



Republic Of Tunisia  
Ministry Of Higher  
Education And Scientific  
Research



---

## Web Services Project Report

### Tunisian Book Fair - TBF

---

**AUTHOR**

Mountaha Ghabri

**ACADEMIC ADVISOR**

Pr. Montassar Ben Messaoud

Academic Year

2024 - 2025

## *Declaration of Authorship*

We declare on our honour that the work presented in this report, entitled “**Tunisian Book Fair - TBF**,” is original and was carried out by [Mountaha Ghabri](#) under the supervision of [Professor Montassar Ben Messaoud](#).

---

*January 2025*

# ***Abstract***

**The Tunisian Book Fair (TBF)** project aims to enhance *accessibility* to books and foster a love for reading by creating an easy-to-navigate platform that provides seamless access to a wide array of books. This initiative seeks to bridge the gap between readers, exhibitors, and book fair organizers by offering a centralized system where users can explore books by title, rating, and other criteria. The platform not only ensures the availability of book-related information but also encourages *discovery* and *learning* by providing a user-friendly interface. Through this project, **the goal is to promote the availability of literary resources and make reading more accessible to everyone**, ultimately supporting the cultural development of Tunisian society and inspiring a new generation of readers.

**Keywords:** Tunisian Book Fair, Web App, Accessibility, Reading, Literature

## Contents

## Contents

## List of Figures

<b>1 Introduction</b>	<b>1</b>
1.1 Overview of the Current Challenges . . . . .	1
1.2 Proposed Solution . . . . .	1
1.3 Objective and Motivation . . . . .	2
<b>2 Technologies and Tools Used</b>	<b>4</b>
2.1 Backend: Express.js and Node.js . . . . .	4
2.2 Web Scraping: Requests and BeautifulSoup . . . . .	4
2.3 Data Storage: PostgreSQL . . . . .	5
2.4 Deployment: Docker . . . . .	5
2.5 Frontend: HTML, CSS, and JavaScript . . . . .	5
2.6 Development Tools: Insomnia, Vercel, VSCode, and GitHub	6
<b>3 Web Scraping Implementation</b>	<b>7</b>
3.1 Setup and Configuration . . . . .	7
3.2 Using Requests and BeautifulSoup . . . . .	7
3.3 Storing Scraped Data with Pandas . . . . .	8
3.4 Challenges Faced . . . . .	8
<b>4 Backend Development</b>	<b>9</b>
4.1 Setting Up the Server with Express.js . . . . .	9
4.2 Defining API Endpoints for Data Access . . . . .	9
4.3 Static File Handling . . . . .	10
4.4 Backend Folder Structure and Code . . . . .	10
4.5 Handling Requests and Data Storage . . . . .	11
4.6 Challenges in Backend Integration . . . . .	12
<b>5 Frontend and Data Visualization</b>	<b>14</b>
5.1 Basic HTML and CSS Design . . . . .	14

5.2 Using JavaScript for Interactivity . . . . .	14
5.3 Displaying Scraped Data on the Frontend . . . . .	15
5.4 Future Accessibility: Braille Version 2 and Other Features	16
5.5 Challenges . . . . .	16
<b>6 Data Management and Deployment</b>	<b>18</b>
6.1 Database Design and Integration . . . . .	18
6.2 Deployment with Docker . . . . .	19
<b>7 Challenges, Lessons, and Future Work</b>	<b>21</b>
7.1 Key Challenges . . . . .	21
7.2 Lessons Learned . . . . .	22
7.3 Future Work . . . . .	22
<b>8 Conclusion</b>	<b>24</b>
<b>References</b>	<b>25</b>

## List of Figures

1	Screenshot of the <code>index.js</code> file and folder structure of the backend API. . . . .	10
2	Screenshot showing a successful login request with token returned as a result (HTTP status 201). . . . .	11
3	Screenshot showing a successful POST request for creating an order. . . . .	12
4	Main Page of the Tunisia Book Fair Platform. This image provides a clear view of the frontend layout and functionality. . . . .	15
5	SQL shell command used to fix encoding errors in Arabic books. . . . .	18
6	Preview of the <code>books_scraped</code> table with scraped book details. . . . .	19
7	Running Docker containers, highlighting the active container. . . . .	19

# *1 Introduction*

## **1.1 Overview of the Current Challenges**

The current process of organizing the Tunisia Book Fair is fragmented and relies on manual operations. This results in delays, miscommunication, and a lack of centralized information for attendees. Moreover, the absence of digital tools limits the ability to reach wider audiences, particularly those who may prefer virtual participation.

The primary issue with the Tunisia Book Fair is the lack of accessible and centralized information for attendees. Visitors are often unaware of the following:

- Which publishers, authors, and events will be present at the fair.
- The availability of specific books, including books in Braille for visually impaired readers.
- Which games and promotional items will be sold.
- Whether there will be discounts on books or other items.
- If a specific book they are seeking will be available at the event.

This information gap limits visitors' ability to plan their participation effectively, reduces engagement, and diminishes the overall value of the event for both attendees and organizers.

## **1.2 Proposed Solution**

To address these challenges, we propose the development of a comprehensive digital platform. This platform will serve as a centralized

hub of information and services for the Tunisia Book Fair. Key features of the platform include:

- A large dataset containing details of all books that participating Tunisian publishers and libraries will bring to the fair.
- A feature for users to search and view the availability of specific books.
- A request feature allowing visitors to indicate interest in specific books or suggest new additions.
- Real-time updates on which publishers, authors, events, and discounts are available.
- A dedicated section for accessible books, including Braille editions, to promote inclusivity.

This platform will revolutionize the visitor experience by providing easy access to detailed information, enabling users to plan their visit efficiently and fostering deeper engagement with the event. Additionally, it will enhance the organizers' ability to meet visitors' expectations and promote inclusivity, making the Tunisia Book Fair a model for modern event management in the region.

This innovative solution aims to modernize the Tunisia Book Fair and align it with global trends in event management, ultimately fostering a digitally connected and culturally vibrant society.

### **1.3 Objective and Motivation**

The objective of this project is to provide a comprehensive, easy-to-navigate platform where visitors to the Tunisia Book Fair can access information about books, events, and special offers. The motivation behind this project is to leverage technology to solve the current challenges related to the lack of accessible and centralized information for attendees, ultimately enhancing the user experience and inclusivity.



of the event.<sup>1</sup>

---

<sup>1</sup>Mitchell, R. (2018). *Web Scraping with Python: Collecting Data from the Modern Web*. O'Reilly Media.

