

SILIGURI INSTITUTE OF TECHNOLOGY

PROJ- CS881 FUNDAMENTALEUREKA

BY

IT_PROJ_2025_06

Name of Students	Roll No.
BIKASH SAH	11900222040
SOU MODEEP ROY	11900222027
PRAHLAD BISWAS	11900222029
BAIDURJYA SIKDER	11900222026

Under the Guidance

of

Dr. Sirshendu Sekhar Ghosh

Submitted to the Department of **Information Technology** in partial fulfillment of the requirements
for the award of the degree Bachelor of Technology in **Information Technology**



Siliguri Institute of Technology

P.O. SUKNA, SILIGURI, DIST. DARJEELING, PIN: 734009

Tel: (0353)2778002/04, Fax: (0353) 2778003

Year of Submission: 2025

DECLARATION

This is to certify that Report entitled “**FundaMentalEureka**” which is submitted by me in partial fulfillment of the requirement for the award of degree B.Tech. in Information Technology at Siliguri Institute of Technology under Maulana Abul Kalam Azad University of Technology, West Bengal. We took the help of other materials in our dissertation which have been properly acknowledged. This report has not been submitted to any other Institute for the award of any other degree.

Date:

SN	Name of the Student	Roll No	Signature
1	Bikash Sah	11900222040	
2	Soumodeep Roy	11900222027	
3	Prahlad Biswas	11900222029	
4	Baidurjya Sikder	11900222026	

CERTIFICATE

This is to certify that the project report entitled “FundamentalEureka” submitted to **Department of Information Technology of Siliguri Institute of Technology** in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Information Technology** during the academic year **2024-25**, is a bona fide record of the project work carried out by them under my guidance and supervision.

Project Group Number:

SN	Name of Students	Registration No	Roll No.
1	BIKASH SAH	221190120354	11900222040
2	SOUМОDEEP ROY	221190120367	11900222027
3	PRAHLAD BISWAS	221190120357	11900222029
4	BAIDURJYA SIKDER	221190120353	11900222026

Signature of Project Guide

Name of the Guide:

Dr. Sirshendu Sekhar Ghosh

Signature of the HOD

Department of Information Technology

Acknowledgement

We would like to thanks, our professor-in-charge and our principal, for their support and guidance in completing our project on the topic (**Fundamental Eureka**). Also, our sincere thanks to all the faculty and staff members of the Department of Information Technology for providing extensive help, useful information and materials. It was a great learning experience and we extend our sincere gratitude to our project coordinator **Dr. Sirshendu Sekhar Ghosh** for his consistent support, encouragement, and valuable input throughout the project.

Signature of all the group members with date

- 1.
- 2.
- 3.
- 4.

Table of Contents	Page No
Introduction	7
1 System Analysis	8-16
1.1 Identification of Need	
1.2 Preliminary Investigation	
1.3 Project Planning	
1.4 Software requirement specifications (SRS)	
1.5 Software Engineering Paradigm applied	
1.6 Flow diagrams	
2 System Design	17-21
2.1 Modularization details	
2.2 Data integrity and constraints	
2.3 Database design	
2.4 User Interface Design	
3 Coding	22-31
3.1 Include important modules' codes only.	
3.2 Error handling	
3.3 Parameters calling/passing	
3.4 Validation checks	
4 Testing	32-34
4.1 Testing techniques and testing strategies used along with the test case designs and test reports.	
4.2 Debugging and Code improvement	
5 System security measures	35-36
5.1 Database security	
5.2 Creation of User profiles and access rights	
6 Cost Estimation of the Project	37-38
6.1 Reports	
6.2 Gantt chart	
7 Conclusion & Future Scope	39-39
8 References and Appendices	40-43

Abstract

This project is designed to address the increasing demand for efficient and reliable solutions within the domain of user skill development and career readiness. Its primary objective is to build a system that not only tackles real-world challenges but also showcases practical application through the integration of modern tools and technologies. By focusing on enhancing user competencies and preparing them for interviews, the project aims to contribute meaningfully to individual career growth and the evolving needs of the skill-based industry.

Following this, the design phase involved selecting appropriate methodologies and tools to build the proposed system. The insert method used Web development lifecycle was implemented to structure the workflow. Data collection, system modeling, and performance testing were done to evaluate the effectiveness of the developed solution.

The tools and technologies used include [HTML/CSS, MONGO DB, AI etc.] Which played a crucial role in system development and integration? Regular testing and validation ensured the reliability and accuracy of the final output.

In conclusion, the project successfully achieved its intended objectives by delivering a functional and effective solution. The outcomes indicate potential for further development and scalability, providing a foundation for future research or practical deployment in real-world scenarios. The project holds promising scope in addressing similar problems across varied domains.

Introduction

“FundaMentalEureka” is an innovative web-based educational platform designed to meet the evolving needs of students and learners across all academic streams in today’s competitive, skill-driven industry. The platform offers a comprehensive and interactive learning environment that supports both academic development and career readiness. Its core focus is to help users strengthen their foundational knowledge, prepare effectively for job interviews, and stay updated with current career trends. By integrating structured academic content with real-world applications, AI-powered responses, and user-driven contributions, “FundaMentalEureka” encourages active learning and collaborative engagement. The platform also offers AI-assisted resume creation tools that help users craft professional resumes, which can be securely stored in the cloud for easy access and future updates. Additionally, it provides essential tools for interview preparation and skill assessments, making it a reliable companion for continuous personal and professional growth. Built using modern web technologies, the platform ensures accessibility, responsiveness, and an engaging user experience tailored to the dynamic needs of learners and job seekers alike.

1 System Analysis

System Analysis is the process of studying and understanding a system to identify its components, functions, and interactions. It involves examining current systems, gathering requirements, and defining system goals to propose effective solutions. The main objective is to improve efficiency and functionality by analyzing data flow, processes, and user needs. System analysis is essential in software development, as it helps in planning, designing, and implementing reliable and efficient systems. It also ensures that the final product meets both technical and business requirements, laying the foundation for successful system design and development.

1.1 Identification of Need

The development of “**FundaMentalEureka**” stems from the growing demand for a unified platform that supports academic learning, career preparation, and professional development. Many students and job seekers face difficulties in accessing reliable resources that offer both conceptual understanding and practical tools such as resume building and interview readiness. Additionally, the lack of intelligent systems that provide personalized guidance and support further limits their ability to grow in a competitive environment.

To address these challenges, “**FundaMentalEureka**” combines structured educational content with AI-powered assistance, interactive learning modules, and user-driven discussions. One of its standout features is the AI-based resume builder, which allows users to create polished, industry-ready resumes and store them securely in the cloud. This integrated approach fulfills the pressing need for a flexible, accessible, and intelligent platform that helps users enhance their skills, track their growth, and succeed in the evolving job market.

