CMPE 138 Database Systems I

Spartan Quality Assurance Tool

Team 3

Ankita Arvind Deshmukh (SID: 016029585)
Fnu Butul Parveen (SID: 015918227)
Nalini Mehendale (SID: 015312206)
Vaishak Melarcode Kalllampad (SID: 015017496)
Rushil Shah (SID: 015908789)
Divija Naredla (SID: 015945969)

Table of Contents:

The choice of database project:

Database Engine	Mysql Workbench 8.0.16
DB application Technologies	Node.js and Express.js
Frameworks	React.js
Languages	SQL and Javascript
DB access Technology	REST API and HTTP Protocol

Introduction

Spartan Quality Assurance Tool keeps track of test cases and problems to make the testing process more convenient and transparent. At the same time, it offers a new way of looking at and analyzing the company's testing data at the project and component level. The system is designed to give managers, test leads, and testers a formal framework to view, improve, and streamline the entire testing process.

Testers can create tickets against failed test cases to add, delete, and amend test cases, as well as alter the status of test cases as pass, fail, or N/A. For users to view the testing progress of projects, the system creates status reports and graphical analyses.

Final overall architecture: block/component diagram

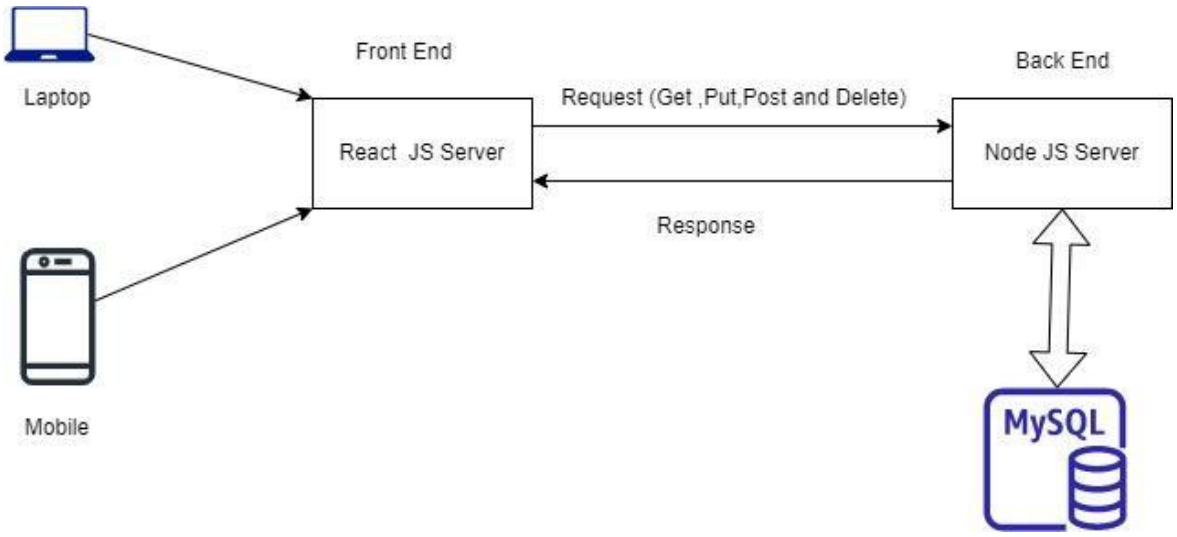


Figure - Block Diagram

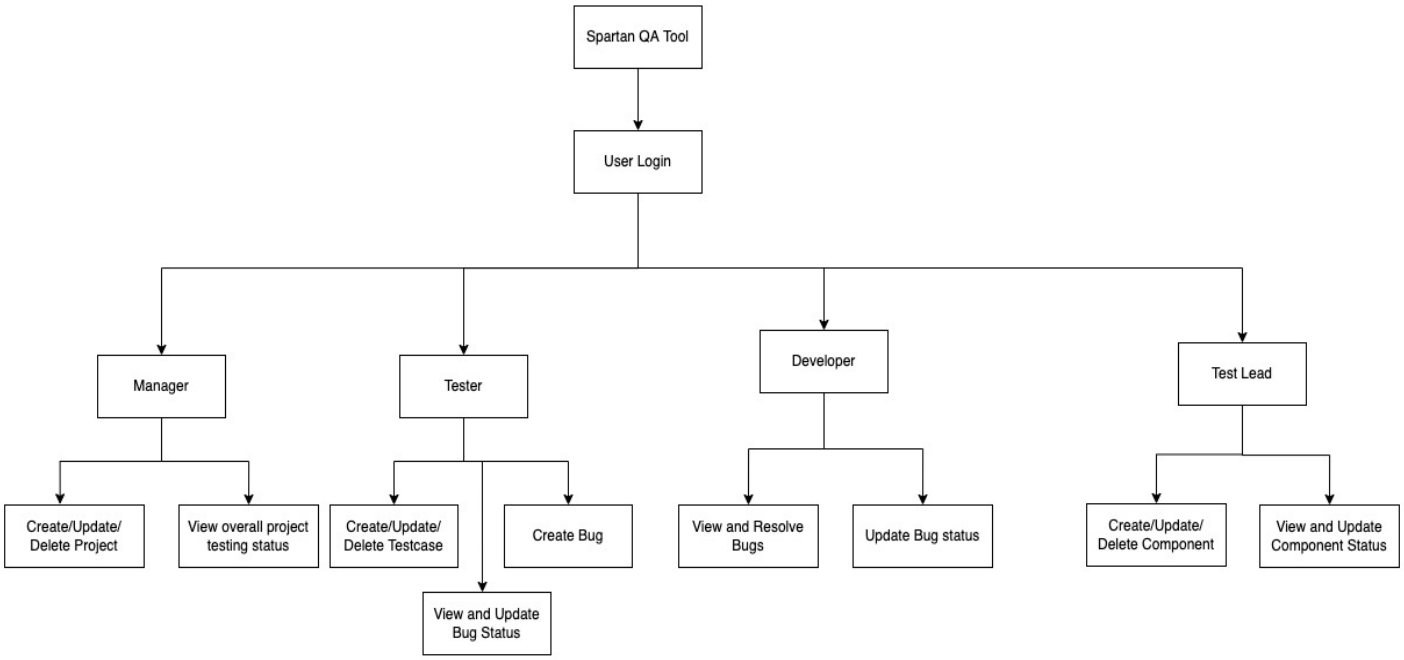


Figure - Architecture Diagram

Final list of functionalities/operations of each role/actor, in list format

Actors

