BENSLIMANE Sarah
ARRIGONI Ambroise
EL GUEDDARI Kawtar

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

Technician internship report

Summer 2025

I. Introduction

As part of our third-year technician internship, we spent eight weeks at the National Institute of Technology, Ibaraki College in Hitachinaka, Japan. The internship aimed to reinforce both technical and soft skills through a combination of teaching assistance and software development tasks. We were expected to apply the knowledge acquired during our engineering education, develop problem-solving abilities, and become proficient with professional and technical tools. The experience also emphasized understanding company operations at economic, organizational, and human levels, as well as awareness of sustainability, quality, and workplace safety. Furthermore, the internship fostered teamwork, communication within a professional environment, autonomy, accountability, and the ability to present ideas clearly both in writing and orally.

The National Institute of Technology, Ibaraki College (NITIC) is a leading Japanese technical institution offering a five-year integrated engineering program like the INSA Rouen; however, it differs in that students are admitted directly after junior high school. It specializes in Industrial Engineering with four majors and provides a two-year advanced course leading to a bachelor's degree. The college emphasizes hands-on, practical education closely aligned with real-world applications, resulting in high employment and university transfer rates. NITIC is internationally active, with partnerships including INSA Rouen in France and institutions in Korea, Mexico, and Canada. It fosters research and innovation, notably in AI education, and regularly undergoes external quality evaluations. Campus life is supported by modern facilities, active student organizations, and on-site housing, making it a dynamic environment for both academic and personal growth. During our internship, we stayed on campus in the student dormitories, which provided a unique and enriching cultural experience. The welcoming atmosphere and daily life on campus made our stay both enjoyable and meaningful.

II. Teaching Assistance

The main goal of this mission was to support students during their courses by assisting them with understanding the material, helping with exercises, and facilitating their learning process. It also aimed to develop our skills in pedagogy and communication.

There were five of us assigned to the teaching assistant role, and each of us was assigned to specific courses. For each course, there were two to three teaching assistants working together to help the students. We assisted students ranging from 1st to 3rd year, aged between 15 and 17 years old.

The courses we supported included Global Life Science and Computer Literacy for 1st-year students, Global Science for 2nd-year students, and Information Processing 1 and Programming 2 for 3rd-year students.

The Global Life Science and Global Science courses focused on scientific culture. At the end, we gave a presentation introducing ourselves and INSA Rouen. It was very interesting to engage in discussions with the students afterward, especially since they will have the opportunity to attend INSA Rouen in the future.

In the Global Science course, the students worked on a semester-long project where they had to develop a solution to a regional problem. All the student groups were competing for the chance to represent their school in a larger competition held in Thailand. We had the chance to discuss with each group and offer advice and guidance throughout their project.

In the Computer Literacy course, our main role was to ensure that students successfully completed their exercises and to assist them when needed. The students programmed using Python in this course.

The Information Processing 1 course was similar, but students coded in C. The course instructor provided us with corrections that we could use to help the students.

Finally, in Programming 2, which was also taught in Python, we prepared the lessons in advance and had access to the exercise corrections. Overall, this teaching assistance experience was incredibly enriching for us

Beyond the technical content, this role taught us valuable skills in communication and pedagogy. We learned how to explain complex concepts in simple and clear terms, adapting our explanations to different levels of understanding. Reformulating questions and ideas became essential to help students grasp difficult topics. It also developed our patience and empathy, as supporting students requires listening carefully to their difficulties and encouraging their progress. This experience improved our ability to work as a team and reinforced the importance of collaboration when assisting learners. Overall, being a teaching assistant helped us grow not only academically but also personally and professionally.

III. Project

1. General overview of the project

As part of our internship, we were also required to carry out a team project. We were free to choose the theme, and after discussing several options, we decided to focus on software development.

We chose the software development theme because we wanted to put into practice the knowledge and skills we had acquired during our third year at INSA, particularly in web development and programming. This path appeared both more challenging and more relevant to our future careers, as it aligned with the type of work we could be doing in tech companies or research labs.

