SKILLFORGE: AN AI-POWERED JOB READY PREPARATION PLATFORM

A Full-Stack Web Application Project Report

Submitted By:

Bhavesh Dilip Khandare

Roll No: 23040

Guided by: Priti Jadhav Mam

Department of Computer Science

Swaraj Collage of Arts, Commerce and Science

Certificate

This is to certify that the project entitled 'SkillForge' is a record of bonafide work carried out by me under the guidance of my project supervisor. This project has not been submitted elsewhere for any other degree or diploma.

_____ _____

Principal Signature External Examiner

Declaration

I hereby declare that the project work entitled 'SkillForge' is my own work and effort. The information used in this project has been obtained from authentic sources and all helping materials have been acknowledged.

_____

(Signature of Student)

Date: February 23, 2026

Acknowledgement

I would like to start by thanking my project coordinator and my teachers for giving me the chance to work on such an interesting project. Building 'SkillForge' was a great learning experience. I also want to thank my parents and friends for their support and suggestions during the development process. Especially, I am thankful for the open-source community whose libraries helped me build this application efficiently.

I would also like to thank my college computer lab assistants who gave me extra hours to test the connectivity of the backend server. Creating the dark mode UI with orange buttons was a suggestion from one of my friends, and I am glad I listened to him because it looks much better than my original white theme design.

Abstract

SkillForge is a modern web application designed to help students and job seekers prepare for technical interviews. In today's world, just knowing how to code is not enough; one needs to explain logic and solve problems under pressure. My project provides a complete environment where users can practice coding, ask questions to an AI mentor, and track their daily progress. The frontend is built using React with a unique black and orange dark theme. The backend uses FastAPI for high performance. The core feature is the AI integration which allows users to get instant explanations for tough coding problems. We also have a PDF generation feature that lets users save their practice sessions. The goal of SkillForge is to make interview preparation organized and less stressful.

In this project, I used modern technologies like React and Python (FastAPI). The main idea was to give students a platform that is not just a coding site but also a learning site. By including AI models, I ensured that a user doesn't have to wait for anyone to get their logic cleared. The dashboard keeps track of daily tasks and the PDF generator makes it easy to keep records. Overall, it's a complete ecosystem for interview preparation.

Table of Content

Certificate 2

Declaration 3

Acknowledgement 4

Abstract 5

Table of Content 6

4. Feasibility Study 11

5. System Requirements 12

6. System Architecture 13

7. Technology Stack Explanation 14

8. Module Description 15

9. User Interface Design 20

11. User Interface Design 21

12. AI Integration in Depth .....**Error! Bookmark not defined.**

13. Security Considerations 23

14. Testing the System 24

15. Limitations of the Project 25

16. Future Enhancements 26

17. Conclusion 27

18. Bibliography / References 28

Introduction

1.1 What is SkillForge?

SkillForge is basically a one-stop platform for anyone looking to clear a tech interview. When I started my own prep, I realized that I had to jump between five different websites— one for coding, one for theory, one for company-specific questions, and another for sticking notes. SkillForge brings all of this together. It is a full-stack website where you can write code, chat with an AI about your doubts, and see your growth on a dashboard.

SkillForge is designed for a generation that loves dark mode and fast interfaces. I tried to make every feature as accessible as possible. For example, instead of hiding the important coding problems deep in menus, they are right there on the dashboard. It is essentially a bridge between a confused student and a prepared job candidate.

1.2 Why this Project is Needed?

The main reason I built this is that existing platforms are either too expensive or too professional to the point where they feel boring. Most students need a platform that feels 'cool' and easy to use. I wanted to build something that motivates a student to spend more time practicing. Also, with the rise of AI, it is now possible to have a personal tutor available 24/7. SkillForge uses this technology to help students when they get stuck on a bug at 2 AM.

The current job market is very competitive. Companies don't just ask if you can code; they ask if you can optimize. Existing websites often throw too much information at you. I wanted to build something that tells the user exactly what they need to focus on today. This focused approach is the biggest 'Why' behind SkillForge.