- **Manager :** User who creates a new project in the system. The manager can view the status of different components of projects. The manager can also view the entire testing status in the form of pie charts.

Functionality	Status
SignUp/Login to the system as a Manager	Done
Create Project	Done
Assigning test tasks for different projects to the test leads	Not Implemented
View overall testing Status of the Project in form of pie charts	Done

- **Test Lead :** User who creates components for a particular project and updates the status of the components. The Test Lead can view the status of test cases of each component (number of test cases passed, failed etc.)

Functionality	Status
SignUp/Login to the system as a Test Lead	Done
Create Component	Done
Update Component	Done
Delete Component	Done
Assign the Component to tester and developer	Not Implemented
View Status of Test Case on component level	Done
Update the status of the component	Done
Assigns bugs to developer	Not Implemented
Execute Test Run	Not Implemented
View status of test runs	Not Implemented

- **Tester :** User who creates, updates and deletes test cases for a particular component of a particular project. The tester can raise a bug if the test case is not successful.

Functionality	Status
SignUp/Login to the system as a Tester	Done
Create Test case	Done
Update Test Case	Done
Delete Test Case	Done
Raise Bug	Done
View Status of Test Run	Not Implemented
View Status of Bugs Raised	Done

- **Developer :** User who resolves the bugs raised by the tester. The developer can view the list of bugs raised and can update the status of the bugs once solved.

