Project Report

Title of the Project: AI Enhanced Career Guide Application

Course Code: CSE226

Course Title: Android App Development

Submitted By: Deepak Singla

Name: Deepak Singla

Reg No: 12220033

Submitted To: Mr. Vikas Sharma

Faculty Name and designation

LOVELY PROFESSIONAL UNIVERSITY

LOVELY PROFESSIONAL UNIVERSITY

DECLARATION

I, **Deepak Singla**, a student of Bachelor of Technology under CSE discipline at Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own work and is genuine

Your name: **Deepak Singla**

Registrartion Number: **12220033**

Table of Contents

1.Project Description

Objective:

The primary objective of the **AI Enhanced Career Guidance App (CareerBot)** is to develop a comprehensive, modern, and highly personalized mobile platform that leverages **Generative AI** to assist users with their **career planning, educational choices, skill development, and interview preparation.** It aims to replace static, generalized career advice with dynamic, interactive, and context-aware guidance, all within a secure and intuitive Android application.

Core Features:

🧠 AI-Powered Career Chatbot (Core Feature)

- **Technology:** Integrated with the **Gemini API** (via Google AI Studio/Firebase AI Logic).
- **Functionality:** Provides **real-time conversational guidance** in the chat section.
- **Scope:** The chatbot is strictly configured to answer prompts related to **study, career paths, and interview help** (e.g., *"What to do after 12th?"*, *"Career in Data Science"*). It provides a polite, automated response if a non-career-related question is asked.
- **Data Persistence:** All chat conversations and history are securely **saved in MongoDB** for continuity and analysis.

💡 AI Quiz Generator

- **Functionality:** Allows users to **search any topic** (e.g., *"Basic Java"*, *"Kotlin"*).
- **Outcome:** The app generates a **basic quiz** dynamically on the requested topic, providing an immediate knowledge assessment

tool. This showcases a practical, on-demand use case of Generative AI beyond simple chat.

📈 Progress Tracker and Goal Setting

- **Functionality:** A dedicated section where users can **add, track, and monitor** their professional and educational **goals** (e.g., *"Complete Python Course"*, *"Practice 5 Mock Interviews"*).
- **Purpose:** Promotes user engagement and transforms the app into an actionable tool for personal development.

📝 Career Tips and Insights

- **Functionality:** Provides curated and potentially personalized **career tips** to aid the user's development, complementing the advice given by the chatbot.

🔐 Secure User Authentication

- **System:** Includes dedicated **Login and Sign-up pages**.
- **Process:** Users create an account using their **email ID and password**.
- **Security:** All user data, including login credentials and profile information, is **saved securely in MongoDB**.

📱 Modular and User-Centric Interface

- **Navigation:** Features a clean dashboard with **multiple navigation buttons** for quick access, a **settings icon** on top, and an accessible **profile section**.
- **UX Features:** Includes a **Profile** page and a **Dark Mode** toggle in the Settings for enhanced user experience and accessibility.

🎯 Challenges Addressed

- **Generalized Advice:** Moving from static, one-size-fits-all career books to **personalized, real-time advice** via Generative AI.
- **Lack of Practice:** Providing an **on-demand quiz tool** for immediate knowledge testing and skill validation.

- **Disorganized Planning:** Offering a **Progress Tracker** to help users formalize, organize, and monitor their career goals.
- **Technology Integration:** Successfully integrating complex technologies like the **Gemini API** and a NoSQL database (**MongoDB**) into a single, cohesive Android application.

👩‍💻 **End Users**

- **Students:** Seeking guidance on educational paths (e.g., after 12th grade, college majors).
- **Job Seekers:** Needing interview preparation, resume advice, and job market insights.
- **Working Professionals:** Looking for upskilling tips, career change advice, and professional goal tracking.

2. Technology Used (Front End/Back End)

The **AI Enhanced Career Guidance App (CareerBot)** is architected using a modern and scalable stack, combining native Android development tools with powerful cloud-based and database services.

📱 **Front End: Android Studio (Kotlin + XML)**

The client-side application is built natively to ensure high performance, device compatibility, and a premium user experience.

Technologies and Tools:

- **Kotlin:**
  The primary programming language. Chosen for its modern features, conciseness, safety, and official support by Google for Android development, which simplifies the integration of the Gemini API.

- **XML (Extensible Markup Language):**
  Used exclusively for defining the **layout and structure of the User Interface (UI)**, including the dashboard, chat screen, profile, and navigation components.

- **Material Design Components:**
  Principles applied across the UI (buttons, navigation bars, cards) to ensure a **clean, consistent, and intuitive user experience** with features like the **Dark Mode** option in Settings.

- **Android Libraries:**
  Utilized components like **RecyclerView** (for displaying chat history and tips) and **Navigation Components** (for managing screen flow).

Back End: MongoDB + Google AI Studio (Gemini API)
The backend employs a hybrid architecture, using MongoDB for robust data management and the Gemini API for intelligence.

MongoDB
The primary NoSQL database. Used for:
User Account Data: Secure storage of login/signup credentials (email/password).
Chat History: Storing the log of conversations with the AI chatbot.
Progress Data: Saving user-defined goals and progress status.
Quiz History: Storing records of completed AI quizzes.
Advantage: Its flexible document model is ideal for handling the diverse, unstructured nature of chat logs and dynamic quiz data.

