Aspireit

Full Stack Engineer Internship

We are excited to announce a 3-month remote internship opportunity for a Full Stack Developer at Aspireit. This isn't just any internship—it's a unique chance to work directly alongside our founder on cutting-edge AI-driven products, placing you at the forefront of the industry's shift towards AI innovation.

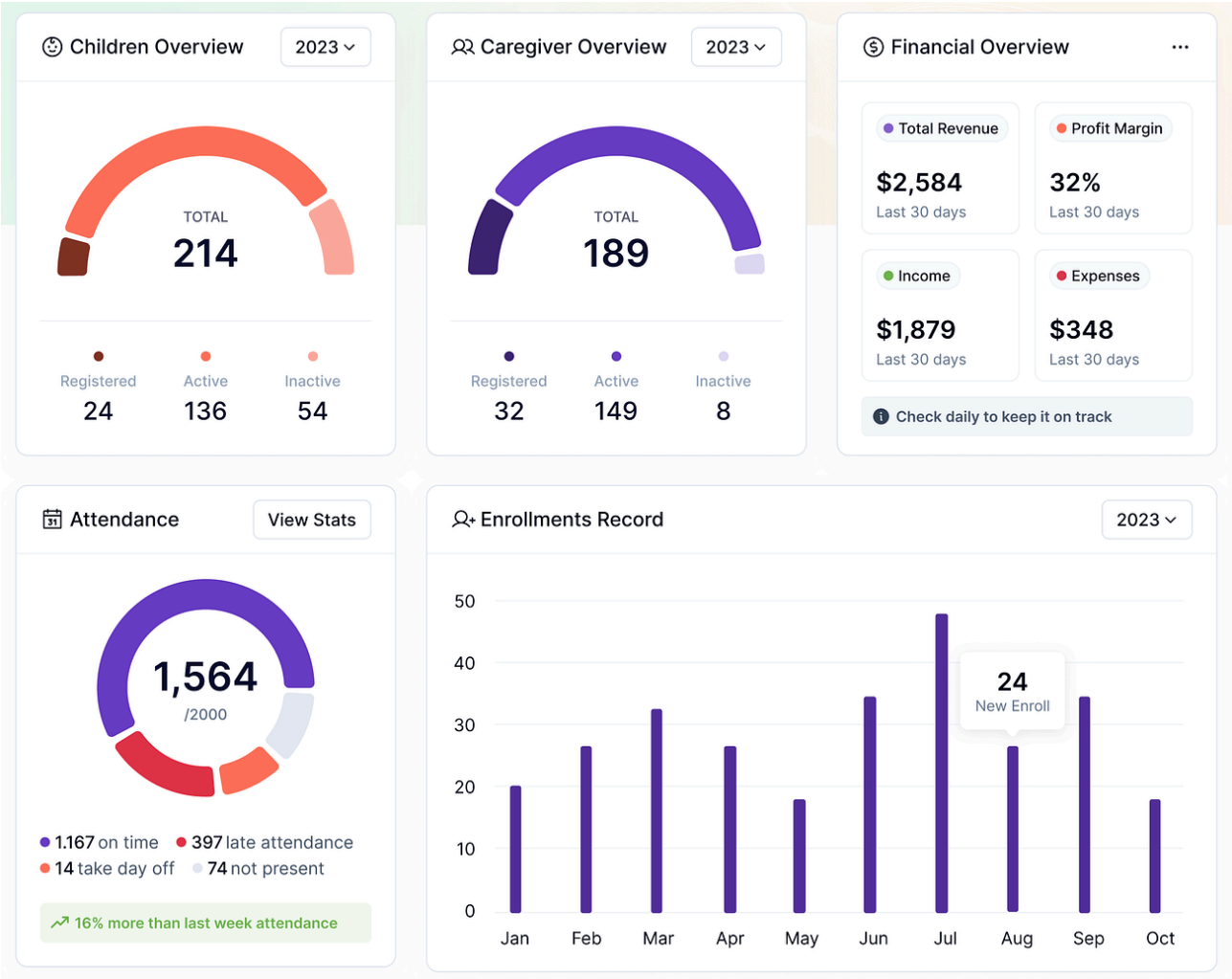
During this internship, you will be a key member of our team, **contributing to the development of three innovative AI products. You'll gain invaluable experience by working closely with our AI engineers, integrating AI models on the backend, and bringing them to life on the front-end.** This hands-on involvement will give you the practical exposure needed to build next-generation products and prototypes.

Important Note: You are encouraged to use ChatGPT or any other LLM for code generation. As a forward-thinking company, we are focused on efficiency and innovation, and we'd like to assess how quickly you can complete and submit tasks.

Additionally, if you are selected for the internship, you will gain valuable exposure to ADVANCED AI TOOLS FOR CODE WRITING AND GENERATION, providing you with essential industrial knowledge. As the industry evolves, this experience will position you at the forefront of where technology is heading.

The task is on the next page.