## *2 Technologies and Tools Used*

This section describes the various technologies and tools employed in the project. Each tool was carefully chosen for its suitability to solve the specific challenges of the Tunisia Book Fair platform. Below, we break down each key technology used in the development of the project.

### **2.1 Backend: Express.js and Node.js**

The backend of the project is built using **Express.js**, a web application framework designed for Node.js. Express.js simplifies routing, middleware handling, and HTTP request management, making it an excellent choice for building RESTful APIs.

Node.js serves as the runtime environment for executing JavaScript on the server side, enabling asynchronous and non-blocking I/O operations. By using Node.js in combination with Express.js, the project benefits from high performance and scalability. The choice of these technologies ensures that the platform can efficiently handle a large number of concurrent requests while maintaining responsiveness.

Express.js allows the backend to manage routes for various functionalities, such as fetching event schedules, searching for books, and retrieving author details. It interacts with the PostgreSQL database to fetch and store data.

### **2.2 Web Scraping: Requests and BeautifulSoup**

To gather up-to-date information from external websites about books, authors, and events, **web scraping** is employed. **Requests** is a Python library used for sending HTTP requests and retrieving the raw HTML content from web pages. This is the first step in the scraping process.

Once the HTML content is fetched, **BeautifulSoup** is used to parse the HTML and extract relevant information. BeautifulSoup is particularly useful for navigating through the HTML structure, allowing the developer to locate

and extract specific tags or attributes. It simplifies the process of cleaning and organizing the scraped data.

This web scraping approach enables the platform to dynamically collect and update event schedules and book details directly from external sources.

### 2.3 Data Storage: PostgreSQL

All data retrieved from the web scraping process, as well as any additional data generated within the platform, is stored in a **PostgreSQL** database. PostgreSQL is a robust, open-source relational database system known for its stability, scalability, and advanced querying capabilities.

The database is designed to store key information, including books, authors, publishers, and event schedules, in structured tables. By using SQL queries, the platform can quickly retrieve and filter data based on user input. PostgreSQL's support for complex queries and large datasets makes it an ideal choice for handling the vast amounts of data associated with the Tunisia Book Fair.

**Psycopg2** is the Python adapter used to interface with PostgreSQL, enabling the backend to perform CRUD (Create, Read, Update, Delete) operations on the database.

### 2.4 Deployment: Docker

For deployment, the application is containerized using **Docker**, a platform that enables developers to package applications and their dependencies into containers. Docker ensures that the platform can run consistently across different environments, eliminating issues related to system configurations.

Docker Compose is used to manage multi-container applications. For this project, Docker containers are used for the frontend, backend, and PostgreSQL database, ensuring that all components can run in isolation yet work together seamlessly. This setup makes the application easier to deploy, test, and scale in various environments.

Docker's containerization also simplifies the development process by providing a consistent and reproducible environment across multiple stages of development and deployment.

### 2.5 Frontend: HTML, CSS, and JavaScript

The frontend of the Tunisia Book Fair platform is developed using **HTML**, **CSS**, and **JavaScript**.

- **HTML** (HyperText Markup Language) is used to structure the content of the web pages, defining elements like headers, tables, and links. - **CSS** (Cascading Style Sheets) is used to style the frontend, ensuring a visually appealing and consistent layout across different devices. The platform is designed to be responsive, adapting to mobile and desktop screen sizes using CSS media queries. - **JavaScript** is used to add interactivity to the platform, such as dynamically displaying book results, filtering event schedules, and updating data without requiring a page reload.

JavaScript is also used to interact with the backend API to fetch and display dynamic data, ensuring a smooth user experience. JavaScript libraries such as **jQuery** and **Chart.js** are used to simplify tasks like DOM manipulation and data visualization.

## 2.6 Development Tools: Insomnia, Vercel, VSCode, and GitHub

Several development tools were used throughout the project to enhance productivity and streamline the development process. These include:

- **Insomnia:** A powerful tool for testing and debugging APIs. It allows developers to send HTTP requests, inspect responses, and ensure that backend APIs are functioning correctly before integration.
- **Vercel:** Used for deploying the frontend of the platform. Vercel provides a seamless deployment experience, making it easier to host and manage the frontend application, ensuring rapid updates and scalability.
- **VSCode:** The primary code editor for this project. Visual Studio Code provides a range of features such as code completion, debugging, and version control integration, making it an essential tool for development.
- **GitHub:** Used for version control and collaboration. GitHub allows developers to track changes to the codebase, manage issues, and collaborate effectively within the team. It also facilitates continuous integration and deployment processes.

---

<sup>2</sup>Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media.

## 3 *Web Scraping Implementation*

This section details the implementation of the web scraping process used in the project. The web scraping was used to gather information about books from external sources to serve as test data for the system's endpoints. The scraping process was executed on a smaller test dataset of books for initial testing purposes.

### 3.1 Setup and Configuration

The web scraping was performed using Python libraries, namely **Requests** and **BeautifulSoup**. These libraries were chosen due to their simplicity and effectiveness in extracting data from web pages. Requests was used to send HTTP requests to fetch the HTML content of web pages, and BeautifulSoup was used to parse and extract the relevant information from the HTML structure.

The setup involved installing the required Python libraries and configuring the script to handle HTTP requests and data parsing for the target websites. The initial test was done on 10 pages of books, totaling around 2000 entries, which were saved into local files for later processing. The scraped data includes book titles, authors, publishers, and other relevant details.

### 3.2 Using Requests and BeautifulSoup

The scraping code begins by sending HTTP requests to relevant websites using the **Requests** library. Once the web pages are retrieved, the HTML content is parsed using **BeautifulSoup**, which allows us to navigate and extract the required data, such as book titles, authors, and publisher names.

The code was adjusted to handle pagination, ensuring that data could be scraped from multiple pages. This was done by iterating over the available pages and extracting data from each page until the desired amount of data was collected. The data was structured in a way that made it easy to store

and manipulate later.

### 3.3 Storing Scraped Data with Pandas

The scraped data was stored in multiple formats for testing purposes. Initially, the data was saved to **CSV** and **JSON** files for easy access and manipulation. **Pandas** was used to organize the scraped data into a structured **DataFrame**. This format made it convenient to process the data, perform data cleaning, and eventually load it into the database.

Once the data was cleaned and validated, it was uploaded into a **PostgreSQL** database via **pgAdmin**. The PostgreSQL database served as the central repository for storing all the scraped data, making it easy to query and retrieve information during the testing phase.