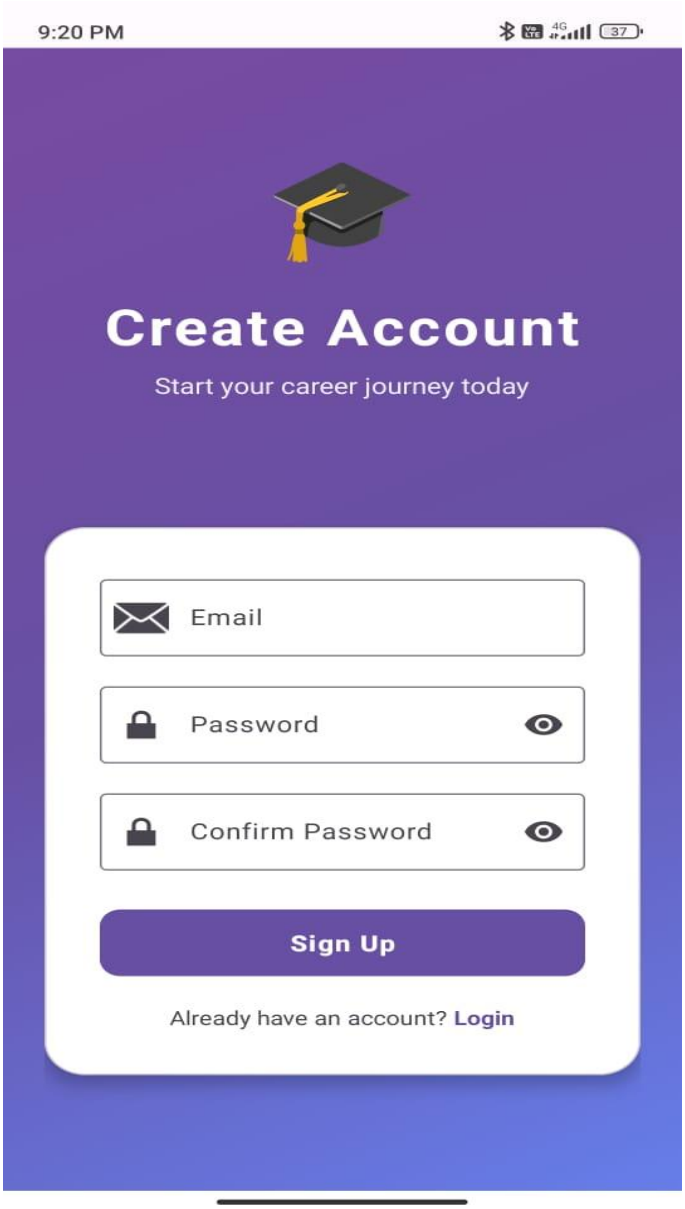
Gemini API
The core Artificial Intelligence engine, accessed via Google AI Studio or its dedicated SDK. Used for:
Generative Chat: Powering the real-time, contextual conversations for career and education advice.

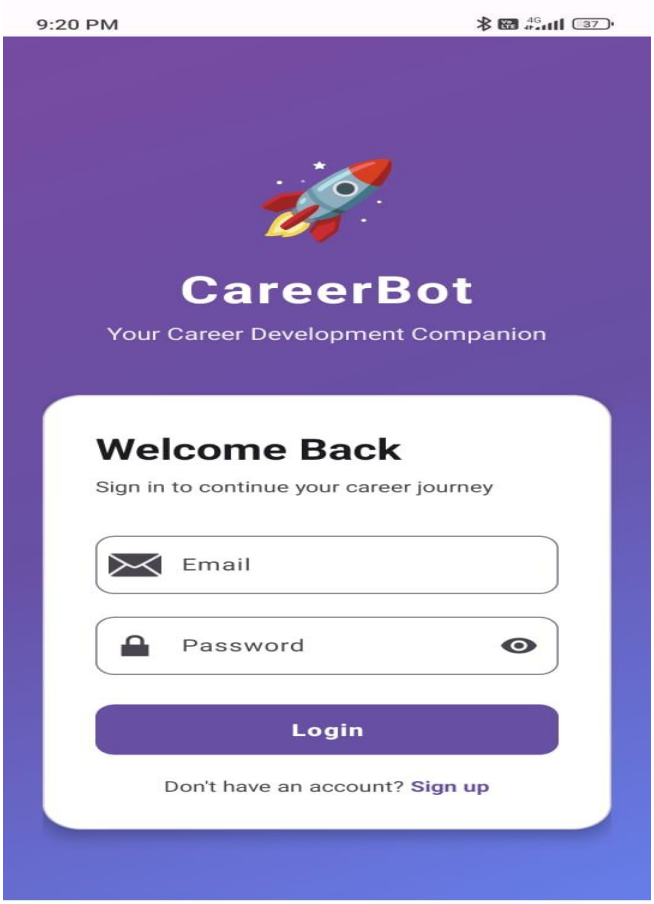
Key Highlight: The use of Gemini demonstrates integration of cutting-edge Generative AI technology.

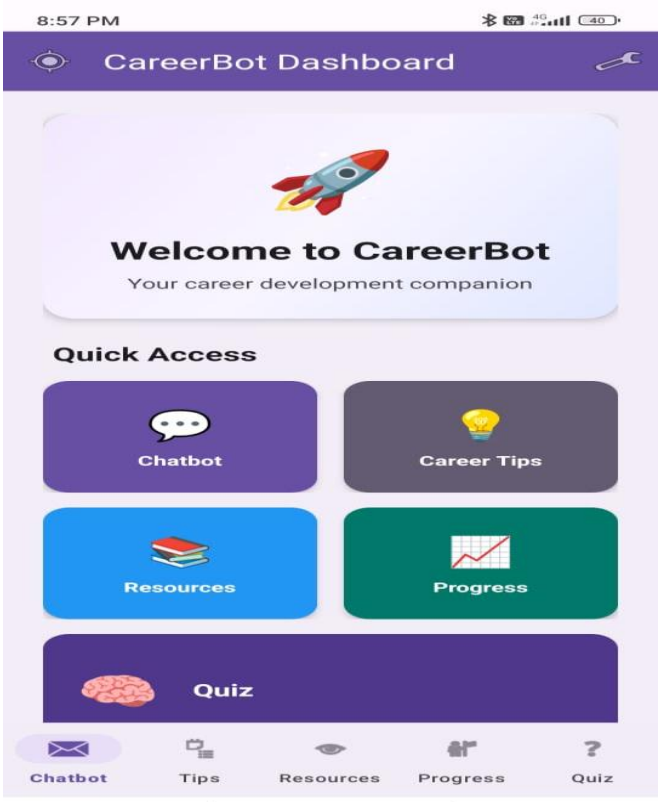
Authentication System

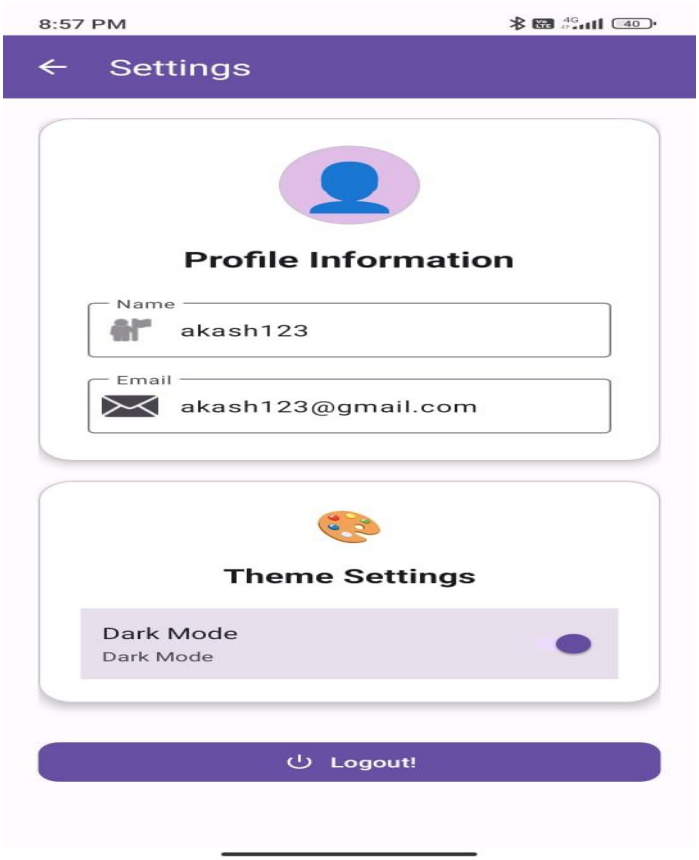
The app's custom login/signup system is built around securing user accounts and validating credentials against the data stored in MongoDB.