Assignment Overview:



1. Backend Development Evaluation Task: Dashboard Creation

Objective:

Develop a robust, scalable, and secure backend system using **Flask, Express.js (Node.js)**, or **Django** that communicates effectively with a React frontend dashboard. The dashboard provides an overview of various metrics related to children, caregivers, finances, attendance, and enrollments. Your goal is to build RESTful APIs that support the management of these metrics, integrating business logic and a machine learning model to enhance data insights and predictions.

Task Requirements:

1. Set Up Backend Environment:

- **Framework**: You may use **Flask (Python)**, **Express.js (Node.js)**, or **Django (Python)** to set up the backend server.
- **RESTful APIs**: Develop RESTful APIs to manage data points related to children, caregivers, attendance, finances, and enrollments.
- **Security**: Ensure all APIs are secure, efficient, and follow RESTful best practices.

2. Design and Implement RESTful APIs:

Create the following APIs to manage the data points:

- **Children Overview API**:
  Handle the total count of children, and break them into registered, active, and inactive categories.
- **Caregiver Overview API**:
  Similar to the children API, manage caregiver totals and categorize them into registered, active, and inactive groups.
- **Financial Overview API**:
  API to calculate total revenue, profit margin, income, and expenses over the last 30 days.
- **Attendance Tracking API**:
  Track attendance metrics such as on-time, late, day-off, and not-present data points.
- **Enrollments Record API**:
  Manage and provide the number of new enrollments each month, dynamically updating with new data.

Ensure that data modifications (additions, updates, or deletions) are reflected in real-time and accurately retrieved by the React frontend.

3. Database Management:

- **Database**: Use a relational database such as **PostgreSQL**, **MySQL**, or **MongoDB** to store structured data.
- **Schema Design**: Create tables for the following:
  - Children (ID, Name, Status [Active/Inactive])
  - Caregivers (ID, Name, Status [Active/Inactive])
  - Attendance (ID, Status [On-time, Late, Absent])
  - Financial Records (Revenue, Expenses, Profit Margin)
  - Enrollments (Date, New Enrollments Count)
- Ensure the schema is designed to support efficient data retrieval, storage, and real-time updates.
- **Justification**: In your submission, provide a brief explanation of why you chose the specific database and how your schema supports efficient data handling.

4. Implement Backend Business Logic:

- **Children and Caregiver Overview**: Develop logic to calculate totals, as well as active and inactive counts.
- **Financial Overview**: Implement logic to dynamically calculate total revenue, profit margin, income, and expenses over the last 30 days.

- **Attendance Metrics**: Calculate counts for on-time, late, day-off, and not-present children.
- **Enrollment Records**: Track and calculate new enrollments dynamically, updating the totals for each month.

5. Security and Authentication:

- **JWT Authentication**: Implement user authentication and authorization using **JWT (JSON Web Tokens)** to secure the backend.
- **Endpoint Protection**: Ensure all endpoints are protected, allowing access only to authenticated users.
- **Security Measures**: Implement protections against common vulnerabilities such as SQL injection and XSS attacks.

6. Error Handling and Validation:

- Implement robust error handling and input validation across all API endpoints.
- Provide meaningful error messages and ensure the system manages API failures gracefully.

7. Testing:

- Write unit tests for API endpoints and business logic.
- Test the integration of the ML model using tools like **Postman** to validate the correct functionality of the backend system.

8. Additional Instructions:

- **Backend Framework**: Use **Flask**, **Express.js**, or **Django** for the backend, and ensure the APIs integrate seamlessly with the React frontend.
- **Documentation**: Provide clear documentation for your APIs, explaining how they interact with the frontend.
- **README**: Include setup instructions for running the backend locally in a README file.
- **Design Justification**: Include reasoning for the architectural and design choices made during development.
- **Caching**: Implement caching strategies (e.g., Flask-Caching with Redis, or caching mechanisms in Express.js/Django) to optimize performance and handle real-time updates.

2. Frontend Development Evaluation Task: Dashboard Creation

Objective:

Develop a responsive and interactive dashboard frontend using **React** that provides an overview of various metrics related to children, caregivers, finances, attendance, and enrollments. The dashboard should align with the provided design reference and dynamically reflect data, including predictions from a simple machine learning model integrated into the backend.

Task Requirements:

1. Develop the Dashboard:

- Use the attached screenshot as a reference for building the dashboard's layout and design. Ensure the visual elements, such as charts, cards, and overview components, closely match the reference.
- Implement a responsive design that provides an optimal viewing experience across different devices (desktops, tablets, and smartphones).

2. Choose Your Tools and Libraries:

- Use **React** as the primary frontend framework.
- Decide on additional tools or libraries for state management, API handling, and data visualization that best suit the requirements (e.g., Redux for state management, Axios for API calls, Chart.js or D3.js for data visualization).
- Explain your choices and how they contribute to the application's scalability, maintainability, and performance.

3. Implement Dynamic Data Handling:

- Use **Axios** or **Fetch API** to make asynchronous API calls to the backend.
- Ensure the dashboard can fetch and display data from the backend seamlessly and dynamically, including real-time data changes and predictions from the machine learning model.

