Development Report: Purdue ECE Faculty Finder

By: Hitaesh Saravanarajan, hitaesh.sara@gmail.com , hsarava@purdue.edu

Link: https://purdue-faculty-finder.vercel.app/

The objective of this project was to build a website that allows users to search Purdue ECE faculty by name or research area, with partial matches supported. The system also needed to return each faculty member's website and research description, and it had to be deployed on a free cloud tier.

Approach

Tech Stack:

I selected FastAPI for the backend and Next.js for the frontend. My previous experience with both technologies allowed fast development and simplified deployment. For the database, I chose SQLite because the ECE faculty directory contains only about 134 entries, which I verified by scraping the official site. A lightweight embedded database avoids the overhead of managing an external service, keeps the system portable, and allows the repository to remain self-contained.

Web Scraper:

My first step was to write a dedicated scraper that collects faculty names, profile URLs, personal webpages, and research areas. The scraper produces a JSON file that serves as the canonical data snapshot for the application. During startup, the FastAPI service reads this JSON and initializes the SQLite database. This approach keeps initialization fast, deterministic, and deployable without extra infrastructure.

User Interface Design:

The user interface was designed around a single-page layout for clarity. As the user types in either the name field or the research field, the frontend issues debounced queries to the backend, avoiding unnecessary request bursts. The backend responds immediately due to the small SQLite dataset, so the experience feels like a real time lookup. When a search returns exactly one result, the UI automatically opens the detailed view for that professor to meet the project specification. In practice, this can feel slightly abrupt, and the behavior can be toggled if needed based on usability feedback.

Other Considerations:

I considered additional features but did not include them to avoid deviating from the assignment. One idea was a combined search across both name and research interests, which would require a small extension of the API. Another option was listing all faculty alphabetically on load, but this risked cluttering the single page interface.

Additional Enhancements

Data Updates:

The application supports live data updating through a protected `/update` endpoint. Given admin credentials, the system automatically re-scrapes the faculty directory and refreshes the database. For this implementation, the admin username is `eceadmin` and the password is `ecefaculty1234!`. These are fixed for the MVP; a future improvement would be a full authentication system with role-based access

Testing and Security:

I also added unit tests for backend logic, including the scraper, authentication, and database ingestion. Endpoint behavior was tested continuously with Postman during development. The backend uses SQLAlchemy ORM, which provides structured queries and reduces risk of SQL injection.

Conclusion

Overall, I believe this project meets the full requirements of the assignment, is deployable on a free tier, and is structured for future expansion with minimal effort. I had a lot of fun making it and would love the opportunity to work on similar project for the ECE Corporate Partnerships Office. I am excited for this website to be reviewed, and any feedback and questions are welcome at my personal or school email!