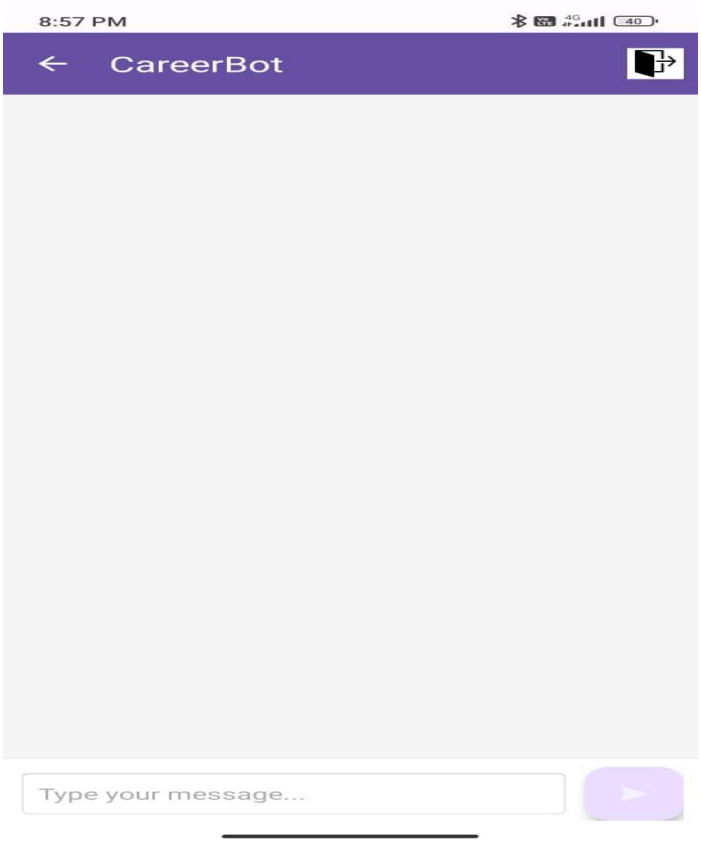
3.Screenshots of the project

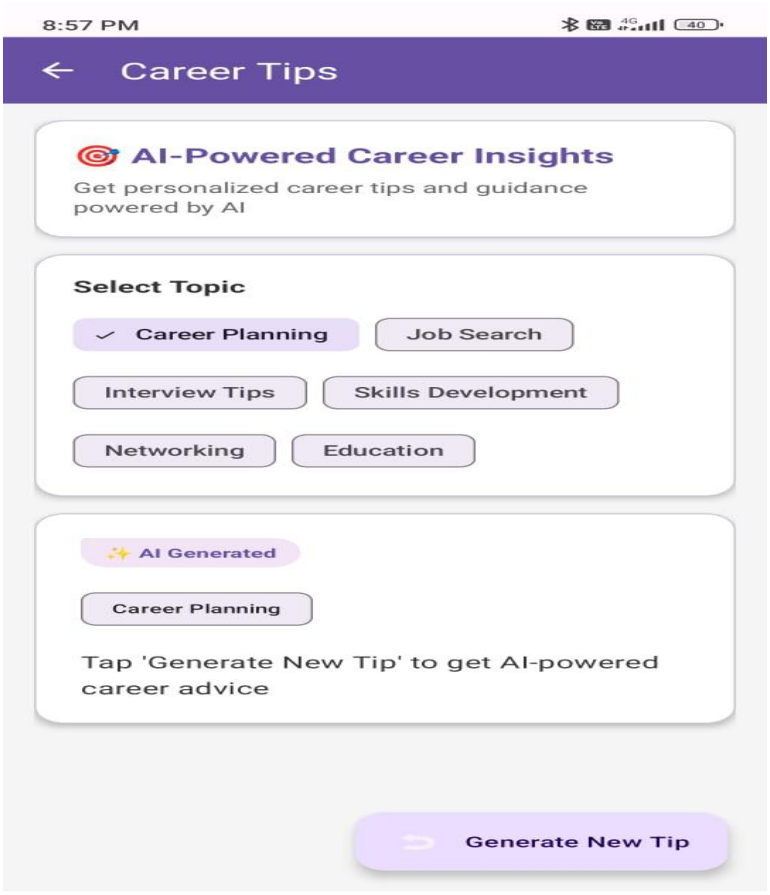


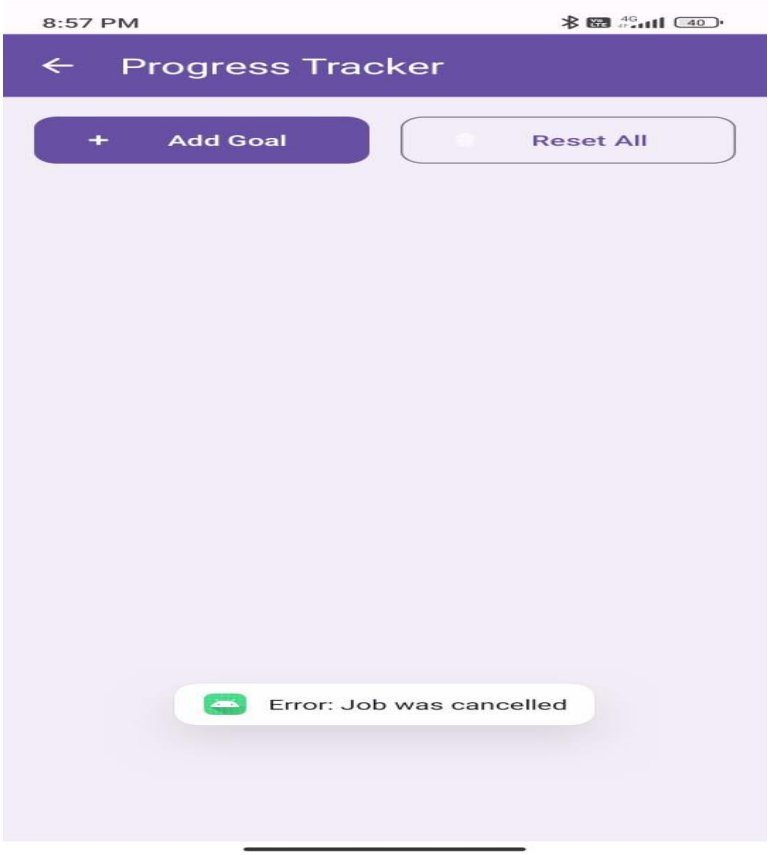

9:20 PM
CareerBot
Your Career Development Companion
Welcome Back
Sign in to continue your career journey
Email
Password
Login
Don't have an account? Sign up

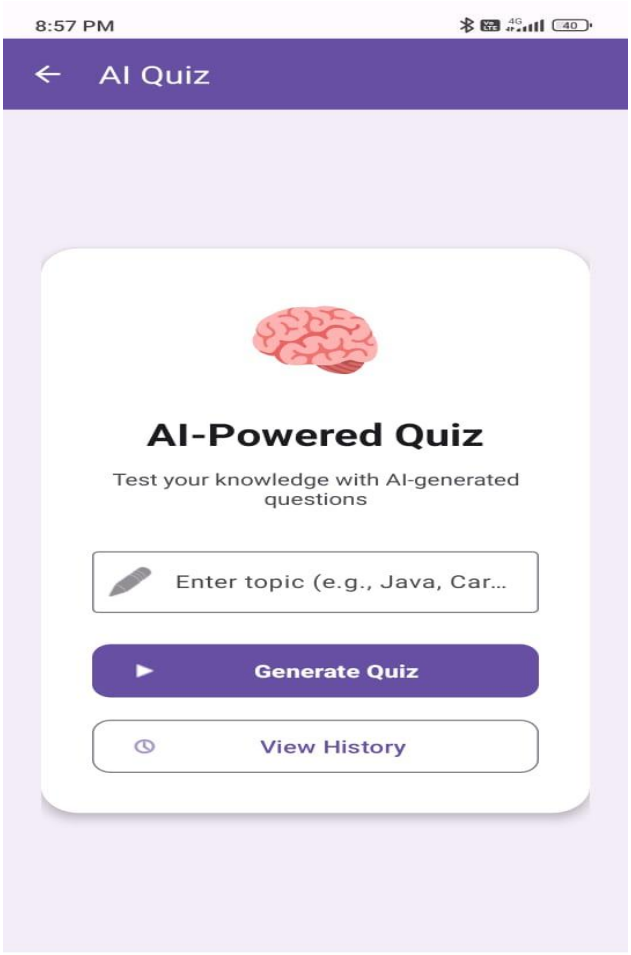

