SOHAM MHATRE
FRONTEND DEVELOPER

CONTACT

9518521612

sohammhatre1907@gmail.com

'Soham' Bunglow,
Plot No-33&34,
Behind Fatima High School
Belavli, Badlapur(West),421503

PROFILE SUMMARY

I am a student specializing in Artificial Intelligence & Data Science, portraying myself as a responsible and organized individual. Eagerly awaiting my first professional experience to bridge theoretical knowledge with practical implementation

EDUCATION

2008 - 2020
Fatima High School
- SSC - 83.60%

2020-2022
LSMT. CHANDIBAI HIMATHMAL MANSUKHANI COLLEGE
- HSC (PCMB) - 51.83%

2022 - Present
KONKAN GYANPEETH
COLLEGE OF ENGINEERING
B.E. Artificial Intelligence & Data Science (4rd Year)
Pursuing

TECHNICAL SKILLS

- C
- My SQL
- Python
- JavaScript
- HTML
- CSS
- MSCIT certified

WORK EXPERIENCE

PROJECTS

- **Sound Detecting System**
  - A sound detecting system captures and analyzes sound using microphones or sensor arrays.
  - It converts analog sound to digital via an ADC, applies filtering and amplification, and then analyzes frequencies and patterns to identify specific sounds.
  - Applications range from security and surveillance to wildlife monitoring and industrial automation.

- **Password Strengthening Site**

  Using HTML, CSS and JavaScript
  - A password-strengthening site uses HTML for structure, CSS for styling, and JavaScript for real-time functionality.
  - HTML: Provides an input field for the password and a display area for feedback.
  - CSS: Styles the page and changes colors based on password strength (e.g., red for weak, green for strong).
  - JavaScript: Analyzes the password based on criteria like length, character variety, and common patterns, and updates the feedback dynamically as the user types.

- **Chat Bot**

  Using Python, NLTK, TensorFlow, NumPy, and JSON:
  - Python (Core Logic): Handles chatbot training, processes user input, and generates appropriate responses using NLP and a neural network.
  - NLTK (Processing): Performs tokenization, stemming, and preprocessing of text to prepare input for the intent classification model.
  - TensorFlow/Keras (Model): Trains and runs a simple neural network that classifies intents based on user input using a bag-of-words approach.

- **Web Chatbot**

  Using HTML, CSS ,JavaScript and Python

  1. Python (Backend)
     - Handles the core chatbot logic — processes user messages, fetches data from APIs (like weather, jokes, or Wikipedia), and generates appropriate responses.
     - Built using Flask, a lightweight Python web framework that serves responses and provides API endpoints.
  2. HTML (Structure)
     - Defines the layout and structure of the chatbot interface.
     - Includes elements like the chat window, input box, send button, voice toggle, and theme switcher.
  3. CSS (Styling)
     - Provides styling for the entire chat interface — sets colors, fonts, spacing, button styles, and light/dark themes.
     - Ensures a smooth and responsive user experience across screen sizes.
  4. JavaScript (Client-side Logic)
     - Handles user interaction without reloading the page — listens for user input, sends it to the server via fetch(), and dynamically displays responses.
     - Also manages features like text-to-speech, theme switching, and copying text.
  5. External APIs (Data Fetching)
     - Integrates various public APIs:
     - OpenWeatherMap – for real-time weather updates
     - JokeAPI – for fetching jokes
     - Wikipedia API – for summaries
     - Currency & Unit Converters – for real-time conversions
     - Deep Translator – for multi-language support
     - 6. Gunicorn (Production WSGI Server)
     - Used to serve the Flask app in production environments like Render.
     - It acts as a bridge between Flask and the web server.

- **Cloud Contact Form**

  Using HTML, Tailwind CSS, JavaScript, Google Apps Script (GAS):

  - HTML (Structure): Provides the markup for a modern, responsive contact form interface that includes input fields for name, email, and message.
  - Tailwind CSS (Styling): Used for utility-first styling, creating a clean, elegant, and responsive design with minimal custom CSS.
  - JavaScript (Client Logic):
  - Captures the form submit event and prevents the default browser behavior.
  - Sends the form data to a server endpoint using fetch() and FormData.
  - Provides user feedback through alert() based on the submission result.
  - Google Apps Script (Backend Logic):
  - Handles HTTP POST requests from the web form.
  - Extracts submitted data (name, email, message) and appends it to a connected Google Sheet along with a timestamp.
  - Acts as a serverless backend, making this solution free to host and easy to deploy via Google Workspace.

.

- **Responsive Landing Page**

  Using HTML, CSS, and JavaScript:

  1. HTML (Structure):
     - Provides the basic layout of the landing page, including navigation bar, hero section, feature highlights, and call-to-action areas.
     - Ensures semantic structure for accessibility and SEO.
  2. CSS (Styling):
     - Handles the visual design with responsive layout adjustments for various screen sizes.
     - Implements modern design principles such as flexbox and media queries to ensure fluid grid systems.
     - Custom fonts, button styles, background gradients, and hover effects enhance the UI.
  3. JavaScript (Interactions):
     - Adds basic dynamic behavior to elements, such as toggling menus or scroll-based effects (if applicable).
     - May include enhancements for smooth scrolling or responsive navbar toggles on smaller screens.

