

# SE 3XA3: Test Report

## GoDBMS

Team #7, Databased  
Faiq Ahmed, ahmedf46  
Eesha Qureshi, qureshe  
Kevin Kannammalil, kannammk

April 12th, 2022

# Contents

<b>1</b>	<b>Functional Requirements Evaluation</b>	<b>3</b>
1.1	Input Parsing . . . . .	3
1.2	Saving Data . . . . .	9
<b>2</b>	<b>Non Functional Requirements Evaluation</b>	<b>20</b>
2.1	Look and Feel Testing . . . . .	20
2.2	Usability and Humanity Testing . . . . .	21
2.3	Performance Testing . . . . .	21
2.4	Performance Testing . . . . .	22
<b>3</b>	<b>Comparison to Existing Implementation</b>	<b>23</b>
<b>4</b>	<b>Unit Testing</b>	<b>23</b>
<b>5</b>	<b>Changes Due To Testing</b>	<b>23</b>
<b>6</b>	<b>Automated Testing</b>	<b>23</b>
<b>7</b>	<b>Trace to Requirements</b>	<b>25</b>
<b>8</b>	<b>Trace To Modules</b>	<b>29</b>

# List of Tables

<b>1</b>	<b>Revision History</b> . . . . .	<b>ii</b>
<b>2</b>	<b>Table of Abbreviations</b> . . . . .	<b>1</b>
<b>3</b>	<b>Table of Definitions</b> . . . . .	<b>2</b>
<b>4</b>	<b>Trace Between Requirements and Modules</b> . . . . .	<b>29</b>

# List of Figures

<b>1</b>	<b>Traceability Matrix for Functional Requirements in BE1 to BE4</b> . . . . .	<b>25</b>
<b>2</b>	<b>Traceability Matrix for Functional Requirements in BE5 to BE8</b> . . . . .	<b>26</b>
<b>3</b>	<b>Traceability Matrix for Nonfunctional Requirements in NFR-1 to NFR-8</b> . . . . .	<b>27</b>

**Table 1: Revision History**

<b>Date</b>	<b>Version</b>	<b>Notes</b>
April 11, 2022	1.0	Initial Document
April 11, 2022	1.2	Started Test Cases
April 12, 2022	1.2	Finished Test Cases
April 12, 2022	1.2	Completed Document

The purpose of this document is to provide the outcome of the testing methods that were laid out in the Test Plan document. This document includes the assessment of how successful the Functional and Non Functional requirements that were created during the development process and the traceability of those requirements throughout the process.

**Table 2: Table of Abbreviations**

<b>Abbreviation</b>	<b>Definition</b>
SRS	Software Requirements Specification; a document that outlines important information about the software project such as stakeholders, functional requirements, non-functional requirements and project scope
POC	Proof of Concept; a prototype, demo or design that provides evidence of the project's feasibility
DBMS	<b>Database</b> Management System; software that allows a user to perform <b>database</b> transactions and operations
CLI	Command Line Interface; the point of communication between the software and the user
Go	GoLang

Table 3: **Table of Definitions**

Term	Definition
<b>Database</b>	A set of <b>records</b> that store information organized by keys
<b>Command Line</b>	The point of communication between user and internal computer instructions
<b>Table</b>	An organized collection of <b>records</b> in a <b>database</b>
<b>Record</b>	A single row of data in a <b>database</b>
<b>Primary Key</b>	A unique identifier for a <b>record</b> in a <b>database</b>
<b>Struct</b>	A modular collection of fields in GoLang
<b>Schema</b>	An outline of fields, table names, and relationships in a <b>database</b>
<b>Columns</b>	Vertical aggregations of data in a table
<b>Catalog</b>	A list storing the names of all the tables in the database. This list can be encoded and decoded from a file to allow persistent storage.
<b>Tuple</b>	An immutable finite sequence of elements
<b>Message Passing</b>	The method of invoking a process in a computer
<b>Concurrency</b>	The use of threads to run multiple computer processes at the same time
<b>GoLang</b>	A compiled programming language