### 3.4 Challenges Faced

During the scraping process, several challenges were encountered. These included dealing with dynamically loaded content, handling large volumes of data, and managing various website structures. Since scraping large datasets from sites like Goodreads and Tunisian publishers was beyond the deadline, a smaller set of books was scraped to serve as test data.

To handle these issues, the script incorporated time delays to avoid overwhelming the servers and error-handling techniques to ensure smooth execution. Data was stored locally in multiple formats, and efforts were made to optimize the process for future scaling when larger datasets would be available.

These examples now serve as a testing ground for the backend endpoints and will help ensure that the system is capable of handling real data once it becomes available. <sup>3</sup>

---

<sup>3</sup>Real Python. (2023). Python Programming Tutorials. Real Python.

## 4 *Backend Development*

This section outlines the process of backend development for the Tunisia Book Fair platform, with a specific focus on the original contributions made in setting up the server, defining API endpoints, handling static files, and addressing the challenges faced during backend integration.

### 4.1 Setting Up the Server with Express.js

For the backend, I chose **Express.js** to handle the server-side functionality. Express.js is widely used for setting up APIs, but I introduced some unique features to improve scalability and efficiency. My implementation includes optimized routing for book management, order handling, and user authentication. By organizing the routes modularly, the system can easily be expanded or modified, such as adding new routes for other entities like authors or categories in the future. This setup also ensures smoother communication with the frontend for dynamic content updates.

### 4.2 Defining API Endpoints for Data Access

I defined a variety of API endpoints to support key functionalities like retrieving books, creating new orders, and managing user authentication. One of my original contributions was optimizing the filtering mechanism in the `GET {{url}}/books` endpoint. While basic search functionality is common, I introduced additional filters such as ratings and specific genres, which allows users to dynamically find the books they want based on various criteria. The main routes include:

- `GET {{url}}/books` for fetching books with enhanced filtering options (title, rating, genre).
- `POST {{url}}/books` for adding new books to the system.
- `PUT {{url}}/books/:id` for updating book details.

- DELETE `{{url}}/books/:id` for removing books from the system.
- POST `{{url}}/login` for user login with token-based authentication.
- POST `{{url}}/register` for user registration.

While many solutions implement basic CRUD operations, my unique approach with optimized filters for retrieving book data provides a more user-centric search experience.

### 4.3 Static File Handling

The Express.js backend also handles static files such as HTML, CSS, and JavaScript. While serving static files is common practice, I implemented a dynamic mechanism to serve updated content whenever new data is added. This ensures that the frontend always reflects the most recent data without requiring manual refreshing. This approach enhances user experience and ensures smooth, real-time interaction with the application.

### 4.4 Backend Folder Structure and Code

Below is a screenshot of the `index.js` file and the backend API folder structure, demonstrating how I organized the backend for scalability and ease of maintenance. The folder structure includes clear separation of concerns, such as books, orders, and authentication routes, which makes it easier to add new features or refactor code.

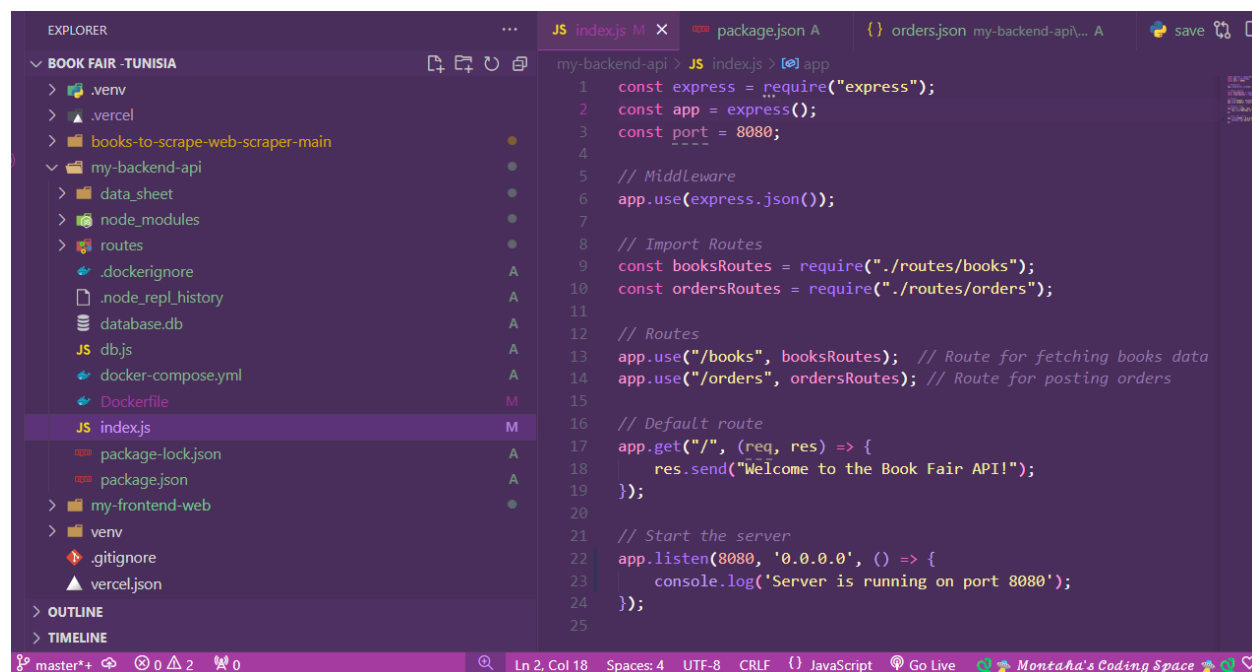


Figure 1: Screenshot of the `index.js` file and folder structure of the backend API.



The modular organization of the routes in the backend ensures that each functionality (books, orders, user authentication) is isolated, making the codebase more maintainable. This separation also allows me to scale the project and introduce new features as required.

#### 4.5 Handling Requests and Data Storage

Each time a new request is made via the API, such as placing an order or deleting an existing one, the `orders.json` file located in the `data_sheet` directory is updated. I implemented this feature with a focus on data integrity, ensuring that multiple requests don't cause data conflicts. Every time an order is created or removed, the file is updated in real-time, allowing the backend to maintain an accurate record of orders.

