.json`

This file will be created for you when you run `npx create-react-app .`, but you will need to add the dependencies. Run the installation commands from Part 3, which will populate this file correctly. For reference, it will look something like this:

```
{
  "name": "icecd-platform",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@mui/icons-material": "^5.14.19",
    "@mui/material": "^5.14.20",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.20.1",
    "react-scripts": "5.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": { "extends": ["react-app"] },
  "browserslist": {
    "production": [">0.2%", "not dead", "not op_mini all"],
    "development": ["last 1 chrome version", "last 1 firefox version", "last 1 sa
  }
}
```

`public/index.html`

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/icecd-logo.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="ICECD Global Community Platform"
    />
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;
    <title>ICECD Community Platform</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

(Action): Download a logo for ICECD, rename it to `icecd-logo.png`, and place it in the `public/` folder.

`src/index.css`

```
body {
  margin: 0;
```

```css
  font-family: 'Roboto', -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  background-color: #f4f6f8;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

src/App.css

```css
.app-container {
    display: flex;
    flex-direction: column;
    min-height: 100vh;
}

main {
    flex-grow: 1;
    padding: 24px;
}
```

src/index.js

```javascript
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { BrowserRouter } from 'react-router-dom';
import { AuthProvider } from './context/AuthContext';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <AuthProvider>
        <App />
      </AuthProvider>
    </BrowserRouter>
  </React.StrictMode>
);
```

src/App.js

```
import React from 'react';
import { Routes, Route } from 'react-router-dom';
import LandingPage from './pages/LandingPage';
import LoginPage from './pages/LoginPage';
import TraineeDashboard from './pages/TraineeDashboard';
import GraduateDashboard from './pages/GraduateDashboard';
import AdminDashboard from './pages/AdminDashboard';
import CommunityPage from './pages/CommunityPage';
import Header from './components/Header';
import ProtectedRoute from './components/ProtectedRoute';
import { useAuth } from './context/AuthContext';
import { Box, CssBaseline } from '@mui/material';
import './App.css';

function App() {
  const { user } = useAuth();

  return (
    <>
      <CssBaseline />
      <div className="app-container">
        {user && <Header />}
        <main>
          <Routes>
            <Route path="/" element={<LandingPage />} />
            <Route path="/login/:role" element={<LoginPage />} />

            <Route path="/dashboard" element={
              <ProtectedRoute>
                {user?.role === 'trainee' && <TraineeDashboard />}
                {user?.role === 'graduate' && <GraduateDashboard />}
                {user?.role === 'admin' && <AdminDashboard />}
              </ProtectedRoute>
            } />

            <Route path="/community" element={
              <ProtectedRoute>
                <CommunityPage />
              </ProtectedRoute>
            } />

            {/* In a real app, a chat page would be here: e.g., /chat/:chatId */}
          </Routes>
        </main>
      </div>
    </>
  );
}

export default App;
```

src/data/mockUsers.js

```
export const mockUsers = [
    // --- Trainees ---
    {
        id: 'trainee1',
        phone: '9876543210',
        role: 'trainee',
        name: 'Anika Sharma',
        tags: ['Beauty'],
        assignedMentorId: 'grad2',
        courseProgress: 2, // 2 out of 5 courses completed
    },
    {
        id: 'trainee2',
        phone: '9876543211',
        role: 'trainee',
        name: 'Suresh Kumar',
        tags: ['Groceries'],
        assignedMentorId: 'grad3',
        courseProgress: 4,
    },
    // --- Graduates ---
    {
        id: 'grad1',
        phone: '8765432109',
        role: 'graduate',
        name: 'Raj Patel',
        tags: ['Furniture', 'Success'],
        mentoring: [],
        followUpStatus: 'Month 34/34',
        successStory: 'I started with a small workshop and now I employ 15 people
    },
    {
        id: 'grad2',
        phone: '8765432108',
        role: 'graduate',
        name: 'Priya Singh',
        tags: ['Beauty'],
        mentoring: ['trainee1'],
        followUpStatus: 'Month 20/34',
        successStory: null,
    },
    {
        id: 'grad3',
        phone: '8765432107',
        role: 'graduate',
        name: 'Amit Verma',
        tags: ['Groceries', 'Success'],
        mentoring: ['trainee2'],
        followUpStatus: 'Month 40/34',
        successStory: 'My grocery delivery service is now profitable and has expa
    },
    // --- Admin ---
    {
        id: 'admin1',
        phone: '7654321098',
        role: 'admin',
        name: 'Anjali Mehta',
    },
];
```

src/data/mockCourses.js

