**Final Project**

Project Title

**Resources to Enhance Academic Quality for Higher Education Institutions in the United States**

Team members

**Venkata Akhil Kumar Gummadi, Du Huynh, Mary Nelson**

# Application Content (links):

Application: <https://duhuynh.us/final_project/>

Video <https://youtu.be/G_pPrsNsKZU>

# Files: <https://github.com/huynhthedu/Final_project>

# **Purpose/Audience:**

The purpose of this project is to analyze and enhance the understanding of academic resource distribution—particularly instructional staff and library assets—across higher education institutions in the U.S. The project leverages structured data from the Integrated Postsecondary Education Data System (IPEDS) to support data-driven decision-making in academic planning, resource allocation, and policy development.

The intended audience includes:

* Leadership and administrative decision-makers at colleges and universities
* Education policymakers
* Institutional researchers
* Academic planners aiming to benchmark or improve institutional performance
* Data analysts in higher education institutions

The system provides them with visualizations, dashboards, and query-based insights that compare instructional and library resources across institutions and sectors, supporting evidence-based strategic decisions.

# **App Functionalities:**

**Database**

The data is hosted in Postgres-DigitalOcean. There are three data sources: IPEDS for main data imported to Postgres with Pandas; dimension tables created with csv file and imported to Postgres; and Models in Django connected with Postgres. The conceptual diagram/schema for database is as figure 1.

**Figure 1. Conceptual Diagram/Schema**

A computer screen shot of a computer

AI-generated content may be incorrect.

*Sources: Authors’*

* Live IPEDS Data Extraction: The backend downloads and extracts ZIP-compressed CSV files from IPEDS (e.g., hd2023.csv, al<year>.csv, sal<year>\_is.csv) to process data for the years 2019 to 2023.
* Instructional and Library Analytics:
  + Library data includes metrics like physical books, e-books, media, and serials.
  + Instructional data includes faculty categories, duration types, and gender identifiers.
* Dimension Table Integration: Lookup tables such as LibraryD, Instructional\_D, and Tenure\_D are used to enrich raw data with human-readable descriptors.
* State and Sector Mapping: Institution data is mapped to U.S. states and sectors (e.g., public, private), using dimension tables (state, sector) to enable grouped analysis.

Two views (faculty\_view, and library\_view) were created as in Figure 2.

**Figure 2. Views in Database**

A screenshot of a computer program

AI-generated content may be incorrect.

*Source: Authors*

Just library\_view was used for demonstration.

**Application**

The output is a webpage for users to create, edit and delete data (evaluation and grading higher education institutions); and reference to visualizations for library resources such as number of collections, and expenditures for library.

The web application provides an interactive platform to explore and manage data related to instructional and library resources. It integrates real-time data extraction, local database mapping, and dynamic visualizations. The core functionalities include:

* Institution Selector: Users can select a U.S. college or university from a dropdown. The default view loads Indiana University - Bloomington, but users can dynamically change the selection.
* Institution CRUD Features:
  + Create: Users can add new institutions using a form that supports dropdowns for UNITID and institution name.
  + Edit: Existing institutions can be updated, including fields: comments/evaluations, and grade (grade is selected from a 1–7 dropdown).
  + Delete: Institutions can be deleted individually, removing them from the list.
  + These operations are backed by Django forms and templates for user interaction and validation.
* Interactive Dashboard (final\_project.html): A central HTML page renders the processed results for the selected institution using visual tables or charts, ready for expansion into graphical output (e.g., using Matplotlib or Chart.js).
* Auto-refresh on Selection: When an institution is selected, the dashboard automatically refreshes to show the updated, institution-specific instructional and library data.

Main steps are as below:

[The main page](https://duhuynh.us/final_project/) allows users to choose to create new evaluations, edit evaluations, visualizations for individual institutions, and visualizations for aggregation level (Figure 3).

**Figure 3. Main Page**

A screenshot of a computer

AI-generated content may be incorrect.

*Source: Authors’*

Choosing Evaluation>> will pop up [create evaluation page](https://duhuynh.us/final_project/create-evaluation/) (Figure 4). It will allow users to choose an institution to comments/evaluate, then give grade with Likert 1-7.