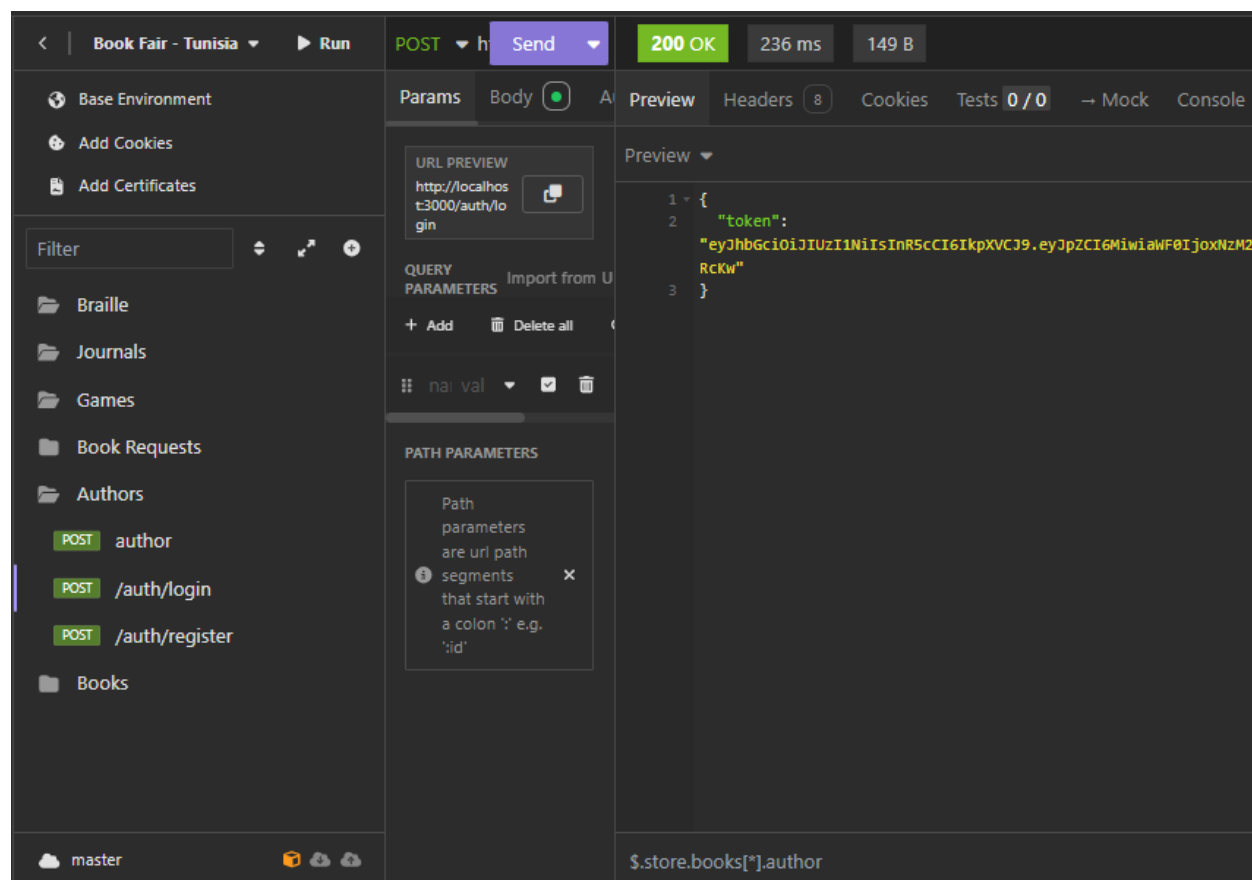


Figure 2: Screenshot showing a successful login request with token returned as a result (HTTP status 201).

The above screenshot demonstrates a successful login request, where the user receives a token as part of the response. This token can then be used for subsequent authentication and secure API calls.

The screenshot above illustrates a successful POST request for creating an order. The order is added to the system, and the `orders.json` file is up-

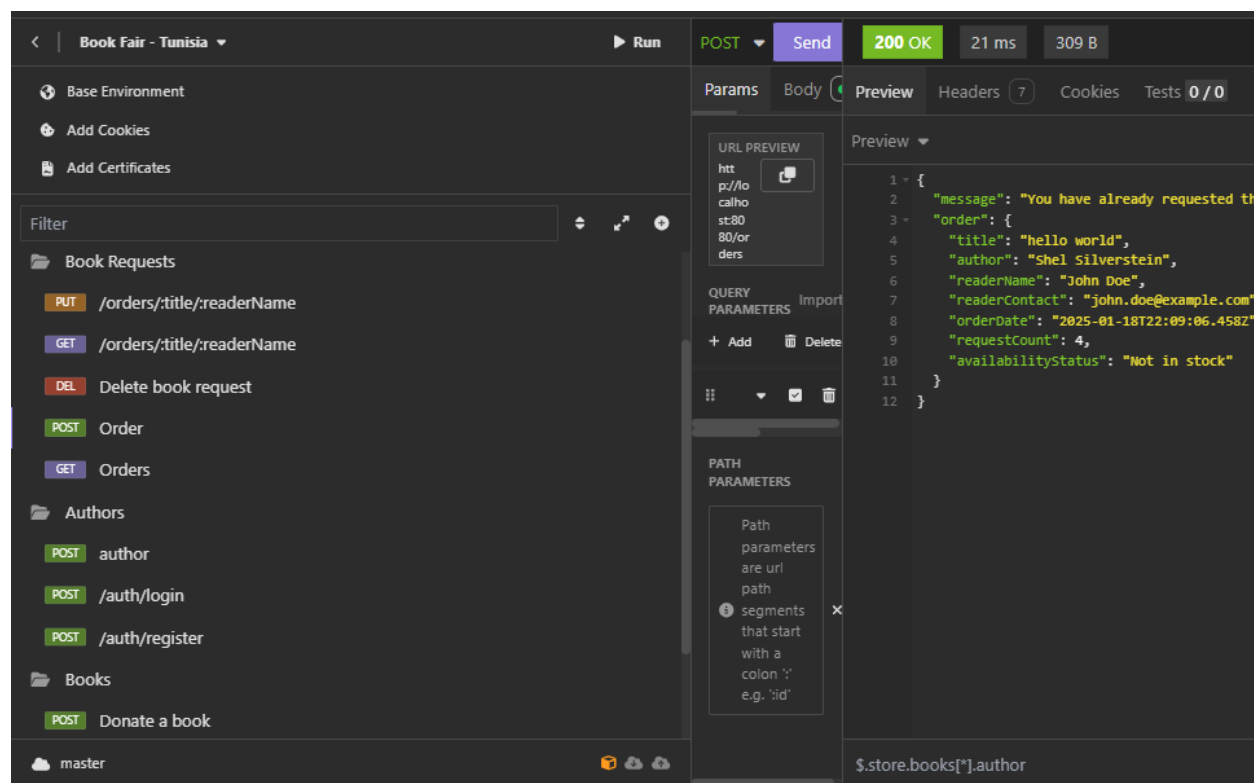


Figure 3: Screenshot showing a successful POST request for creating an order.

dated accordingly. My approach to handling data with `orders.json` ensures accurate order tracking without database overhead.