Key Needs Addressed:

- A centralized platform for both learning and career preparation.
- Personalized, AI-driven tools for resume creation and interview readiness.
- Cloud-based storage for easy access and future updates of user-created resumes.

1.2 Preliminary Investigation

Before initiating the development of “**FundaMentalEureka**” a preliminary investigation was conducted to understand the existing gaps in current educational and career development platforms. The investigation involved market research, competitor analysis, and direct feedback from students, educators, and job seekers to evaluate the need for an integrated system that addresses both academic and career readiness requirements.

The research revealed that while numerous platforms offer online learning or job preparation individually, very few combine both aspects into a single, user-friendly solution. Students often resort to using multiple fragmented tools for learning fundamentals, preparing for interviews, and building resumes. This not only leads to inefficiency but also creates confusion and inconsistency in the learning and preparation process.

In addition, the lack of interactive features, personalized guidance, and real-time feedback in most existing systems further highlights the need for an intelligent, comprehensive platform. Many users expressed the need for a system that could adapt to their learning pace, help them prepare for interviews, and generate resumes quickly and professionally using AI.

Based on these findings, the development of “**FundaMentalEureka**” was proposed as a solution to bridge this gap. The goal was to design a unified, web-based platform that integrates fundamental subject learning, interview preparation, AI-based resume creation, and cloud storage—all within an intuitive and accessible user interface.

This preliminary investigation confirmed the feasibility, relevance, and high demand for a solution like “**FundaMentalEureka**” providing a strong foundation for further system analysis, design, and implementation.

1.3 Project Planning

Project planning for “**FundaMentalEureka**” involved a structured and strategic approach to ensure that the system meets its functional objectives, is delivered on time, and remains scalable for future enhancements. The planning phase focused on defining project goals, identifying key deliverables, allocating resources, and establishing a clear development timeline.

➤ **Objectives**

- To develop a unified web platform for fundamental learning, skill development, and interview preparation.
- To integrate AI-based tools for personalized learning support and resume creation.
- To ensure scalability, accessibility, and user engagement across all devices.

➤ **Project Phases**

1. Requirement Gathering

Collected input from target users including students, educators, and job seekers through surveys and interviews to define key features and functionalities.

2. Feasibility Study

assessed the technical, operational, and economic feasibility of the platform using cost-benefit analysis and risk assessments.

3. System Design

Designed both front-end and back-end architectures. Created wireframes, UI/UX mockups, and a database schema suitable for content management and user interaction.

4. Technology Stack Selection

Chose modern web technologies such as HTML, CSS, JavaScript (React.js), Node.js, and Mongo DB for efficient development and deployment. AI functionalities were planned using NLP models and cloud integration.

5. Development Phase

adopted an iterative development model with modular coding, regular testing, and feature-wise implementation to ensure quality and flexibility.

6. Testing and Debugging

Performed unit testing, integration testing, and user acceptance testing to identify and fix issues early in the development cycle.

7. Deployment and Cloud Integration

Deployed the application on a secure web server with resume storage capabilities in cloud services for seamless user access.

8. Documentation and Training

Prepared user manuals, developer documentation, and training material to support platform usage and maintenance.

9. Maintenance and Feedback

Established a feedback loop for continuous improvement based on user suggestions, analytics, and future updates.

1.4 Software requirement specifications (SRS)

➤ **Next Js**

Next.js is a React framework for building full-stack web applications. You use React Components to build user interfaces, and Next.js for additional features and optimizations.

It also automatically configures lower-level tools like bundlers and compilers. You can instead focus on building your product and shipping quickly.

➤ **Tailwind CSS**

Tailwind CSS is an open-source CSS framework. Unlike other frameworks, like Bootstrap, it does not provide a series of predefined classes for elements such as buttons or tables. Instead, it creates a list of "utility" CSS classes that can be used to style each element by mixing and matching.

➤ **Shadcn**

A set of beautifully-designed, accessible components and a code distribution platform. Works with your favorite frameworks. Open Source. Open Code.

➤ **Authjs**

Auth.js is a popular open-source authentication library used for JavaScript-based applications. It provides a flexible and secure way to handle user authentication.

➤ **Mongo DB**

MongoDB is an open-source NoSQL database management program that uses JSON-like documents with optional schemas.

It is classified as a NoSQL database product and is designed to manage document-oriented information, store or retrieve information.

Mongo DB is a source-available, cross-platform, document-oriented database program.

➤ **Vercel**

Vercel is a cloud platform for deploying and managing web applications, designed for frontend frameworks and static sites. It offers a simple workflow that improves performance, scalability, and developer experience.

➤ **GitHub**

GitHub is a proprietary developer platform that allows developers to create, store, manage, and share their code. It uses Git to provide distributed version control bug tracking, software feature requests, task management, continuous integration

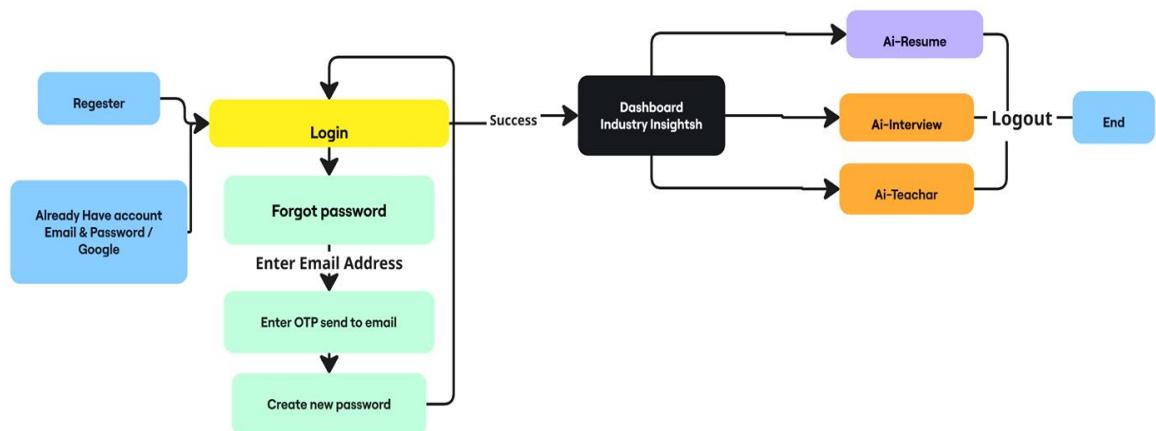
➤ **Gemini**

The Gemini API lets you access the latest generative models from Google. This API reference provides detailed information for the classes and methods available in the Gemini API SDKs.