Functionality	Status
SignUp/Login to the system as a Developer	Done
View and resolve bugs	Done
Update the status of bug	Done

Member's contributions: list project major areas/tasks

- **Ankita Arvind Deshmukh (SID: 016029585)**
  - Proposal - 02/28
  - EERD - 03/09
  - Reports/slides - 05/02
  - Implementation - 05/02
    - Worked on Project Table
    - Functional Dependencies, Normalization and SQL Queries

- **Fnu Butul Parveen (SID: 015918227)**
  - Proposal - 02/28
  - EERD - 03/09
  - Reports/slides - 05/02
  - Implementation work - 05/02
    - Worked on Component Table
    - Functional Dependencies, Normalization and SQL Queries

- **Nalini Mehendale (SID: 015312206)**
  - Proposal - 02/28
  - EERD - 03/09
  - Reports/slides - 05/02
  - Implementation work - 05/02
    - Worked on User Table
    - Functional Dependencies, Normalization and SQL Queries

- **Vaishak Melarcode Kalllampad (SID: 015017496)**
  - Proposal - 02/28

- EERD - 03/09
- Reports/slides - 05/02
- Implementation work - 05/02
  - Worked on TestCase Table
  - Functional Dependencies, Normalization and SQL Queries

❖ **Rushil Shah (SID: 015908789)**

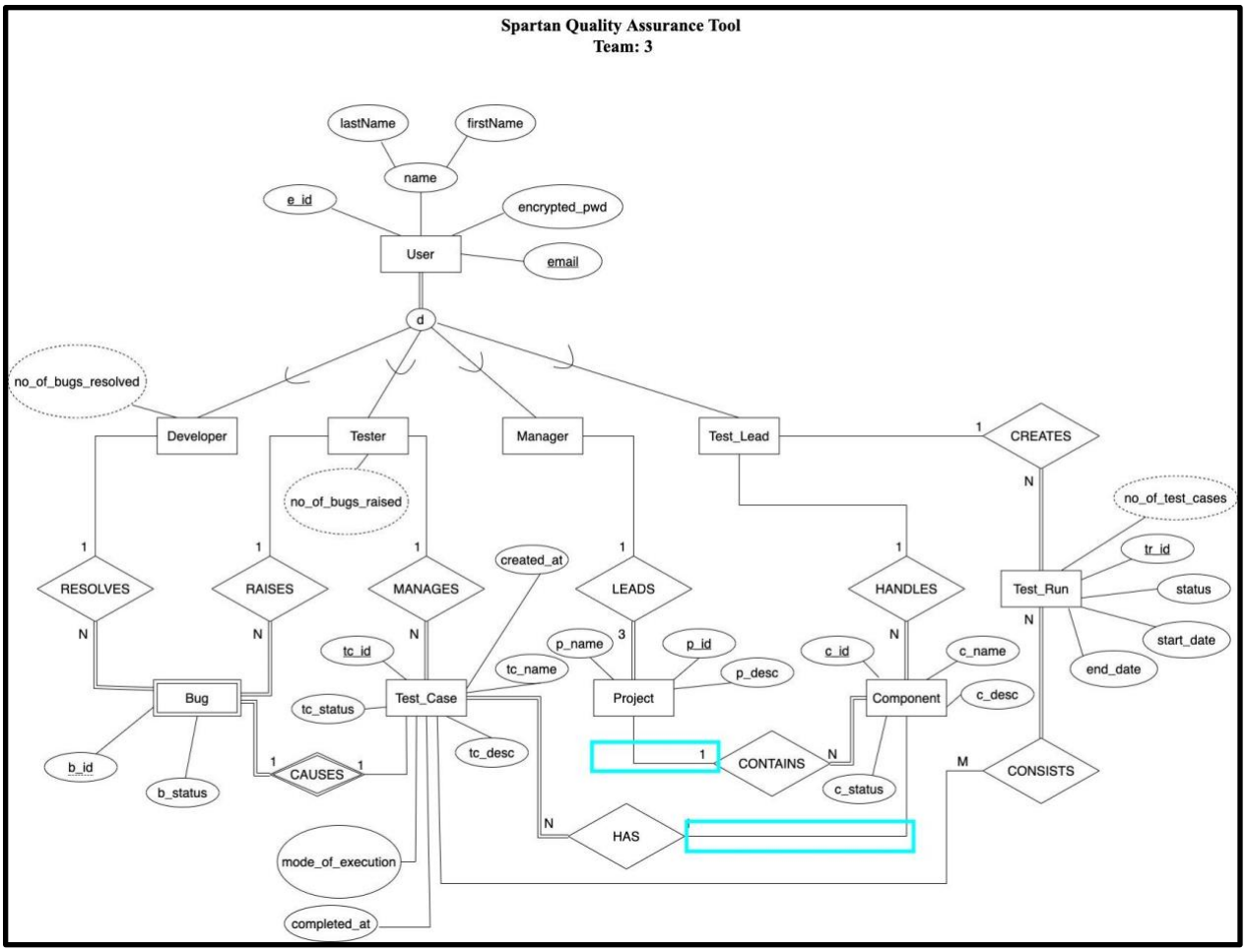
- Proposal - 02/28
- EERD - 03/09
- Reports/slides - 05/02
- Implementation work - 05/02
  - Worked on Bug Table
  - Functional Dependencies, Normalization and SQL Queries