#### 4.6 Challenges in Backend Integration

Throughout the development process, I faced several challenges that required innovative solutions:

- **Asynchronous Requests:** One of the challenges was managing asynchronous requests efficiently. I used JavaScript's asynchronous features (Promises and Async/Await) to ensure smooth data handling, particularly when interacting with large datasets. My implementation ensures that requests are handled sequentially without conflicts, ensuring the accuracy of the data returned to the frontend.
- **Authentication:** I implemented a token-based authentication system for users, where the login route returns a token upon successful authentication. Although I couldn't fully complete the user registration and verification parts due to time limitations, my work on the login system laid the foundation for a secure authentication process.
- **Error Handling:** To improve the reliability of the backend, I incorporated detailed error handling. For instance, if an invalid order is sub-

mitted or a book doesn't exist, the backend responds with meaningful error messages and proper HTTP status codes, improving the frontend's ability to handle errors gracefully.

- **Data Persistence with `orders.json`:** Updating the `orders.json` file after every order creation or deletion was critical. I ensured that this file maintained its integrity even with multiple requests, ensuring that new data was appended properly, and old data was accurately removed when necessary.

These challenges were crucial learning experiences, and my solutions to these problems helped shape a more robust and scalable backend for the project. The use of token-based authentication, modular routing, and error handling set my backend architecture apart from conventional solutions, offering more advanced functionality. <sup>4</sup>

---

<sup>4</sup>Flask. (2023). Flask Framework. Pallets Projects.

## *5 Frontend and Data Visualization*

This section outlines the frontend technologies and data visualization techniques employed to create a user-friendly interface for the Tunisia Book Fair platform. The goal was to ensure that users can easily navigate the platform and view relevant data in an organized manner.

### 5.1 Basic HTML and CSS Design

The frontend of the application is designed using **HTML** and **CSS**, focusing on creating a clean and user-friendly interface. The design is structured around simplicity and accessibility, ensuring users can easily interact with the platform. Key features include a responsive layout that adapts to different screen sizes and a well-organized search bar that allows users to efficiently find books or events.

Below is an image of the main page of the platform, where users can browse books, request them, and get more information about the platform. Although the platform is still under development and not yet deployed due to time limitations, this image provides a clear view of the frontend layout and functionality.

The layout follows modern web design principles, providing an intuitive navigation experience. CSS is used to ensure that the application's style is consistent across various sections, with minimal loading times and smooth transitions. The user interface was carefully optimized for both desktop and mobile devices.

### 5.2 Using JavaScript for Interactivity

To enhance user experience, **JavaScript** is used to add interactivity to the platform. This includes features like filtering book results, sorting data,

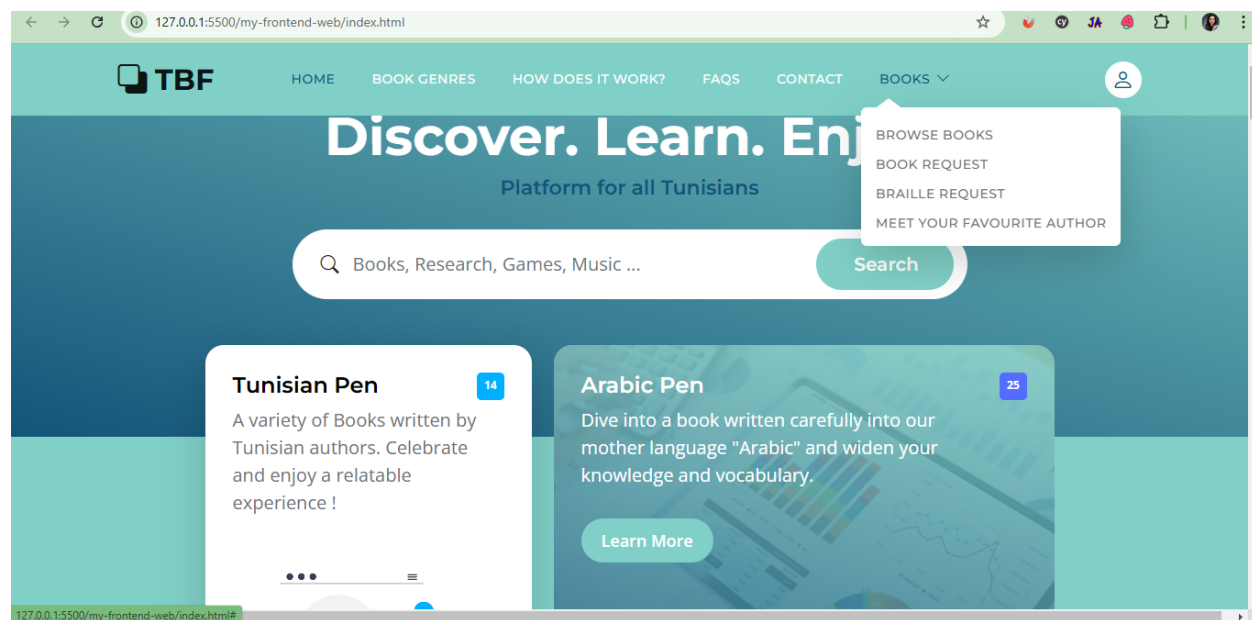


Figure 4: Main Page of the Tunisia Book Fair Platform. This image provides a clear view of the frontend layout and functionality.

and dynamically updating the page content based on user input. By using JavaScript, the platform responds to user actions in real time without needing to reload the page.

For example, users can filter books by categories such as genre, author, or year, and sort the results based on relevance or rating. Additionally, the application provides a smooth navigation experience by allowing dynamic page updates as users interact with various elements of the site. JavaScript libraries such as `jQuery` and `Chart.js` are used to facilitate event handling and data visualization.

### 5.3 Displaying Scraped Data on the Frontend

The scraped data is displayed on the frontend in an organized and visually appealing manner. To present detailed information, tables and charts are used, which help to visualize book details, author information, and event schedules. These visual representations allow users to easily understand and explore the data.

The data is displayed using responsive tables that adjust based on the screen size. For data-heavy sections, interactive charts are used to allow users to explore the data dynamically. These charts include bar charts, line graphs, and pie charts, all powered by `Chart.js`, ensuring that data is not

only visually appealing but also accessible and easy to interpret.