# 1 Functional Requirements Evaluation

## 1.1 Input Parsing

<b>Test #1:</b>	<b>IP-T1</b>
<b>Type:</b>	Functional, Dynamic, Manual
<b>Initial State:</b>	The command line interface has been started and is waiting for a user input
<b>Input:</b>	User enters an input that is not a create table, insert, delete, list or search statement
<b>Output:</b>	Invalid query error appeared
<b>Expected:</b>	The system must print an error back to the user stating that their query syntax is invalid
<b>Result:</b>	PASS

<b>Test #2:</b>	<b>IP-BE1-T1</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to create a table will be passed in to the parser testing function with a missing table name
<b>Output:</b>	Missing table name error appeared
<b>Expected:</b>	The function must return an error stating that the table name is missing
<b>Result:</b>	PASS

<b>Test #3:</b>	<b>IP-BE1-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to create a table will be passed in to the parser testing function with a missing columns
<b>Output:</b>	Missing columns error appeared
<b>Expected:</b>	The function must return an error stating that the table columns is missing
<b>Result:</b>	PASS

<b>Test #4:</b>	<b>IP-BE1-T3</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to create a table will be passed in to the parser testing function with a missing primary key
<b>Output:</b>	Missing primary key error appeared
<b>Expected:</b>	The function must return an error stating that the table primary key is missing
<b>Result:</b>	PASS

<b>Test #5:</b>	<b>IP-BE1-T4</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to create a table will be passed in to the parser testing function with the correct syntax
<b>Output:</b>	Struct of the query was retrieved
<b>Expected:</b>	The function must return a struct that contains the schema of the create table statement
<b>Result:</b>	PASS

<b>Test #6:</b>	<b>IP-BE2-T1</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to delete a table will be passed in to the parser testing function with a missing table name
<b>Output:</b>	Missing table name error appeared
<b>Expected:</b>	The function must return an error stating that the table name is missing
<b>Result:</b>	PASS

<b>Test #7:</b>	<b>IP-BE2-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to delete a table will be passed in to the parse testing function with the correct syntax
<b>Output:</b>	Received a struct with the schema of the delete table statement
<b>Expected:</b>	The function must return a struct that contains the schema of the delete table statement
<b>Result:</b>	PASS

<b>Test #8:</b>	<b>IP-BE4-T1</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to insert a record will be passed in to the parser testing function with a missing table name
<b>Output:</b>	Missing table name error appeared
<b>Expected:</b>	The function must return an error stating that the table name is missing
<b>Result:</b>	PASS

<b>Test #9:</b>	<b>IP-BE4-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to insert a tuple will be passed in to the parser testing function with the missing columns
<b>Output:</b>	Missing columns error appeared
<b>Expected:</b>	The function must return an error stating that the record columns are missing
<b>Result:</b>	PASS

<b>Test #10:</b>	<b>IP-BE4-T3</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to insert a tuples will be passed in to the parser testing function with missing record data
<b>Output:</b>	Missing record data error appeared
<b>Expected:</b>	The function must return an error stating that the record data is missing
<b>Result:</b>	PASS

<b>Test #11:</b>	<b>IP-BE4-T4</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to insert a record will be passed in to the parser testing function with the correct syntax
<b>Output:</b>	Received struct with the schema of the insert record statement
<b>Expected:</b>	The function must return a struct that contains the schema of the insert record statement
<b>Result:</b>	PASS

<b>Test #12:</b>	<b>IP-BE5-T1</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to modify a record will be passed in to the parser testing function with a missing table name
<b>Output:</b>	Missing table name error appeared
<b>Expected:</b>	The function must return an error stating that the table name is missing
<b>Result:</b>	PASS

<b>Test #13:</b>	<b>IP-BE5-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to modify a record will be passed in to the parser testing function with a missing primary key to uniquely identify the tuple
<b>Output:</b>	Missing primary key error appeared
<b>Expected:</b>	The function must return an error stating that the record primary key is missing
<b>Result:</b>	PASS

<b>Test #14:</b>	<b>IP-BE5-T3</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to insert a tuple will be passed in to the parser testing function with the missing data to update
<b>Output:</b>	Missing record data error appeared
<b>Expected:</b>	The function must return an error stating that the record data to update is missing
<b>Result:</b>	PASS

