Contents

[Requirement 2](#_Toc119086421)

[Setup 2](#_Toc119086422)

[Prerequisite 2](#_Toc119086423)

[Installation 2](#_Toc119086424)

[Testing core functionality 3](#_Toc119086425)

[Scenario 1: There is indicator that the API is listening Result: Pass 4](#_Toc119086426)

[Scenario 2: Post Customer/ save customer to DB as per swagger Result: Pass 4](#_Toc119086427)

[Scenario 3: Get Customer/{id} record as per swagger Result: Pass 4](#_Toc119086428)

[Scenario 4: Deletion record \*optional Requirement Result: Pass 4](#_Toc119086429)

[Test Error Handling. 3](#_Toc119086430)

[Scenario 1. Response 500 for All internal errors including external connectivity errors. Result: Pass 3](#_Toc119086431)

[Scenario 2. Response 400 for Any generic client errors (e.g. missing mandatory request parameters). Result: Pass 4](#_Toc119086432)

[Mocha Test Case 5](#_Toc119086433)

ANZ coding challenge

# Requirement

**Instructions**

Please follow the Swagger spec and build a service using Nodejs. The application should run on a local machine using npm start command.

**Acceptance criteria**

·         Service should validate input and on input errors a BadRequest should be returned as per Swagger.

·         Service should save data in DB (in memory DB or file or whatever you like) and return success or failure as per error codes in Swagger

·         Service should retrieve data from DB using the customer id passed in the path params (refer to Swagger)

·         Service needs to have test cases (Unit and Integration) written using Mocha and Chai. You can mock the responses where you deem necessary.

**Other considerations**

·         Use Promises or async/ await for IO operations

·         Apply best coding practices like DRY, KISS, DI etc

·         Write test case for as much code as possible

o   Use mocking for unit tests

·         Write comments where it makes sense.

·         Create a read me file with instructions on how to run the program

# Setup

## Prerequisite

* Windows OS
* NodeJS
* SQL Server Express, with mix SQL Server and Windows Authentication
* Postman (optional)

Installation

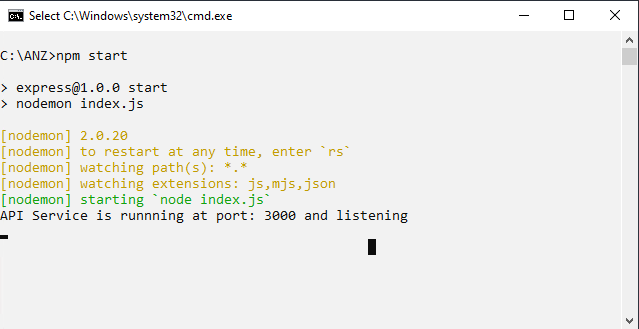
1. Download source code from GitHub.  
   https://github.com/andrewcsa/ANZ\_CodeChellenge

|  |  |
| --- | --- |
| **File name** | **Description** |
| index.js | The main script in the runtime to execute API method. |
| dbconfig.js | hold DB connection value |
| dboperation.js | functions of SQL statement to be executed when the API method is called. |
| commonclass.js | Collection of Class constructors |
| commonlib.js | Collection of functions |
| DB\_prep folder >  Create\_CustomerTable.sql | to create customer table with column as per swagger. |
| DB\_prep folder >  Create\_sp\_InsertCustomer.sql | to create store Prod for inserting new record. |
| Test > test\_customerApi.js | Mocha test script |

1. Execute 2 SQL files from folder DB\_Prep.
2. Using command prompt go into the downloaded foot folder and install package nodejs module dependency “npm install”
3. Open dbconfig.js and enter the DB connection details.
4. Run application with “npm start”