❖ **Divija Naredla (SID: 015945969)**

- Proposal - 02/28
- EERD - 03/09
- Reports/slides - 05/02
- Implementation work - 05/02
  - Integration/Login
  - Functional Dependencies, Normalization and SQL Queries

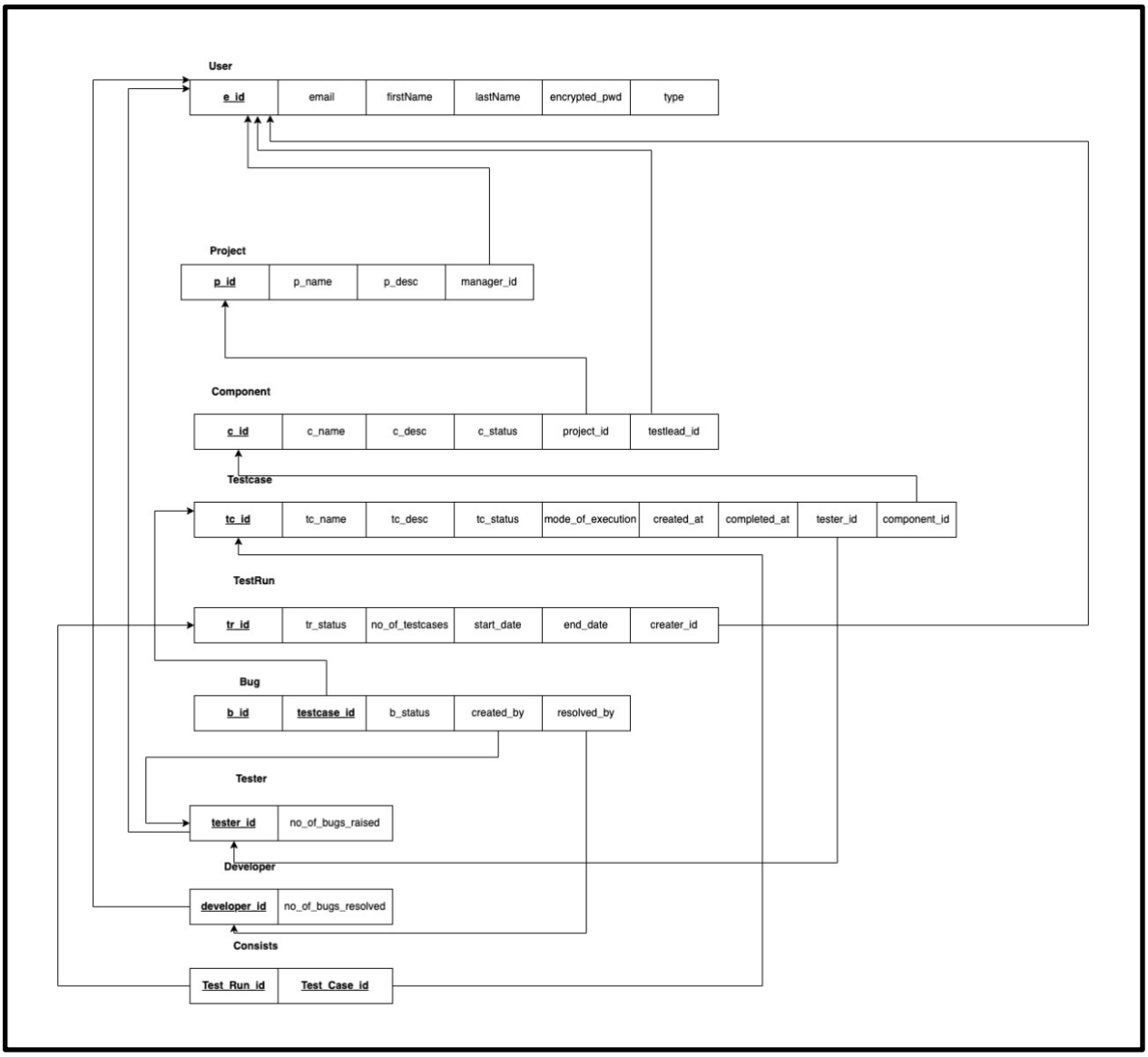
<u>Final design of database portion</u>:

ER/EER diagram:



There are minor modifications in EERD from the approved one. Originally, the relationship between Project and Component, and Component and Test_Case was of total participation which is now changed to partial participation. This is because a newly created project may not have the components and newly created components may not have test cases linked to them.

Schema diagram:



Specification of each DB object :

- **User Table:**

User

e_id	email	firstName	lastName	encrypted_pwd	type

```
CREATE Table User(e_id INT NOT NULL AUTO_INCREMENT,
email VARCHAR(30) NOT NULL,
firstName VARCHAR(30) NOT NULL,
lastName VARCHAR(30),
encrypted_pwd BINARY(64) NOT NULL,
type VARCHAR(10) NOT NULL,
PRIMARY KEY(e_id),
UNIQUE(email));
```

- **e_id (PK) :** Each User must have a primary key unique e_id(employee id) for identifying the employee.
- **email :** It stores the unique email id of each user.
- **firstName :** It stores the first name of each user.
- **lastName :** It stores the last name of each user.
- **encrypted_pwd :** Each user must have a password which is encrypted and not viewable by anyone.
- **type :** Each user is either a tester, or developer, or test lead or manager

- **Project Table:**

Project

p_id	p_name	p_desc	lead_by

```
CREATE TABLE Project ( p_id INT AUTO_INCREMENT,
p_name VARCHAR(30) NOT NULL,
p_desc VARCHAR(50) NOT NULL,
manager_id INT NOT NULL,
PRIMARY KEY (p_id),
CONSTRAINT fk_project_user FOREIGN KEY (manager_id)
REFERENCES User(e_id)
on delete cascade
);
```

- **p_id(PK)** : Each project must have a unique p_id (project id) for identification.

- ➢ **p_name :** Each Project must have a project name for identifying the project title..
- ➢ **p_desc:**It stores the description of the project.
- ➢ **manager_id :** This stores the employee id of the manager leading this project.

- **Component Table:**

Component

c_id	c_name	c_desc	c_status	project_id	testlead_id

```
CREATE TABLE Component (c_id INT NOT NULL AUTO_INCREMENT,
 c_name VARCHAR(20) NOT NULL,
 c_desc VARCHAR(300) NOT NULL,
 c_status VARCHAR(10) NOT NULL,
 project_id INT NOT NULL,
 testlead_id INT NOT NULL,
 PRIMARY KEY(c_id),
 CONSTRAINT fk_proj_id FOREIGN KEY (project_id) references Project(p_id) on delete cascade,
 CONSTRAINT fk_test_lead FOREIGN KEY (testlead_id) references User(e_id) on delete cascade
 );
```

- ➢ **c_id(PK) :** Each component must have a unique primary key attribute c_id (component_id) for identification.
- ➢ **c_name :** It stores the name of the component.
- ➢ **c_desc :** It stores the description of the component.
- ➢ **c_status :** It stores the testing status of the component.
- ➢ **project_id :** It stores the id of the project to which the component belongs.
- ➢ **testlead_id :** The testlead_id stores the employee id of the test lead handling this component.

- **TestCase Table:**

Testcase

tc_id	tc_name	tc_desc	tc_status	mode_of_execution	created_at	completed_at	tester_id	component_id

```
CREATE TABLE TestCase (tc_id INT NOT NULL AUTO_INCREMENT,
tc_name VARCHAR(20) NOT NULL,
tc_desc VARCHAR(300) NOT NULL,
tc_status VARCHAR(10) NOT NULL,
mode_of_execution VARCHAR(20) NOT NULL,
created_at DATETIME NOT NULL,
completed_at DATETIME NOT NULL,
tester_id INT NOT NULL,
component_id INT NOT NULL,
PRIMARY KEY(tc_id),
CONSTRAINT fk_tester FOREIGN KEY (tester_id) references tester(tester_id) on delete cascade,
CONSTRAINT fk_comp_id FOREIGN KEY (component_id) REFERENCES Component(c_id) on delete cascade
);
```

