Port La Goulette Management RESTful API Report

LA-GOULETTE-PORT

Your doorway to Tunisia

Author:
Laabidi Rayen

Submitted to:
Prof. Montassar Ben Messaoued

January 6, 2025

0.1 Abstract

Port La Goulette, a cornerstone of Tunisia's maritime infrastructure, is not just a transit point but a vital gateway connecting Tunisia to the world. Despite its importance, the port faces critical challenges, including the lack of up-to-date information regarding delays in cruise schedules and the difficulty passengers face in finding transportation upon arrival. These gaps not only hinder operational efficiency but also create a disconnect between the port and its users, leading to frustration and missed opportunities for service providers and travelers alike.

To address these issues, the proposed solution is a RESTful API specifically designed to enhance operations and improve user experience at Port La Goulette. This API centralizes crucial information and automates real-time updates about cruise delays, ensuring passengers and stakeholders stay informed. Additionally, the API integrates features to assist passengers in locating and booking transportation options upon arrival, reducing stress and improving accessibility.

Beyond solving immediate challenges, the API offers advanced functionalities such as user management, ship scheduling, booking services, and interactive maps. By providing stakeholders with tools to manage and access critical data efficiently, the API fosters better coordination and engagement between the port, its users, and service providers.

This project is not just a technical upgrade but a transformative initiative aimed at modernizing Port La Goulette's operations and enhancing its digital presence. By addressing key pain points and offering innovative solutions, the API strengthens the port's connection with its users and reaffirms its role as a crucial gateway to Tunisia, paving the way for a more efficient, connected, and user-friendly maritime hub.

Contents

0.1 Abstract . . . .

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Statement 1
1.3 Solution 1
1.4 Background and Context 2
1.5 Scope of the Project 2
1.5.1 The scope of the project 2
1.5.2 Work-in-Progress 3
1.5.3 Out of Scope: 3
1.6 Taxonomy 3
1.6.1 Users 3
1.6.2 Entities 3
1.6.3 Processes 3
1.7 Study of Similar APIs and Originality 4
1.7.1 Existing Projects 4
1.7.2 Research on Similar APIs 4
1.7.3 Adaptation to Tunisian Context 5

2 System Architecture **6**
2.1 Backend Models 6
2.2 Database Design 6
2.2.1 Tables 6
2.2.2 Entity-Relationship Diagram 7
2.2.3 Data Collection and Usage 7
2.3 Methodology 8
2.3.1 API Endpoints 8
2.3.1.1 User Endpoints 8
2.3.1.2 Booking Endpoints 8
2.3.1.3 Parking Endpoints 8
2.3.1.4 Services Endpoints 9
2.3.1.5 Port Facility Endpoints 9
2.3.1.6 Ship Endpoints 9
2.3.1.7 Schedule Endpoints 9
2.3.1.8 Feedback Endpoints 10
2.3.2 Error Handling Strategies 10
2.3.3 Reuse of Existing APIs 10

3 Implementation and Deployment **11**
3.1 Security and User Authentication . . . 11
3.1.1 Authentication Methods . . . 11
3.1.2 Encryption and Data Protection Strategies . . . 11
3.2 Technology Stack . . . 12
3.2.1 Languages . . . 12
3.2.1.1 Python . . . 12
3.2.1.2 Front-end development . . . 12
3.2.2 Frameworks and Libraries . . . 12
3.2.2.1 Flask . . . 12
3.2.2.2 SQLAlchemy . . . 13
3.2.2.3 Alembic . . . 13
3.2.2.4 Werkzeug . . . 13
3.2.2.5 SQLite . . . 13
3.2.3 Documentation . . . 13
3.2.3.1 Swagger . . . 13
3.2.3.2 Pygments . . . 14
3.2.4 Environment and Tools . . . 14
3.2.4.1 Virtualenv . . . 14
3.2.4.2 Pip . . . 14
3.2.4.3 VS Code . . . 14
3.2.4.4 Git -for version control . . . 14
3.3 Perspective . . . 15
3.4 Conclusion . . . 15
3.5 Future Enhancements . . . 15

Chapter 1

Introduction

1.1 Motivation

Managing port operations efficiently is essential not only for economic growth but also for ensuring smooth and seamless service delivery to users. As a critical gateway for trade and travel, Port La Goulette plays a pivotal role in Tunisia's maritime activities. However, the current system relies heavily on outdated manual processes that are prone to inefficiencies and errors. These limitations create delays, hinder effective communication, and disrupt the overall experience for port users, businesses, and stakeholders.