<b>Test #15:</b>	<b>IP-BE5-T4</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to modify a record will be passed in to the parser testing function with the correct syntax
<b>Output:</b>	Received the struct of the schema for the modify record statement
<b>Expected:</b>	The function must return a struct that contains the schema of the modify record statement
<b>Result:</b>	PASS

<b>Test #16:</b>	<b>IP-BE6-T1</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to delete a record will be passed in to the parser testing function with a missing table name
<b>Output:</b>	Missing table name error appeared
<b>Expected:</b>	The function must return an error stating that the table name is missing
<b>Result:</b>	PASS

<b>Test #17:</b>	<b>IP-BE6-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to delete a record will be passed in to the parser testing function with a missing primary key to uniquely identify the tuple
<b>Output:</b>	Missing primary key error appeared
<b>Expected:</b>	The function must return an error stating that the record primary key is missing
<b>Result:</b>	PASS

<b>Test #18:</b>	<b>IP-BE6-T3</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to delete a record will be passed in to the parser testing function with the correct syntax
<b>Output:</b>	Received a struct of the schema for the delete record statement
<b>Expected:</b>	The function must return an error stating that contains the schema of the delete record statement
<b>Result:</b>	PASS

<b>Test #19:</b>	<b>IP-BE7-T1</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to search for records will be passed in to the parser testing function with missing table names
<b>Output:</b>	Missing table name error appeared
<b>Expected:</b>	The function must return an error stating that the table names to search from is missing
<b>Result:</b>	PASS

<b>Test #20:</b>	<b>IP-BE7-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to search for records will be passed in to the parser testing function with missing table columns to return
<b>Output:</b>	Missing table column error appeared
<b>Expected:</b>	The function must return an error stating that the table columns to return are missing
<b>Result:</b>	PASS

<b>Test #21:</b>	<b>IP-BE7-T3</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The parser has been initialized
<b>Input:</b>	A string query to search for records will be passed in to the parser testing function with the correct syntax
<b>Output:</b>	Received struct of the schema for the search record statement
<b>Expected:</b>	The function must return a struct that contains the schema fo the search record statement
<b>Result:</b>	PASS

## 1.2 Saving Data

<b>Test #22:</b>	<b>SD-BE1-T1</b>
<b>Type:</b>	Functional, Dynamic, Manual
<b>Initial State:</b>	The command line interface has been started and is waiting for a user input, and the database has been initialized
<b>Input:</b>	User enters an input to create a table with the correct syntax and a unique table name
<b>Output:</b>	Message appeared indicating that the table was created successfully
<b>Expected:</b>	The system must print a success message back to the user informing them that their table has been created
<b>Result:</b>	PASS

<b>Test #23:</b>	<b>SD-BE1-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized
<b>Input:</b>	A struct to create a table has been passed in with the correct syntax and a unique table name
<b>Output:</b>	Verified that the table does exist
<b>Expected:</b>	The system must create the specified table in the catalog
<b>Result:</b>	PASS

<b>Test #24:</b>	<b>SD-BE1-T3</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized and already has tables added to it
<b>Input:</b>	A struct to create a table has been passed in with the correct syntax and a table name already exists
<b>Output:</b>	Received duplicate table name error
<b>Expected:</b>	The system must return an error stating that a table by the specified name already exists
<b>Result:</b>	PASS

<b>Test #25:</b>	<b>SD-BE2-T1</b>
<b>Type:</b>	Functional, Dynamic, Manual
<b>Initial State:</b>	The command line interface has been started, is waiting for a user input, and the database already has tables added to it
<b>Input:</b>	User enters an input to delete a table with the correct syntax and an existing table name
<b>Output:</b>	Received success message and verified that table does not exist
<b>Expected:</b>	The system must print a success message back to the user informing them that their table has been deleted
<b>Result:</b>	PASS

<b>Test #26:</b>	<b>SD-BE2-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized and already has tables added to it
<b>Input:</b>	A struct to delete a table has been passed in with the correct syntax and an existing table name
<b>Output:</b>	Verified that the table does not exist anymore
<b>Expected:</b>	The system must delete the specified table in the catalog
<b>Result:</b>	PASS