1.5 Software Engineering Paradigm applied

The development of **FundaMentalEureka** follows the Iterative and Incremental Development Paradigm, a modern approach that emphasizes continuous improvement and feedback. This paradigm is highly suitable for dynamic web applications like FundaMentalEureka, where user needs and feature requirements can evolve during development. In the iterative model, the system is developed through repeated cycles (iterations), where each cycle adds new features or refines existing ones. This allows for progressive enhancement of functionality such as interactive learning modules, AI-based interview preparation, and resumes building tools. Early prototypes of modules are presented to stakeholders, and feedback is incorporated into subsequent iterations. The incremental approach ensures that the platform is built and delivered in parts or increments. For instance, the initial increment may deliver the user registration and content browsing features, followed by interview preparation tools, and later the resume builder and cloud integration. This modular rollout not only helps in better testing and debugging but also allows the team to prioritize the most critical features first. By combining iterative feedback with incremental growth, the project ensures high adaptability, efficient development, and timely delivery, while maintaining quality and alignment with user expectations. This paradigm also supports continuous integration and deployment, vital for modern web platforms.

1.6 Flow diagrams



Visualizing the core logic behind FundaMentalEureka's Project workflow

2 System Design

The system design of **FundaMentalEureka** follows a three-tier architecture consisting of the presentation layer, application layer, and data layer. The front-end is built with modern web technologies to ensure a responsive and user-friendly interface. The back-end handles business logic, user management, AI integration, and secure API interactions. The database layer stores user data, content, and resumes, while cloud services manage file storage. Key modules include user authentication, interactive learning, AI-powered interview preparation, and resume generation. The design is modular, scalable, and secure, ensuring efficient performance, easy maintenance, and seamless integration of future features for learners and job seekers.

2.1 Modularization details

FundaMentalEureka is designed using a modular architecture that divides the system into independent, functional units or modules. This approach improves code maintainability, scalability, and simplifies debugging and future enhancements. Each module focuses on a specific task and interacts with others through defined interfaces.

1. **User Management Module:** Handles user registration, login, authentication, and profile management. It supports role-based access (e.g., admin and user).
2. **Learning Module:** Delivers structured academic content categorized by subjects. It enables users to browse, read, and contribute to various engineering topics, promoting interactive and collaborative learning.
3. **Interview Preparation Module:** Uses AI to simulate interview scenarios. It allows users to practice technical and behavioral questions based on selected domains, improving their job readiness.
4. **Resume Builder Module:** Helps users input their details and generate professional resumes using AI. Completed resumes are stored securely in the cloud and available for download.

2.2 Data integrity and constraints

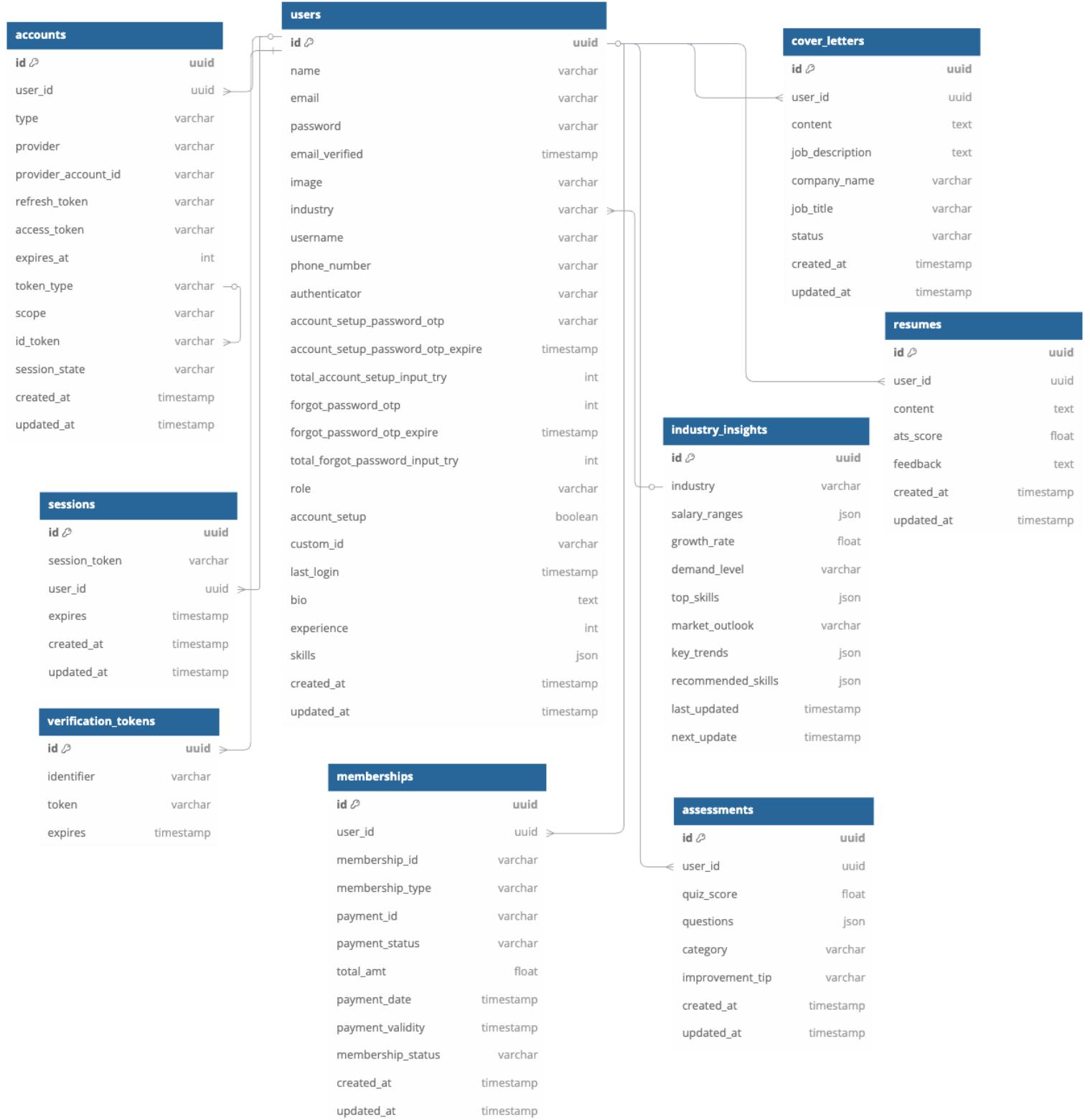
FundaMentalEureka ensures data integrity by maintaining accuracy, consistency, and reliability of information throughout the system. Data validation is implemented at both front-end and back-end levels to prevent incorrect or malicious input. All user inputs, such as login credentials, resume details and content submissions, are sanitized and verified before being stored in the database. Redundant data entries are avoided using unique constraints (like unique email IDs), and transactions are handled atomically to ensure consistency during concurrent operations. Regular backups and version control ensure that no critical information is lost or corrupted.