8:57 PM
CareerBot Dashboard
Welcome to CareerBot
Your career development companion
Quick Access
Chatbot
Career Tips
Resources
Progress
Quiz
Chatbot
Tips
Resources
Progress
Quiz

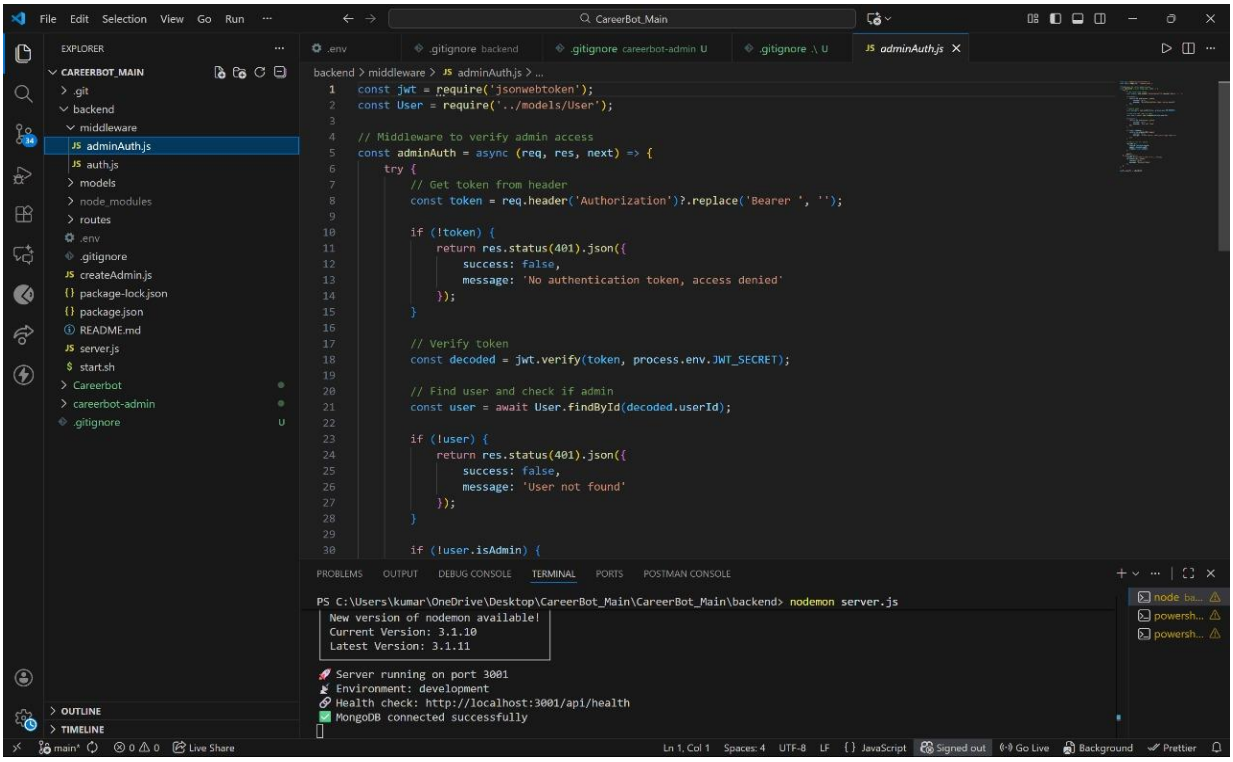

8:57 PM
Settings
Profile Information
Name
akash123
Email
akash123@gmail.com
Theme Settings
Dark Mode
Dark Mode
Logout!

