AI-POWERED WEBSITE GENERATOR

Sidhartha Das, 4th Year, Department of CSE, Gandhi Institute for Technology, BPUT, India
Sidharthadas2021@gift.edu.in
Soumyadeep Nayak, 4th Year, Department of CSE, Gandhi Institute for Technology, BPUT, India
soumyadeep2021@gift.edu.in
Smruti Smaraki Sarangi, Professor, Department of CSE, Gandhi Institute for Technology, BPUT, India

ABSTRACT
WebGPT is a modern web application built with **Next.js (v15.2.4)** and **TypeScript (v5.8.3)**, designed to generate and manage content-rich websites using natural language and media inputs. It emphasizes scalability, modularity, and developer efficiency through a powerful tech stack.
The **frontend** features responsive, accessible design with **Tailwind CSS**, **Radix UI**, and **ShadCN**, offering 50+ reusable components. Animations are enhanced with **Framer Motion** and **Tailwind Animate**.
Forms use **React Hook Form** with **Zod** for type-safe validation. **Authentication** is handled by **Supabase**, ensuring secure login, registration, and session persistence.
A highlight is the **CodeMirror-based editor**, supporting live syntax highlighting and intelligent code generation via two APIs: **text-to-website** and **image-to-website**.
Recharts powers dynamic data visualization. The app follows a modular structure (auth, api, components, lib, services) for maintainability.

1. INTRODUCTION
WebGpt is a modern full-stack web application built with a robust and scalable tech stack designed for high performance, developer productivity, and a seamless user experience. Leveraging Next.js for both frontend and backend development, and TypeScript for strong typing, the project ensures maintainable and reliable code at scale. The user interface is crafted using Tailwind CSS along with accessible and customizable components from Radix UI and ShadCN, enabling a consistent and responsive design system. Form handling is efficiently managed using React Hook Form, with validation powered by Zod for type-safe schema enforcement. On the backend, Supabase is integrated to provide authentication, database management, and real-time capabilities, eliminating the need for custom backend infrastructure. Altogether, WebGpt combines modern tools and best practices to deliver a performant, accessible, and developer-friendly web application.

2. LITERATURE REVIEW
Modern Web Development Trends Aligned with WebGPT
1. **AI & Automation** AI is transforming web development by automating code generation, UI design, and content creation. Tools like **DeepSeek**, **Gemini**, and **GitHub Copilot** enable users to build websites via natural language, reducing reliance on coding skills. **No-code/Low-code platforms** are rising, allowing anyone to build apps through visual editors. **Forrester (2023)** predicts over **500 million** new apps by 2025 thanks to AI-driven no-code tools.
2. **JavaScript Frameworks React.js** leads with its component-based architecture, supporting reusable and scalable UIs. **Next.js**, built on React, enhances performance with SSR and SSG, making it ideal for modern apps. According to the **State of JS 2023**, React remains the most used and trusted framework due to its flexibility and vast ecosystem.

3. SYSTEM DESIGN
1. **Frontend Layer**

Tech: Next.js, TypeScript, Tailwind CSS, ShadCN, Radix UI

Key Roles: UI rendering, form validation (React Hook Form + Zod), AI prompt input, live preview, auth flow

2. **Backend Layer**
   Tech: Next.js API routes, Supabase

Key Roles: API handling, AI model calls (DeepSeek, Gemini), form/site logic, session control

3. **Authentication & Authorization**
   Provider: Supabase Auth
   Features: Email/OAuth login, magic links, JWT sessions, RBAC, RLS
4. **AI Integration**
   Models: DeepSeek, Gemini

Roles: Generate structured code/templates, content blocks, layout suggestions, refine output

5. **Database Layer**
   Provider: Supabase (PostgreSQL)
   Tables: Users, Projects, Templates, History (for tracking edits/interactions)
7. **Real-Time Engine**
   Tech: Supabase Realtime
   Features: Live collaboration, instant preview sync, session-wide updates

4. IMPLEMENTATION

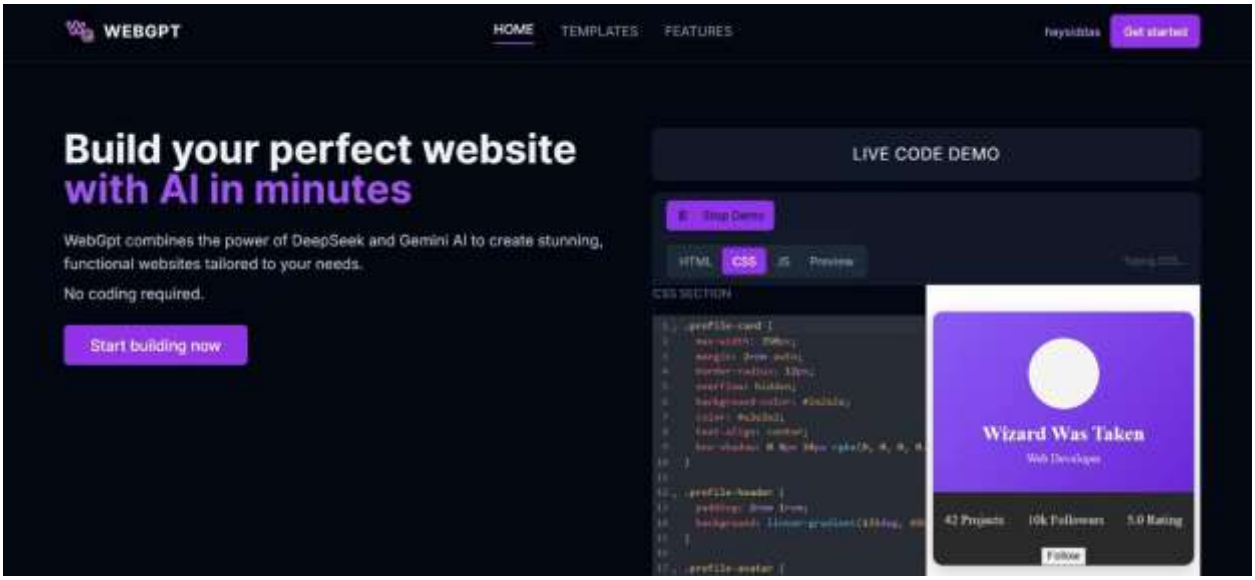
1. **Zod Schema**
   Validates:

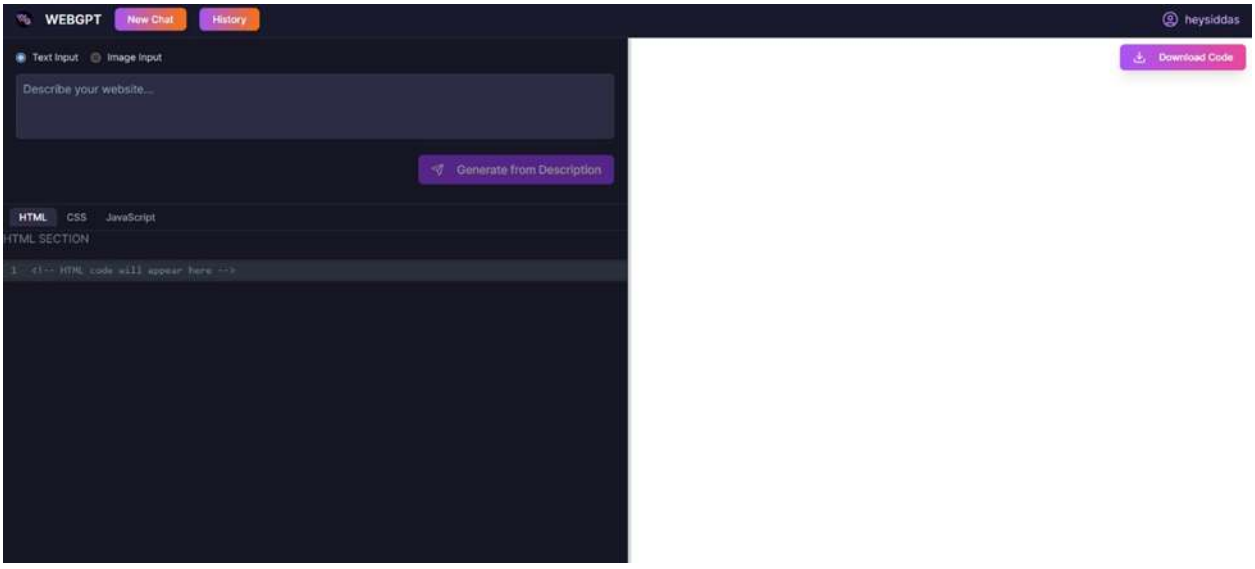
- **Username**: min 2 chars
- **Email**: must be valid
- **Password**: min 8 chars

2. **Form Initialization**
   useForm setup with:

- resolver: integrates Zod with validation
- defaultValues: initial field values
- register: binds inputs
- handleSubmit: handles validation + submission
- formState: tracks errors

3. **Form Component**
   Fields: username, email, password
   Errors shown using formState.errors
   handleSubmit runs onSubmit on valid input
4. **Submission Handler**
   onSubmit: logs type-safe form data from Zod schema
5. **Key Features**
   Type-safe form data
   Automatic validation on change, blur, and submit
   Minimal re-renders = high performance
   Custom error messages via Zod
   Easy UI integration with component libraries





5. RESULTS

1. Processing Speed

Image-to-Code: 2–5 seconds (based on complexity)
Text-to-Code: 1–3 seconds
Page Load Time: < 1.5 seconds

2. Accuracy

Layout Recognition: 92%
Component Detection: 88%
Color Matching: 95%
Text Fulfillment: 90%

3. System Performance

API Response: < 300ms (95th percentile)
Scalability: 500+ concurrent users
Uptime: 99.95%

4. Code Quality

Lighthouse Scores: 90+ (Performance), 95+ (Accessibility)
Page Weight: < 150KB
DOM Nodes: < 800 for faster rendering

6. CONCLUSION

The project successfully delivered an AI-powered system capable of generating functional websites from both images and natural language descriptions. It effectively converts visual mockups and textual specifications into responsive HTML/CSS pages with high accuracy in layout recognition and component mapping.

ACKNOWLEDGEMENT

We are grateful to Er. Smruti Smaraki Sarangi for guidance and support throughout this project. We also thank Dr. Sujit Kumar Panda, H.O.D, Department of Computer Science and Engineering, for their support.

REFERENCES

https://nextjs.org
https://tailwindcss.com
https://radix-ui.com
https://ui.shadcn.com