Backend Internship Assignment – Solution Report

Candidate Name: Spandan Bhattarai
Date: 2025-07-25
Role: Backend Intern
Tech Stack: Python (Flask), SQLite, Postman

Assignment Overview

The test consisted of two tasks:

1) Design and implement RESTful blog APIs with CRUD capabilities.
2) Write a function to find a pair in an array that sums to a given target.

Task 1: Blog API Development

Each blog should contain:

- title (string)
- description (string)
- category (string; one per blog)

API Endpoints Implemented

Endpoint	Method	Description
/blogs	GET	Get all blogs
/blogs/<id>	GET	Get a blog by its ID
/blogs	POST	Create a new blog
/blogs/<id>	PUT	Update an existing blog

Code Snippet: Flask App

```python
from flask import Flask, request, jsonify, abort
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///blogs.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

class Blog(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    description = db.Column(db.Text, nullable=False)
    category = db.Column(db.String(50), nullable=False)

    def to_dict(self):
        return {"id": self.id, "title": self.title, "description": self.description, "category": self.category}

@app.before_first_request
def create_tables():
    db.create_all()

@app.route("/blogs", methods=["GET"])
def get_all_blogs():
    blogs = Blog.query.all()
    return jsonify([blog.to_dict() for blog in blogs])

@app.route("/blogs/<int:id>", methods=["GET"])
def get_blog(id):
    blog = Blog.query.get_or_404(id)
    return jsonify(blog.to_dict())

@app.route("/blogs", methods=["POST"])
def create_blog():
    data = request.get_json()
    new_blog = Blog(title=data["title"], description=data["description"], category=data["category"])
    db.session.add(new_blog)
    db.session.commit()
    return jsonify(new_blog.to_dict()), 201

@app.route("/blogs/<int:id>", methods=["PUT"])
def update_blog(id):
    blog = Blog.query.get_or_404(id)
    data = request.get_json()
    blog.title = data.get("title", blog.title)
    blog.description = data.get("description", blog.description)
    blog.category = data.get("category", blog.category)
    db.session.commit()
    return jsonify(blog.to_dict())

if __name__ == "__main__":
    app.run(debug=True)

```

Example POST Request Body (Postman)

```
{
  "title": "Intro to Flask",
  "description": "Flask is a lightweight WSGI web application framework.",
  "category": "Python"
}
```

Task 2: Find Pair with Given Sum

Problem Statement

Given an unsorted list of integers and a target sum, find whether a pair exists that adds up to the target.

Optimized Approach

- Time complexity: O(n) using a hash set().
- Avoids nested loops by storing seen values and checking complements.

Code Implementation

```
def find_pair_with_sum(nums, target):
    seen = set()
    for num in nums:
        complement = target - num
        if complement in seen:
            print(f"Pair found ({complement}, {num})")
            return
        seen.add(num)
    print("Pair not found.")
```

Example Inputs and Outputs

```python
# Example usage
find_pair_with_sum([8, 7, 2, 5, 3, 1], 10)
find_pair_with_sum([5, 2, 6, 8, 1, 9], 12)
find_pair_with_sum([1, 2, 3, 4, 5], 9)
find_pair_with_sum([10, 20, 30, 40, 50], 100)
find_pair_with_sum([-3, -1, 2, 5, 8], 7)

```

```
PS C:\Users\sbhat\Downloads\backend_assignment_solution> python pair_sum.py
Pair found (8, 2)
Pair not found.
Pair found (4, 5)
Pair not found.
Pair found (2, 5)
PS C:\Users\sbhat\Downloads\backend_assignment_solution>
```