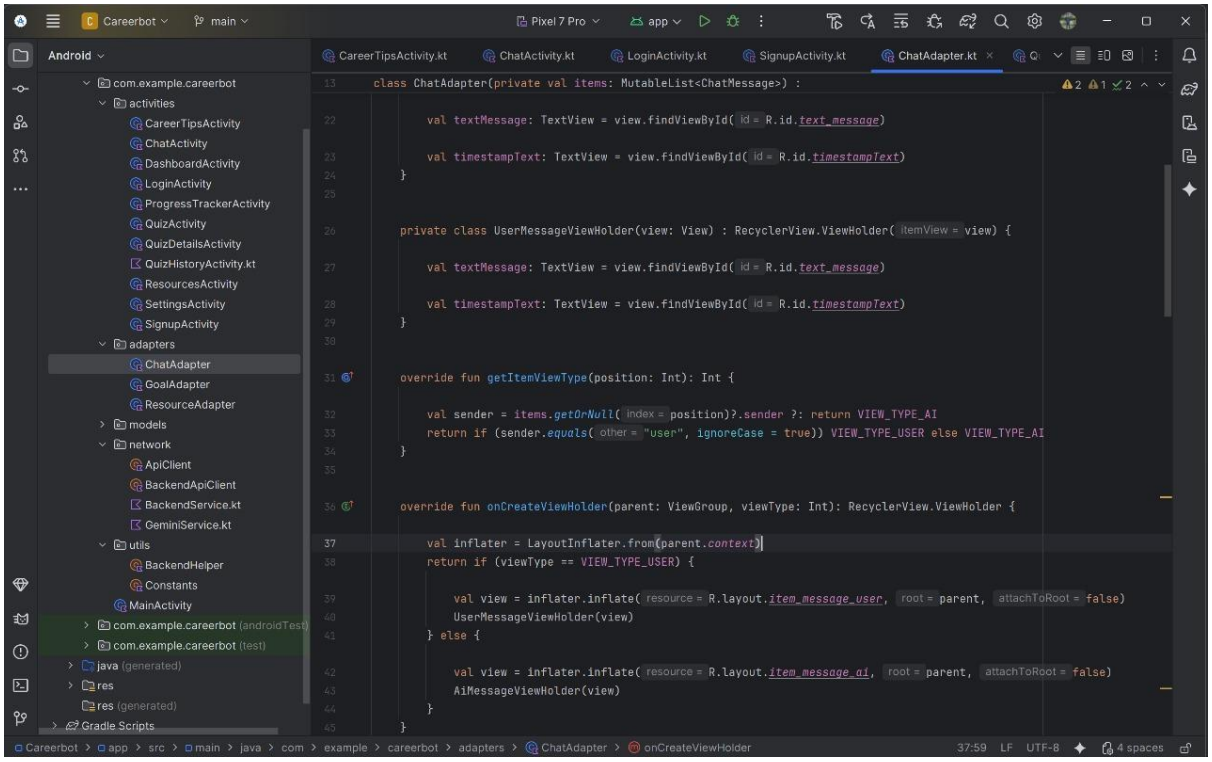

8:57 PM
CareerBot
Type your message...

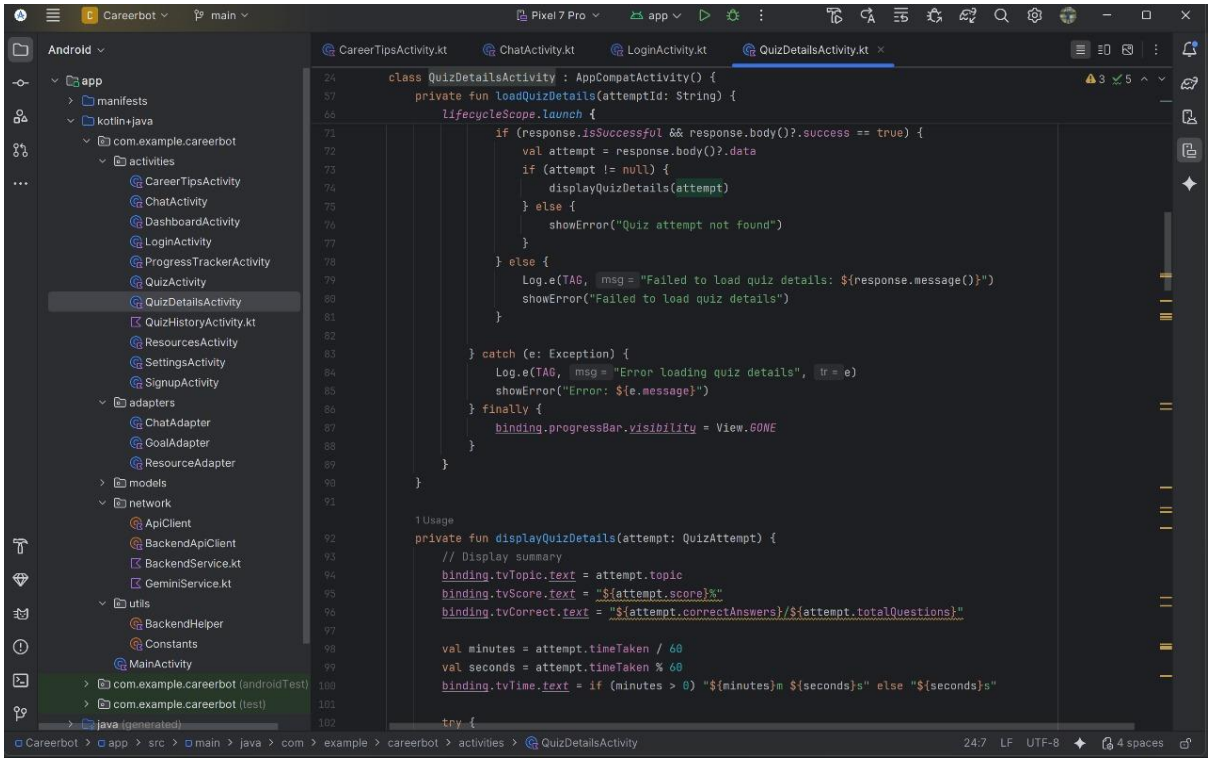

8:57 PM
Career Tips
AI-Powered Career Insights
Get personalized career tips and guidance powered by AI
Select Topic
Career Planning
Job Search
Interview Tips
Skills Development
Networking
Education
AI Generated
Career Planning
Tap 'Generate New Tip' to get AI-powered career advice
Generate New Tip