```
export const mockCourses = {
    trainee: [
        { id: 't1', title: 'Module 1: Business Fundamentals' },
        { id: 't2', title: 'Module 2: Sourcing Materials & Suppliers' },
        { id: 't3', title: 'Module 3: Basic Accounting & Finance' },
        { id: 't4', title: 'Module 4: Customer Service Excellence' },
        { id: 't5', title: 'Module 5: Digital Marketing for Beginners' },
    ],
    refresher: [
        { id: 'r1', title: 'Advanced Digital Marketing & SEO' },
        { id: 'r2', title: 'GST and Tax Compliance Update' },
        { id: 'r3', title: 'Scaling Your Production & Operations' },
        { id: 'r4', title: 'Advanced Employee Management' },
    ]
};
```

src/data/mockCommunity.js

```
export const mockCommunityPosts = [
    {
        id: 1,
        authorName: 'Anika Sharma',
        authorRole: 'trainee',
        title: 'Need help with financial projections for a skincare line.',
        query: 'I have a good product idea but I am struggling to create realisti
        sectorTag: 'Beauty',
    },
    {
        id: 2,
        authorName: 'Suresh Kumar',
        authorRole: 'trainee',
        title: 'Best way to manage fresh produce inventory?',
        query: 'For my grocery business, minimizing waste is key. What are the be
        sectorTag: 'Groceries',
    },
    {
        id: 3,
        authorName: 'Raj Patel',
        authorRole: 'graduate',
        title: 'Sharing a great contact for raw wood suppliers in Gujarat.',
        query: 'For anyone in the furniture business, I have a fantastic and reli
        sectorTag: 'Furniture',
    }
];
```

src/data/mockChats.js

```
export const mockChats = [
    {
        id: 'chat1',
        name: 'Mentorship: Anika & Priya',
        members: ['trainee1', 'grad2', 'admin1'],
        messages: [
            { sender: 'Anjali Mehta (Admin)', text: 'Welcome Anika and Priya! Pri
            { sender: 'Priya Singh', text: 'Of course! Happy to help, Anika. Let
        ]
    },
    {
        id: 'chat2',
        name: 'Groceries Cohort Q&A',
        members: ['trainee2', 'grad3', 'admin1'],
        messages: [
            { sender: 'Anjali Mehta (Admin)', text: 'This is the group for our gr
        ]
    }
]
```

src/context/AuthContext.js

```
import React, { createContext, useState, useContext } from 'react';
import { useNavigate } from 'react-router-dom';
import { mockUsers } from '../data/mockUsers';

const AuthContext = createContext(null);
```

```
export const AuthProvider = ({ children }) => {
    const [user, setUser] = useState(null);
    const navigate = useNavigate();

    const login = (phone, role) => {
        // Prototype logic: Find user by phone and role. In real app, this would
        const foundUser = mockUsers.find(u => u.phone === phone && u.role === rol
        if (foundUser) {
            setUser(foundUser);
            navigate('/dashboard');
        } else {
            alert('User not found for this role. Please check the phone number an
        }
    };

    const logout = () => {
        setUser(null);
        navigate('/');
    };

    return (
        <AuthContext.Provider value={{ user, login, logout }}>
            {children}
        </AuthContext.Provider>
    );
};

export const useAuth = () => {
    return useContext(AuthContext);
};
```

src/components/ProtectedRoute.js