1.2 Problem Statement

Port La Goulette currently lacks a centralized and up-to-date system for managing its diverse and dynamic operations. Key challenges include the absence of real-time updates on cruise delays and the difficulties passengers face in finding suitable transportation upon arrival. The reliance on manual processes exacerbates these issues, making operations time-consuming, error-prone, and insufficient for modern demands. These inefficiencies not only compromise the port's operational capacity but also negatively impact the user experience and stakeholder engagement.

1.3 Solution

The proposed solution is the development of a comprehensive RESTful API tailored specifically to the needs of Port La Goulette. This API aims to modernize port operations by centralizing and automating key processes, providing stakeholders with a streamlined and efficient way to manage activities. Core features of the API include:

- User Authentication and Management: Ensuring secure access to port services and enabling personalized user experiences.
- Ship Management: Facilitating real-time updates on ship arrivals, departures, and delays, improving communication and reducing uncertainty.
- Booking Services: Allowing users to reserve port services such as cargo handling, berthing, or passenger facilities with ease.

- Schedules and Notifications: Providing automated and accurate schedules, along with instant notifications for updates or changes.
- Interactive Maps: Assisting users in navigating the port and finding transportation options upon arrival.

By integrating these features, the API not only addresses existing challenges but also enhances operational efficiency and user satisfaction. It empowers stakeholders with data-driven insights and real-time updates, ensuring that the port remains competitive and customer-focused in a rapidly evolving digital landscape.

1.4 Background and Context

1.5 Scope of the Project

The project focuses on developing a RESTful API to serve as the backbone for modernizing and digitizing operations at Port La Goulette.

1.5.1 The scope of the project

- User Management:

  Secure user authentication and role-based access control for port staff, travelers, and transportation providers.

- Ship Management:

  Real-time tracking and management of ship schedules, including arrival, departure, and delays. Information retrieval for ships by type, status, or schedules.

- Booking System:

  Automating the booking process for port services, such as berth allocation and cargo handling. Integration of passenger booking for smoother travel experiences.

- Transportation Coordination:

  Providing real-time information about available transportation options for passengers arriving at the port. Ensuring travelers can plan their journeys efficiently.

- Interactive Features:

  An interactive map for visualizing port facilities, transportation hubs, and service points.

- Automation and Data Flow:

  Automating manual processes to minimize delays and errors.

1.5.2 Work-in-Progress

Building Front-End User Interfaces: Development of a front-end interface to utilize the API is underway. While this component remains a work-in-progress, it will eventually serve as a user-friendly platform for engaging with the API's features.

1.5.3 Out of Scope:

Large-scale infrastructure changes at the port itself (e.g., physical IoT installations). Real-time integration with external systems like customs or shipping company APIs (planned for future phases).

1.6 Taxonomy

1.6.1 Users

The API supports multiple user groups with distinct needs:

- **Passengers**: Travelers using port services who require updates on delays, bookings, and transportation options.
- **Port Staff**: Employees managing operations such as ship schedules, berth allocations, and passenger services.
- **Transportation Providers**: Service operators offering taxis, buses, and shuttles for passenger convenience.
- **Administrators**: Responsible for maintaining and supervising the system's data and functionality.

1.6.2 Entities

The API manages several critical entities, forming the backbone of port operations:

- **Ships**: Data on vessel types, schedules, statuses (e.g., delayed, on-time)...
- **Schedules**: Timetables for ship arrivals, departures, and updates on delays.
- **Bookings**: Reservations for passenger travel, cargo handling, or port facilities.
- **Transportation Options**: Availability and status of taxis, shuttles, and other transport services.
- **Facilities**: Details on port amenities, such as parking, lounges...

1.6.3 Processes

The API enables efficient automation and management of port activities through the following processes:

- **Real-Time Updates**: Tracking and communicating delays, departures, and arrivals to ensure accurate information dissemination.