1.3 Problems in Existing Learning Platforms

Current platforms have a few issues. First, they lack real-time help. If you don't understand an error, you have to search on forums like Stack Overflow. Second, they don't help much with the 'explanation' part of an interview. You might solve a problem, but can you explain it to a recruiter? Third, UI consistency is often missing. Some sites look very old and are hard to navigate on mobile phones.

I also found that many platforms are very slow. They take 5-10 seconds to run a simple Python script. By using FastAPI in my backend, I managed to cut down the waiting time. Also, many popular sites don't have a good mobile view. Many of us study while travelling in buses or trains, so a good mobile UI is a must.

1.4 How SkillForge Solves These Problems

SkillForge solves this by including an 'AI Ask' feature. If you don't understand a problem, the AI explains it like a teacher. It also uses a dark theme with orange accents which is easy on the eyes during long night study sessions. Since it is built with responsive UI (using React and Swiper.js), a student can even revise questions on their phone while traveling.
We also added a PDF generator so people can take their notes offline.

2. Project Objective

- To build a full-stack platform where students can prepare for their tech interviews.
- To solve the problem of not getting help while coding by using AI integration (Groq/ Gemini).
- To create a dark-themed UI that is responsive on every device, from mobile to desktop.
- To allow users to save their answers in PDF format for offline revision using ReportLab.
- To provide a progress tracking system that motivates students with a dashboard and a calendar.
- To give separate roles to free and premium users so the platform can have a business side as well.

My main personal goal while building this was to understand how to bridge the gap between AI and regular web apps. I didn't want the AI to be a separate thing; I wanted it to be part of the whole problem-solving experience.

3. Scope of the Project

The scope of SkillForge starts from a student landing on the home page and continues till they successfully land a job. It is not just about solving one coding problem. It covers looking at past mistakes, asking the AI when you get stuck, and seeing how many problems you solved each day. This project is for individual students, small coding groups, and even training institutes who want an easy-to-use platform for their students. In the future, it can be expanded to include mock interviews with real humans too.

The scope also includes a feedback loop. When a user solves a problem, they aren't just done. They can go back and see their code history, which helps them remember how they solved dynamic programming or tree problems which are usually very hard to grasp in one go.

4. Feasibility Study

4.1 Technical Feasibility

Is it possible to build this with current tools? Yes. React is excellent for building the UI and managing state. FastAPI is very efficient in handling requests from the frontend and talking to the Python-based AI libraries. Since I am using APIs for LLMs like Groq or Gemini, the heavy AI processing is done on their servers, which means my app can run fast on simple hosting too.

I spent about two weeks just researching libraries. I looked at various PDF converters before choosing ReportLab and looked at many CSS frameworks before deciding to stick with custom CSS to keep it unique. The tech stack is solid and can scale well if more users join.

4.2 Economic Feasibility

Is it worth the money and time? Definitely. The cost of running this project is very low because many tools are open-source. For example, Python and React are totally free. I am using the free/low-cost tiers for databases and AI APIs initially. If this becomes popular, we can start a premium subscription for advanced interview questions, making it a profitable project.

4.3 Operational Feasibility

Can people easily use it? Yes. The interface is simple. We don't have confusing complicated menus. The sidebar has everything, and the dashboard tells you exactly what to do next. Even a beginner who has just started coding can navigate through SkillForge without any tutorial.

5. System Requirements

5.1 Hardware Requirements

To develop this, you need at least 8GB of RAM and a decent processor (i3/i5 or equivalent). For the user, the requirements are very low—any smartphone or computer with a working internet connection and a browser like Chrome or Firefox will work perfectly.