Our goal was to create a web platform designed to help students learn to code in Python through interactive exercises. The idea was to allow students to progress independently, at their own pace, by offering them structured exercises and immediate feedback. We wanted the platform to be intuitive, educational, and accessible to beginners, particularly those from 1st to 3rd year.

The website is fully responsive, meaning it works seamlessly on both computers and mobile devices, ensuring accessibility for all users. The final version of the platform is available online at: Python Learning Site

This project had both a technical and educational purpose: on one hand, we were developing a complete web application; on the other, we were designing an effective learning tool tailored to the needs of young learners. This double objective made the project especially motivating and meaningful for us.

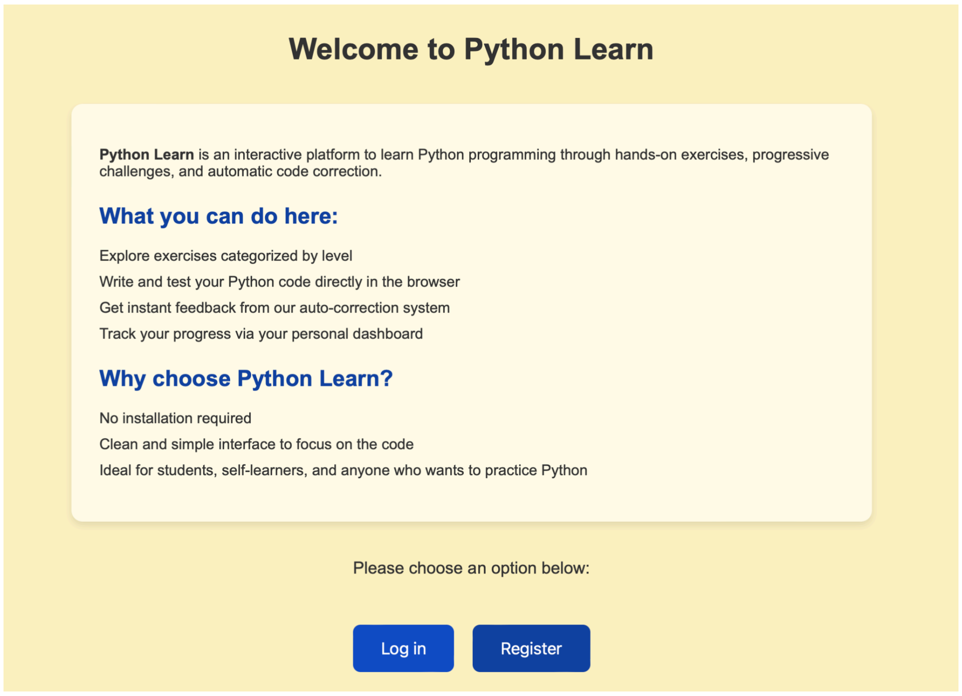
2. Frontend

For the frontend development, we used standard web technologies:

- HTML5 for page structure
- CSS for layout and styling
- JavaScript for interactivity and communication with the backend using fetch API calls

The user interface of the platform is based on several main pages:

- index.html: This is the main landing page, offering options for login and registration.



Welcome to Python Learn

Python Learn is an interactive platform to learn Python programming through hands-on exercises, progressive challenges, and automatic code correction.

What you can do here:

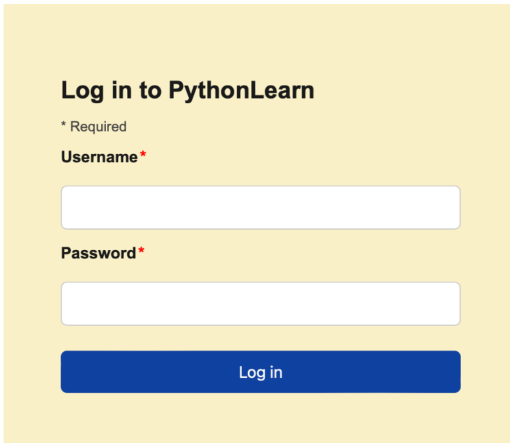
Explore exercises categorized by level
Write and test your Python code directly in the browser
Get instant feedback from our auto-correction system
Track your progress via your personal dashboard

Why choose Python Learn?