- **Data Flow and Synchronization**: Centralized coordination of information across users and entities.
- **Decision-Making Support**: Tools to assist staff in optimizing berth assignments, scheduling, and transportation coordination.
- **Booking Automation**: Streamlined booking workflows for passengers and operational services.

1.7 Study of Similar APIs and Originality

1.7.1 Existing Projects

Several international port management platforms provide comprehensive services for port operations and user engagement. These platforms often serve as benchmarks for functionality, offering insights into best practices. Some notable examples include:

- **Port of San Diego** (https://www.portofsandiego.org/): Focuses on offering streamlined port services and community engagement.
- **Galataport Istanbul** (https://galataport.com/): Known for its integration of port services with cultural and tourism features.
- **Maritime and Port Authority of Singapore** (https://www.mpa.gov.sg/home): A leader in digital transformation and real-time updates for port activities.
- **Port Canaveral** (https://www.portcanaveral.com/): Offers a range of services including cruise management, cargo operations, and real-time information.
- **La Goulette Cruise Port** (https://www.lagoulettecruiseport.com/): The current platform for La Goulette port

While these platforms offer valuable features, they often face limitations in addressing specific cultural and logistical challenges, especially for ports in developing regions like Tunisia. Common gaps include limited real-time updates, lack of transportation service integration, and restricted user interaction capabilities.

1.7.2 Research on Similar APIs

To design a robust API for port management, a detailed analysis of APIs for transportation, scheduling, and logistics was conducted. Lessons learned include:

- The importance of providing real-time data updates, as seen in the Singapore Maritime and Port Authority system.
- Integration with external services such as transportation providers and tourist activities, as demonstrated by Galataport.

- Comprehensive user engagement features that cater to diverse user groups, including passengers, port staff, and transportation providers.

These insights emphasize the need for an adaptable API that not only automates processes but also connects multiple user groups through a unified interface.

1.7.3 Adaptation to Tunisian Context

Tunisia's unique cultural, logistical, and economic landscape necessitates significant adaptations to existing port management frameworks:

- **Cultural Challenges**: High reliance on manual processes and limited digital engagement among some user groups.
- **Logistical Challenges**: Frequent delays in cruise schedules and lack of real-time updates, leading to passenger dissatisfaction.
- **Economic Considerations**: The need for cost-efficient solutions that can integrate seamlessly with existing infrastructure.

To address these challenges, the proposed API incorporates features such as real-time schedule updates, transportation service integration, and a user-friendly interface to enhance engagement and efficiency. This ensures the API is not only functional but also relevant to the specific needs of Tunisia's port operations.

Chapter 2

System Architecture

This chapter discusses the backend models, database design, methodology, and API endpoints. It provides an in-depth explanation of the architecture used to develop the Port La Goulette Management API.

2.1 Backend Models

- **User Model**: Manages user authentication and roles (e.g., passenger, staff, admin).
- **Ship Model**: Tracks ship details such as name, type, capacity, and status (e.g., arrived, delayed, departed).
- **Booking Model**: Captures bookings made by users, including the service type and status.
- **Schedule Model**: Stores ship schedules, arrival and departure times.
- **Parking Model**: Represents parking slots and their availability at the port.
- **Feedback Model**: Collects user feedback for services or experiences.
- **Services Model**: Manages all port-related services, including transportation, logistics support, and administrative assistance.
- **Port Facility Model**: Represents infrastructure-related facilities at the port, such as restrooms, waiting areas, and cargo storage.

2.2 Database Design

2.2.1 Tables

- **Users**: Stores user information (e.g., name, email, role).
- **Ships**: Stores ship details (e.g., name, type, capacity).
- **Bookings**: Tracks bookings and their statuses.
- **Schedules**: Contains ship arrival and departure schedules.

- **Parking**: Manages vehicle parking slots and availability.
- **Feedback**: Stores user feedback and ratings.
- **Services**: Manages available services at the port, such as transportation and logistics support.
- **Port Facilities**: Tracks details of port infrastructure facilities (e.g., capacity, availability).

2.2.2 Entity-Relationship Diagram

The class diagram below illustrates the relationships between key components of the database.