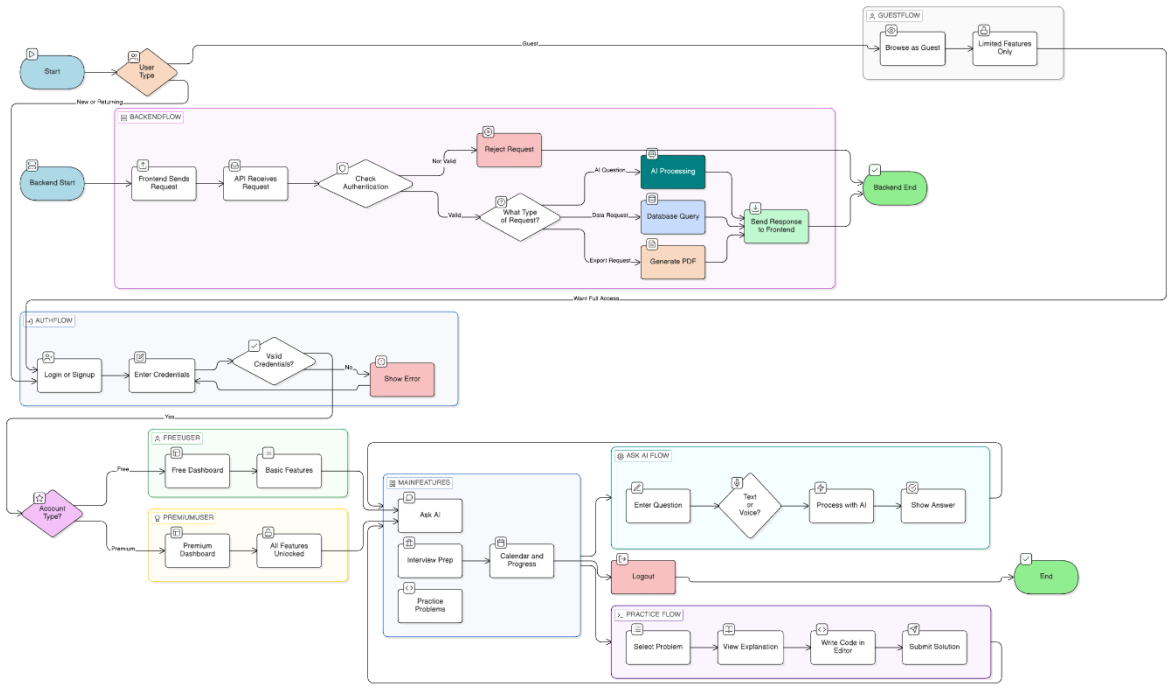
5.2 Software Requirements

The frontend is built on Node.js/React environment. For backend, Python 3.10+ is needed with FastAPI. We also need VS Code for writing code and a modern browser for testing the UI. For the database, we can use something like PostgreSQL or MongoDB during the final launch.

6. System Architecture

The architecture follows a standard 'Client-Server' model. When the user clicks a button (Client/React), it sends a request to the API (Server/FastAPI). The server checks the database to get information about the user or the problem. If the user asks a question, the server sends a request to the AI model. Once the AI answers, the server sends that text back to the React app. We also have a PDF generator module on the server that creates a document and sends it to the user's browser. It's like a team where everyone has a specific job to do.

To put it simply, my project is like a restaurant. React is the 'Waiter' who takes your order. FastAPI is the 'Chef' in the kitchen who cooks the data. The Database is the 'Fridge' where all the ingredients are kept. And the AI is like an expert consultant the chef calls when he needs a special recipe!



7. Technology Stack Explanation

7.1 Frontend: React.js

I chose React because it is component-based. This means I can create a 'Sidebar' component once and use it on every page without rewriting code. It helps in making the web app feel like a mobile app because it doesn't refresh the whole page every time you
click a link.

7.2 Backend: FastAPI (Python)

Many people use Flask or Django, but I picked FastAPI because it is very modern and really fast (as the name says). It also creates automatic documentation for the backend API, which saved me a lot of time while testing the connection between frontend and backend.

7.3 UI Theme: Black & Orange Dark Mode

The black and orange theme is my own choice. Black looks premium and orange gives a punchy feel to the important buttons. I used Swiper.js for the sliders because it has very smooth touch effects for mobile users.