No installation required
Clean and simple interface to focus on the code
Ideal for students, self-learners, and anyone who wants to practice Python

Please choose an option below:

- login.html and register.html: These pages allow users to log in or create an account. They interact with the backend via fetch requests to a REST API connected to our PostgreSQL database hosted on Railway.

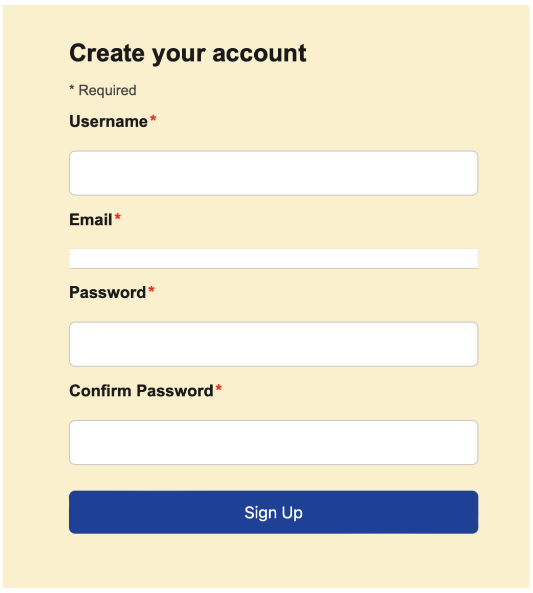


Log in to PythonLearn

* Required

Username*

Password*



Create your account

* Required

Username*

Email*

Password*

Confirm Password*

- After logging in, the user is redirected to acceuil.html, which serves as the main dashboard to access exercises.

- The acceuil.html page displays a list of Python exercises, categorized by type (e.g. *discovery*, *lesson*, *practice*, *validation*, *challenge*), creating a structured learning path.
- When an exercise is opened, the coding interface (defined in interfacecode.js) is loaded, allowing students to write, run, and submit Python code directly in the browser.

Some of the key interactive frontend features include:

- Dynamic forms for login and registration, using fetch to send data asynchronously without refreshing the page
- Conditional redirection to the main dashboard (acceuil.html) upon successful login
- A coding interface integrated into the exercise pages, allowing users to:
    - Write Python code
    - Execute it directly in the browser
    - Submit it for validation and save their progress
- Use of badges and labels (via HTML/CSS) to visually differentiate exercise types and guide the user's learning journey

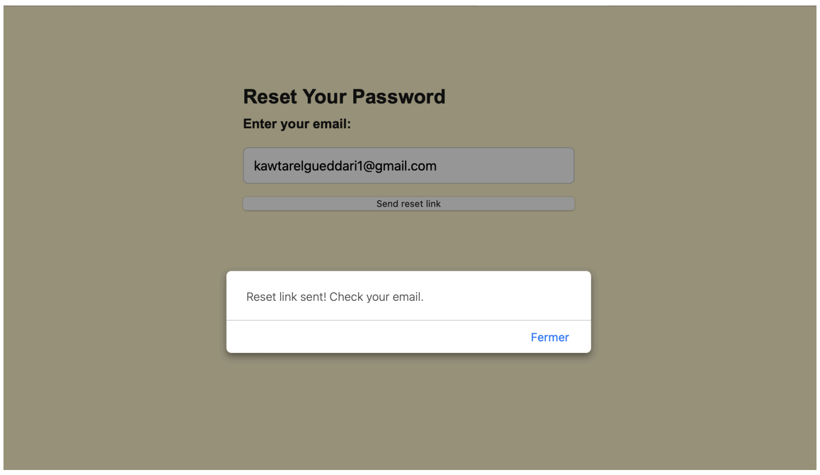
The password reset feature allows a user to securely reset their forgotten password by following these steps:

1. Forgot Password Request
   The user clicks a "Forgot Password?" link on the login page and enters their email.
2. Generate Token and Send Email
    - A unique reset token (usually a JWT) is generated and stored in the database with an expiration time (e.g., 15 minutes).
    - A reset link containing this token is sent to the user's email.
3. Reset Password Form
    - When the user clicks the link, they are taken to a secure page where they can enter a new password.
    - The form includes the token (hidden) and the new password.
4. Validation and Update
    - The backend verifies that the token is valid and not expired.
    - If valid, the new password is hashed and updated in the database.
    - The used token is then deleted to prevent reuse.