```
import React from 'react';
import { Navigate } from 'react-router-dom';
import { useAuth } from '../context/AuthContext';

const ProtectedRoute = ({ children }) => {
    const { user } = useAuth();

    if (!user) {
        // If no user is logged in, redirect to the landing page
        return <Navigate to="/" />;
    }

    return children;
};
```

```
export default ProtectedRoute;
```

src/components/Header.js

```
import React from 'react';
import { AppBar, Toolbar, Typography, Button, Box, Avatar } from '@mui/material';
import { useAuth } from '../context/AuthContext';
import { useNavigate } from 'react-router-dom';

const Header = () => {
    const { user, logout } = useAuth();
    const navigate = useNavigate();

    return (
        <AppBar position="static" sx={{ backgroundColor: '#004c8f' }}>
            <Toolbar>
                <Avatar src="/icecd-logo.png" sx={{ mr: 2 }} />
                <Typography variant="h6" component="div" sx={{ flexGrow: 1 }}>
                    ICECD Community Platform
                </Typography>
                <Box>
                    <Button color="inherit" onClick={() => navigate('/dashboard')
                    <Button color="inherit" onClick={() => navigate('/community')
                    <Button color="inherit" onClick={logout} variant="outlined">L
                </Box>
            </Toolbar>
        </AppBar>
    );
};

export default Header;
```

src/pages/LandingPage.js

```
import React from 'react';
import { useNavigate } from 'react-router-dom';
import { Container, Typography, Box, Card, CardContent, CardActionArea, Avatar }
import PeopleIcon from '@mui/icons-material/People';
import SchoolIcon from '@mui/icons-material/School';
import WorkspacePremiumIcon from '@mui/icons-material/WorkspacePremium';

const LandingPage = () => {
    const navigate = useNavigate();

    const roles = [
        { name: 'For Our Team', role: 'admin', icon: <PeopleIcon fontSize="large"
        { name: 'For Trainees', role: 'trainee', icon: <SchoolIcon fontSize="larg
        { name: 'For Graduates', role: 'graduate', icon: <WorkspacePremiumIcon fo
    ];

    return (
        <Container maxWidth="md">
            <Box sx={{ my: 4, textAlign: 'center' }}>
                <Avatar src="/icecd-logo.png" sx={{ width: 80, height: 80, mx: 'a
                <Typography variant="h4" component="h1" gutterBottom>
                    Welcome to the ICECD Global Community
                </Typography>
                <Typography variant="h6" color="text.secondary">
                    Connecting entrepreneurs across the globe.
                </Typography>
            </Box>
            <Box sx={{ display: 'flex', justifyContent: 'center', gap: 4, flexWra
                {roles.map((item) => (
                    <Card key={item.role} sx={{ minWidth: 275, maxWidth: 300 }}>
                        <CardActionArea onClick={() => navigate(`/login/${item.ro
                            <CardContent sx={{ textAlign: 'center', p: 3 }}>
                                <Box sx={{ mb: 2 }}>{item.icon}</Box>
                                <Typography variant="h5" component="div">
                                    {item.name}
                                </Typography>
                                <Typography sx={{ mt: 1.5 }} color="text.secondar
                                    {item.description}
                                </Typography>
                            </CardContent>
                        </CardActionArea>
                    </Card>
                ))}
            </Box>
        </Container>
    );
};

export default LandingPage;
```

src/pages/LoginPage.js

```
import React, { useState } from 'react';
import { useParams, useNavigate } from 'react-router-dom';
import { useAuth } from '../context/AuthContext';
import { Container, Box, Typography, TextField, Button, Card, CardContent } from

const LoginPage = () => {
    const { role } = useParams();
    const { login } = useAuth();
    const navigate = useNavigate();
    const [phone, setPhone] = useState('');
    const [otp, setOtp] = useState('');

    // Based on our mock data, pre-fill for easy demo
    const placeholderPhone = {
        admin: '7654321098',
        trainee: '9876543210',
        graduate: '8765432109',
    };

    const handleLogin = (e) => {
        e.preventDefault();
        // Prototype logic: OTP can be any value, just need the phone number to b
        login(phone || placeholderPhone[role], role);
    };