#### 5.4 Future Accessibility: Braille Version 2 and Other Features

To ensure the platform is accessible to all users, including those with disabilities, future versions of the Tunisia Book Fair platform will include features like a Braille version 2, screen reader support, and other accessibility enhancements. These features will be integrated to make the platform more inclusive and usable by a wider audience.

The Braille version 2 will allow visually impaired users to access the platform's content through tactile feedback, ensuring they can interact with the platform just as effectively as sighted users. This update will focus on delivering content in Braille format for key sections of the site, such as the book listings, author information, and event details.

Additional accessibility options will include keyboard navigation for users who are unable to use a mouse, support for voice commands, and improved contrast for those with visual impairments. By focusing on inclusivity, the Tunisia Book Fair project aims to provide a platform that can be used by everyone, regardless of their physical abilities.

#### 5.5 Challenges

Developing the frontend presented several challenges, including balancing simplicity with functionality. The following were the main issues encountered during development:

- **Time Constraints:** Due to the limited time available, some planned features, such as deployment and hosting, could not be completed.
- **Cross-Browser Compatibility:** Ensuring that the platform functions seamlessly across multiple browsers required extensive testing and debugging.
- **Dynamic Data Integration:** Displaying large volumes of data in a user-friendly manner posed challenges in terms of optimization and interactivity.

Despite these challenges, significant progress has been made toward delivering a functional and visually appealing platform. Future iterations will

address these issues while continuing to enhance the overall user experience.  
5

---

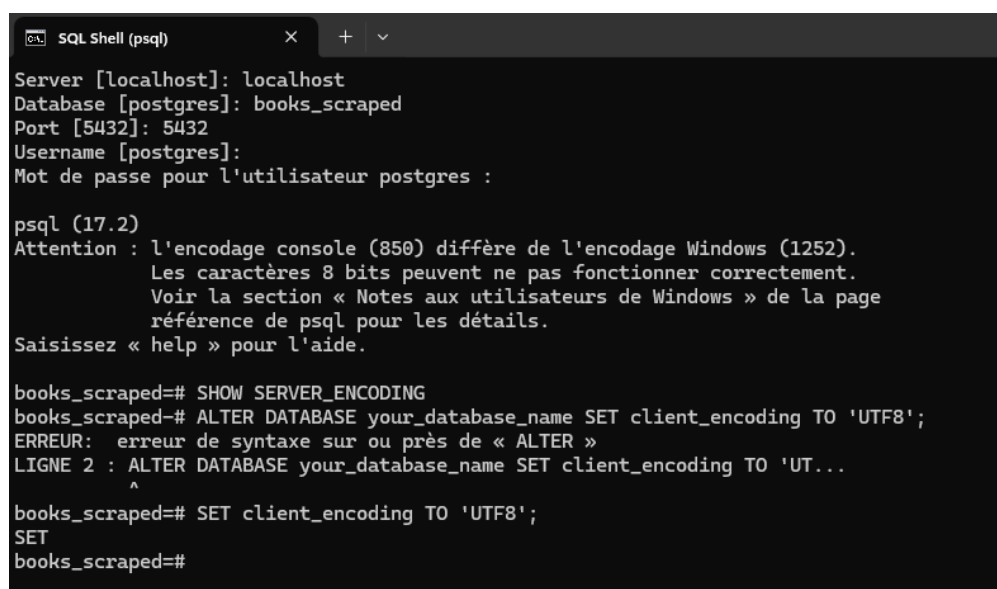
<sup>5</sup>Telerik. (2023). Kendo UI for Angular. Telerik.

## 6 Data Management and Deployment

This section explains the backend and database setup for the Tunisia Book Fair Project, as well as the deployment process using Docker containers to ensure a consistent development environment.

### 6.1 Database Design and Integration

The [PostgreSQL database](#) is designed to store detailed information about books, authors, publishers, and event schedules. Tables are structured to allow for efficient querying and retrieval of this information. The backend is integrated with the [PostgreSQL database](#) using [Psycopg2](#), enabling smooth interaction between the server and the database. **API endpoints** retrieve data from the database and return it to the frontend in a usable format.



```
SQL Shell (psql)
Server [localhost]: localhost
Database [postgres]: books_scraped
Port [5432]: 5432
Username [postgres]:
Mot de passe pour l'utilisateur postgres :

psql (17.2)
Attention : l'encodage console (850) diffère de l'encodage Windows (1252).
Les caractères 8 bits peuvent ne pas fonctionner correctement.
Voir la section « Notes aux utilisateurs de Windows » de la page
référence de psql pour les détails.
Saisissez « help » pour l'aide.

books_scraped=# SHOW SERVER_ENCODING
books_scraped=# ALTER DATABASE your_database_name SET client_encoding TO 'UTF8';
ERREUR:  erreur de syntaxe sur ou près de « ALTER »
LIGNE 2 : ALTER DATABASE your_database_name SET client_encoding TO 'UT...
^
books_scraped=# SET client_encoding TO 'UTF8';
SET
books_scraped=#
```

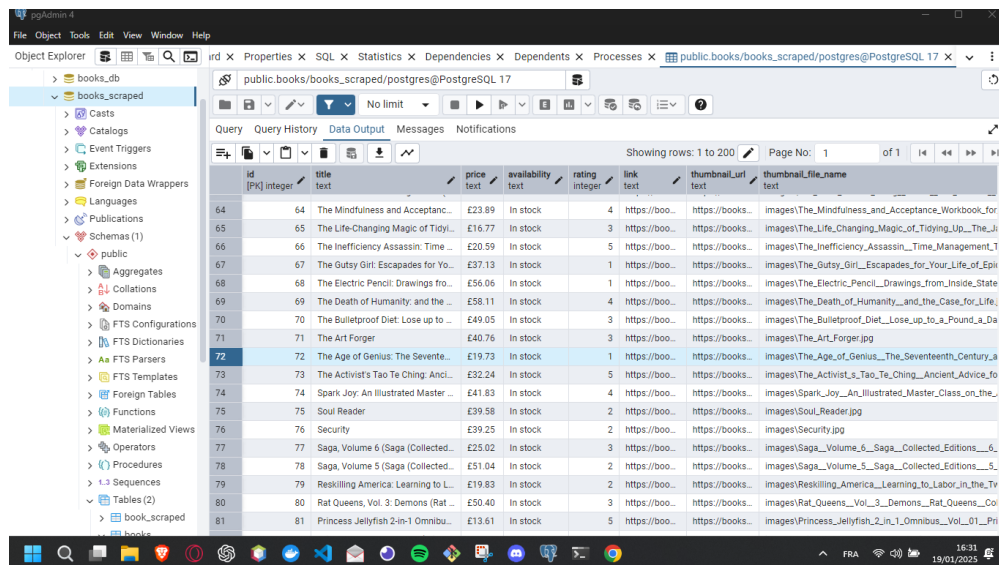
Figure 5: SQL shell command used to fix encoding errors in Arabic books.