- **tc_id(PK):**Test Case id is the unique primary key attribute for identifying the test case.
- **tc_name:**For identifying the test case with respect to the test case name.
- **tc_desc:**It stores the description of the test case.
- **tc_status:**Test case status will store the status of the test case either passed or failed etc.
- **mode_of_execution :** The mode_of_execution stores the mode of execution of a test case – either automatic or manual.
- **created_at :** Its stores the timestamp at which the test case is created.
- **completed_at :** Its stores the timestamp at which the test case is passed.
- **tester_id :** The tester_id stores the employee id of the tester managing this test case.
- **component_id :** It stores the component_id of the component for which this test case is created.

- **TestRun Table:**

TestRun

tr_id	tr_status	no_of_testcases	start_date	end_date	creater_id

```
CREATE TABLE TestRun (tr_id INT NOT NULL AUTO_INCREMENT,
tr_status VARCHAR(10) NOT NULL,
no_of_testcases INT NOT NULL,
start_date DATETIME NOT NULL,
end_date DATETIME NOT NULL,
creator_id INT NOT NULL,
PRIMARY KEY(tr_id),
CONSTRAINT fk_creator FOREIGN KEY (creator_id) references User(e_id) on delete cascade
);
```

- **tr_id(PK) :** TestRun_id is a unique primary key attribute for identifying each Test Run.
- **tr_status :** It stores the status of a test run as 'All Passed' or 'Pending'. If all test cases in the Test Run are passed, the status is 'All Passed' or else 'Pending'.
- **no_of_testcases :** This attribute contains the number of test cases considered for the test run. All test cases created between a given start_date and end_date will be considered for a given test run.
- **start_date :** It indicates the start date selected for a test run.
- **end_date :** It indicates the end date selected for a test run.
- **creator_id :** It contains the employee id of the test lead who creates this test run.

- **Bug Table:**

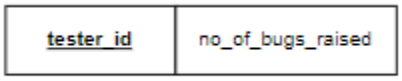
b_id	testcase_id	b_status	created_by	resolved_by

```
CREATE TABLE Bug (b_id INT NOT NULL,
testcase_id INT NOT NULL,
b_status VARCHAR(10) NOT NULL,
created_by INT NOT NULL,
resolved_by INT ,
PRIMARY KEY(testcase_id, b_id),
CONSTRAINT fk_createdby FOREIGN KEY (created_by) references tester(tester_id) on delete cascade,
CONSTRAINT fk_resolver FOREIGN KEY (resolved_by) references developer(developer_id) on delete cascade,
CONSTRAINT fk_tcid FOREIGN KEY (testcase_id) REFERENCES TestCase(tc_id) on delete cascade
);
```

- **b_id :**It is a unique bug_id for a given test case.
- **Testcase_id :** It contains test case id of the related the test case.

- ➢ **b_status :** It contains the status of a bug – Resolved, InProgress or Unresolved.
- ➢ **created_by:** It indicates the employee id of the tester who creates the bug.
- ➢ **resolved_by :** It indicates the employee id of the developer who resolves the bug.

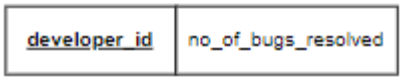
- **Tester Table:**




```
CREATE TABLE tester(
tester_id INT NOT NULL,
no_of_bugs_raised INT DEFAULT 0,
PRIMARY KEY(tester_id),
CONSTRAINT testerid FOREIGN KEY(tester_id) REFERENCES User(e_id) on delete cascade
);
```

- ➢ **tester_id:**It is a unique primary key attribute for the tester table.
- ➢ **no_of_bugs_raised :** It contains the number of bugs raised by the tester.

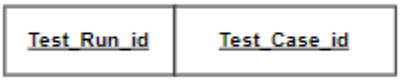
- **Developer Table:**




```
CREATE TABLE developer(
developer_id INT NOT NULL,
no_of_bugs_resolved INT DEFAULT 0,
PRIMARY KEY(developer_id),
CONSTRAINT developerid  FOREIGN KEY(developer_id) REFERENCES User(e_id) on delete cascade
);
```

- ➢ **developer_id:**It is a unique primary key attribute for the developer table.
- ➢ **no_of_bugs_resolved :** It contains the number of bugs resolved by the developer.

