# Smart Cab Allocation

https://github.com/harshal-1/Smart-Cab

**Table of Contents**



**1. Introduction**

The Smart Cab application is a RESTful API for managing cab allocation and searching for nearby cabs. It utilizes JWT for authentication and Redis for caching cab locations, providing an efficient and secure solution for cab management.

**2. Features**

- User authentication with JWT

- Cab allocation based on employee location

- Search for nearby cabs

- Caching of cab locations using Redis

- Generation of random real-time cab locations

**3. Technologies Used**

- Flask: A lightweight WSGI web application framework for Python.

- Flask-JWT-Extended: Extension for Flask that provides JWT support.

- Redis: In-memory data structure store for caching.

- Random: Python module to generate random real-time locations.

**4. Application Structure**

- **app.py**: Main application file containing the API endpoints.

```python
from flask import Flask, jsonify, request
from flask_jwt_extended import JWTManager, create_access_token, jwt_required
from auth import authenticate_user
from cab_allocation import allocate_cab, search_nearby_cabs
from monitoring import log_event

app = Flask(__name__)
app.config['JWT_SECRET_KEY'] = 'your_secret_key'
jwt = JWTManager(app)
```

```python
@app.route('/login', methods=['POST'])
def login():
    username = request.json.get('username')
    password = request.json.get('password')
    if authenticate_user(username, password):
        access_token = create_access_token(identity=username)
        log_event(f"User {username} logged in")
        return jsonify(access_token=access_token)
    return jsonify({"msg": "Bad username or password"}), 401

@app.route('/allocate_cab', methods=['POST'])
@jwt_required()
def allocate():
    employee_location = request.json.get('employee_location')
    allocated_cab = allocate_cab(employee_location)
    return jsonify(allocated_cab=allocated_cab)

@app.route('/search_nearby_cabs', methods=['GET'])
@jwt_required()
def search():
    employee_location = request.args.get('location')
    cabs = search_nearby_cabs(employee_location)
    return jsonify(cabs=cabs)
```

- **auth.py**: Handles user authentication.

```python
users = {"admin": "password123"}  # Example user database

def authenticate_user(username, password):
    return users.get(username) == password
```

- **cache.py**: Provides caching functionality for cab locations.

```python
import redis

cache = redis.StrictRedis(host='localhost', port=6379, db=0)

def cache_cab_location(cab_id, location):
    cache.set(cab_id, location)

def get_cached_cab_location(cab_id):
    return cache.get(cab_id)
```

- **cab_allocation.py**: Manages cab allocation and searches for nearby cabs.

```python
import heapq

cabs = {
    "cab1": (10, 20),
    "cab2": (15, 25),
    "cab3": (30, 35),
}

def find_nearest_cab(cabs, location):
    nearest_cabs = []
    for cab_id, cab_location in cabs.items():
        distance = ((cab_location[0] - location[0]) ** 2 +
                    (cab_location[1] - location[1]) ** 2) ** 0.5
        heapq.heappush(nearest_cabs, (distance, cab_id))
    return heapq.heappop(nearest_cabs)[1]  # Return nearest cab ID

def allocate_cab(employee_location):
    # This function allocates a cab based on nearest cab to the employee's location
    employee_location = tuple(map(int, employee_location.split(',')))
    return find_nearest_cab(cabs, employee_location)

def search_nearby_cabs(employee_location):
    employee_location = tuple(map(int, employee_location.split(',')))
    return [cab for cab in cabs if find_nearest_cab(cabs, employee_location)]
```

- **real_time_location.py**: Simulates the generation of random real-time locations for cabs.