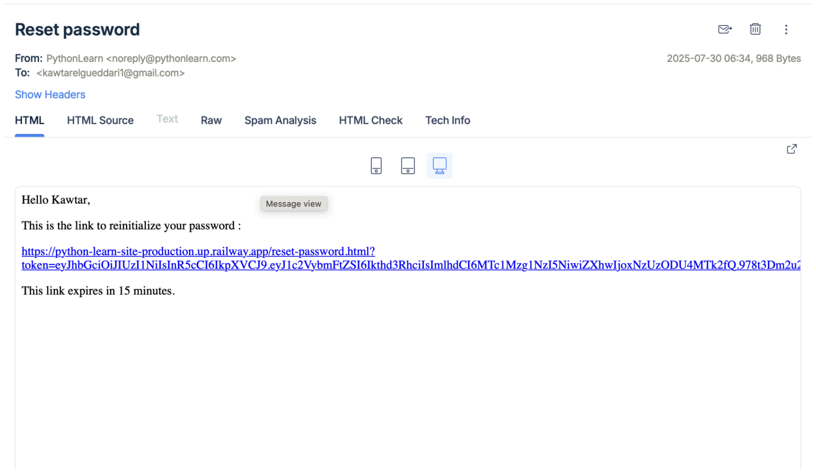
Since we’re working in a development environment, we couldn't send real emails to external addresses without setting up an actual SMTP service.
To simulate email delivery safely, we used Mailtrap; a tool designed to catch outgoing emails for testing purposes.

Using Mailtrap allowed us to:

- Verify that the reset email is being sent correctly
- View the contents of the reset email (including the link and token)
- Test the full reset flow without needing a real Gmail or production mail server



Reset link sent! Check your email.



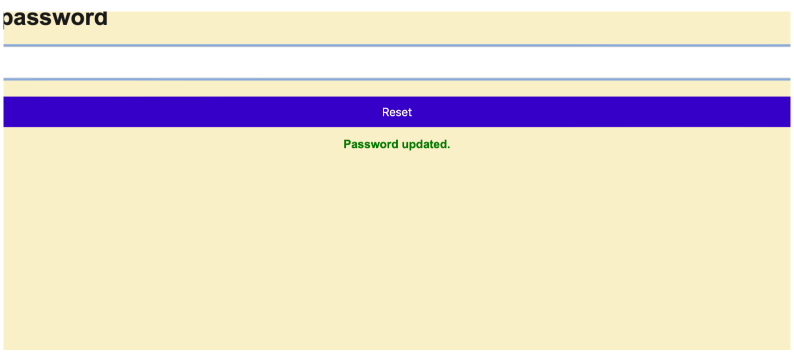
From: PythonLearn <noreply@pythonlearn.com>
To: <kawtarelgueddari1@gmail.com>
2025-07-30 06:34, 968 Bytes

Hello Kawtar,

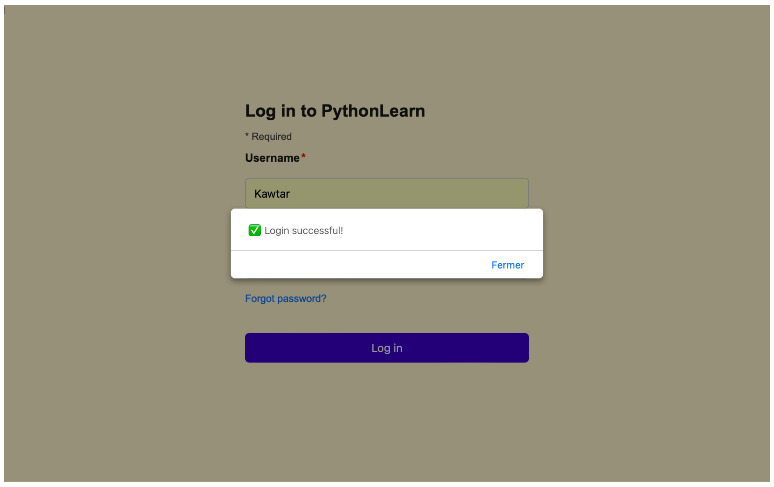
This is the link to reinitialize your password :