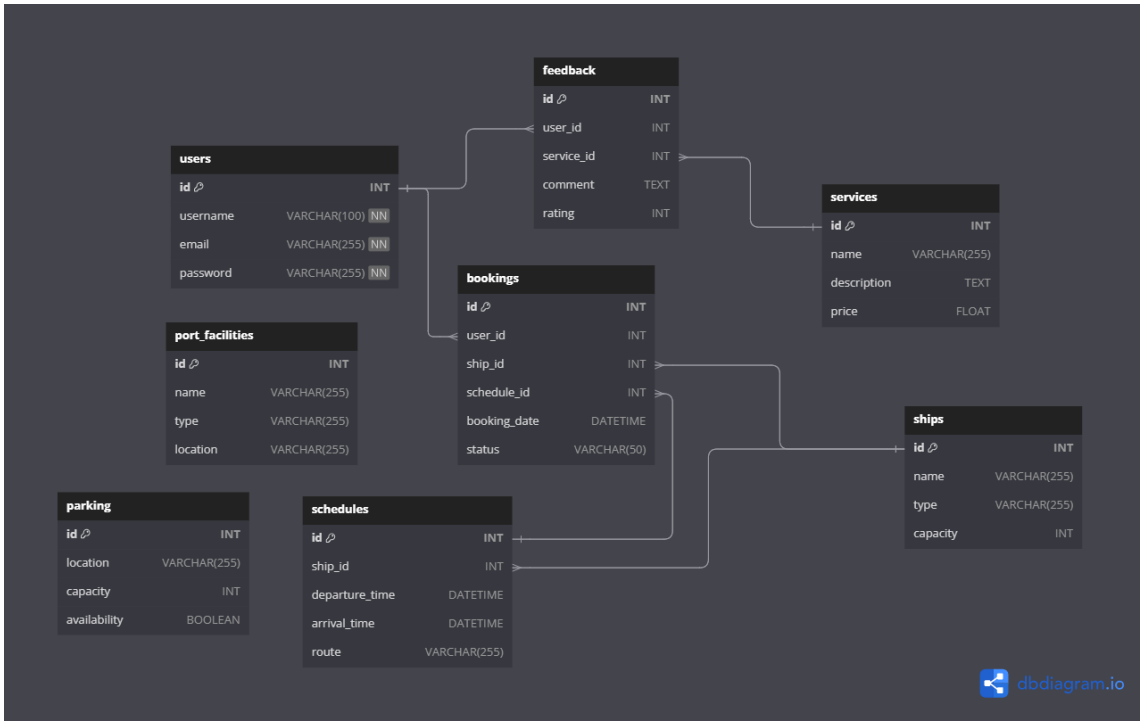

Figure 2.1: Entity-Relationship Diagram for the Port La Goulette Management API.

2.2.3 Data Collection and Usage

Data used in the API design was gathered from:

- Feedback from port users regarding common challenges.
- Integration of data from external sources such as the Google Maps API for interactive maps.
- Collaboration with platforms like inDrive for transportation-related services.

2.3 Methodology

This section describes the methodology adopted in the development of the Port La Goulette Management API. It covers the API's endpoint design, error handling strategies, and reuse of existing APIs to deliver an efficient, scalable, and user-centric system.

2.3.1 API Endpoints

The API is structured around modular and well-defined endpoints, categorized based on functionality:

2.3.1.1 User Endpoints

- **POST /users/register**: Registers a new user. This endpoint uses `werkzeug.security` for hashing user passwords. The hashing mechanism ensures that user passwords are stored securely in the database by converting them into irreversible hash values. Even in the event of a database breach, plaintext passwords cannot be retrieved.
- **POST /users/login**: Authenticates a user using their credentials, this endpoint validates the provided password against the hashed password stored in the database and returns an appropriate response to the user upon successful validation.
- **PUT /users/update/<int:user_id>**: Updates user information.
- **DELETE /users/<int:user_id>**: Deletes a user.

2.3.1.2 Booking Endpoints

- **POST /bookings**: Creates a new booking.
- **GET /bookings**: Retrieves all bookings.
- **DELETE /bookings/:id**: Cancels a booking.
- **GET /bookings/ship/:ship_id**: Retrieves all bookings for a specific ship.
- **GET /bookings/service/:service_id**: Retrieves all bookings for a specific service.