<b>Test #27:</b>	<b>SD-BE2-T3</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized
<b>Input:</b>	A struct to delete a table has been passed in with the correct syntax and a table name does not exist
<b>Output:</b>	Received error indicating there is no table with that name
<b>Expected:</b>	The system must return an error stating that a table by the specified name does not exist
<b>Result:</b>	PASS

<b>Test #28:</b>	<b>SD-BE3-T1</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized
<b>Input:</b>	A string query to list all tables has been passed in to the parser testing function with the correct syntax
<b>Output:</b>	Received the existing tables
<b>Expected:</b>	The function must return all tables in the catalog
<b>Result:</b>	PASS

<b>Test #29:</b>	<b>SD-BE4-T1</b>
<b>Type:</b>	Functional, Dynamic, Manual
<b>Initial State:</b>	The command line interface has been started, is waiting for a user input, and the database already has tables added to it
<b>Input:</b>	User enters an input to insert a record with the correct syntax, an existing table name, and a primary key that does not already exist
<b>Output:</b>	Received success message
<b>Expected:</b>	The system must print a success message back to the user informing them that their record has been inserted
<b>Result:</b>	PASS

<b>Test #30:</b>	<b>SD-BE4-T1</b>
<b>Type:</b>	Functional, Dynamic, Manual
<b>Initial State:</b>	The command line interface has been started, is waiting for a user input, and the database already has tables added to it
<b>Input:</b>	User enters an input to insert a record with the correct syntax, an existing table name, and a primary key that does not already exist
<b>Output:</b>	Received success message and verified that the record exists
<b>Expected:</b>	The system must print a success message back to the user informing them that their record has been inserted
<b>Result:</b>	PASS

<b>Test #29:</b>	<b>SD-BE4-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized and already has tables added to it
<b>Input:</b>	A struct to insert a record has been passed in with the correct syntax, an existing table name, and a primary key that does not already exist
<b>Output:</b>	Verified that the record exists in the table
<b>Expected:</b>	The system must insert the specified tuple in the database
<b>Result:</b>	PASS

<b>Test #31:</b>	<b>SD-BE4-T3</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized
<b>Input:</b>	A struct to insert a record has been passed in with the correct syntax and a table name that does not already exist
<b>Output:</b>	Received error indicating the table does not exist
<b>Expected:</b>	The system must return an error stating that a table by the specified name does not exist
<b>Result:</b>	PASS

<b>Test #32:</b>	<b>SD-BE5-T1</b>
<b>Type:</b>	Functional, Dynamic, Manual
<b>Initial State:</b>	The command line interface has been started, is waiting for a user input, and the database already has tables and records to it
<b>Input:</b>	User enters an input to modify a record with the correct syntax, an existing table name, and a primary key to uniquely identify the record
<b>Output:</b>	Verified that the record has been modified correctly
<b>Expected:</b>	The system must print a success message back to the user informing them that their record has been modified
<b>Result:</b>	PASS

<b>Test #33:</b>	<b>SD-BE5-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized and already has tables and records added to it
<b>Input:</b>	A struct to modify a record has been passed in with the correct syntax, an existing table name, and a primary key to uniquely identify the record
<b>Output:</b>	Verified that the record has been modified correctly
<b>Expected:</b>	The system must modify the specified tuple in the database
<b>Result:</b>	PASS

<b>Test #34:</b>	<b>SD-BE5-T3</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized and already has tables and records added to it
<b>Input:</b>	A struct to modify a record has been passed in with the correct syntax and a table name that does not exist
<b>Output:</b>	Received incorrect table name error
<b>Expected:</b>	The system must return an error stating that a table by the specified name does not exist
<b>Result:</b>	PASS

<b>Test #35:</b>	<b>SD-BE5-T4</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized and already has tables and records added to it
<b>Input:</b>	A struct to modify a record has been passed in with the correct syntax, an existing table name, and a primary key that does not exist
<b>Output:</b>	Received incorrect primary key error
<b>Expected:</b>	The system must return an error stating that a table by the specified primary key does not exist
<b>Result:</b>	PASS

<b>Test #36:</b>	<b>SD-BE6-T1</b>
<b>Type:</b>	Functional, Dynamic, Manual
<b>Initial State:</b>	The command line interface has been started, is waiting for a user input, and the database already has tables and records added to it
<b>Input:</b>	User enters an input to delete a record with the correct syntax, an existing table name, and a primary key to uniquely identify
<b>Output:</b>	Received a success message and verified that the record has been deleted
<b>Expected:</b>	The system must print a success message back to the user informing them that their record has been deleted
<b>Result:</b>	PASS