    return (
        <Container component="main" maxWidth="xs">
            <Box sx={{ marginTop: 8, display: 'flex', flexDirection: 'column', al
                <Card sx={{ minWidth: 400 }}>
                    <CardContent sx={{ p: 3 }}>
                        <Typography component="h1" variant="h5" align="center" gu
                            Login as a {role.charAt(0).toUpperCase() + role.slice
                        </Typography>
                        <Box component="form" onSubmit={handleLogin} noValidate s
                            <TextField
                                margin="normal"
                                required
                                fullWidth
                                id="phone"
                                label="Phone Number"
                                name="phone"
                                autoComplete="tel"
                                autoFocus
                                value={phone}
                                onChange={(e) => setPhone(e.target.value)}
                                placeholder={placeholderPhone[role]}
                            />
                            <TextField
                                margin="normal"
                                required
                                fullWidth
                                name="otp"
                                label="OTP"
                                type="password"
                                id="otp"
                                value={otp}
                                onChange={(e) => setOtp(e.target.value)}
                                placeholder="Enter any 4 digits (e.g., 1234)"
                            />
                            <Button
                                type="submit"
                                fullWidth
                                variant="contained"
                                sx={{ mt: 3, mb: 2, backgroundColor: '#004c8f' }}
                            >
                                Login
                            </Button>
                        </Box>
                    </CardContent>
                </Card>
            </Box>
        </Container>
    );
};

export default LoginPage;
```

src/pages/TraineeDashboard.js

```
import React, { useState } from 'react';
import { useAuth } from '../context/AuthContext';
import { mockCourses } from '../data/mockCourses';
import { mockChats } from '../data/mockChats';
import { Box, Typography, Card, CardContent, Checkbox, FormControlLabel, LinearPr

const TraineeDashboard = () => {
    const { user } = useAuth();
    const totalCourses = mockCourses.trainee.length;
    const [completed, setCompleted] = useState(
        Array(totalCourses).fill(false).map((_, i) => i < user.courseProgress)
    );

    const handleCheckboxChange = (index) => {
        const newCompleted = [...completed];
        newCompleted[index] = !newCompleted[index];
        setCompleted(newCompleted);
    };

    const completedCount = completed.filter(Boolean).length;
    const progressPercentage = (completedCount / totalCourses) * 100;

    const myChats = mockChats.filter(chat => chat.members.includes(user.id));

    return (
        <Box>
```

```
            <Typography variant="h4" gutterBottom>Welcome, {user.name}!</Typograp
            <Typography variant="body1" color="text.secondary" sx={{mb: 3}}>Your 

            <Paper elevation={3} sx={{ p: 2, mb: 4 }}>
                <Typography variant="h5" gutterBottom>My Learning Path</Typograph
                <Box sx={{ display: 'flex', alignItems: 'center', mb: 2 }}>
                    <Box sx={{ width: '100%', mr: 1 }}>
                        <LinearProgress variant="determinate" value={progressPerc
                    </Box>
                    <Box sx={{ minWidth: 35 }}>
                        <Typography variant="body2" color="text.secondary">{`${Ma
                    </Box>
                </Box>
                {mockCourses.trainee.map((course, index) => (
                    <Card key={course.id} sx={{ mb: 1 }}>
                        <CardContent>
                            <FormControlLabel
                                control={<Checkbox checked={completed[index]} onC
                                label={course.title}
                            />
                        </CardContent>
                    </Card>
                ))}
            </Paper>

            <Paper elevation={3} sx={{ p: 2 }}>
                <Typography variant="h5" gutterBottom>My Mentor Groups</Typograph
                <List>
                    {myChats.length > 0 ? myChats.map((chat, index) => (
                        <React.Fragment key={chat.id}>
                            <ListItem button>
                                <ListItemText primary={chat.name} secondary={`Las
                            </ListItem>
                            {index < myChats.length - 1 && <Divider />}
                        </React.Fragment>
                    )) : <Typography>You are not in any groups yet. Post a query 
                </List>
            </Paper>
        </Box>
    );
};

export default TraineeDashboard;
```

src/pages/GraduateDashboard.js

```
import React from 'react';
import { useAuth } from '../context/AuthContext';
import { mockCourses } from '../data/mockCourses';
import { mockChats } from '../data/mockChats';
import { mockUsers } from '../data/mockUsers';
import { Box, Typography, Paper, List, ListItem, ListItemText, Divider, Card, Car
import CheckCircleIcon from '@mui/icons-material/CheckCircle';

const GraduateDashboard = () => {
    const { user } = useAuth();
    const myChats = mockChats.filter(chat => chat.members.includes(user.id));
    const mentees = mockUsers.filter(u => u.role === 'trainee' && user.mentoring.
    const isSuccess = user.tags.includes('Success');

    return (
        <Box>
            <Box sx={{ display: 'flex', alignItems: 'center', gap: 2, mb: 1 }}>
                <Typography variant="h4">Welcome back, {user.name}!</Typography>
                {isSuccess && <Chip icon={<CheckCircleIcon />} label="Certified S
            </Box>
            <Typography variant="body1" color="text.secondary" sx={{mb: 3}}>Your 