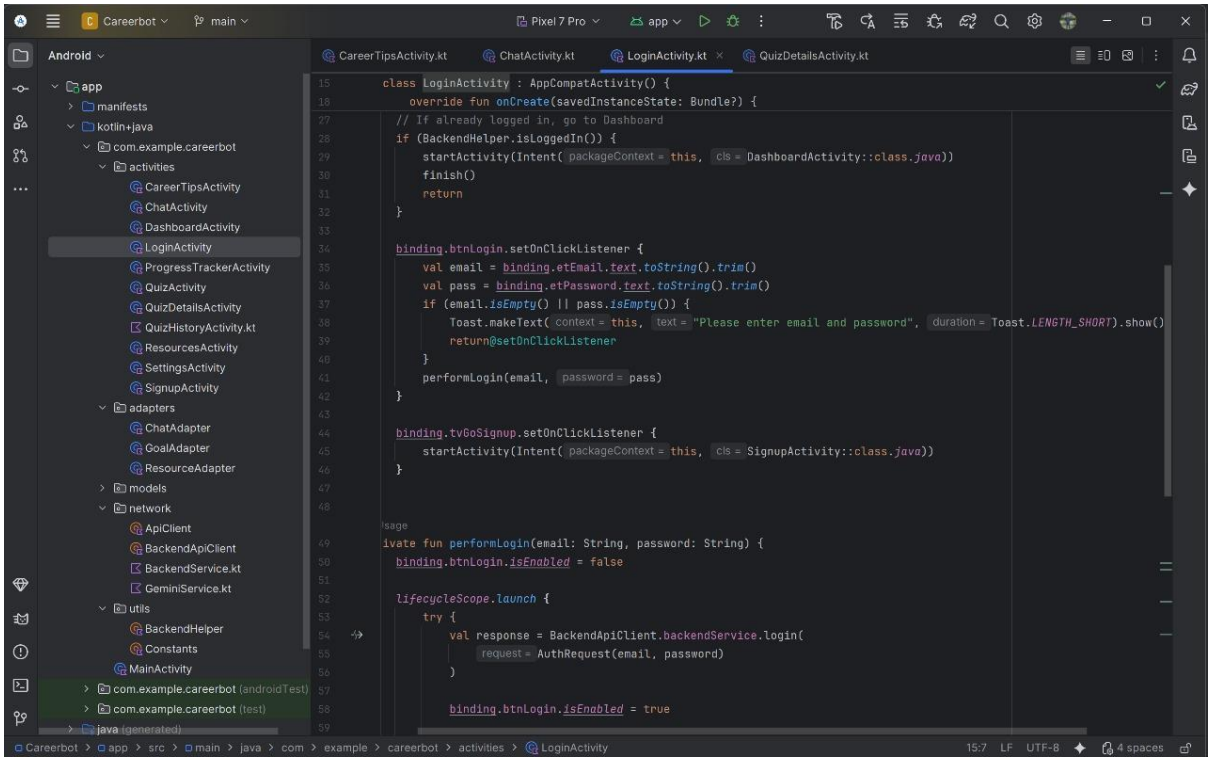

8:57 PM
Progress Tracker
Add Goal
Reset All
Error: Job was cancelled

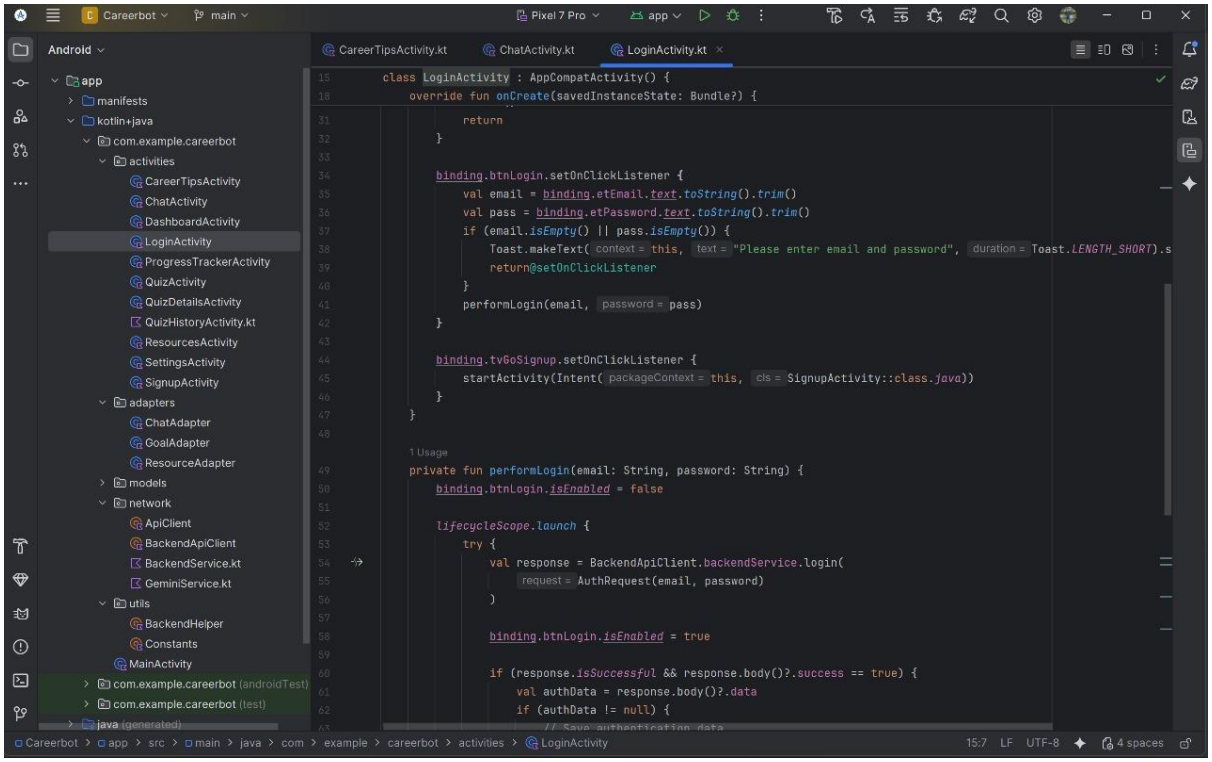

8:57 PM
AI Quiz
AI-Powered Quiz
Test your knowledge with AI-generated questions
Enter topic (e.g., Java, Car...
Generate Quiz
View History