https://python-learn-site-production.up.railway.app/reset-password.html?
token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Ikthd3RhciIsImlhdCI6MTc1Mzg1NzI5NiwiZXhwIjoxNzUzODU4MTk2fQ.978t3Dm2u2

This link expires in 15 minutes.



Password updated.



One of the main challenges was managing user session persistence on the frontend without using a complex framework. We implemented a lightweight solution using localStorage to store basic user data (such as the user ID and name) and share it across pages.

Another difficulty was ensuring smooth communication between the frontend and backend, especially for retrieving and storing exercise progress. We had to carefully structure our fetch calls to handle authentication and data transmission securely and efficiently.

3. Backend

The backend of our web platform was developed using Node.js with the Express.js framework. Our main responsibilities involved setting up the server, managing HTTP routes, handling API requests, and ensuring secure and structured communication with the frontend and the database.

We organized our backend into different modules to improve clarity and maintainability:

- Authentication routes (/auth): Managed user registration and login. These routes were connected to our PostgreSQL database to verify credentials securely.
- Exercise management routes (/api/exercises): Enabled retrieving and saving exercises, including student code submissions and feedback.
- Static routes: We also served static HTML files (such as the coding interface) using Express's express.static() and res.sendFile() methods.

We made use of CORS and express.json() to allow smooth communication between frontend and backend, especially during local development and deployment on Railway. We also used dotenv to manage environment variables securely, such as database credentials.

The backend followed a RESTful API architecture, which means it used standard HTTP methods (GET, POST, etc.) to interact with well-defined resources (such as exercises or users). This approach ensured that data exchange between client and server remained stateless, modular, and easily scalable. For example, the frontend would make a POST request to /api/compile to send Python code to the server and receive the output in return.

Although the backend was relatively lightweight, it played a crucial role in:

- Serving the frontend content
- Exposing RESTful APIs
- Ensuring modular separation of concerns
- Logging server activity
- Managing the structure and flow of data between client and server

This backend setup allowed us to deploy the project seamlessly on Railway and test it in both development and production environments.