            {isSuccess && (
                <Card sx={{ mb: 4, backgroundColor: '#e8f5e9' }}>
                    <CardContent>
                        <Typography variant="h6">Your Success Story</Typography>
                        <Typography variant="body2" sx={{ fontStyle: 'italic', my
                        <Button variant="contained" color="success">Edit/Update Y
                    </CardContent>
                </Card>
            )}

            <Paper elevation={3} sx={{ p: 2, mb: 4 }}>
                <Typography variant="h5" gutterBottom>Mentor Center</Typography>
                <Typography variant="subtitle1" gutterBottom>Trainees you are men
                {mentees.length > 0 ? mentees.map(mentee => (
                    <Typography key={mentee.id}>- {mentee.name} ({mentee.tags.joi
                )) : <Typography>You are not currently mentoring any trainees.</T

                <Typography variant="h6" sx={{mt: 2}} gutterBottom>Your Groups</T
                <List>
                    {myChats.map((chat, index) => (
                        <React.Fragment key={chat.id}>
                            <ListItem button>
                                <ListItemText primary={chat.name} />
                            </ListItem>
                            {index < myChats.length - 1 && <Divider />}
                        </React.Fragment>
                    ))}
                </List>
            </Paper>
            <Paper elevation={3} sx={{ p: 2 }}>
                <Typography variant="h5" gutterBottom>Skill Refreshers</Typograph
                {mockCourses.refresher.map(course => (
                    <Card key={course.id} variant="outlined" sx={{ mb: 1 }}>
                        <CardContent>
                            <Typography>{course.title}</Typography>
                        </CardContent>
                    </Card>
                ))}
            </Paper>
        </Box>
    );
};

export default GraduateDashboard;
```

src/pages/AdminDashboard.js

```
import React from 'react';
import { useAuth } from '../context/AuthContext';
import { mockUsers } from '../data/mockUsers';
import { mockChats } from '../data/mockChats';
import { Box, Typography, Paper, Table, TableBody, TableCell, TableContainer, Tab

const AdminDashboard = () => {
    const { user } = useAuth();
    const trainees = mockUsers.filter(u => u.role === 'trainee');
    const graduates = mockUsers.filter(u => u.role === 'graduate');

    const getMentorName = (mentorId) => {
        const mentor = mockUsers.find(u => u.id === mentorId);
        return mentor ? mentor.name : 'Not Assigned';
    };

    const handleCreateGroup = (traineeName) => {
        alert(`Prototype Action: Opening form to create a group chat for ${trainee
    }

    const handleAwardSuccess = (gradName) => {
        alert(`Prototype Action: The "Success" tag has been awarded to ${gradName
    }

    return (
        <Box>
            <Typography variant="h4" gutterBottom>Admin Dashboard</Typography>
            <Typography variant="h6" color="text.secondary">Welcome, {user.name}<
```

```
                {/* Trainee Management */}
                <Paper elevation={3} sx={{ p: 2, my: 4 }}>
                    <Typography variant="h5" gutterBottom>Trainee Management</Typogra
                    <TableContainer>
                        <Table>
                            <TableHead>
                                <TableRow>
                                    <TableCell>Name</TableCell>
                                    <TableCell>Sector</TableCell>
                                    <TableCell>Course Progress</TableCell>
                                    <TableCell>Assigned Mentor</TableCell>
                                    <TableCell>Actions</TableCell>
                                </TableRow>
                            </TableHead>
                            <TableBody>
                                {trainees.map(trainee => (
                                    <TableRow key={trainee.id}>
                                        <TableCell>{trainee.name}</TableCell>
                                        <TableCell>{trainee.tags.join(', ')}</TableCe
                                        <TableCell>{trainee.courseProgress} / 5</Tabl
                                        <TableCell>{getMentorName(trainee.assignedMen
                                        <TableCell>
                                            <Button variant="outlined" size="small" o
                                        </TableCell>
                                    </TableRow>
                                ))}
                            </TableBody>
                        </Table>
                    </TableContainer>
                </Paper>