File Edit Selection View Go Run
CareerBot_Main
EXPLORER
CAREERBOT_MAIN
.git
backend
middleware
adminAuth.js
auth.js
models
node_modules
routes
.env
.gitignore
createAdmin.js
package-lock.json
package.json
README.md
server.js
start.sh
Careerbot
careerbot-admin
.gitignore
OUTLINE
TIMELINE
.env
.gitignore backend
.gitignore careerbot-admin U
.gitignore .\ U
adminAuth.js
backend > middleware > adminAuth.js > ...
const jwt = require('jsonwebtoken');
const User = require('../models/User');
// Middleware to verify admin access
const adminAuth = async (req, res, next) => {
try {
// Get token from header
const token = req.header('Authorization')?.replace('Bearer ', '');
if (!token) {
return res.status(401).json({
success: false,
message: 'No authentication token, access denied'
});
}
// Verify token
const decoded = jwt.verify(token, process.env.JWT_SECRET);
// Find user and check if admin
const user = await User.findById(decoded.userId);
if (!user) {
return res.status(401).json({
success: false,
message: 'User not found'
});
}
if (!user.isAdmin) {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
PS C:\Users\kumar\OneDrive\Desktop\CareerBot_Main\CareerBot_Main\backend> nodemon server.js
New version of nodemon available!
Current Version: 3.1.10
Latest Version: 3.1.11
Server running on port 3001
Environment: development
Health check: http://localhost:3001/api/health
MongoDB connected successfully
node ba...
powersh...
powersh...
main*
Live Share
Ln 1, Col 1
Spaces: 4
UTF-8
LF
JavaScript
Signed out
Go Live
Background
Prettier


Careerbot
main
Pixel 7 Pro
app
Android
CareerTipsActivity.kt
ChatActivity.kt
LoginActivity.kt
SignupActivity.kt
ChatAdapter.kt
com.example.careerbot
activities
CareerTipsActivity
ChatActivity
DashboardActivity
LoginActivity
ProgressTrackerActivity
QuizActivity
QuizDetailsActivity
QuizHistoryActivity.kt
ResourcesActivity
SettingsActivity
SignupActivity
adapters
ChatAdapter
GoalAdapter
ResourceAdapter
models
network
ApiClient
BackendApiClient
BackendService.kt
GeminiService.kt
utils
BackendHelper
Constants
MainActivity
com.example.careerbot (androidTest)
com.example.careerbot (test)
java (generated)
res
res (generated)
Gradle Scripts
class ChatAdapter(private val items: MutableList<ChatMessage>) :
val textMessage: TextView = view.findViewById(id = R.id.text_message)
val timestampText: TextView = view.findViewById(id = R.id.timestampText)
private class UserMessageViewHolder(view: View) : RecyclerView.ViewHolder(itemView = view) {
override fun getItemViewType(position: Int): Int {
val sender = items.getOrNull(index = position)?.sender ?: return VIEW_TYPE_AI
return if (sender.equals(other = "user", ignoreCase = true)) VIEW_TYPE_USER else VIEW_TYPE_AI
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {
val inflater = LayoutInflater.from(parent.context)
return if (viewType == VIEW_TYPE_USER) {
val view = inflater.inflate(resource = R.layout.item_message_user, root = parent, attachToRoot = false)
UserMessageViewHolder(view)
} else {
val view = inflater.inflate(resource = R.layout.item_message_ai, root = parent, attachToRoot = false)
AiMessageViewHolder(view)
Careerbot › app › src › main › java › com › example › careerbot › adapters › ChatAdapter › onCreateViewHolder
37:59 LF UTF-8 4 spaces


Careerbot
main
Pixel 7 Pro
app
Android
CareerTipsActivity.kt
ChatActivity.kt
LoginActivity.kt
QuizDetailsActivity.kt
app
manifests
kotlin+java
com.example.careerbot
activities
CareerTipsActivity
ChatActivity
DashboardActivity
LoginActivity
ProgressTrackerActivity
QuizActivity
QuizDetailsActivity
QuizHistoryActivity.kt
ResourcesActivity
SettingsActivity
SignupActivity
adapters
ChatAdapter
GoalAdapter
ResourceAdapter
models
network
ApiClient
BackendApiClient
BackendService.kt
GeminiService.kt
utils
BackendHelper
Constants
MainActivity
com.example.careerbot (androidTest)
com.example.careerbot (test)
java (generated)
class QuizDetailsActivity : AppCompatActivity() {
private fun loadQuizDetails(attemptId: String) {
lifecycleScope.launch {
if (response.isSuccessful && response.body()?.success == true) {
val attempt = response.body()?.data
if (attempt != null) {
displayQuizDetails(attempt)
} else {
showError("Quiz attempt not found")
} else {
Log.e(TAG, msg = "Failed to load quiz details: ${response.message()}")
showError("Failed to load quiz details")
} catch (e: Exception) {
Log.e(TAG, msg = "Error loading quiz details", tr = e)
showError("Error: ${e.message}")
} finally {
binding.progressBar.visibility = View.GONE
1 Usage
private fun displayQuizDetails(attempt: QuizAttempt) {
// Display summary
binding.tvTopic.text = attempt.topic
binding.tvScore.text = "${attempt.score}%"
binding.tvCorrect.text = "${attempt.correctAnswers}/${attempt.totalQuestions}"
val minutes = attempt.timeTaken / 60
val seconds = attempt.timeTaken % 60
binding.tvTime.text = if (minutes > 0) "${minutes}m ${seconds}s" else "${seconds}s"
try {
Careerbot › app › src › main › java › com › example › careerbot › activities › QuizDetailsActivity
24:7 LF UTF-8 4 spaces


Careerbot
main
Pixel 7 Pro
app
Android
CareerTipsActivity.kt
ChatActivity.kt
LoginActivity.kt
QuizDetailsActivity.kt
app
manifests
kotlin+java
com.example.careerbot
activities
CareerTipsActivity
ChatActivity
DashboardActivity
LoginActivity
ProgressTrackerActivity
QuizActivity
QuizDetailsActivity
QuizHistoryActivity.kt
ResourcesActivity
SettingsActivity
SignupActivity
adapters
ChatAdapter
GoalAdapter
ResourceAdapter
models
network
ApiClient
BackendApiClient
BackendService.kt
GeminiService.kt
utils
BackendHelper
Constants
MainActivity
com.example.careerbot (androidTest)
com.example.careerbot (test)
java (generated)
class LoginActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
// If already logged in, go to Dashboard
if (BackendHelper.isLoggedIn()) {
startActivity(Intent(packageContext = this, cls = DashboardActivity::class.java))
finish()
return
}
binding.btnLogin.setOnClickListener {
val email = binding.etEmail.text.toString().trim()
val pass = binding.etPassword.text.toString().trim()
if (email.isEmpty() || pass.isEmpty()) {
Toast.makeText(context = this, text = "Please enter email and password", duration = Toast.LENGTH_SHORT).show()
return@setOnClickListener
}
performLogin(email, password = pass)
}
binding.tvGoSignup.setOnClickListener {
startActivity(Intent(packageContext = this, cls = SignupActivity::class.java))
}
sage
ivate fun performLogin(email: String, password: String) {
binding.btnLogin.isEnabled = false
lifecycleScope.launch {
try {
val response = BackendApiClient.backendService.login(
request = AuthRequest(email, password)
)
binding.btnLogin.isEnabled = true
Careerbot > app > src > main > java > com > example > careerbot > activities > LoginActivity
15:7 LF UTF-8 4 spaces