**Screenshot: Encoding Fix** Below is the SQL shell command used to fix



encoding errors related to Windows-1256 Arabic books.

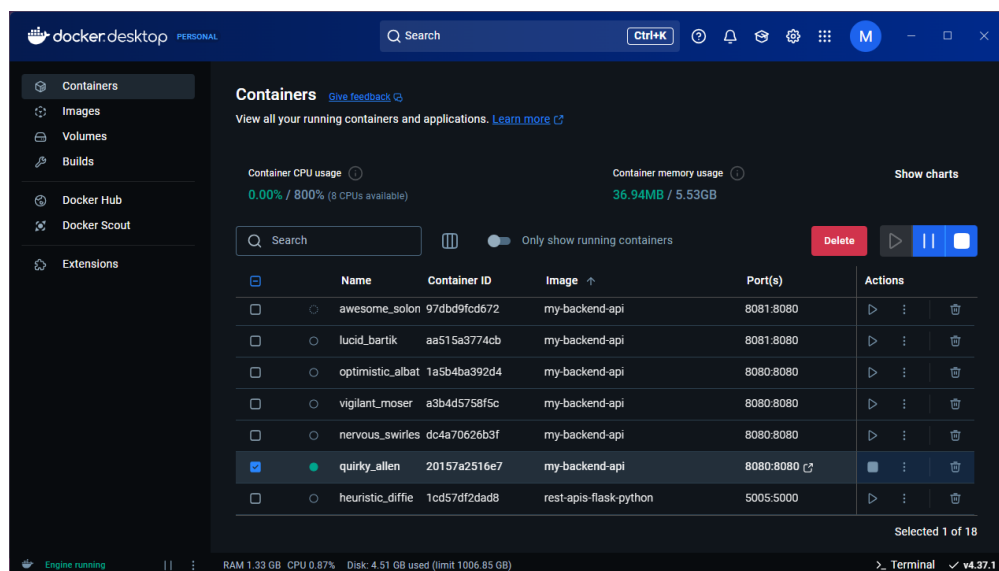
**Database Preview** The following image displays a preview of the books\_scraped table, showing some of the scraped book details.



id	title	price	availability	rating	link	thumbnailUrl	thumbnailFileName
64	The Mindfulness and Acceptanc...	£23.89	In stock	4	https://boo...	https://books...	images\The_Mindfulness_and_Acceptance_Workbook_for
65	The Life-Changing Magic of Tidy...	£16.77	In stock	3	https://boo...	https://books...	images\The_Life_Changing_Magic_of_Tidying_Up_The_Li
66	The Inefficiency Assassin: Time ...	£20.59	In stock	5	https://boo...	https://books...	images\The_Inefficiency_Assassin_Time_Management_1
67	The Gutsy Girl: Escapades for Yo...	£37.13	In stock	1	https://boo...	https://books...	images\The_Gutsy_Girl_Escapades_for_Your_Life_of_Epi
68	The Electric Pencil: Drawings fro...	£56.06	In stock	1	https://boo...	https://books...	images\The_Electric_Pencil_Drawings_from_Inside_State
69	The Death of Humanity: and the ...	£58.11	In stock	4	https://boo...	https://books...	images\The_Death_of_Humanity_and_the_Case_for_Life
70	The Bulletproof Diet: Lose up to ...	£49.05	In stock	3	https://boo...	https://books...	images\The_Bulletproof_Diet_Lose_up_to_a_Pound_a_Da
71	The Art Forger	£40.76	In stock	3	https://boo...	https://books...	images\The_Art_Forgery.jpg
72	The Age of Genius: The Sevente...	£19.73	In stock	1	https://boo...	https://books...	images\The_Age_of_Genius_The_Seventeenth_Century_a
73	The Activist's Tao Te Ching: Anci...	£32.24	In stock	5	https://boo...	https://books...	images\The_Activist's_Tao_Te_Ching_Ancient_Advice_fo
74	Spark Joy: An Illustrated Master ...	£41.83	In stock	4	https://boo...	https://books...	images\Spark_Joy_An_Illustrated_Master_Class_on_the
75	Soul Reader	£39.58	In stock	2	https://boo...	https://books...	images\Soul_Reader.jpg
76	Security	£39.25	In stock	2	https://boo...	https://books...	images\Security.jpg
77	Saga, Volume 6 (Saga (Collecte...	£25.02	In stock	3	https://boo...	https://books...	images\Saga_Volume_6_Saga_Collected_Editions_6
78	Saga, Volume 5 (Saga (Collecte...	£51.04	In stock	2	https://boo...	https://books...	images\Saga_Volume_5_Saga_Collected_Editions_5
79	Reskilling America: Learning to L...	£19.83	In stock	2	https://boo...	https://books...	images\Reskilling_America_Learning_to_Labor_in_the_Tv
80	Rat Queens, Vol. 3: Demons (Rat...	£50.40	In stock	3	https://boo...	https://books...	images\Rat_Queens_Vol_3_Demons_Rat_Queens_Co
81	Princess Jellyfish 2-in-1 Omnibu...	£13.61	In stock	5	https://boo...	https://books...	images\Princess_Jellyfish_2_in_1_Omnibus_Vol_01_Pri

Figure 6: Preview of the books\_scraped table with scraped book details.

## 6.2 Deployment with Docker



Name	Container ID	Image	Port(s)	Actions
awesome_solon	97dbd9fcd672	my-backend-api	8081:8080	Stop, Restart, Delete
lucid_bartik	aa515a3774cb	my-backend-api	8081:8080	Stop, Restart, Delete
optimistic_albat	1a5b4ba392d4	my-backend-api	8080:8080	Stop, Restart, Delete
vigilant_moser	a3b4d5758f5c	my-backend-api	8080:8080	Stop, Restart, Delete
nervous_swirls	dc4a70626b3f	my-backend-api	8080:8080	Stop, Restart, Delete
quirky_allen	20157a2516e7	my-backend-api	8080:8080	Stop, Restart, Delete
heuristic_diffie	1cd57df2dad8	rest-api-flask-python	5005:5000	Stop, Restart, Delete

Figure 7: Running Docker containers, highlighting the active container.