Data Constraints in FundaMentalEureka

To maintain data integrity, the following constraints are applied:

- **Primary Key Constraint:** Ensures each record (e.g., user ID, resume ID) is uniquely identifiable.
- **Foreign Key Constraint:** Maintains referential integrity between tables (e.g., linking resumes to users).
- **Not Null Constraint:** Mandatory fields like email, password, and resume sections must be filled.
- **Unique Constraint:** Prevents duplicate entries for unique fields like email.
- **Check Constraint:** Validates data ranges, such as password length or age limits.

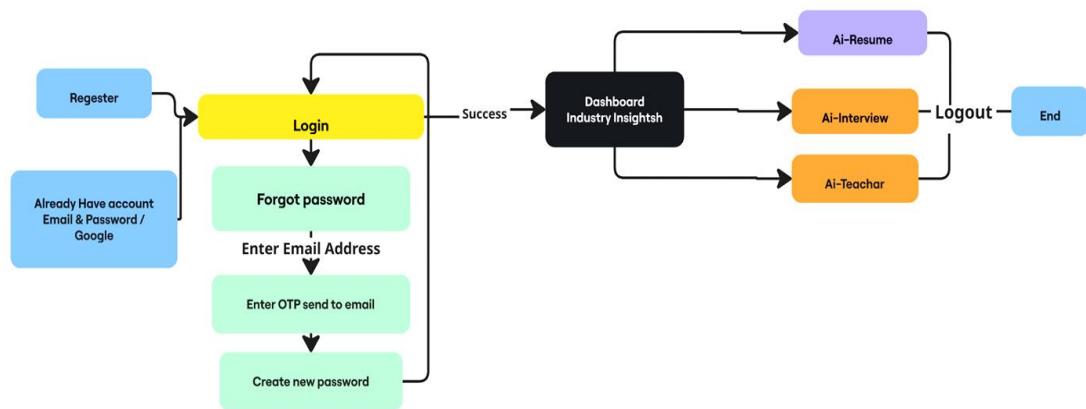
2.3 Database design



Visualizing the database design of FundaMentalEureka

2.4 User

Interface Design



Visualizing the user interface design of

3 Coding

Next.js

- A powerful React-based framework used for building the front-end.
- Enables server-side rendering (SSR) and static site generation (SSG) for better performance and SEO.
- Supports modular, component-based architecture for scalable UI development.

TypeScript

- A superset of JavaScript that adds static typing to the codebase.
- Improves code quality, reduces runtime errors, and enhances developer productivity.
- Helps maintain clean, organized, and scalable code across modules.

Tailwind CSS

- A utility-first CSS framework used for responsive and modern UI design.
- Enables rapid development of custom user interfaces with pre-defined classes.
- Reduces the need for custom CSS and supports mobile-first design.

REST API

- Facilitates communication between the front-end and back-end.
- Used to handle user data, content management, resume creation, and more.
- Ensures secure, scalable, and modular interaction between client and server.

3.1 Include important modules' codes only

```
import Image from "next/image";
import { faqs, features, howItWorks, testimonial } from "@/data/data";
import {
Accordion,
AccordionContent,
AccordionItem,
AccordionTrigger,
} from "@/components/ui/accordion";
import Link from "next/link";
import { Button } from "@/components/ui/button";
import { ArrowRight } from "lucide-react";
import { Card,CardContent } from "@/components/ui/card";
import HeroSection from "@/components/HeroSection";
import { getUserAccountSetup } from "@/actions/user";
import { redirect } from "next/navigation";
import { auth } from "@/auth";
export default async function Home() {
const session = await auth();
if (session) {
const { isAccountSetup } = await getUserAccountSetup();
if (!isAccountSetup) {
redirect("/account-setup");} }
return (
<div>
<div className="grid-background"></div>
<HeroSection />
<section className="w-full py-12 md:py-24 lg:py-32 bg-transparent">
<div className="container mx-auto px-4 md:px-6">
<h2 className="text-3xl font-bold tracking-tighter text-center mb-12">
Powerful Features for your Career Growth
</h2>
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 max-w-6xl mx-auto">
{features.map((feature, i) => {
return (
<Card
key={i}
className="border-2 hover:border-cyan-200/50 transition-colors duration-300"
>
<CardContent className="pt-6 text-center flex flex-col items-center">
<div className="flex flex-col items-center justify-center ">
<span className="text-3xl">{feature.icon}</span>
<h3 className="text-2xl font-bold my-3">
```

```
{feature.title}  
</h3>  
<p className="text-muted-foreground">  
{feature.description}  
</p>  
</div>  
</CardContent>  
</Card>  
);  
})}  
</div>  
</div>  
</section>  
<section className="w-full py-12 md:py-24 lg:py-32 bg-muted/50">  
<div className="container mx-auto px-4 md:px-6">  
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 max-w-6xl mx-auto">  
<div className="flex flex-col items-center justify-center space-y-2">  
<h3 className="text-4xl font-bold">50+</h3>  
<p className="text-muted-foreground">Subject Covered</p>  
</div>  
<div className="flex flex-col items-center justify-center space-y-2">  
<h3 className="text-4xl font-bold">1000+</h3>  
<p className="text-muted-foreground">Interview Questions</p>  
</div>  
<div className="flex flex-col items-center justify-center space-y-2">  
<h3 className="text-4xl font-bold">95%</h3>  
<p className="text-muted-foreground">Success Rate</p>  
</div>  
<div className="flex flex-col items-center justify-center space-y-2">  
<h3 className="text-4xl font-bold">24/7</h3>  
<p className="text-muted-foreground">Support</p>  
</div>  
</div>  
</div>  
</section>
```