**Figure 4. Create Evaluation**

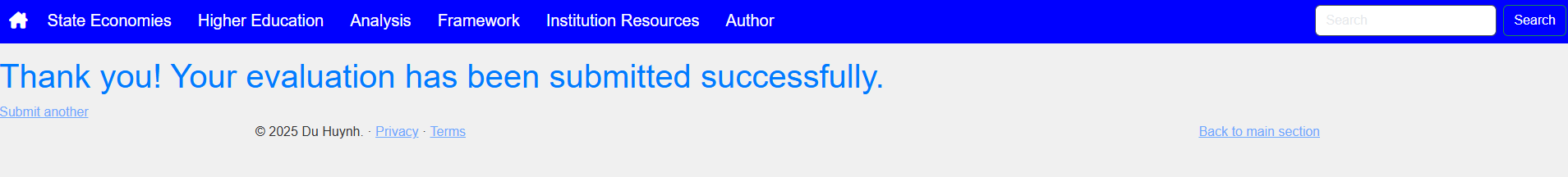
A screenshot of a computer

AI-generated content may be incorrect.

*Source: Authors’*

After submitting, it leads to the page in Figure 5.

**Figure 5. Thank you page**



*Source: Authors’*

It allows users to submit another or back to the main section.

Choosing Edit evaluation in the main section will lead to List of Evaluation (Figure 6).

**Figure 6. List of Evaluation**

A screenshot of a computer

AI-generated content may be incorrect.

*Source: Authors’*

Users can choose to edit or delete any row. If delete, it will lead to a confirmation message. If edit, it will lead to Figure 7.

**Figure 7. Edit Evaluation**

A screenshot of a computer

AI-generated content may be incorrect.

*Source: Authors’*

Users can go with similar steps as creating evaluations.

Choosing Visualization in the main section will lead to Figure 8.

**Figure 8. Institution Resources**

A screenshot of a computer

AI-generated content may be incorrect.

*Source: Authors’*

Users can choose institutions, years and items.

At the main section, users can choose “Exploring all institutions” to Figure 9.

**Figure 9. Dynamic Visualization for Aggregation Level**

A screenshot of a computer

AI-generated content may be incorrect.

*Source: Authors’*

# **Link and Challenges, Issues Faced:**

<https://duhuynh.us/final_project/>

During the development and deployment of the Django-based web application with Postgres SQL in DigitalOcean, several technical and configuration-related challenges were encountered. Some of these were specific to our project, while others are common when working with Django and web-based systems in general.

Since our database hosting limitation, we just demonstrated the library data while both instructional and library data have been extracted, transformed and loaded in Postgres.

#### 1. **Initial Environment Setup**

* Installing Django and setting up the virtual environment required configuring correct Python versions, resolving permission issues on macOS, and managing package installations using pip within isolated environments.
* File encoding issues occurred during CSV imports (e.g., utf-8 vs. latin1), causing runtime exceptions while loading IPEDS datasets into memory.

#### 2. **Model Integration and Migration Conflicts**

* Updating or refactoring models after initial development led to migration mismatches and occasional schema conflicts.
* Since some tables (e.g., LibraryD, Instructional\_D) were externally managed, Django’s managed=False models required special handling to prevent migration attempts on existing database tables.

#### 3. **Data Consistency**

* Mapping external IPEDS datasets to internal dimension tables required careful data cleaning and transformation to maintain referential integrity.
* Dropping or reloading large sets of CSV data increased the risk of duplication or foreign key mismatches during testing phases.

#### 4. **User Interface Integration**

* Dynamically populating dropdowns for institution fields (e.g., UNITID, INSTNM, Grade) required customizing Django forms and passing querysets with unique values.
* Displaying visual elements like charts or structured tables required coordination between the backend data processing logic and front-end template rendering.

#### 5. **Deployment Issues**

* Deploying to a public host (e.g., duhuynh.us) involved configuring static files, database connections, and environment variables correctly for production.
* Common Django deployment issues encountered included:
  + Static files not rendering correctly without running collectstatic.
  + Host/domain validation errors due to missing ALLOWED\_HOSTS in Django settings.
  + Debug mode needing to be disabled for production security.
  + Database permissions and path differences between local development and server environments.

#### 6. **Security and Access**

* Securing admin access and protecting against unauthorized data edits required setting up Django’s authentication and permission mechanisms.
* While not part of the project scope, preparing the system for production-level deployment highlighted the need for secure SSL setup and HTTPS enforcement.

# **Technical Development and Contributions:**

***Technical development***

The project was developed using the Django web framework in Python and focused on building a scalable, data-driven application that integrates external datasets, structured models, and user-facing CRUD functionalities. The backend connects to a SQL-based database, where normalized tables such as Institution, Library, Instructional, and supporting dimension tables (LibraryD, Instructional\_D, Tenure\_D, State, and Sector) are defined using Django’s ORM. Data was fetched from IPEDS ZIP-compressed CSV files and processed using Pandas to extract, transform, and load institutional and instructional metrics.

Custom Django management commands were written to automate data ingestion from the downloaded datasets into the application database. The front-end was built using Django templates, rendering user-friendly forms and views that allow institution-level record creation, editing, and deletion. Dropdowns and field validations were implemented through Django Forms to ensure data consistency and improve user experience.