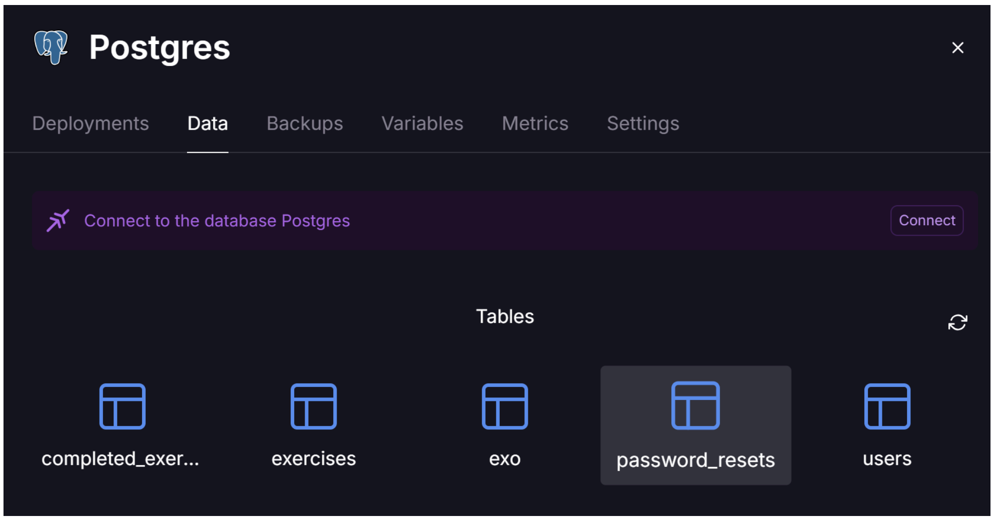
4. Database

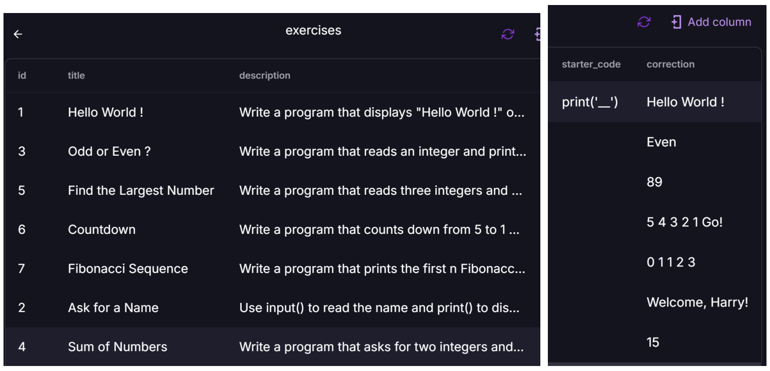
For this project, we chose to use PostgreSQL as our database management system. To simplify deployment and access for all team members, we hosted the database using Railway, a cloud platform that allows easy management of backend services like databases and APIs. We also used a PostgreSQL extension on VSCodium to handle the data on the tables using queries.

Railway offers a user-friendly interface, automatic deployments, and integration with GitHub, which made it particularly convenient for a student project. It allowed us to focus on development rather than infrastructure setup, while still working in a realistic production-like environment.

We designed a relational database to store the essential elements of our application, including:

- **Users**: to identify students using the platform
- **Exercises**: with titles, descriptions, difficulty levels, and sample input/output
- **Exo:** with id of the exercises, a story, a subject, a tip, a story correction, and the actual correction of the code
- **Completed_exercises**: to keep track of which exercises each user has completed
- **password_resets:** with an id and a token, as well as an expiration date




exercises
Add column

id	title	description	starter_code	correction
1	Hello World !	Write a program that displays "Hello World !" o...	print('__')	Hello World !
3	Odd or Even ?	Write a program that reads an integer and print...		Even
5	Find the Largest Number	Write a program that reads three integers and ...		89
6	Countdown	Write a program that counts down from 5 to 1 ...		5 4 3 2 1 Go!
7	Fibonacci Sequence	Write a program that prints the first n Fibonacc...		0 1 1 2 3
2	Ask for a Name	Use input() to read the name and print() to dis...		Welcome, Harry!
4	Sum of Numbers	Write a program that asks for two integers and...		15

exercises table

Most of the queries were related to:

- Fetching exercises for a user based on difficulty
- Storing and retrieving user submissions
- Checking which exercises a user had completed
- Managing user accounts and login sessions

5. Global Integration

Our platform consists of three main components: the frontend (interface and user interaction), the backend (logic and routing), and the PostgreSQL database (data storage). While each component was developed independently, the strength of the project lies in how we successfully integrated them into a single, cohesive system.

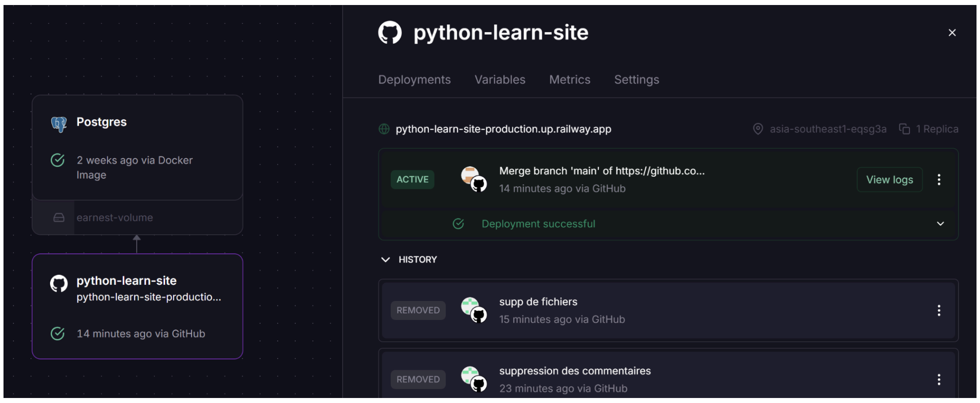
Component Interaction:

The different parts of the application communicate through a well-defined structure of RESTful APIs. These APIs allow the frontend to trigger backend operations such as authentication, code execution, and exercise retrieval. This architecture enabled clean separation between user interface, logic, and data layers, making development and debugging more manageable.

Instead of repeating backend logic, we focused here on ensuring that all modules worked together reliably and responded correctly across environments, especially during deployment.

Deployment with Railway:

To make the platform accessible online and simulate a real-world usage scenario, we deployed the full stack on Railway, a cloud hosting platform tailored for web apps.



Merge branch 'main' of https://github.co...
14 minutes ago via GitHub
Deployment successful

supp de fichiers
15 minutes ago via GitHub

suppression des commentaires
23 minutes ago via GitHub

We selected Railway because it offered:

- Straightforward deployment from GitHub (CI/CD workflow),
- Built-in PostgreSQL hosting, simplifying database connection,
- Secure environment variable management, essential for protecting sensitive data,
- Quick iteration cycles, allowing us to test changes directly in production.

Railway served both our static frontend files and dynamic backend routes, ensuring a unified deployment. This gave us a realistic environment to test features as end users would experience them.

Functional Testing:

Once deployed, we conducted full end-to-end manual testing to ensure:

- Consistent behavior between local and cloud versions,
- Proper flow of data between UI, backend logic, and the database,
- Robust handling of user input and Python code execution,
- Secure authentication and error management.

We also simulated typical student usage patterns to confirm that the platform could support its educational goals effectively.

6. Project Review

Overall, the project was a success both technically and educationally. The team collaborated efficiently, dividing tasks between frontend, backend, and database development while maintaining consistent communication. The platform's core features, including user authentication, exercise management, and the coding interface with real-time code execution, were successfully implemented and deployed. In addition, the integration with a remote execution API ensured that Python code could be compiled and executed directly in the browser, significantly enhancing the user experience.

Despite these achievements, certain aspects could have been improved. The project initially faced difficulties with database connectivity and deployment on Railway, which caused delays in development. The code structure and separation of responsibilities between frontend and backend could also have been more refined, which would have facilitated scalability and future maintenance. Furthermore, while the platform allowed users to solve exercises and receive immediate feedback, it lacked advanced features such as detailed progress tracking and automated grading for more complex tasks. These improvements could be considered in future iterations of the project.

Through this project, we gained several valuable lessons. We strengthened our practical experience in full-stack web development using Node.js, Express, PostgreSQL, and frontend technologies. We learned the importance of structured planning, debugging, and version control in collaborative environments. The project also provided us with hands-on experience in managing deployment challenges in a cloud environment and enhanced our problem-solving abilities. More broadly, it improved our teamwork skills and our capacity to adapt quickly to unforeseen technical issues, which are essential competencies in professional software development.

IV. Conclusion

This internship provided a comprehensive and enriching experience that combined teaching assistance with a technical development project. The teaching role strengthened our ability to communicate complex concepts clearly, adapt to different learning styles, and work collaboratively with both students and faculty members.

From a technical standpoint, building the Python learning platform allowed us to consolidate and expand our skills in full-stack development, database management, and software deployment. We also gained a deeper understanding of educational technology and learned how to design interactive tools that enhance student learning.

Overall, this experience contributed significantly to our personal and professional growth. It helped us develop technical skills—such as programming, database design, and cloud deployment—and soft skills—such as teamwork, communication, and project management. These competencies will be highly valuable for our future careers in engineering, software development, or research, and they have provided a strong foundation for tackling more advanced projects and professional challenges.