3.2 Error handling

```
"use server";

import { auth } from "@/auth";
import { prisma } from "@/lib/prisma";
import { generateAiInsights } from "./dashboard";
import sendEmail from "@/hooks/sendEmail";
import accountSetupTemplate from "@/components/ui/accountSetupTemplate";
import { compare, hash } from "bcryptjs";
interface UpdateResult{ updatedUser?: any; industryInsight?: any; }
interface Updatinguser{ industry: string; subIndustry?: string; bio?: string; experience?: number; skills?: Array; }
/***
Updates user profile with industry-related info and fetches or generates industry insights.
@param data - User's updated data (industry, experience, bio, skills).
@returns Updated user object.
@throws Error if user not found or update fails.
*/
export async function updateUser(data: Updatinguser): Promise { const session = await auth(); const email = session?.user?.email as string;
const user = await prisma.user.findFirst({ where: { email }, include: { IndustryInsight: true } });
if (!user) throw new Error("User not found");
try { const result: UpdateResult = await prisma.$transaction(async (tx) => { let
industryInsight = await tx.industryInsight.findFirst({ where: { industry: data.industry }, });
if (!industryInsight) {
const insights = await generateAiInsights(data?.industry);
industryInsight = await prisma.industryInsight.create({
data: {
industry: data?.industry,
...insights,
}
nextUpdate: new Date(Date.now() + 7 * 24 * 60 * 60 * 1000), // 7 days later
}, });
const updatedUser = await tx.user.update({
where: { email },
data: {
```

```

        industry: data.industry,
        experience: data.experience,
        bio: data.bio,
        skills: data.skills,},});
    return { updatedUser, industryInsight };
}, { timeout: 1000 });
return user;
} catch { throw new Error("Something went wrong"); } }
/**

Checks if the current user has completed onboarding (i.e., has selected an industry).
@returns Object with onboarding status.
@throws Error if user not found or DB call fails.

*/ export async function getUserOnBoardingStatus() { const session = await auth(); const email = session?.user?.email as string;
const user = await prisma.user.findFirst({ where: { email } }); if (!user) throw new Error("User not found");
try { const user = await prisma.user.findFirst({ where: { email }, select: { industry: true } },);
return { isOnBoarded: !!user?.industry || false };
} catch { throw new Error("Error fetching user onboarding status"); } }
/**

Checks if the user's account setup process is complete.
@returns Object with account setup status.
@throws Error if user not found or DB call fails.

export async function getUserAccountSetup() { const session = await auth(); const email = session?.user?.email as string;
const user = await prisma.user.findFirst({ where: { email } }); if (!user) throw new Error("User not found");
try { const user = await prisma.user.findFirst({ where: { email }, select: { accountSetup: true } },);
return { isAccountSetup: !!user?.accountSetup || false };
} catch { throw new Error("Error fetching user Account setup"); } }
/**

Generates a 6-digit OTP, stores it hashed, sets expiration, and emails the user.

```

```
@returns void
@throws Error if user not found or OTP process fails.

*/ export async function generateOtp() { const session = await auth(); const email =
session?.user?.email as string;

const user = await prisma.user.findFirst({ where: { email } }); if (!user) throw new
Error("User not found");

try { constotp = Math.floor(Math.random() * 900000) + 100000; const hwOtp = await
hash(otp.toString(), 10); const expireTime = new Date().getTime() + 60 * 60 * 1000; // 1
hour const totalForgotPasswordInputTry = 3;
const update = await prisma.user.update({
    where: { id: user.id },
    data: {
        accountSetupPasswordOtp: hwOtp,
        accountSetupPasswordOtpExpire: new Date(expireTime).toISOString(),
        totalaccountSetupInputTry: totalForgotPasswordInputTry,
    }
});
if (update) {
    await sendEmail({
        sendTo: email,
        subject: "Account Setup OTP",
        html: accountSetupTemplate({ name: "User", otp }),
    });
}
catch { throw new Error("Error generating OTP"); } }

/**
```

3.3 Parameters calling

```
import { getIndustryInsights } from "@/actions/dashboard";
import {
  getUserAccountSetup,
  getUserOnBoardingStatus,
  getUserRole,
} from "@/actions/user";
import { redirect } from "next/navigation";
import React from "react";
import DashboardView from "./_components/DashboardView";

type SalaryRange = {
  role: string;
  min: number;
  max: number;
  median: number;
};

type Insights = {
  id: string;
  salaryRanges: SalaryRange[];
  marketOutlook: string | null;
  lastUpdated: Date;
  nextUpdate: Date | null;
  industry: string;
  growthRate: number | null;
  demandLevel: string | null;
  topSkills: string[];
  keyTrends: string[];
  recommendedSkills: string[];
};

async function Page() {
  const { isOnBoarded } = await getUserOnBoardingStatus();
  const { isAccountSetup } = await getUserAccountSetup();
  const role = await getUserRole();
  if (!isAccountSetup) {
    redirect("/account-setup");
  }
  if (!isOnBoarded) {
    redirect("/complete-profile");
  }
  if (role === "ADMIN") {
    redirect("/admin/dashboard");
  }
  const insights = await getIndustryInsights();
  return (
    <div className="container mx-auto">
      <DashboardView insights={insights as Insights} />
    </div>);
}

export default Page;
```

3.4 Validation checks

```
import NextAuth, {  
  type User as AuthUser,  
  NextAuthConfig,  
  AuthError,  
} from "next-auth";  
  
import CredentialsProvider from "next-auth/providers/credentials";  
import GoogleProvider from "next-auth/providers/google";  
import { compare } from "bcryptjs";  
import { prisma } from "./lib/prisma";  
import { PrismaAdapter } from "@auth/prisma-adapter";  
  
const adapter = PrismaAdapter(prisma);  
const authOptions: NextAuthConfig = {  
  adapter,  
  providers: [  
    GoogleProvider({  
      clientId: process.env.AUTH_GOOGLE_ID || "",  
      clientSecret: process.env.AUTH_GOOGLE_SECRET || "",}),  
    CredentialsProvider({  
      name: "Credentials",  
      credentials: {  
        email: { label: "email", type: "text" },  
        password: { label: "Password", type: "password" },  
      },  
      async authorize(credentials): Promise<AuthUser | null> {  
        try {  
          const email = credentials?.email as string | undefined;  
          const password = credentials?.password as string | undefined;  
          if (!email || !password) {  
            throw new Error("Please enter a valid email and password.");}  
          const user = await prisma.user.findUnique({ where: { email } });  
          if (!user) throw new Error("Invalid user credentials.");  
          if (!user.password) throw new Error("Please set a password.");  
        } catch (error) {  
          console.error(error);  
        }  
      },  
    }),  
  ],  
};  
  
export default NextAuth(authOptions);
```

```

const isPasswordMatch = await compare(password,user.password as string);
if (!isPasswordMatch) throw new Error("Invalid email or password.");
return { name: user.name, email: user.email };
} catch (error) {
throw new Error("Unable to proceed, Please try after some time.");
}},]),

callbacks: {

signIn: async ({ user, account }) => {
if (!user || !user.email) return false;

const existingUser = await prisma.user.findUnique({
where: { email: user.email },
});

if (existingUser) {
await prisma.user.update({
where: {email: user.email},
data: {
lastLogin: new Date(),}})
return true;
} else {
if (account?.provider === "google") {
await prisma.user.create({
data: {
name: user.name,
email: user.email,
image: user.image,
Authenticator: "google",
accounts: {
create: {
provider: "google",
providerAccountId: account.providerAccountId,
type: account.type,
access_token: account.access_token,
id_token: account.id_token,},},},});
```

```
return true;}}
```

```
return false;},
```

```
session: async ({ session, token }) => {
```

```
return {
```

```
...session,
```

```
user: {
```

```
...session.user,
```

```
id: token.sub, // Ensure user ID is included in the session},};},
```

```
jwt: async ({ user, token }) => {
```

```
if (user) {
```

```
token.sub = user.id;}
```

```
return token;},},
```

```
session: {
```

```
strategy: "jwt",
```

```
maxAge: 30 * 24 * 60 * 60, // 30 days
```

```
updateAge: 24 * 60 * 60, // 24 hours
```

```
},
```

```
pages: {
```

```
signIn: "/auth/login",
```

```
newUser: "/auth/signup",},
```

```
secret: process.env.AUTH_SECRET,
```

```
} satisfies NextAuthConfig;
```



```
export const { handlers, auth, signIn, signOut } = NextAuth(authOptions);
```

4 Testing

4.1 Testing techniques and testing strategies used along with the test case designs and test reports

Testing techniques and testing strategies

➤ Unit Testing

- Tests individual components such as user login, content rendering, and resume form handling.
- Ensures each function/module performs as expected in isolation.

