Project Test Plan

1.0 Introduction

This document outlines the testing strategy for the Internship Placement Manager API. The goal is to ensure the API's functionality, security, and data integrity meet the project's requirements. All tests are conducted using the **Pytest** framework.

2.0 Testing Environment

- **Testing Framework:** Pytest
- **Database:** An in-memory **SQLite** database is used for testing, ensuring a clean state for each test run. This prevents tests from interfering with each other or with the production database.
- **Test Client:** FastAPI's TestClient is used to simulate HTTP requests to the API endpoints.

3.0 Test Approach

The testing approach focuses on **integration testing** of the API endpoints. This involves testing the full request-response cycle, from the client request through the API router, business logic, and database interaction. The current test suite, tests/test_endpoints.py, covers the following:

3.1 Test Cases Implemented

The test plan is structured to validate the **CRUD (Create, Read, Update, Delete)** operations for each of the major resources.

Test Case	**Description**	**Expected Result**
test_root	Verifies that the root endpoint (/) returns a welcome message.	HTTP Status Code 200 with the expected JSON message.
test_students_crud	Tests the creation, retrieval, updating, and deletion of a student.	Successful status codes (201, 200, 204) and correct data payloads. A 404 status code is expected after deletion.
test_mentors_crud	Tests the CRUD operations for mentors.	Same as student tests, confirming successful

		POST, GET, PATCH, and DELETE.
test_employers_crud	Tests the CRUD operations for employers.	Same as student and mentor tests.
test_placement_and_evaluation_crud	An end-to-end test that ensures placements and evaluations can be created and managed.	Successful CRUD operations for both placements and evaluations, and correct relationships between entities.

4.0 Future Enhancements

- **Authentication and Authorization Tests:** Add tests to ensure that only authenticated and authorized users can access specific endpoints. For example, verify that an unauthenticated request to a protected route returns a 401 Unauthorized status code.
- **Negative Test Cases:** Implement tests that send invalid data (e.g., incorrect email format, missing required fields) to ensure the API handles errors gracefully and returns appropriate 400 Bad Request responses.
- **Edge Case Testing:** Test scenarios like trying to delete a user who has active placements or evaluations to ensure the application's logic handles these dependencies correctly.