2.3.1.3 Parking Endpoints

- **GET /parkings**: Retrieves all parking spaces.
- **POST /parkings**: Creates a new parking space.
- **GET /parkings/<int:parking_id>**: Retrieves details of a specific parking space.
- **PUT /parkings/<int:parking_id>**: Updates a specific parking space.

- **DELETE /parkings/<int:parking_id>**: Deletes a specific parking space.
- **POST /parkings/book**: Books a parking space.

2.3.1.4 Services Endpoints

- **GET /services**: Lists all available services at the port.
- **POST /services**: Creates a new service.
- **GET /services/<int:service_id>**: Retrieves details of a specific service.
- **PUT /services/<int:service_id>**: Updates a specific service.

2.3.1.5 Port Facility Endpoints

- **GET /facilities**: Retrieves information about available port facilities.
- **POST /facilities**: Adds a new port facility.
- **GET /facilities/<int:facility_id>**: Retrieves details of a specific port facility.
- **PUT /facilities/<int:facility_id>**: Updates a specific port facility.

2.3.1.6 Ship Endpoints

- **GET /ships**: Lists all ships.
- **POST /ships**: Creates a new ship.
- **GET /ships/<int:ship_id>**: Retrieves details of a specific ship.
- **GET /ships/status/<string:status>**: Lists ships by status.
- **PUT /ships/<int:ship_id>/status**: Updates the status of a specific ship.
- **POST /ships/<int:ship_id>/service/<int:service_id>**: Links a ship to a service.

2.3.1.7 Schedule Endpoints

- **GET /schedules**: Lists all schedules.
- **POST /schedules**: Creates a new schedule.
- **GET /schedules/<int:schedule_id>**: Retrieves details of a specific schedule.
- **PUT /schedules/<int:schedule_id>**: Updates a specific schedule.
- **DELETE /schedules/<int:schedule_id>**: Deletes a specific schedule.
- **GET /schedules/ship/<int:ship_id>**: Retrieves schedules for a specific ship.

2.3.1.8 Feedback Endpoints

- **GET /feedbacks**: Lists all feedbacks.
- **POST /feedbacks**: Adds new feedback.
- **DELETE /feedbacks/<int:feedback_id>**: Deletes a specific feedback.

2.3.2 Error Handling Strategies

To ensure a reliable and user-friendly API, the following error-handling strategies were adopted:

- **Custom Error Codes**: Standardized error codes are used to help users and developers quickly identify and resolve issues. Each type of error is associated with a specific HTTP status code and a descriptive error message. For example:
  - `400 Bad Request`: Returned when the input data does not meet the API's requirements, such as missing mandatory fields.
  - `404 Not Found`: Used when a requested resource, like a specific booking or ship, cannot be found.
  - `500 Internal Server Error`: Indicates an unexpected error on the server, often logged for developer investigation.
- **Validation**: Input validation ensures that all data sent to the API meets predefined standards before being processed. Examples include:
  - Validating user email formats during registration (e.g., rejecting `"user@domain"` but accepting `"user@domain.com"`).
  - Rejecting empty or improperly formatted JSON payloads, such as missing required keys like `"ship_id"` or `"service_id"`.
- **Error Messages**: Clear and informative error messages improve user experience by providing actionable insights into what went wrong. Examples include:
  - A response like `"Invalid email format. Please provide a valid email address."` for input validation failures.
  - `"Booking ID 1234 does not exist."` for a 404 error when attempting to retrieve a nonexistent booking.

2.3.3 Reuse of Existing APIs

To enrich the functionality of the API and improve efficiency, external APIs were integrated:

- **Google Maps API**: Enables interactive map generation for navigation and transportation routes.
- **inDrive Platform API**: Facilitates transportation booking and management services.

Chapter 3

Implementation and Deployment

3.1 Security and User Authentication

Ensuring the security of user data and safeguarding the system from unauthorized access are critical aspects of the Port La Goulette Management API. This section outlines the authentication methods and data protection strategies employed in the project.

3.1.1 Authentication Methods

The project employs **Werkzeug**, a Python library widely used for secure password hashing. By using the `generate_password_hash` and `check_password_hash` functions provided by Werkzeug, passwords are never stored in plaintext, significantly reducing the risk of data breaches.