<b>Test #37:</b>	<b>SD-BE6-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized and already has tables and records added to it
<b>Input:</b>	A struct to delete a record has been passed in with the correct syntax, an existing table name, and a primary key to uniquely identify the record
<b>Output:</b>	Verified the record does not exist in the table
<b>Expected:</b>	The system must delete the specified tuple in the database
<b>Result:</b>	PASS

<b>Test #38:</b>	<b>SD-BE6-T3</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized and already has tables and records added to it
<b>Input:</b>	A struct to delete a record has been passed in with the correct syntax and a table name that does not exist
<b>Output:</b>	Received missing table name error
<b>Expected:</b>	The system must return an error stating that a table by the specified name does not exist
<b>Result:</b>	PASS

<b>Test #39:</b>	<b>SD-BE7-T1</b>
<b>Type:</b>	Functional, Dynamic, Manual
<b>Initial State:</b>	The command line interface has been started, is waiting for a user input, and the database already has tables and records added to it
<b>Input:</b>	User enters an inputs to search for records with the correct syntax, and an existing table name
<b>Output:</b>	Received all the records in the table
<b>Expected:</b>	The system must print back all records found from their search query back to the user
<b>Result:</b>	PASS

<b>Test #40:</b>	<b>SD-BE7-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The command line interface has been started, is waiting for a user input, and the database already has tables and records added to it
<b>Input:</b>	A struct to search for records has been passed in with the correct syntax and a table name that does not exist
<b>Output:</b>	Received all the records in the table
<b>Expected:</b>	The system must return an error stating that a table by the specified name does not exist
<b>Result:</b>	PASS

<b>Test #41:</b>	<b>SD-BE8-T1</b>
<b>Type:</b>	Functional, Dynamic, Manual
<b>Initial State:</b>	Testers have access to the source code for the transactions in the database management system
<b>Input:</b>	The testers do a code walk through to ensure that the system correctly locks the record during a write
<b>Output:</b>	The testers were able to confirm that the locking mechanism is programmed correctly
<b>Expected:</b>	We are able to determine whether structurally the code will prevent writes and other transactions at the same time
<b>Result:</b>	PASS

<b>Test #42:</b>	<b>SD-BE8-T2</b>
<b>Type:</b>	Functional, Dynamic, Automated
<b>Initial State:</b>	The database has been initialized and already has tables and records added to it
<b>Input:</b>	The testers do a code walk through to ensure that the system correctly locks the record during a write
<b>Output:</b>	The testers were able to confirm that the locking mechanism is programmed correctly
<b>Expected:</b>	We are able to determine whether structurally the code will prevent writes and other transactions at the same time
<b>Result:</b>	PASS

<b>Test #43:</b>	<b>SD-BE8-T3</b>
<b>Type:</b>	Functional, Dynamic, Manual
<b>Initial State:</b>	The database has been initialized and already has tables and records added to it
<b>Input:</b>	The testers will manually pass in multiple read and write queries to the database through multiple CLIs
<b>Output:</b>	The testers verified the result of the queries were correct
<b>Expected:</b>	The system will output the result of all passed in queries
<b>Result:</b>	PASS

## 2 Non Functional Requirements Evaluation

### 2.1 Look and Feel Testing

<b>Test #44:</b>	<b>NFR-1-LF1</b>
<b>Type:</b>	Dynamic, Manual
<b>Description:</b>	A manual test to determine whether the appearance of the client for the database is up to par
<b>Tester(s):</b>	Group of students with at least a basic understanding of databases
<b>Pass:</b>	Average survey score of at least 70%
<b>Result:</b>	PASS

<b>Test #45:</b>	<b>NFR-2-LF2</b>
<b>Type:</b>	Dynamic, Manual
<b>Description:</b>	A manual test to determine whether the styling of the client is good
<b>Tester(s):</b>	Group of students with at least a basic understanding of databases
<b>Pass:</b>	Average survey score of at least 70%
<b>Result:</b>	PASS