                {/* Graduate Management */}
                <Paper elevation={3} sx={{ p: 2, mb: 4 }}>
                    <Typography variant="h5" gutterBottom>Graduate Management</Typogr
                    <TableContainer>
                        <Table>
                            <TableHead>
                                <TableRow>
                                    <TableCell>Name</TableCell>
                                    <TableCell>Sector</TableCell>
                                    <TableCell>Follow-up Status</TableCell>
                                    <TableCell>Success Tag</TableCell>
                                    <TableCell>Actions</TableCell>
                                </TableRow>
                            </TableHead>
                            <TableBody>
                                {graduates.map(grad => (
                                    <TableRow key={grad.id}>
                                        <TableCell>{grad.name}</TableCell>
                                        <TableCell>{grad.tags.filter(t => t !== 'Succ
                                        <TableCell>{grad.followUpStatus}</TableCell>
                                        <TableCell>
                                            {grad.tags.includes('Success')
                                                ? <Chip label="Yes" color="success" s
                                                : <Chip label="No" size="small" />}
                                        </TableCell>
                                        <TableCell>
                                            <Button variant="outlined" size="small" o
                                        </TableCell>
                                    </TableRow>
                                ))}
                            </TableBody>
                        </Table>
                    </TableContainer>
                </Paper>
            </Box>
        );
    };

    export default AdminDashboard;
```

src/pages/CommunityPage.js

```
import React, { useState } from 'react';
import { mockCommunityPosts } from '../data/mockCommunity';
import { Box, Typography, Card, CardContent, CardActions, Button, TextField, Pape

const CommunityPage = () => {
    const [posts, setPosts] = useState(mockCommunityPosts);
    const [filter, setFilter] = useState('All');
    const allSectors = ['All', ...new Set(mockCommunityPosts.map(p => p.sectorTag

    const filteredPosts = filter === 'All' ? posts : posts.filter(p => p.sectorTa

    const handlePostQuery = () => {
        // In a real app, this would submit the form data to a backend.
        // For the prototype, we just show an alert.
        alert('Your query has been posted to the community! An admin will review
    };

    return (
        <Box>
            <Typography variant="h4" gutterBottom>Open Community</Typography>

            {/* Post a New Query */}
            <Paper elevation={3} sx={{ p: 2, mb: 4 }}>
                <Typography variant="h6" gutterBottom>Have a question? Ask the co
                <TextField label="Query Title" fullWidth sx={{ mb: 2 }} />
                <TextField label="Describe your question in detail..." multiline
                <Button variant="contained" onClick={handlePostQuery}>Post Query<
            </Paper>

            {/* Community Feed */}
            <Paper elevation={3} sx={{ p: 2 }}>
                <Typography variant="h6" gutterBottom>Community Feed</Typography>
                <Box sx={{ mb: 2 }}>
                    {allSectors.map(sector => (
                        <Chip
                            key={sector}
                            label={sector}
                            onClick={() => setFilter(sector)}
                            color={filter === sector ? 'primary' : 'default'}
                            sx={{ mr: 1, mb: 1 }}
                        />
                    ))}
                </Box>
                {filteredPosts.map(post => (
                    <Card key={post.id} sx={{ mb: 2 }}>
                        <CardContent>
                            <Typography variant="h5" component="div">
                                {post.title}
                            </Typography>
                            <Typography sx={{ mb: 1.5 }} color="text.secondary">
                                by {post.authorName} ({post.authorRole}) - Sector
                            </Typography>
                            <Typography variant="body2">
                                {post.query}
                            </Typography>
                        </CardContent>
                        <CardActions>
                            <Button size="small">Reply</Button>
                            <Button size="small" onClick={() => alert('An admin h
                                Request Admin Help
                            </Button>
                        </CardActions>
                    </Card>
                ))}
            </Paper>
        </Box>
    );
};

export default CommunityPage;
```

.gitignore

This file will be created by `create-react-app`. It's important to keep it as is. It prevents you from committing the `node_modules` folder.