- **Password Hashing:** When a user registers, their password is hashed using Werkzeug's robust hashing algorithms (e.g., PBKDF2). The hash is then stored in the database instead of the plaintext password. For example, a password like `"MySecureP@ssw0rd"` would be hashed into a string like `"pbkdf2:sha256:150000`$ZKs1XeCh$`e5af..."`.
- **Password Verification:** During login, the hashed password stored in the database is retrieved and compared with the user-provided password using Werkzeug's `check_password_hash` function. This ensures that even if the database is compromised, the actual passwords remain secure. For instance, if a user enters their password as `"MySecureP@ssw0rd"`, it will be hashed again and matched with the stored hash to validate the login request.

3.1.2 Encryption and Data Protection Strategies

To protect user data and ensure secure communication, the project implements several encryption and data protection measures:

- **Secure Storage of Sensitive Data:** Sensitive information, such as passwords, is securely hashed and stored using tools like Werkzeug. Non-sensitive data, such as usernames and emails, are stored in plaintext but protected by access controls to prevent unauthorized retrieval.

- **Regular Security Updates:** The application is built with dependencies and libraries that are regularly updated to address known vulnerabilities. This ensures that the API remains secure against emerging threats.
- **Preventing SQL Injection:** All database queries are parameterized, ensuring that user inputs cannot be used to execute malicious SQL commands.

3.2 Technology Stack

3.2.1 Languages

3.2.1.1 Python

Python is a high-level, interpreted programming language known for its ease of use and readability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is widely used in web development, data analysis, machine learning, and automation. In this project, Python is used for developing the back-end logic and the API using the Flask framework.

3.2.1.2 Front-end development

HTML, CSS, and JavaScript are the foundational technologies for building the front-end of web applications. HTML is used to structure the web page, CSS is used to style it, and JavaScript is used to make it interactive. In this project, HTML, CSS, and JavaScript are used to build the user interface that communicates with the Flask API. These technologies are currently being used to create the front of the project(work in process).

3.2.2 Frameworks and Libraries

3.2.2.1 Flask

Flask is a lightweight web framework for Python, designed for building web applications and APIs quickly and with minimal overhead. It is based on the WSGI toolkit and the Jinja2 template engine, allowing developers to create dynamic websites and RESTful APIs. Flask is highly extensible and provides tools, libraries, and technologies to handle a wide range of development tasks. This project uses Flask for the creation of the web application and API endpoints.

3.2.2.2 SQLAlchemy

SQLAlchemy is an open-source ORM (Object Relational Mapper) for Python that allows developers to interact with relational databases using object-oriented programming principles. It provides a set of high-level API methods for connecting to and querying databases while abstracting away low-level SQL operations. SQLAlchemy simplifies database interactions and increases productivity for Python developers. In this project, it is used for handling database models and migrations.

3.2.2.3 Alembic

Alembic is a lightweight database migration tool for use with SQLAlchemy. It allows developers to manage changes to the database schema over time by generating migration scripts. Alembic is used in this project to handle schema changes and ensure the database structure is up-to-date as the project evolves.

3.2.2.4 Werkzeug

Werkzeug is a comprehensive library for Python that provides utilities for building web applications. It includes tools for request and response handling, URL routing, security features like password hashing, and much more. In this project, Werkzeug is used for password hashing, where it provides secure utilities such as PBKDF2 for hashing user passwords before storing them in the database.

3.2.2.5 SQLite

SQLite is a self-contained, serverless, and zero-configuration relational database engine. It is widely used for embedded applications and is an excellent choice for projects with low to moderate data storage needs. In this project, SQLite is used to store and manage the data for the web application, as it is lightweight and integrates well with SQLAlchemy.

3.2.3 Documentation

3.2.3.1 Swagger

Swagger is a set of open-source tools that helps design, build, document, and consume RESTful APIs. It enables developers to describe the functionality of an API in a standard format (OpenAPI Specification) and automatically generates interactive documentation that can be used to test the API directly from a web browser. In this project, Swagger is used for API documentation, making it easier to understand and interact with the various endpoints of the system.