I manually wrote the CSS variables for the colors so that if I want to change the shade of orange later, I only have to change it in one place. I also used Swiper.js to handle the slideout panels and the image carousels in the dashboard.

7.4 AI Integration: Groq/Gemini

Instead of writing my own AI from scratch, I used Groq and Gemini APIs. This gives my small project the power of a giant model that can understand human language and code
logic very well.

8. Module Description

8.1 Login & Signup Module

This is the entry point of the application. I kept it very clean. Users can either create a new account or log in if they already have one. We use security measures to make sure passwords are not saved as plain text. Once you log in, you get a special 'token' which tells the website that you are a verified user.

8.2 Dashboard Module

The dashboard is the main home for the user. It has a sidebar for navigation. In the middle, you see your recent activity. On the right side, there is a small panel showing your current streak and quick stats like 'Problems Solved'. It feels like a command center where you decide what to study today.

8.3 Sidebar & Navbar

The sidebar stays on the left and has icons for Home, Practice, AI Ask, and Settings. The navbar at the top has a search bar where you can quickly find any problem by typing its name. I made sure that on smaller screens the sidebar hides away to save space.

8.4 AI Ask & Voice Support

This is my favorite module. If you have a doubt, you can type it. But I also added a voice input button because sometimes it's easier to talk than to type a long paragraph. The AI takes your text and gives a detailed explanation. It can even explain code line by line like a real mentor would do.

The voice support uses the Web Speech API. I think this is very helpful for users who might have trouble typing or just want to quickly dictate their doubt while looking at the screen.

8.5 Practice Problems Module

This module looks a bit like LeetCode. It lists problems based on categories like 'Arrays', 'Strings', or 'Recursion'. Each problem has a difficulty tag: Easy, Medium, or Hard. This helps the student start with easy ones and build confidence.

8.6 Code Editor & Explanation

The code editor is where the real work happens. It supports multiple languages like Python, Java, and C++. It has syntax highlighting, which means the code looks colorful and easy to read. You can write your solution here and then get feedback.

Beside the code editor, there is a tab called 'Explanation'. If you are stuck and can't figure out the logic, you can open this. It breaks down the logic into simple steps. I didn't want students to just copy the code, so I made sure the explanation teaches the concept first.

8.7 Calendar & Activity Tracking

To keep users coming back, I added a calendar. Every day you solve a problem, that day gets highlighted on the calendar. If you miss a day, you can see the gap in your progress.
It's a great way to stay disciplined.

8.8 Interview Planning by Company

This section organizes problems by famous companies like Google, Amazon, or Microsoft. If a student has an interview next week at a specific company, they can just click that company's logo and see all the questions asked there in the past.

9. Database Design

The database is the backbone of SkillForge. I spent a lot of time mapping out the relationships between different tables.

- **User Table:** Stores basic things like username, email (hashed), password, and whether the user is free or premium. This is the most important table.
- **Task Table:** Records every time a user starts or finishes a problem. It helps us calculate the time taken by the user.
- **History Table:** Whenever a user solves a problem, we save a copy of their code here. This allows them to look back and see how they solved it last month.
- **Progress Table:** This is used for the calendar. It stores daily solved counts so we can draw the activity heat-maps.

I also used indexes on the 'user_id' column to make sure that when a user logs in, the query to find their data happens almost instantly.

8.9 PDF Generation Engine

I noticed that students often like to take prints or save notes. I added a feature using ReportLab in the backend. When a user completes a problem, they can click 'Download PDF'. It generates a nicely formatted document with the problem statement, their code, and the AI's explanation.

8.10 Premium Access Control

To make it more professional, I added a role-based system. Some advanced features, like company-specific mock tests or expert AI hints, are locked for premium users. This is a very common business model in the EdTech industry

10. AI Integration

To keep the platform safe and functional, I defined three main roles:

Quest (Public): Can only see the landing page and some basic information. They cannot solve problems or ask the AI questions.

Free User: Can solve standard problems and use the AI a few times a day. They have access to the basic dashboard.