➤ Integration Testing

- Verifies the interaction between modules like front-end forms and back-end APIs.
- Ensures data flows correctly between modules (e.g., resume builder to cloud storage).

➤ System Testing

- Checks the complete functionality of the application as a whole.
- Ensures the system meets all specified requirements.

➤ Security Testing

- Verifies protection of user data (e.g., login credentials, resumes).
- Checks for vulnerabilities such as SQL injection and XSS attacks.

➤ Testing strategies

• Black Box Testing

- Focuses on input/output without knowing internal code structure.

• White Box Testing

- Tests internal logic, code paths, and loops for bugs.

• Regression Testing

- Ensures new features or updates don't break existing functionality.

Test case designs and test reports

Test Case ID	Description	Input Data	Excepted Result	Actual Result	Status
1	Enter Valid Email Id	Email ID	Success & Redirect to password Input Page	As expected	Pass
2	Enter Valid Email & Password	Email and Password	Success & Redirect to Industry Insight	As expected	Pass
3	If user is new after successful login user verify OTP and then complete profile with (Industry Years of Experience, Skills, Professional Bio)	OTP Industry Years of Experience Skills Professional Bio	Success & Redirect to Industry Insight	As expected	Pass
4	Resume Builder	Enter all Details	Success	As expected	Pass
5	Ask Ai Question	Ask Question	Success	As expected	Pass
6	Ai Interview & Store the result in database	Auto Generated MCQ based Questions	Success	As expected	Pass
7	Unauthorized Access	User try to go dashboard without login	Access denied & redirected to login page	As expected	Pass

4.2 Debugging and Code improvement

➤ Debugging Techniques Used

- **Console Logging and Browser Dev Tools-** Used extensively during front-end development to track UI issues, state changes, and component rendering problems.
- **Breakpoints and Step Execution-** Applied in IDEs like VS Code to step through back-end logic (API calls, authentication, data handling) and locate logical errors.
- **Network Monitoring-** Inspecting API responses and payloads using Chrome Developer Tools helped in resolving data mismatch and response delay issues.

➤ Code Improvement Strategies

- **Code Refactoring-** Improved readability, modularity, and reusability by breaking large functions into smaller ones and using clean coding principles.
- **TypeScript Implementation -** Enforced strong typing to reduce runtime bugs and improve IDE support with better code suggestions and error detection.
- **Reusable Components -** Created generic UI components (buttons, modals, input fields) to reduce duplication and increase consistency across the UI.
- **API Optimization -** Minimized payload sizes, added loading indicators, and handled errors gracefully to improve UX during network delays.
- **Code Linting and Formatting -** Integrated tools like **ES Lint** and **Prettier** to maintain consistent code style and catch syntax or logical issues early.

5 System security measures

5.1 Database security

➤ Secure ORM with Prisma

- Prisma enforces structured access to the database using schemas, reducing chances of unauthorized or malformed queries.
- Type-safe queries prevent injection attacks and coding errors.

➤ Environment-Based Configuration

- Database connection strings and credentials are stored securely using environment variables (.env files) and not hardcoded in source files.
- Access to .env is restricted in production environments.

➤ Authentication & Access Control

- Role-based access control (RBAC) is implemented—admins and users have different data access levels.
- Sensitive routes (e.g., resume storage or content editing) are protected with token-based authentication (JWT).

➤ Input Validation & Sanitization

- All user inputs are validated on both client and server sides to prevent **No SQL injection** or malformed data entry.
- Middleware handles sanitization before any Prisma database query is executed.

➤ Encrypted Data Storage

- Fields like passwords are hashed using **bcrypt** before storage.
- Resumes stored in the cloud are secured with access tokens.

➤ Database Access Control

- Mongo DB database is hosted on a secure server with IP white listing and user authentication.
- Read/write permissions are restricted per database user.

➤ Regular Backups & Monitoring

- Periodic backups ensure data recovery.
- Monitoring tools may be integrated for real-time anomaly detection.

5.2 Creation of User profiles and access rights

User Profile Creation

User Registration

- Users sign up using Full Name,
- Phone number,
- Email,
- Password
- Confirm password

Complete your profile

- Industry
- Years of Experience
- Skills
- Professional Bio

Access Rights and Role-Based Permissions

- **Roles Defined:**
 - **Admin**
 - Can manage users, content, and moderate discussions.
 - Access to analytics and database-related controls.
 - **User (Student/Enthusiast)**
 - Can view learning materials, create resumes, and take mock interviews.
 - Access limited to their own data and public resources.
- **Authorization Middleware**
 - API routes are protected using middleware to check roles via **JWT tokens**.
 - Prevents unauthorized access to admin or restricted features.