## 2.2 Usability and Humanity Testing

<b>Test #45:</b>	<b>NFR-3-UH1</b>
<b>Type:</b>	Dynamic, Manual
<b>Description:</b>	A manual test to determine whether the database is intuitive enough for beginners
<b>Tester(s):</b>	Group of students with at least a basic understanding of databases
<b>Pass:</b>	Average survey score of at least 70%
<b>Result:</b>	PASS

<b>Test #46:</b>	<b>NFR-4-UH2</b>
<b>Type:</b>	Dynamic, Manual
<b>Description:</b>	A manual test to determine whether how easily comprehensive the language in the database client is
<b>Tester(s):</b>	Group of students with at least a basic understanding of databases
<b>Pass:</b>	Average survey score of at least 70%
<b>Result:</b>	PASS

## 2.3 Performance Testing

<b>Test #47:</b>	<b>NFR-5-P1</b>
<b>Type:</b>	Dynamic, Automated
<b>Description:</b>	An automated test to determine whether the database provides responses to the user in a reasonably fast time. This is done by timing how long it takes for responses to multiple requests to be made in the database
<b>Input:</b>	Multiple search queries are run on the database
<b>Expected:</b>	The time recorded is less than or equal to the maximum accepted response time of 5 seconds
<b>Result:</b>	PASS

<b>Test #48:</b>	<b>NFR-6-P2</b>
<b>Type:</b>	Dynamic, Manual
<b>Description:</b>	A manual test to determine whether the database maintains the precision of the numbers that are stored
<b>Tester(s):</b>	Group of students with at least a basic understanding of databases
<b>Pass:</b>	The testers confirm whether or not the precision of the initial input data has been maintained
<b>Result:</b>	PASS

<b>Test #49:</b>	<b>NFR-7-P3</b>
<b>Type:</b>	Dynamic, Manual
<b>Description:</b>	A manual test to determine whether the <i>.data</i> file has been generated in the Data folder of the local directory to indicate that the data is persistent and being stored locally
<b>Tester(s):</b>	Group of students with at least a basic understanding of databases
<b>Pass:</b>	The testers confirm whether or not the data file has been generated
<b>Result:</b>	PASS

## 2.4 Performance Testing

<b>Test #50:</b>	<b>NFR-8-OE1</b>
<b>Type:</b>	Dynamic, Manual
<b>Description:</b>	A manual test to determine whether all the functionalities of the database still work as expected without an internet connection
<b>Tester(s):</b>	Group of students with at least a basic understanding of databases
<b>Pass:</b>	The testers confirm the database still works as normal without an internet connection
<b>Result:</b>	PASS

### **3 Comparison to Existing Implementation**

The original implementation was well tested but there were no documentations to support the tests. Additionally the original implementation also did not have any non functional tests. Though the original implementation was well tested from a functional testing standpoint, the lack of traceability to the requirements did not raise confidence in the correctness of the software. In our implementation, the requirements were well documented and divided up into modules that were tested to an equal degree to the original implementation. Due to the use of documentation and explicitly identifying our requirements, which allowed us to trace our tests back to the requirements and be confident that we had successfully achieved our requirements. Our implementation also included various NFR tests which also increased our confidence in the correctness of the implementation.

### **4 Unit Testing**

Unit testing was heavily implemented to ensure confidence in the functionality of each component of our code. As the individual software units comprised a much larger system, it was important to build confidence in the lower level functionality so that error tracing efforts would be efficient for system wide testing.

### **5 Changes Due To Testing**

Testing did not uncover a cause to deeply reframe the architectural decisions or design patterns. However, rigorous test cases revealed local code bugs such as syntax errors, type mismatches and minor logic errors that were promptly addressed. Testing also revealed edge cases and necessary error handling.

### **6 Automated Testing**

Our test plan was heavily automated testing using the Go test library. This reduced manual labour efforts and time resource consumption, and provided instant results right in the IDE. We used nearly 700 lines of automated testing scripts to ensure thorough testing.