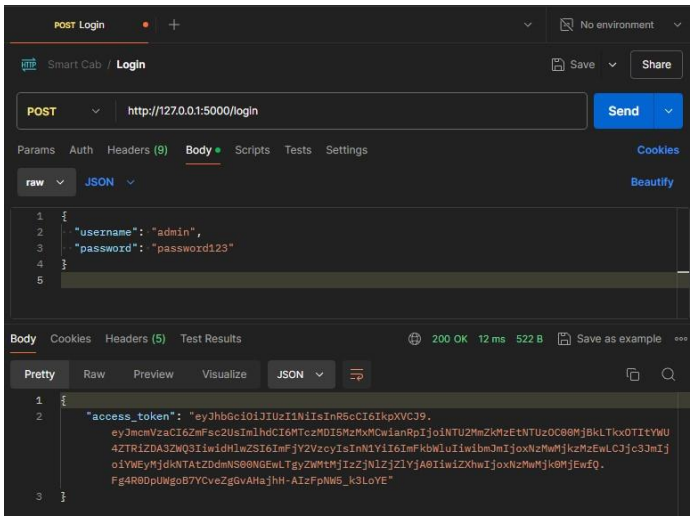
```python
def get_real_time_location():
    latitude = round(random.uniform(-90, 90), 5)
    longitude = round(random.uniform(-180, 180), 5)
    return latitude, longitude

# Example usage
cabs = {
    "cab1": get_real_time_location(),
    "cab2": get_real_time_location(),
    "cab3": get_real_time_location(),
}
```

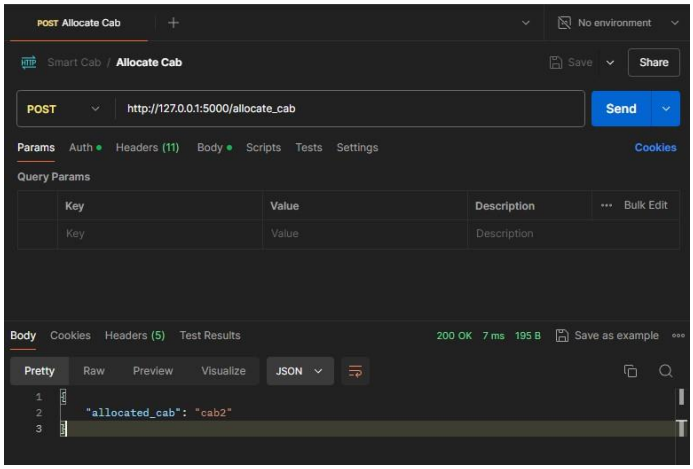
- **monitoring.py**: (Assumed to handle logging events related to user actions).

## 5. API Endpoints

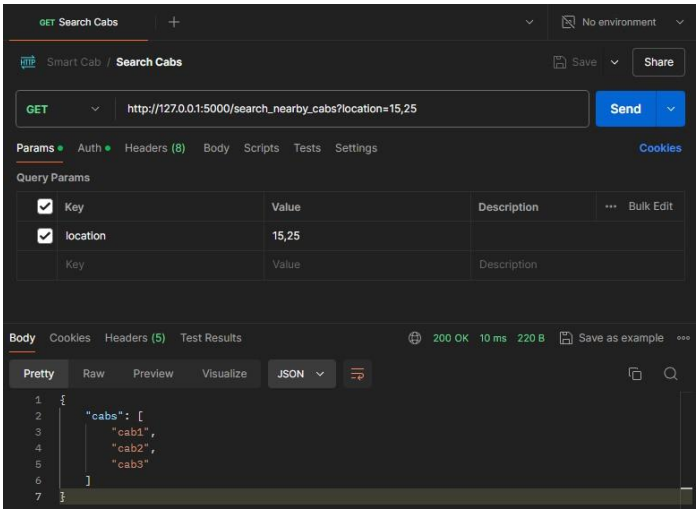
### Login :




### Allocate Cab :




### Search Cab :



## 6. Conclusion

The Smart Cab Application provides a robust solution for cab management, with features that enhance user experience through efficient cab allocation and searching capabilities. Future improvements could include integrating real-time tracking and expanding the user authentication system.