6 Cost Estimation of the Project

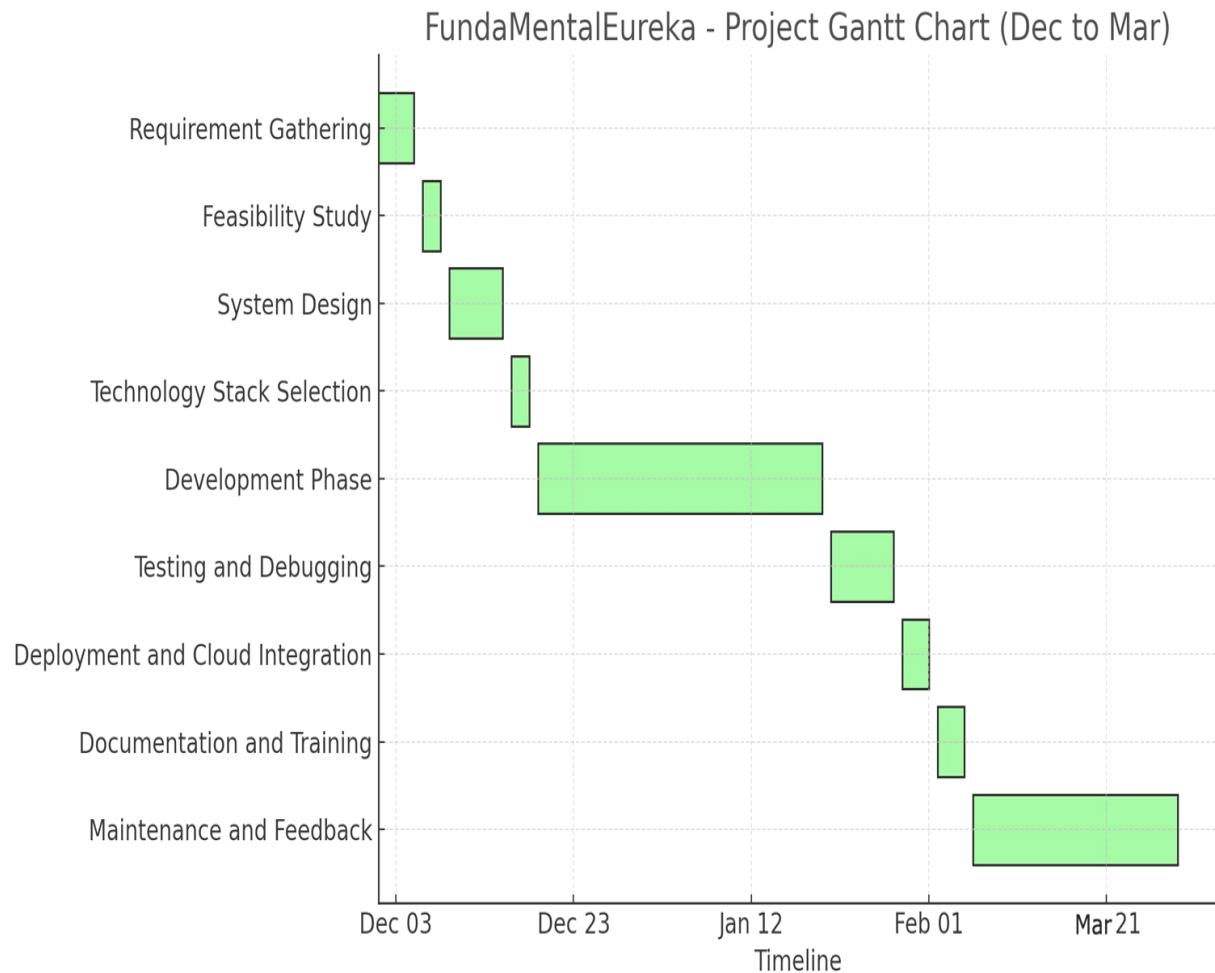
6.1 Reports

FundaMentalEureka is a modern web-based learning platform designed to bridge the gap between academic knowledge and real-world skill requirements. Tailored for engineering students and job aspirants, it provides an interactive environment to learn core technical concepts, enhance professional skills, and prepare for interviews. The platform combines structured educational content, AI-powered responses, and user-contributed discussions to support collaborative and active learning.

Key features include AI-assisted interview preparation, resume generation with cloud storage, and subject-based learning modules. Built using **Next.js**, **TypeScript**, **Tailwind CSS**, **REST APIs**, and **Mongo DB with Prisma**, it offers a responsive, scalable, and secure experience. Security is enforced through data validation, JWT authentication, role-based access control, and encrypted storage.

Users can create personalized profiles, track their progress, and access curated resources to support continuous development. Admins manage content and oversee community interactions. With real-time feedback, interactive tools, and intelligent assistance, FundaMentalEureka stands out as a comprehensive platform that supports both academic growth and career advancement in today's competitive landscape.

6.2 Gantt chart



7 Conclusion and Future Scope

FundaMentalEureka is a web-based educational platform developed to meet the evolving needs of students and job aspirants by integrating academic learning with professional development. The project began with a clear identification of the need for a centralized platform that supports subject understanding, skill enhancement, AI-powered resume building, and interview preparation. The system was carefully designed using modern technologies like **Next.js**, **TypeScript**, **Tailwind CSS**, and **Mongo DB** ensuring flexibility, scalability, and security.

Each chapter of the report—from **system analysis**, **requirement specification**, and **system design to testing, implementation, and debugging**—played a vital role in shaping the application into a robust solution. The modular design ensured easier maintenance and future enhancements, while role-based access control and data security helped maintain user trust.

FundaMentalEureka successfully combines theoretical learning with real-world application, providing a complete environment for users to grow academically and professionally. It stands as a reliable and scalable platform capable of evolving with future learning needs.

Future scope and further enhancement of the project

- Expand content to include more engineering domains and interdisciplinary subjects.
- Integrate **live mentoring or chat support** for personalized guidance.
- Regularly update content and features based on user feedback to maintain relevance.
- Enhance the AI chat.
- Add subscription.

8 References and Appendices

References

MDN Web Docs – <https://developer.mozilla.org/>

(For JavaScript, Typescript, CSS, and web development documentation)

Next.js Documentation – <https://nextjs.org/docs>

(Used for implementation of the frontend framework)

Tailwind CSS Documentation – <https://tailwindcss.com/docs>

(Used for responsive UI design and component styling)

Prisma ORM Documentation – <https://www.prisma.io/docs>

(Used for interaction between the application and Mongo DB database)

Mongo DB Documentation – <https://www.mongodb.com/docs/>

(Database management, schema design, and security practices)

Gemini Documentation – <https://ai.google.dev/gemini-api/docs>

(Used for integrating AI-based features like answer generation)

W3Schools – <https://www.w3schools.com/>

(General reference for HTML, CSS, JavaScript, and responsive design)

OWASP Foundation – <https://owasp.org/>

(Reference for implementing web security standards and best practices)

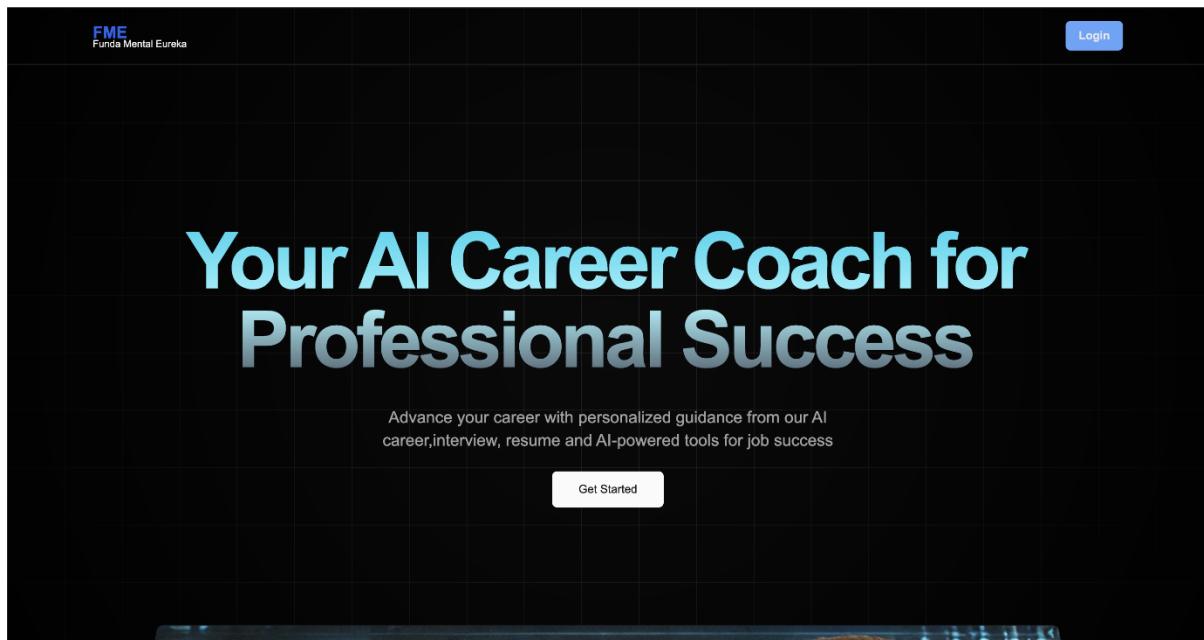
GeeksforGeeks – <https://www.geeksforgeeks.org/>