```
# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies
/node_modules
/.pnp
.pnp.js

# testing
/coverage

# production
/build

# misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*
```

README.md

```
# ICECD Community Platform - Hackathon Prototype

This project is a React-based prototype for the JPMC Code for Good hackathon, des

## Core Features

- **Role-Based Access:** Separate, tailored dashboards for Admins, Trainees, and
- **Learning Management:** Trainees can access courses and track their progress.
- **Mentorship Hub:** Graduates can mentor trainees, access refresher courses, an
- **Admin Control Panel:** Admins can oversee the entire ecosystem, manage users,
- **Community Forum:** An open space for users to ask questions, which can be esc
- **Progress Tracking:** A simulated system for tracking graduate success via mon

## Tech Stack

- **Frontend:** React.js
- **UI Library:** Material-UI (MUI)
- **Routing:** React Router
- **State Management:** React Context API
- **Backend/DB (Simulated):** Local mock data files to allow for rapid, offline-f

## Setup and Running the Project

Follow the setup instructions in the provided documentation to get the project ru
```

Okay, this is a good mock implementation. I tried it and everything is in place.
I ran it and it seems to have things but it is not really implementing a lot of things as you and I know.
If you relook at the flow that I had provided, The problem is that as both you and I know, a lot of things that you have provided are just mock implementations and they are just prototype actions. We have to now step it up, this is for an NGO and we would be pitching. The things right now that are prototype actions, especially the success story button not working, the group creation not working etc. NEED to be upgraded and fixed. The group chat should be like a WhatsApp group chat too, so that too to be kept in mind. Rego through the plan above and now give me UPDATED full directory structure and complete filled codes (for all the files, don't skip any) so that we can now upgrade and step up. Please, very essentially.
Also, the frontend seems to be looking pretty vague right now, we need to step up that too with nice progress bars, boxes and others that look sleek and modern.

Currently we are in JPMC CFG hackathon, we have chosen the problem statement:
"Organization: ICECD
Website: https://icecd.org/
Mission
ICECD's mission is to create an "Entrepreneurial Ecosystem", through increasing access and control over productive resources like study materials for the needy to facilitate their economic, social and political empowerment through Human Resource Development by upskilling and training their beneficiaries in vital life and career skills, keeping gender issues as a key focus.
Context
ICECD is currently active in 55 countries with over 38+ years of experience, having impacted 10 million people including slum, rural and urban populations. Having WhatsApp groups for communications and sharing entrepreneurship-based resources on a common platform. Currently, the organization needs a suitable alternative for WhatsApp, which is their main communication channel. The key methodology that enhances ICECD's success rate is following up and tracking the progress of their beneficiaries after their training is completed to ensure the business lasts beyond its critical time ceilings. The organization measures the success of their ventures by tracking the number of people reaching out to their organization, people's stories/testimonies, and the profit made by the organization.
Strategy:
The organization is in an intermediate stage of its tech roadmap, having identified key areas for improvement in networking, two-way community-based communication, and entrepreneurship curriculum refreshers via a common platform for all. The next stage involves tracking growth for the recipients of their training program beyond their graduation and developing a digital solution that can bring entrepreneurs separated by distance but united by profession together to build a global community of shared interests and mutual struggles. The solution should also have the capacity to allow users to engage creatively with the curriculum and content covered in the training program as a refresher or continuous learning journey.
Stakeholder Universe:
Stakeholders include the nonprofit team, trainees, and graduates of the entrepreneur program. The nonprofit team should be able to facilitate and monitor the interactions of the platform while also tracking

and managing existing entrepreneurs within their ecosystem. They should also have the ability to onboard, collate, and prescribe the ecosystem to budding entrepreneurs who will be recipients of their training. The trainees must be able to leverage the existing ecosystem to advance their learning journey and effectively track their engagement and performance. Graduate entrepreneurs must be able to interact with their peers, collaborate across borders effectively, and upskill themselves with the latest developments in the curriculum.
Challenge
Develop a comprehensive digital solution that addresses the ecosystem tracking challenges faced by ICECD organization. The solution should creatively engage the ecosystem through on-platform elements while also allowing for easy access to graduate entrepreneurs within the ecosystem to help their peers and the new recruits in their journey. This solution should help the organization effectively collate, visualize, and derive insights and bring the beneficiaries of their programs under one roof.