Careerbot
main
Pixel 7 Pro
app
Android
CareerTipsActivity.kt
ChatActivity.kt
LoginActivity.kt
app
manifests
kotlin+java
com.example.careerbot
activities
CareerTipsActivity
ChatActivity
DashboardActivity
LoginActivity
ProgressTrackerActivity
QuizActivity
QuizDetailsActivity
QuizHistoryActivity.kt
ResourcesActivity
SettingsActivity
SignupActivity
adapters
ChatAdapter
GoalAdapter
ResourceAdapter
models
network
ApiClient
BackendApiClient
BackendService.kt
GeminiService.kt
utils
BackendHelper
Constants
MainActivity
com.example.careerbot (androidTest)
com.example.careerbot (test)
java (generated)
class LoginActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
return
}
binding.btnLogin.setOnClickListener {
val email = binding.etEmail.text.toString().trim()
val pass = binding.etPassword.text.toString().trim()
if (email.isEmpty() || pass.isEmpty()) {
Toast.makeText(context = this, text = "Please enter email and password", duration = Toast.LENGTH_SHORT).s
return@setOnClickListener
}
performLogin(email, password = pass)
}
binding.tvGoSignup.setOnClickListener {
startActivity(Intent(packageContext = this, cls = SignupActivity::class.java))
}
}
1 Usage
private fun performLogin(email: String, password: String) {
binding.btnLogin.isEnabled = false
lifecycleScope.launch {
try {
val response = BackendApiClient.backendService.login(
request = AuthRequest(email, password)
)
binding.btnLogin.isEnabled = true
if (response.isSuccessful && response.body()?.success == true) {
val authData = response.body()?.data
if (authData != null) {
Careerbot > app > src > main > java > com > example > careerbot > activities > LoginActivity
15:7 LF UTF-8 4 spaces

4. Future Scope

The **AI Enhanced Career Guidance App (CareerBot)** has established a robust foundation by successfully integrating key features like a Gemini-powered chatbot, on-demand AI quizzing, a progress tracker, and MongoDB data management. The modular design of the Android application (Kotlin/XML) ensures high scalability.

To further solidify CareerBot's position as a leading digital career development tool, the following future enhancements are planned:

🧠 1. Advanced AI Personalization and Analytics

The core AI functionality can be deepened to offer a truly tailored experience:

- **Adaptive Guidance:** Utilize the stored user data from **MongoDB** (chat history, quiz results, and progress goals) to refine the Gemini chatbot's responses, providing **more personalized and context-aware advice**.
- **Skill Gap Analysis:** Implement AI logic that analyzes quiz performance and career goals to **identify specific skill gaps** and suggest relevant external courses or learning resources (APIs from MOOC providers like Coursera or edX).
- **Predictive Career Paths:** Based on current market trends and user profile, suggest highly probable and suitable future career transitions.

🌐 2. External API and Job Board Integration

Expand the app's utility by connecting to real-world career resources:

- **Job Listing Integration:** Integrate with popular job board APIs (e.g., Indeed, LinkedIn) to display **live, relevant job opportunities** matching the user's skills and goals directly within the app.

- **Resume Builder Tool:** Introduce an in-app feature that leverages the Gemini API to help users **draft, optimize, and format their resumes** and cover letters based on target job descriptions.
- **External Resource Linking:** Integrate links to reputable courses, training programs, and certification bodies that align with the user's desired career path.

👥 3. Community and Expert Connection

Foster a more comprehensive and interactive user experience:

- **Peer Mentorship:** Introduce a feature for users to connect with peers for study groups or networking.
- **Expert Q&A Forums:** Implement a secure forum where users can pose questions to **verified industry experts or career counselors** (monetization potential).
- **Video Interview Simulator:** Use the Gemini API to generate complex, role-specific interview questions and record the user's response for self-review.

📊 4. Gamification and Progress Visualization

Enhance user engagement and goal achievement through design:

- **Interactive Dashboards:** Develop more detailed visualization tools (using graphs and charts) to show the user's progress on their goals, quiz mastery over time, and time spent on the app.
- **Achievement Badges:** Implement a system of badges or rewards for completing goals, passing quizzes, or reaching milestones to **gamify the learning process**.

💸 5. Monetization and Premium Features

Future development could introduce a premium tier:

- **Ad-free Experience.**
- **Access to Premium AI Models** for more detailed or lengthy consultations.
- **Unlimited Quiz Generation** and advanced analytics reports.

5. Conclusion

The **AI Enhanced Career Guidance App (CareerBot)** represents a successful endeavor in creating an intelligent, scalable, and user-focused tool for career development on the Android platform.

Key Achievements:

- **Seamless AI Integration:** The project successfully implemented the **Gemini API** into the native Kotlin/XML environment, delivering a **powerful, real-time conversational AI chatbot** and an innovative **on-demand AI quiz generator**.
- **Robust Data Management:** The use of **MongoDB** ensures secure and scalable storage for user accounts (login/signup), personalized goal tracking, and critical user data like chat and quiz history, enabling future personalization.
- **Holistic User Experience:** By combining the power of AI guidance with practical tools like the **Progress Tracker** and **Career Tips**, the application provides a comprehensive and actionable solution that addresses the modern challenges of career planning.
- **Technical Proficiency:** The project demonstrates strong proficiency in **native Android development (Kotlin/XML)** and the integration of modern cloud services and APIs, providing a stable, high-performance application.

In conclusion, **CareerBot** is more than just an app; it is a proof-of-concept for how **Generative AI can revolutionize the way individuals approach education and career development.** It provides a solid, expandable platform that is well-positioned for future integration with job boards, mentorship programs, and advanced predictive analytics. The successful deployment of this project showcases the ability to design, develop, and deploy an impactful, feature-rich mobile application using cutting-edge technology.