License Management and Validation System

Hey everyone!

Firstly, I want to apologize for not being able to upload the entire License Management and Validation System application on GitHub.
This is due to the size and some dependencies involved, so I've only added the key code files here to give you a clear understanding of the structure and implementation. **Thanks for understanding! 🙏**

What This Project Is About:

This is a License Management and Validation System that's designed to integrate with Software as a Service (SaaS) products. It ensures the licenses for these products are valid, securely managed, and stored in a centralized manner.
I've broken down the project into the key components, and here's what I've done so far:

Code Files Explanation:

1. `app.py` (Flask Backend for License Management)

This is the main server-side logic written in Python using Flask. It handles all incoming requests, like validating licenses, activating them, and managing the central license database.

- **Endpoints:**
  - `/license/validate`: Verifies if a license is valid and active.
  - `/license/activate`: Allows activation of a license (needs admin authentication).
  - It also connects to a SQLite database to store and retrieve license data.

2. `models.py` (Database Models)

This file defines the database schema.
I've built a simple License model with the following fields:

- **License Key**: The unique identifier for each product.
- **Product ID**: Identifies the associated SaaS product.
- **Status**: Tracks if the license is active or inactive.
- **Expiration Date**: When the license will expire.

It uses SQLAlchemy to create the tables and manage the database interactions.

3. `init_db.py` (Database Initialization Script)

This script is responsible for setting up the initial database schema and populating it with some test license data for quick setup.
Run this script when setting up the database for the first time so that the licenses are created and stored properly. It seeds a few inactive and active licenses into the database.

4. `middleware/licenseValidator.js` (Client-Side Middleware for SaaS Integration)

This is the client-side license validation middleware built for a Node.js/Express SaaS app. I've added this middleware to **validate licenses before any client-side operation is allowed**. It connects to the Flask backend to ensure that the provided license key is legit. If the license is valid, the client is allowed to continue using the SaaS product.

- **How it works**:
  - It reads the license key from the request header (`x-license-key`), then validates it via the Flask API.
  - If the license is valid, the client-side app can proceed with normal operations.

5. `auth.py` (Admin Authentication for License Management)

This file is critical for securing the admin-side operations like activating or revoking licenses. Admins have to **log in using this route** to generate a JWT token, which they'll use to authenticate further actions (like activating licenses).
Why I added this: Only admins should be allowed to activate or revoke licenses, and this ensures that only authorized people can do so.

6. Docker Support (Dockerfile and docker-compose.yml)

I've also provided a **Dockerfile** and **docker-compose.yml** to containerize the app for easier deployment.
This is helpful if you want to run the system across different environments without worrying about installation issues.

- **Dockerfile**: Defines the Python environment and Flask server setup.
- **docker-compose.yml**: Spins up the container and exposes the necessary ports to run the system.

7. Unit Tests (`test_app.py`)

I know how important testing is, so I've also added a simple test file.
This file uses Python's `unittest` module to run tests for the license validation and activation routes. These tests help ensure that the API behaves correctly and licenses are validated properly.

8. Database (SQLite)

The database schema (defined in `models.py`) is automatically generated when the Flask app is started. I've chosen **SQLite** because it's lightweight and easy to manage during

development.
You can switch to a production-level database later like PostgreSQL or MySQL if needed.

How to Run the Project:

1. **Install Dependencies**:
   You can install the required dependencies by running:

   ```
   bash
   Copy code
   pip install -r requirements.txt
   ```

2. **Initialize the Database**:
   Run the database initialization script:

   ```
   bash
   Copy code
   python init_db.py
   ```

3. **Start the Flask Server**:

   ```
   bash
   Copy code
   python app.py
   ```

4. **For Docker Users**:
   If you prefer Docker, just run:

   ```
   bash
   Copy code
   docker-compose up
   ```

5. **Testing**:
   Run the test file to ensure everything is working as expected:

   ```
   bash
   Copy code
   python -m unittest test_app.py
   ```

What's Missing?

Since this is just a part of the full system, there are still some features and files that I couldn't add here (like complete deployment scripts, full frontend, etc.).
If you have any issues or need clarification, feel free to reach out. I’m happy to help! 😊

Future Enhancements:

1. Adding **more comprehensive error handling**.
2. Enhancing the **Admin Dashboard** for better license management.
3. Implementing **role-based access control** for different admin levels.
4. Improving **performance** by scaling the validation process for thousands of licenses.
5. Adding **more detailed tests** to cover edge cases.

Again, thank you for your understanding, and I hope this code gives you a good insight into the License Management System I’ve built.

Cheers,
Lalit Kumar Chauhan