- **Consists Table:**




```
CREATE TABLE CONSISTS(
testrun_id INT NOT NULL,
testcase_id INT NOT NULL,
PRIMARY KEY(testrun_id, testcase_id),
CONSTRAINT testrunid FOREIGN KEY(testrun_id) REFERENCES TestRun(tr_id) on delete cascade,
CONSTRAINT testcaseid FOREIGN KEY(testcase_id) REFERENCES TestCase(tc_id) on delete cascade
);
```

  - ➢ **Testrun_id :** It contains the id the testrun
  - ➢ **Testcase_id :** It contains the id of the test cases which come under the test run.

Functional dependencies of each table and normalization:

- **User(e_id,**email,firstName,lastName,encrypted_pwd)
  PK: e_id
  Functional dependencies:
  e_id->email,
  e_id->firstName,
  e_id->lastName,
  e_id->encrypted_pwd
  1NF: yes - has PK(e_id); all non-PK attributes FD on PK; attributes have single atomic values.
  2NF: yes - is 1NF; as attributes are fully FD on the entire PK (has no partial dependencies)
  3NF: yes - is 2NF; attributes are directly FD on PK (has no transitive FDs)

- **Project(p_id,** p_name,p_desc,manager_id)
  PK:p_id
  Functional dependencies:
  p_id->p_name,
  p-id->p_desc,
  p_id->manager_id

1NF: yes - has PK(p_id); all non-PK attributes FD on PK; attributes have single atomic values.
2NF: yes - is 1NF; as attributes are fully FD on the entire PK (has no partial dependencies)
3NF: yes - is 2NF; attributes are directly FD on PK (has no transitive FDs)

- **Component(c_id**,c_name,c_desc,c_status,project_id,testlead_id)
  PK:c_id
  Functional dependencies:
    c_id->c_name,
    c_id->c_desc,
    c_id->c_status,
    c_id->project_id,
    c_id->testlead_id
  1NF: yes - has PK(c_id); all non-PK attributes FD on PK; attributes have single atomic values.
  2NF: yes - is 1NF; as attributes are fully FD on the entire PK (has no partial dependencies)
  3NF: yes - is 2NF; attributes are directly FD on PK (has no transitive FDs)

- **TestCase(tc_id**,tc_name,tc_desc,tc_status,mode_of_execution,created_at,completed_at,tester_id,component_id)
  PK:tc_id
  Functional dependencies:
    tc_id->tc_name,
    tc_id->tc_desc,
    tc_id->tc_status,
    tc_id->mode_of_execution,
    tc_id->created_at,
    tc_id->completed_at,
    tc_id->tester_id,
    tc_id->component_id
  1NF: yes - has PK(tc_id); all non-PK attributes FD on PK; attributes have single atomic values.
  2NF: yes - is 1NF; as attributes are fully FD on the entire PK (has no partial dependencies)
  3NF: yes - is 2NF; attributes are directly FD on PK (has no transitive FDs)

- **TestRun(<u>tr_id,</u>** tr_status,no_of_testcases,start_date,end_date,creater_id)
  PK:tr_id
  Functional dependencies:
    tr_id->tr_status,
    tr_id->no_of_testcases,
    tr_id->start_date,
    tr_id->end_date,
    tr_id->creater_id
  1NF: yes - has PK(tr_id); all non-PK attributes FD on PK; attributes have single atomic values.
  2NF: yes - is 1NF; as attributes are fully FD on the entire PK (has no partial dependencies)
  3NF: yes - is 2NF; attributes are directly FD on PK (has no transitive FDs)

- **Bug(<u>b_id,</u> <u>testcase_id,</u>** b_status,created_by,resolved_by)
  Composite PK:(b_id,testcase_id)
  Functional dependencies:
    (b_id,testcase_id)->b_status,
    (testcase_id)->created_by,
    (b_id,testcase_id)->resolved_by
  1NF: yes - has PK(b_id,testcase_id); all non-PK attributes FD on PK; attributes have single atomic values.
  2NF: no - is 1NF; however, the attribute created_by is only dependent on key attribute testcase_id therefore has partial dependencies.
  3NF: no - is not 2NF.