- **Stopwatch Web Application**

  Using HTML, CSS, and JavaScript:

  1. HTML (Structure):
     - Builds the visual interface of a stopwatch with controls (Start, Stop, Reset) and a display area showing elapsed time in hours:minutes:seconds:milliseconds.
  2. CSS (Styling):
     - Applies a clean and user-friendly design, possibly using neumorphic or modern UI styling. It ensures responsive layout and visual clarity of buttons and timer display
  3. JavaScript (Logic):
     - Implements the stopwatch functionality: starting, pausing, resetting, and continuous time calculations.
     - Updates the UI in real time using setInterval or requestAnimationFrame.
     - Enhances accuracy and responsiveness, as noted in the v1.2.2 release with bug fixes and timing improvements

- **Tic-Tac-Toe Game**

  Using HTML, CSS, and JavaScript:

  1.HTML (Structure):
     - Provides the basic structure for the game, including the main menu (index.html) and the game screen (game.html). It defines the game board, buttons, and text areas.
  2. CSS (Styling):
     - Styles the visual elements of the game. This includes the layout of the game board, colors, fonts, button appearances, and the implementation of a theme (dark/light mode). It also handles the styling for the winning line animation.
  3. JavaScript (Logic & Interactivity):
     - Core Game Logic (script.js): Manages the primary game flow, such as handling player moves, switching turns, and checking for win or draw conditions. It also updates the scoreboard and handles user interactions like button clicks (restart, undo, etc.).
     - AI Opponent (ai.js): Implements the artificial intelligence for the computer opponent. It includes different difficulty levels ("easy", "medium", "hard") by using a combination of random moves and the Minimax algorithm to determine the optimal move.

- **Live Weather App**

  Using HTML, CSS, JavaScript, and Public APIs:

  1.HTML (Structure):
  - Provides the webpage's framework, including the search bar, buttons for toggling units and themes, and dedicated sections to display weather data like temperature, wind speed, and the hourly forecast.

  2. CSS (Styling):
  - Styles all visual aspects of the application. It manages the layout, fonts, colors, and dynamic background images. It also implements both a dark and a light theme using CSS variables for a better user experience.

  3. JavaScript (Logic & Interactivity):
  - API Integration: Fetches weather data, air quality information, and forecasts from the OpenWeatherMap API. It also retrieves dynamic background images from the Pexels API based on the current weather conditions.
  - Dynamic Content: Updates the webpage in real-time with the fetched data, such as displaying the current temperature, humidity, wind direction, and a multi-hour forecast.
  - User Interaction: Handles events like city searches, toggling between Celsius and Fahrenheit, and getting the user's location via the browser's Geolocation API.

- **IoT-Based Air Pollution Monitoring System**

  Using an ESP-32, various sensors, and the Blynk IoT platform, this system monitors environmental conditions in real-time.

  1.ESP-32 Microcontroller (Core Logic & Connectivity):
  - This low-cost microcontroller serves as the central processing unit of the system. It is programmed using the Arduino IDE to read data from the connected sensors, process it, and use its built-in Wi-Fi to transmit the information to the Blynk cloud platform.

  2.Sensors (Data Collection):
  - MQ-135 & MQ-2 Gas Sensors: These sensors detect the presence and concentration of various harmful gases and pollutants, such as carbon monoxide (CO), carbon dioxide (CO2), and smoke.
  - Flame Sensor: An LED indicator is activated when a fire is detected by the system.

  3. Blynk Cloud (IoT Platform & User Interface):
  - This cloud-based service acts as the backend and user interface for the project. It receives and stores the data sent from the ESP-32, allowing users to remotely monitor air quality levels from anywhere via a mobile device or computer. The system is also configured to send alerts if pollution levels exceed safe thresholds.

  4. LEDs & Buzzer (Local Alerts):
  - The system includes hardware for on-site alerts. LEDs in different colors (green, yellow, red) provide an immediate visual indication of the current air quality status , while a buzzer sounds an alarm during critical events like a fire.

.

- **Personal Finance Dashboard**

  Using HTML, CSS, JavaScript, Firebase, Chart.js, and jsPDF:

  1.HTML (Structure):
  - Provides the structure of the finance dashboard including forms, charts, dropdowns, and tables for transaction history and summary.

  2. CSS (Styling):
  - Styles the dashboard layout, implements light/dark theme support, and ensures responsiveness across devices using Bootstrap.

  3. JavaScript (Core Logic):
  - Handles transaction form input, calculates income, expenses, and savings.
  - Filters and visualizes data based on month/year selection.
  - Renders Pie, Bar, and Line charts to show category-wise breakdown and daily trends.
  - Adds export functionality for transactions (PDF/CSV).
  - Manages dark mode toggle and dynamic UI updates.

  4. Firebase (Authentication + Firestore):
  - Firebase Auth: Enables user signup/login and ensures data is tied to individual users.
  - Cloud Firestore: Stores and retrieves user-specific transaction data, allowing persistent sync across sessions.

  5. Chart.js (Visualization):
  - Generates interactive charts for income/expense categories, daily spending trends, and summaries.
  - Custom tooltips show category-level data for each date.

  6. sPDF (Export Feature):
  - Converts the transaction history into downloadable PDF format with clean layout and text alignment.

OTHER SKILLS

- Drawing
- Playing **Tabla**
- Skating

LANGUAGES

English
Marathi
Hindi

HOBBIES

- Reading Books
- Tabla
- Gaming

EXTRA CURRICULAR ACTIVITIES

TECHNICAL

MSCIT Certification
- 2017 with 91 marks
- Maharashtra State Board of Technical Education (MSBTE)
- Acquired knowledge in mention key skills or areas like computer basics, MS Office, internet usage, etc.

NON TECHNICAL

- Tabla Performance: Successfully passed a formal Tabla exam with an First Class at intermediate level, demonstrating strong proficiency and dedication to classical music.
- Participated in various roller skating events and competitions, demonstrating agility, coordination, and dedication.