# Testing core functionality

## Scenario 1: There is indicator that the API is listening Result: Pass



## Scenario 2: Post Customer/ save customer to DB as per swagger Result: Pass

1. Check the record is not in DB

Graphical user interface, text, application, email

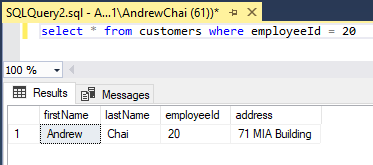
Description automatically generated

1. Run Post API

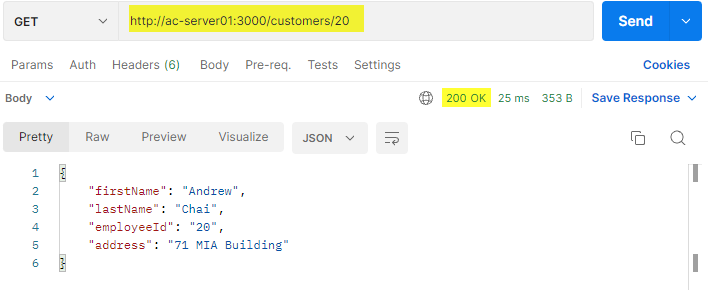
Graphical user interface, text, application, email

Description automatically generated

1. Check the record is in DB



## Scenario 3: Get Customer/{id} record as per swagger Result: Pass



### 

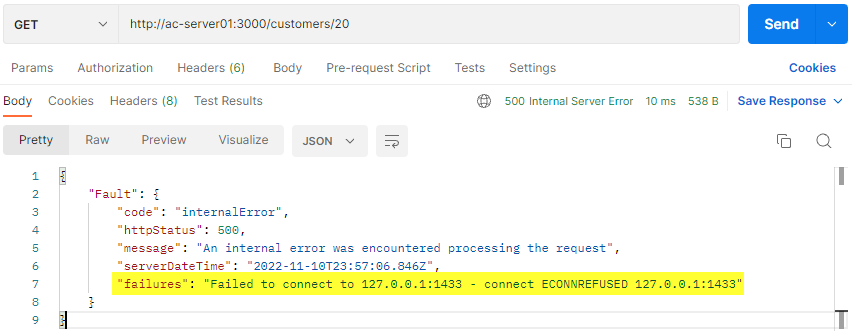
## Scenario 4: Deletion record \*optional Requirement Result: Pass

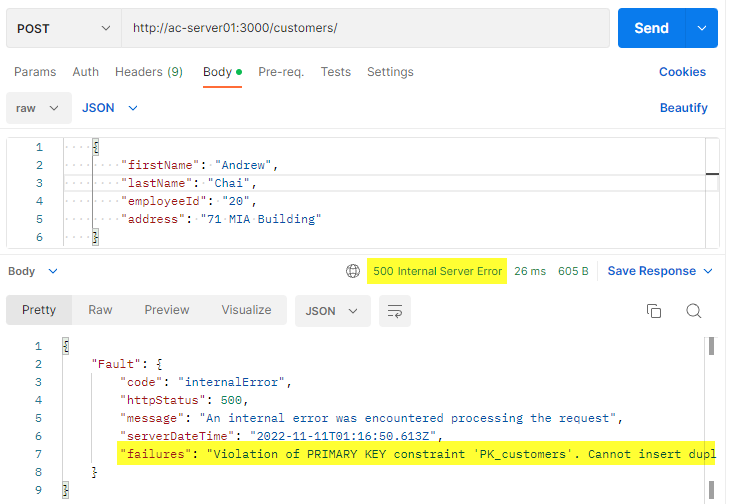
Graphical user interface, text, application, email

Description automatically generated

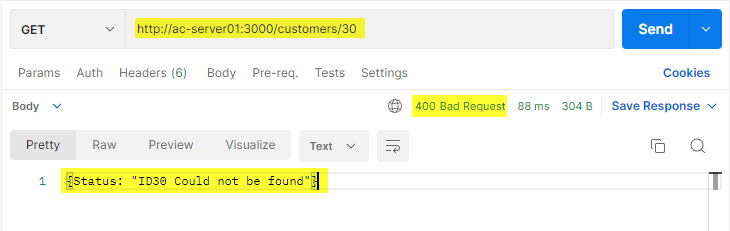
# Test Error Handling.

## Scenario 1. Response 500 for All internal errors including external connectivity errors. Result: Pass

Test 1: Unable to connect to DB  


Test 2: Duplicate record during create  


## Scenario 2. Response 400 for Any generic client errors (e.g. missing mandatory request parameters). Result: Pass



# Mocha Test Case

Text

Description automatically generated