## 7 Trace to Requirements

	Functional Requirements																
	BE1FR1	BE1FR2	BE1FR3	BE1FR4	BE1FR5	BE1FR6	BE1FR7	BE1FR8	BE1FR9	BE2FR1	BE2FR2	BE2FR3	BE2FR4	BE2FR5	BE2FR6	BE2FR7	BE2FR8
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
IP-BE1-11																	
IP-BE1-12	x																
IP-BE1-13	x																
IP-BE1-14		x															
IP-BE1-15			x														
IP-BE1-16			x														
IP-BE1-17			x														
IP-BE1-18			x														
IP-BE1-19			x														
IP-BE1-20			x														
IP-BE1-21			x														
IP-BE1-22			x														
IP-BE1-23			x														
IP-BE1-24			x														
SD-BE1-T1				x													
SD-BE1-T2				x													
SD-BE1-T3				x													
SD-BE2-T1				x													
SD-BE2-T2				x													
SD-BE2-T3				x													
SD-BE3-T1				x													
SD-BE4-T1				x													
SD-BE4-T2				x													
SD-BE4-T3				x													

Figure 1: Traceability Matrix for Functional Requirements in BE1 to BE4

		Functional Requirements														
		BEEFR1	BEEFR2	BEEFR3	BEEFR4	BEEFR5	BEEFR6	BEEFR7	BEEFR8	BEEFR9	BEEFR10	BEEFR11	BEEFR12	BEEFR13	BEEFR14	BEEFR15
Test Cases	Requirements	BEEFR1 BEEFR2 BEEFR3 BEEFR4 BEEFR5 BEEFR6 BEEFR7 BEEFR8 BEEFR9 BEEFR10 BEEFR11 BEEFR12 BEEFR13 BEEFR14 BEEFR15														
I-BE5-T1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
I-BE5-T2																
I-BE5-T3																
I-BE5-T4																
I-BE6-T1																
I-BE6-T2																
I-BE6-T3																
I-BE6-T4																
I-BE6-T5																
I-BE6-T6																
I-BE6-T7																
I-BE6-T8																
I-BE6-T9																
S-BE5-T1	x															
S-BE5-T2																
S-BE5-T3																
S-BE5-T4																
S-BE6-T1																
S-BE6-T2																
S-BE6-T3																
S-BE6-T4																
S-BE6-T5																
S-BE6-T6																
S-BE6-T7																
S-BE6-T8																
S-BE6-T9																

Figure 2: Traceability Matrix for Functional Requirements in BE5 to BE8

	Non Functional Requirements										
	3.1.1.1	3.1.1.2	3.1.2.1	3.2.1.1	3.2.2.1	3.2.3.1	3.2.3.2	3.3.1.1	3.3.3.1	3.3.6.1	3.4.1.1
Test Cases	X										
NFR-1-LF1		X									
NFR-2-LF2			X								
NFR-3-UH1				X							
NFR-4-UH2					X						
NFR-5-P1						X					
NFR-6-P2							X				
NFR-7-P3								X			
NFR-8-OE1									X		

Figure 3: Traceability Matrix for Nonfunctional Requirements in NFR-1 to NFR-8



## 8 Trace To Modules

Req.	Modules
BE1FR1, BE2FR1, BE3FR1, BE4FR1, BE5FR1, BE6FR1, BE7FR1, BE8FR1	m1, m2
BE1FR2, BE2FR2, BE3FR2, BE4FR2, BE5FR2, BE6FR2, BE7FR2	m3 <b>m17</b>
BE1FR3	m6
BE2FR3	m7
BE3FR3	m2
BE4FR3	m5
BE5FR3	m4
BE6FR3	m8
BE7FR3	m9
BE1FR4, BE1FR5, BE2FR4, BE4FR4, BE4FR5, BE5FR4, BE5FR5, BE5FR6, BE6FR4, BE7FR4	m10, m13, m12, m14, m15, m16
BE1FR6, BE1FR8, BE2FR5, BE2FR7, BE3FR4, BE3FR5, BE4FR6, BE4FR8, BE5FR7, BE5FR9, BE6FR5, BE6FR7, BE7FR5, BE7FR6	m1, m2, m10
BE1FR7, BE2FR6, BE4FR7, BE5FR8, BE6FR6	m12, m13, m14, m15, m16
BE8FR2, BE8FR3, BE8FR4, BE8FR5	m11

Table 4: Trace Between Requirements and Modules