## Glimpse of Docker Containers

The image above shows the running Docker containers, with the high-

lighted container representing the active one for this project. **Docker containers** are used to create isolated environments for the backend and database, ensuring consistent development and testing environments. **Docker Compose** manages the multi-container setup, ensuring seamless communication between the backend and database containers. This isolation also helps in eliminating potential conflicts between development environments and production environments.

**Docker Benefits for Deployment:** Using Docker for deployment provides several key benefits, including the ability to ensure that both the backend and database run in isolated environments, which mitigates issues related to configuration discrepancies across different systems. This setup also streamlines the process of scaling applications, as new containers can be easily spun up or removed. Furthermore, with Docker Compose managing the multi-container environment, the setup becomes more maintainable and portable, making it easier to deploy across various environments, such as development, staging, and production. <sup>6</sup>

---

<sup>6</sup>Heroku. (2023). Deploying Python Apps with Heroku. Heroku.

## *7 Challenges, Lessons, and Future Work*

This section discusses the challenges faced during the project, the lessons learned, and potential directions for future work. The project involved a mix of technical hurdles, development strategies, and insights into the publishing industry.

### 7.1 Key Challenges

Continuing with the development process would result in several challenges that might be critical to the success of the Tunisia Book Fair Project. Some of the key challenges include:

- Simplifying **web scraping** for static content. Dynamic content requires additional effort, such as working with APIs or browser automation tools.
- Debugging **integration issues** between the frontend, backend, and database to ensure smooth communication across all components.
- Ensuring **cross-browser compatibility** for the frontend to guarantee that the platform works across various devices and browsers.
- Convincing publishers to collaborate, particularly with digital content. This often involves overcoming resistance to sharing data, especially for small and independent publishers who are often wary of the idea of providing access to their book catalogs.
- The difficulty in ensuring **real-time updates** for book prices, availability, and locations. This is an ongoing issue as publishers often change their pricing or stock availability without notice.
- Managing the infrastructure for continuous scraping and crawling. To keep the data fresh, the scraping process requires high-performance servers

capable of handling large volumes of data and updating the book database frequently.

## 7.2 Lessons Learned

Throughout the project, several lessons were learned that enhanced the development process and deepened our understanding of both technical and business challenges. Some of the major lessons include:

- **Improved Development Workflow:** Using [Docker](#) allowed for creating isolated environments, which streamlined the application setup. This contributed to easier testing and faster development cycles.
- **API Debugging:** The team learned how to resolve integration issues between the frontend and backend more effectively by implementing better **error handling** and **logging** systems. This ensured smoother communication between the various components of the platform.
- **Scraping Complexity:** The complexities of scraping dynamic content from websites became more apparent. In the future, strategies for scraping will be focused on handling these challenges more efficiently, possibly by using headless browsers or more robust API calls.
- **Trust and Data Sensitivity:** A critical realization was the difficulty in gaining users' trust, especially for something as basic as registering and logging in. Many users will definitely be cautious about sharing personal information, and overcoming this skepticism is an ongoing challenge. For some users, the concept of an online platform offering book event schedules and related services is difficult to trust and they deem unworthy of taking the risk.

## 7.3 Future Work

The future direction of the project will focus on both technical improvements and expanding the scope of services provided to users. Key areas of future work include:

- **Enhancing the Frontend:** The frontend interface will be improved by adding more **interactive features**, making it more user-friendly. Future work will include enhancing accessibility features, such as support for [Braille displays](#) and **screen readers**.

- **Backend Scalability:** Improving the backend's scalability is crucial to handle larger datasets and more users. This includes improving database performance and optimizing server infrastructure to handle higher traffic loads, especially as the project expands.
- **Interactive Data Visualizations:** More advanced and interactive data visualizations will be added to the platform. These will help users engage with the data in new and meaningful ways.
- **Expanding Publisher Partnerships:** Efforts will be made to forge more partnerships with both traditional and digital publishers, enabling the platform to offer an even wider range of books and event schedules.
- **Real-Time Updates:** The ability to offer real-time updates on book prices, availability, and locations will be a key area of improvement. This will involve implementing better scraping techniques and potentially working with publishers to get real-time data feeds.

**Personal Reflections:** Throughout this project, I have faced several difficulties related to managing and processing large datasets. I have learned valuable technical skills, especially in web scraping, API integration, and database management. However, the most rewarding aspect of the project was discovering the complexity behind the book publishing industry and the unique challenges involved in digital book distribution. I have also developed a deeper understanding of the importance of user trust and how critical it is for the success of an online platform. <sup>7</sup>

---

<sup>7</sup>Analytics Vidhya. (2023). Data Science Tutorials. Analytics Vidhya.

## 8 Conclusion

**The Tunisian Book Fair** (TBF) was created to address significant challenges in Tunisia's reading landscape, particularly the increasing number of visually impaired individuals who often face barriers in accessing books. The name TBF, or "**To Be Fair**", reflects our vision of creating a platform that is fair and accessible to all, including those with disabilities. With features such as voice reader integration and future support for e-braille, we aim to make the experience of exploring and purchasing books truly inclusive.

By utilizing robust technologies like **Docker** and **Vercel**, TBF hopes to ensure seamless deployment, scalability, and a smooth browsing experience. The platform enables users to easily access and participate in book-related events, with a focus on accessibility and user-centric design. As we continue to improve and expand the platform's features, TBF strives to foster the growth of the Tunisian book market, empower local publishers, and **provide a connected space where users from all walks of life can explore the world of books**. This project stands as a testament to the power of technology, collaboration, and inclusivity in shaping a brighter future for Tunisian readers and preserving national literary and cultural treasures.

## References

- [1] Mitchell, R. (2018). *Web Scraping with Python: Collecting Data from the Modern Web*. O'Reilly Media.
- [2] Swathi, R. (2021). Techniques for Effective Web Scraping. *International Journal of Data Science*.
- [3] Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media.
- [4] Django Software Foundation. (2023). Django Documentation.
- [5] Meyer, E. (2018). *CSS: The Definitive Guide*. O'Reilly Media.
- [6] Bostock, M. (2016). Data-Driven Documents (D3.js). *Journal of Visualization and Interaction Design*.
- [7] Hamilton, M. (2020). *Database Systems: The Complete Guide*. Pearson Education.
- [8] AWS Documentation Team. (2023). AWS Deployment Best Practices.
- [9] McKinney, W. (2022). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter*. O'Reilly Media.
- [10] OpenAI. (2025). GPT Language Model for Assistance and Content Generation.