The application supports dynamic filtering and live dashboard updates based on user-selected institutions. Lookup tables were used to enrich data with readable labels and categories, ensuring meaningful visualization and reporting. The entire application was deployed to a public-facing host, where static assets and database configuration were adjusted for production-readiness.

***Contributions***

Du Huynh: Database design, implementation, data load processing, and architectural design to implement the Django framework for overall project.

Venkata Gummadi: Create, delete, and modify operations on the institutional table by implementing HTML forms and Python; leading the video production.

Mary Nelson: visualizations using Python and Django framework.

# **References:**

**Materials of the course APPLIED DATABASE TECHNOLOGIES at the Luddy School, Indiana University**

**Integrated Postsecondary Education Data System (IPEDS) – U.S. Department of Education**  
Source of institutional, instructional, and library data used in the project.  
 <https://nces.ed.gov/ipeds/datacenter/>

**Django Documentation – Web Framework for Perfectionists with Deadlines** <https://docs.djangoproject.com/en/stable/>  
Referenced for implementing models, forms, admin interfaces, and custom management commands.

**Pandas Documentation – Python Data Analysis Library**https://pandas.pydata.org/docs/  
Used for loading, transforming, and processing data from CSV and ZIP files.

Hu, Y., & Kuh, G. D. (2002). Being (Dis)engaged in Educationally Purposeful Activities: The Influences of Student and Institutional Characteristics.  
Research in Higher Education, 43(5), 555–575.  
Provided context for the importance of institutional resource tracking in academic performance analysis.

Zhang, Y. (2010). The Impact of Academic Library Resources on Students’ Learning.  
Journal of Academic Librarianship, 36(3), 229–236.  
Informed our understanding of how library services contribute to institutional effectiveness.

**Gunicorn & Nginx Deployment Guides**https://docs.gunicorn.org/en/stable/deploy.html  
https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu  
Used to assist with production deployment, static file configuration, and application server setup.

# **Reflections**

This project provided in-depth learning experience in building a full-stack data-driven web application using Postgres and Python (Pandas and Django). One of the most valuable takeaways was understanding how to integrate real-world datasets — specifically, the IPEDS education data — into a relational database and use it to generate meaningful institutional insights. The process of mapping external data sources to normalized database structures taught us the importance of schema design, data validation, and referential integrity. We have learned the key issues below:

* Combinations of Python and SQL to extracting online zip files and transform to Postgres. We have learned skills in dealing with large and sophisticated files, eliminating odds such as null or non-number for values.
* Setting up keys (primary and foreign to enhance data integrity); and creating views in Postgres to streamline the processing.
* Combinations of Django and JavaScripts to create interactive visualizations.

The **most challenging aspect** of the project was handling large-scale data ingestion from ZIP-compressed CSV files across multiple years (2019–2023), while ensuring consistency and correctness in the merged datasets. Initially, we had planned to extract the main data directly from zip files to Django to avoid using database storage. However, it did not work as the process needed huge ram capacity. Furthermore, our plan to use both instructional and library data did not work with the same issue above. Therefore, we only demonstrated with the library data. Another issue is that automating this process using Pandas and custom Django management commands helped us improve efficiency, but required careful handling of encoding issues, missing values, and field mismatches. Additionally, deploying the application to a live server introduced new challenges around environment configuration, static file handling, and security settings — all of which deepened our understanding of backend deployment.

The **most interesting part** was designing interactive forms and CRUD operations for institution management. Enabling users to dynamically add, edit, and delete institution data — along with validation, dropdowns, and UI feedback — made the application more functional and user-friendly. It gave us hands-on experience in combining Django views, forms, and templates effectively.

If we had more time, we would enhance the current application by:

* Figuring solutions for large data sets with limited/cost effective options.
* Adding graphical data visualizations (e.g., charts for year-over-year trends)
* Enforcing stronger database-level constraints (e.g., validation on grade range)
* Implementing user authentication to restrict edit/delete access
* Improving the UI with real-time interactivity using JavaScript frameworks like Vue or React
* Expanding the dashboard to allow institution-to-institution comparisons and historical trends

Acknowledgment of Tools and Assistance. We used ChatGPT (OpenAI) and Google Search as AI-based assistants during the project to support code debugging, resolve framework-related issues, and assist with deployment configurations. These tools were used strictly for troubleshooting and clarification purposes. All ideas, implementation logic, data modeling, and visual design decisions were fully conceived and developed by the project team.

Overall, this project not only strengthened our database and web development skills but also reinforced best practices in data modeling, automation, and deployment. We have learned a great deal with database technology. Thanks so much for the high quality and practical course.