4. Reflect Machine Learning Predictions:

- Display the output of the machine learning model (e.g., predicted future enrollments) prominently on the dashboard.
- Implement components that update dynamically to show ML-based predictions in real-time, such as a dedicated card or graph that displays predicted enrollments or attendance trends.

5. Design for Performance and Scalability:

- Optimize the application for performance by implementing techniques such as lazy loading, code splitting, and caching.
- Minimize re-renders and ensure smooth interactions throughout the dashboard.

6. Implement Animations:

- Incorporate animations to enhance the user experience, such as transitions between different data states, hover effects, and smooth loading animations.
- **Bonus:** Candidates who create dynamic animations (e.g., animated updates for machine learning predictions or real-time data changes) will be given priority consideration.

• Bonus Points:

- Implement a dark mode toggle for the dashboard.

- Add additional animations or transitions for data updates and user interactions, especially those related to ML predictions.
- Provide multilingual support for the dashboard.
- Create a dashboard customization feature (e.g., allowing users to rearrange or hide components).

- **Submission Guidelines:**

  - Provide a GitHub repository link containing your code.
  - Include a README file with instructions to set up and run the frontend locally.
  - Ensure the repository is well-structured, with clear commit messages and organized code.

- **Reference:**

  - **Attached Screenshot:** Use the attached screenshot as the primary visual reference for designing the dashboard. Ensure your implementation closely follows the layout, styles, and design elements depicted.

3. Integration and Functionality:

1. Authentication and Security:

- Implement secure user authentication using JWT tokens obtained from the backend.
- Ensure authenticated routes are handled properly, and access is restricted for unauthorized users.

2. API Documentation:

- Create detailed API documentation using tools like Swagger or Postman.
- Include endpoint descriptions, request and response formats, parameters, and example usage to help developers understand and utilize the backend services effectively.

3. Code Quality:

- Maintain high code quality by adhering to best practices, such as consistent coding standards, modular design, and efficient algorithms.
- Use linting tools, style guides, and automated code analysis to enforce quality standards.
- Conduct regular code reviews to provide feedback, catch errors, and ensure the codebase remains clean, maintainable, and scalable.

4. Testing:

- Write unit tests for each component using a testing framework (e.g., Jest for React).
- Test the application's responsiveness, functionality, and compatibility across different browsers.

Submission Requirements:

- **Provide clear documentation** outlining the architecture, API endpoints, and instructions for running the application locally.
- **Submit the codebase** via a version control system (e.g., GitHub) along with instructions for reviewing the code.
- **After completing all subtasks**, compress your project files into a ZIP folder and include a README.md documenting your approach and any setup instructions.

Submission Guidelines:

- **Deadline:** Your completed assignment should be submitted within **3 days** from the date of this notification.
- **Format:** Compress your project files into a **ZIP** folder and include a README.md with your approach and setup instructions.
- **How to Submit:** Share your ZIP file via the Google form link:

https://forms.gle/2Qn7CKmbJTRqjRYD7

(Please Note: No other form of submission will be accepted; only submit the task via the Google form link provided.)

- **Include your full name and contact details** in the README.md.

Evaluation Criteria:

1. **Code Clarity and Quality:** Evaluate the readability of code, adherence to best practices, and the inclusion of meaningful comments.
2. **API Integration and Deployment Skill:** Assess the effectiveness of infrastructure provisioning and configuration using automation tools.
3. **Database Integration Skill:** Evaluate the implementation of automated testing and monitoring solutions.
4. **Analytical Thinking:** Assess adherence to security best practices and compliance requirements.
5. **Execution:** Ensure the pipeline executes flawlessly, achieving the desired outcomes as per the task description.

Terms of Engagement:

- **Duration:** A 3-month tenure
- **Engagement:** Flexible scheduling.
- **Work Environment:** **Fully Remote**
- **Remuneration -** Below are the terms of engagement to ensure clarity:

    - **Internship Offer Decision:**
        1. Your eligibility for the internship role will be determined based on the quality of your submission and the outcomes of the interview process.

2. The final decision on whether to offer you an internship will be made by the company.
3. This internship is an opportunity for profound learning and professional skill development. It is an unpaid position, intended to provide you with valuable experience and the chance to demonstrate your abilities.
4. We emphasize that this is an unpaid internship, as clearly mentioned in the **job description** and reiterated here to avoid any confusion or misunderstanding.
5. **Benefits:**
   - **Certificate of Completion**: Receive a certificate upon successful completion of the internship, showcasing your achievement.
   - **Letter of Recommendation:** Earn a personalized letter of recommendation based on your performance, enhancing your professional profile.
   - **Referrals:** Gain referrals to three well-connected companies upon successful completion of them internship, expanding your professional network and job opportunities.

Note: We believe in maintaining open communication throughout your internship. This experience is designed to help you gain practical knowledge and showcase your skills, with the potential for a full-time role based on your performance.