Premium User: They have full access. They can ask unlimited questions, see company specific questions, and download unlimited PDFs.

11. User Interface Design

The black and orange theme is great for long coding sessions. Black reduces eye strain, and orange is used for 'Call to Action' buttons. I also used shadows and rounded corners to make the UI look soft and modern.

> *Note: I didn't use Bootstrap here because I wanted to learn pure CSS Flexbox. This helped me understand how elements really move on the screen when we resize the browser window.*

11.1 Animations

I used libraries like Framer Motion or simple CSS transitions. When you switch between tabs, the content slides in smoothly. It doesn't just 'appear' instantly, which makes the app feel premium.

11.2 Responsiveness

I used CSS Flexbox and Grid. This means whether you are using a 27-inch monitor or a 6-inch phone screen, the layout shifts to fit perfectly.

12. AI Integration in Depth

In SkillForge, the AI is not just a chatbot. It is deeply connected with the code editor. When a user writes code, we send that code snippet along with the problem name to the backend. The backend then formats a 'Prompt' (like a set of instructions for the AI) and sends it to Groq or Gemini. The AI reads it and replies. I also made sure that the AI gives encouraging messages so the student doesn't feel bad if they make a mistake.

12.1 Benefits of AI for Students

The biggest benefit is speed. If you ask a real person for help, they might take hours to reply. This AI replies in 2 seconds. It also doesn't get tired of answering the same question ten times. It helps students learn the 'Logic' behind the code, not just the 'Syntax'.

12.2 Known Limitations of AI

Sometimes the AI might give a wrong solution if the problem is very new or extremely complex. This is called 'Hallucination'. I solved this by adding a disclaimer that users should verify the answers.

13. Security Considerations

Security is a priority for me. I am using JWT (JSON Web Tokens) for authentication. When you log in, the server gives you a secret key that is stored in your browser. Also, I never reveal my API keys in the frontend code—they are always hidden on the server. For the database, I made sure that user passwords are encrypted so nobody can read them even if they see the database.

I am also planning to add rate-limiting. This means a single user cannot spam the 'Ask AI' button 100 times in one minute, which could crash the server or cost too much money in API fees.

14. Testing the System

I did different types of testing. First, I did 'Unit Testing' to check if small parts like the calculator or the PDF generator were working. Then I did 'Integration Testing' to see if the React frontend and FastAPI backend were talking correctly. Finally, I did 'Manual Testing' where I asked my friends to use the app and tell me where they got confused. This helped me fix many small bugs and improved the flow of the application.

One funny bug I found during testing was that the PDF generator was crashing if the user typed weird symbols like emojis. I had to update the font settings in ReportLab to handle those properly. It taught me that real users will always find a way to break your code in ways you didn't imagine!

15. Limitations of the Project

Since I am building this alone as a project, there are some limits. For now, it doesn't have a video calling feature for mock interviews. Also, the voice input only works well in quiet environments. The AI uses external APIs, so it needs an active internet connection to work. If there is no internet, the AI features won't work.

16. Future Enhancements

In the future, I want to add a 'Team Coding' feature where two students can solve a problem together on the same screen. I also want to add more languages like Swift and Kotlin. Adding a mobile app version on Play Store using React Native would also be a great next step.

I also want to add a 'Leaderboard' where students can see their rank compared to others in their college. A little competition always makes people work harder.

17. Conclusion

Working on SkillForge has been an amazing journey. I learned how to combine many different technologies like React, Python, and AI to build a real product. It taught me that building a web app is not just about writing code; it's about solving real problems for real people. I hope this platform helps students like me to feel more confident while going for their dream job interviews.

Overall, SkillForge is a project very close to my heart. It is the result of many nights of research and coding. I am happy with how it turned out, even though there is always room to add more features.

18. Bibliography / References

- React Official Documentation (react.dev)
- FastAPI Documentation (fastapi.tiangolo.com)
- W3Schools for CSS and HTML styling tips
- StackOverflow for solving many bugs during production
- YouTube tutorials on AI API integration