Technology Baseline
ICECD are currently using WhatsApp for their communication needs."

Our team discussed over this and we brainstormed to come up with an idea.
The idea includes a flow that I will be putting in:
A page where there would be 3 sections, which correspond to:

1. ICECD Team (Admin)
2. Trainees
3. Graduates (Entrepreneurs)
   Upon clicking on any of these sections, it should:
   Open up a login page, there should be a phone number and OTP option but for now, it should be able to accept any number in OTP as it is a prototype.

Upon login being done, for Trainees:

1. Show some courses, videos and resources to the trainee (for now, keep placeholders blocks) that they can choose from through tabs/sections and it should be able to track them upon their completion which they can provide through checkboxes/etc.
2. Have an open community button which upon clicking should take them to the open community page (which I will provide details later)
3. Show the group chats (which I will provide details later) that they are in, which are added by the Admin.

Upon login being done, for Graduates (Entrepreneurs):

1. It should show them refreshers for the courses, that they can choose out of, in case they want to and again, similar to (1) from Trainees, keep placeholders for the videos/resources etc. that are involved and ensure to have a progress check similar to that.
2. Have the same community button which upon clicking should take them to the open community page (which I will provide details later)
3. Have access to the group chats that they are added to, which would basically mean the trainee that they are assigned to.

Upon login being done, for ICECD Team (Admin):

1. It should show all the trainees and their assigned mentors and their progress.

2. It should also show the Graduates (Entrepreneurs) and their success stories (if any, I will provide details about what a success story is, later) and what is the follow up status (I will give information about this, later)
3. Have the same community button which upon clicking should take them to the open community page (which I will provide details later)
4. Also have access to all of the group chats and create the group chats and add the corresponding trainee and graduates.

Now, let us move into the tags involved.

1. Trainee should have a tag, talking about the business sector that they're part of (for now, choose 4-5 for demo purposes like beauty, groceries, furniture etc.) and this is pre-assigned.
2. Graduates can have the business sector tag and also, a success tag which I will talk about later.
   The tags here play a vital role in being able to filter out people by tag-based searching on a specific sector that they are in.

Now let's move on to the 'discuss later' that I mentioned:

1. Open community: This is a page where the trainees can post their queries and then the admin can form these "group chats".
2. Group chats: These include the admin and the corresponding members added by the admin.
3. Success stories and success tag: There should be a monthly-based SMS that is sent to the graduates pursuing a business, which would include a form that has questions based on: "Indicators include investment and loans secured, number of jobs generated, profitability, startup sustainability, and impact on someone’s life. Intangible outcomes include improved quality of life, increased societal status, better healthcare, and overcoming social taboos." and then once the progress looks good according to the Admin and they receive a positive response for 1000 days then that graduate is qualified to give their success story which would mean that they would have a final SMS that sends them the link to the Graduate portal with the "success" button now in their page, where they can write their stories etc. Additionally, they should also be given a "success" tag now for better monitoring.

Now, here is some information about the linking part:

1. The trainee will basically have the phone number, password and tags in the database (or whatever) and the assigned mentor stored.
2. The graduate will be having the phone number, password and tags stored too but also the trainee that they are assigned to.
3. The admin will be having the access to everything that was mentioned from the points under admin at the top.

Tech stacks:
Use React for frontend and then use anything for backend that is suitable and right and would get it working fast with no errors for the hackathon. We really need to win this, please don't mess anything up. The UI/UX caters to rural, urban and slum area part of
the population, so keep that in your mind but at the same time it should be appealing and make the NGO like it.

My flow in the head:
The person opens the page, chooses the corresponding section and then based on that they proceed with the features linked as already mentioned but the most important part is the open community being a place for query pitch in, the group chat being a place for commonality (with respect to tags) and additionally the success story and post-process linking to be done as the problem statement mentions already.

Now, before you go ahead with the code generation or any of that, please give me your current process flow, from a user point of view so that I can verify if it matches with my vision too and if no, I could give feedback and if yes, I will give you a headsup about code generation.