(Conceptual support for resume building, interview preparation, and data structures)

FreeCodeCamp – <https://www.freecodecamp.org/>

(Tutorials and references for full-stack development)

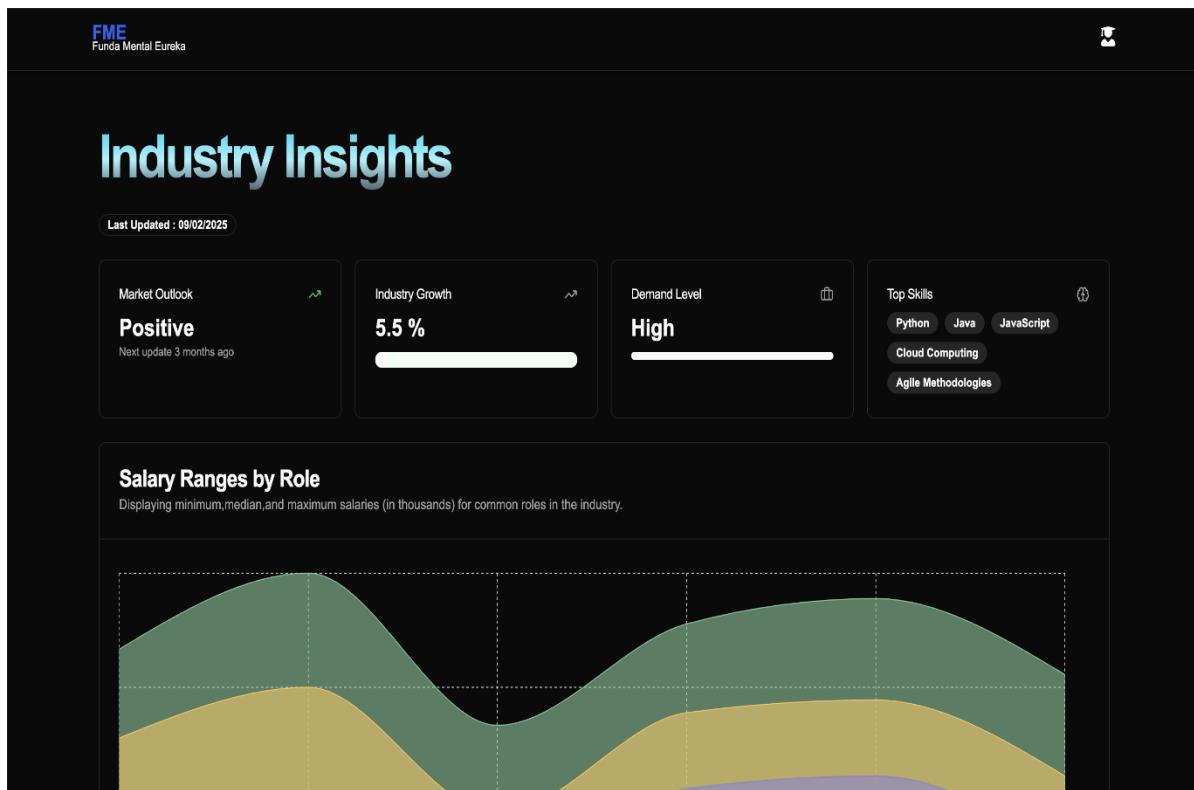
Appendices:



Landing page of FundaMentalEureka

The profile completion page has a dark background. It features a central card with the heading 'Complete Your Profile'. Below it is a sub-instruction: 'Select your industry to get personalized career insights and recommendations.' There are four input fields: 'Industry' (a dropdown menu with 'Select an industry'), 'Years of Experience' (a text input field with placeholder 'Enter years of experience'), 'Skills' (a text input field with placeholder 'e.g., Python, JavaScript, Project Management' and a note 'Separate multiple skills with commas'), and 'Professional Bio' (a text area with placeholder 'Tell us about your professional background...'). At the bottom of the card is a 'Complete Profile' button.

Profile complete page of FundaMentalEureka



Dashboard page of FundaMentalEureka

The resume builder page for FundaMentalEureka allows users to build their resume using a form or Markdown. It includes sections for 'Contact Information' (Email, Mobile Number, LinkedIn URL, Twitter/X Profile) and 'Professional Summary' (a text input field for writing a professional summary). A 'Save' button is located in the top right corner.

Resume builder page of FundaMentalEureka

FME
Funda Mental Eureka

← Back to Interview Preparation

Mock Interview

Test your knowledge with industry-specific questions

Question 1 of 10

What is the primary purpose of the 'useEffect' hook in React?

- To conditionally render components
- To manage component state
- To perform side effects like data fetching or DOM manipulation
- To define component props

Show Explanation

Next Question

© 2025 FundaMentalEureka. All rights reserved.

Mock Interview page of FundaMentalEureka

FME
Funda Mental Eureka

Interview Preparation

Average Score 48.00%
Across All Assessments

Quiz Practiced 5
Total Quiz

Latest Score 80.0%
Most recent quiz score

Performance Trend

Your quiz scores over time

Date	Score (%)
Mar 16	60
Mar 16	75
Mar 16	65
Mar 16	55
Mar 16	45
Mar 16	35
Mar 16	25
Mar 16	15
Mar 16	10
Feb 19	20
Feb 19	15
Feb 19	10
Feb 19	5
Feb 17	10
Feb 17	15
Feb 17	20
Feb 17	25
Feb 17	30
Feb 17	35
Feb 17	40
Feb 17	45
Feb 17	50
Feb 17	55
Feb 17	60
Feb 17	65
Feb 17	70
Feb 17	75
Feb 17	80
Feb 10	80

Interview Dashboard page of FundaMentalEureka