3.2.3.2 Pygments

Pygments is a syntax highlighter for programming languages, markup languages, and other text-based formats. It is used to highlight code snippets in a variety of formats, including HTML, LaTeX, and Markdown. In this project, Pygments is used within the Swagger UI to provide syntax highlighting for API documentation, enhancing readability and user experience.

3.2.4 Environment and Tools

3.2.4.1 Virtualenv

Virtualenv is a tool for creating isolated Python environments. It allows developers to manage project dependencies in separate environments, ensuring that different projects can have different package versions without conflicts. In this project, Virtualenv is used to create an isolated environment for the project dependencies, ensuring that the system's global Python packages are not affected.

3.2.4.2 Pip

Pip is the package installer for Python. It is used to install and manage libraries and dependencies required for Python projects. In this project, Pip is used to install Flask, SQLAlchemy, Alembic, and other necessary dependencies.

3.2.4.3 VS Code

Visual Studio Code (VS Code) is a popular open-source code editor that is widely used for Python development. It supports various extensions for Python, Flask, and other tools, making it an ideal IDE for working on this project. VS Code provides features like syntax highlighting, debugging tools, version control integration, and support for virtual environments, which all contribute to a smooth development experience.

3.2.4.4 Git -for version control

Git is a distributed version control system that allows developers to track changes to their code, collaborate with others, and manage different versions of a project. In this project, Git is used for version control, enabling the development team to manage changes efficiently and collaborate effectively.

3.3 Perspective

The La Goulette Port RESTful API project demonstrates the integration of modern web technologies and logistics management. By using Flask, SQLAlchemy, and SQLite, the project benefits from a scalable and efficient approach to creating RESTful APIs, with Flask enabling quick iteration and SQLAlchemy simplifying database interactions. Werkzeug ensures secure user authentication. The project highlights the importance of efficient logistics, with dynamic route management and real-time updates improving port operations. Ultimately, it showcases how technology can optimize operations, enhance communication, and improve user experience in port management.

3.4 Conclusion

This project has been an incredibly rewarding experience, providing me with valuable insights into open-source projects, API design, and implementation. I gained hands-on experience in developing a RESTful API, managing databases, and ensuring security while using modern technologies like Flask, SQLAlchemy, and Werkzeug. I would like to express my gratitude to Prof. Montassar Ben Messoued for introducing us to the world of APIs and web services, and for offering guidance throughout this project. This journey has not only enhanced my technical skills but has also taught me the importance of creativity and persistence when solving real-world problems.

3.5 Future Enhancements

- User Experience Improvements:
  - Develop a front-end user interface to enhance interaction.
  - Add multi-language support to cater to a wider audience.
- Security Enhancements:
  - Implement two-factor authentication (2FA) to add an extra layer of security during the login process, ensuring that even if a password is compromised, an additional verification step is required.
  - Incorporate OAuth 2.0 for secure authorization, allowing users to sign in using third-party services (like Google or Facebook).
  - Implement rate-limiting and IP blocking to prevent brute-force attacks and ensure that malicious actors cannot overwhelm the system with login attempts or requests.
- email notifications to alert users about important updates
- create a dedicated mobile app

Bibliography

[1] Flask: https://flask.palletsprojects.com/en/2.1.x/

[2] SQLAlchemy: https://www.sqlalchemy.org/

[3] SQLite: https://www.sqlite.org/

[4] Werkzeug: https://werkzeug.palletsprojects.com/en/2.0.x/

[5] Marshmallow: https://marshmallow.readthedocs.io/en/stable/

[6] Alembic: https://alembic.sqlalchemy.org/en/latest/

[7] Swagger: https://swagger.io/

[8] Pygments: https://pygments.org/

[9] Git: https://git-scm.com/

[10] Visual Studio Code: https://code.visualstudio.com/

[11] Python: https://www.python.org/

[12] HTML: https://developer.mozilla.org/en-US/docs/Web/HTML

[13] CSS: https://developer.mozilla.org/en-US/docs/Web/CSS

[14] JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[15] Port of San Diego: https://www.portofsandiego.org/

[16] Galataport Istanbul: https://galataport.com/

[17] Maritime and Port Authority of Singapore: https://www.mpa.gov.sg/home

[18] Port Canaveral: https://www.portcanaveral.com/

[19] La Goulette Cruise Port: https://www.lagoulettecruiseport.com/