# Тест-кейсы для загруженных файлов

## Файл: test\_init\_db.py

### Тест-кейс: Проверка инициализации базы данных

Цель: Убедиться, что таблицы 'weather' и 'cities' создаются корректно.

Шаги:

1. 1. Создать подключение к тестовой базе данных.
2. 2. Вызвать функцию `init\_db()`.
3. 3. Проверить наличие таблиц 'weather' и 'cities'.

Ожидаемый результат: Таблицы 'weather' и 'cities' созданы.

## Файл: test\_get\_weather\_data\_failure.py

### Тест-кейс: Обработка ошибок API

Цель: Проверить, что функция корректно обрабатывает недоступные данные о погоде.

Шаги:

1. 1. Создать заглушку для API, возвращающую код состояния 404.
2. 2. Вызвать функцию `get\_weather\_data()` с несуществующим городом.
3. 3. Проверить, что возвращаемое значение равно `None`.

Ожидаемый результат: Функция возвращает `None` при ошибке API.

## Файл: test\_get\_favorites.py

### Тест-кейс: Получение списка избранных городов

Цель: Убедиться, что функция возвращает корректный список избранных городов.

Шаги:

1. 1. Инициализировать базу данных с помощью `init\_db()`.
2. 2. Добавить города 'Berlin' и 'London' в избранное с помощью `save\_city()`.
3. 3. Вызвать функцию `get\_favorites()` и проверить наличие этих городов.

Ожидаемый результат: Города 'Berlin' и 'London' присутствуют в списке избранных.

## Файл: test\_save\_city.py

### Тест-кейс: Добавление города в избранное

Цель: Убедиться, что город добавляется в избранное и дублирование исключено.

Шаги:

1. 1. Инициализировать базу данных с помощью `init\_db()`.
2. 2. Добавить город 'Paris' в избранное с помощью `save\_city()`.
3. 3. Проверить наличие города 'Paris' в таблице.
4. 4. Повторно добавить город 'Paris' и убедиться в отсутствии дублирования.

Ожидаемый результат: Город 'Paris' добавлен единожды, дублирование отсутствует.

## Файл: test\_get\_weather\_data\_success.py

Тест-кейс: Успешное получение данных о погоде  
  
Цель: Проверить, что функция `get\_weather\_data` корректно получает данные о погоде при успешном запросе.  
  
Шаги:  
1. Создать подключение к тестовой базе данных.  
2. Создать заглушку для API с корректным ответом (код состояния 200 и данные о погоде).  
3. Вызвать функцию `get\_weather\_data()` для города 'Moscow'.  
4. Проверить, что возвращены корректные данные о температуре и влажности.  
  
Ожидаемый результат: Функция возвращает данные о погоде для города 'Moscow' с температурой 10°C и влажностью 80%.