  We have decide not to Normalize this table because it would split the table into 2 table:
  - Table1(**<u>b_id,</u> <u>testcase_id,</u>** b_status,resolved_by)
  - Table2(**<u>testcase_id</u>**, created_by).

  This relationship has already been covered in “TestCase” table, so creating a new table for this is not needed.

- **Tester(<u>tester_id</u>**,no_of_bugs_raised)
  PK:tester_id
  Functional dependencies:
    tester_id->no_of_bugs_raised

1NF: yes - has PK(tester_id); all non-PK attributes FD on PK; attributes have single atomic values.
2NF: yes - is 1NF; as attributes are fully FD on the entire PK (has no partial dependencies)
3NF: yes - is 2NF; attributes are directly FD on PK (has no transitive FDs)

- **Developer**(**developer_id**,no_of_bugs_resolved)
  PK:developer_id
  Functional dependencies:
  tester_id->no_of_bugs_resolved
  1NF: yes - has PK(developer_id); all non-PK attributes FD on PK; attributes have single atomic values.
  2NF: yes - is 1NF; as attributes are fully FD on the entire PK (has no partial dependencies)
  3NF: yes - is 2NF; attributes are directly FD on PK (has no transitive FDs)

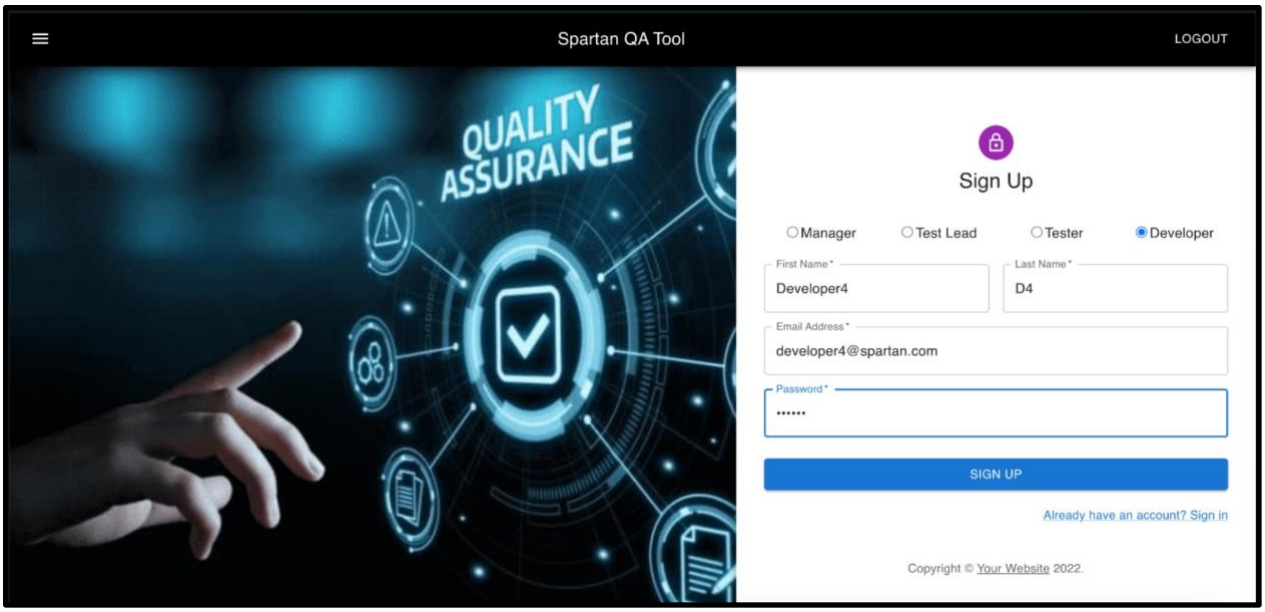
- **Consists**(**tr_id,tc_id**)
  Composite PK:(tr_id,tc_id)
  Functional Dependencies: no functional dependencies in consists table
  1NF: yes - has PK(tr_id,tc_id); no FDs; attributes have single atomic values.
  2NF: yes - is 1NF; no FDs(no partial FDs);
  3NF: yes - is 2NF; no FDs(no transitive FDs);

There is no denormalization based on the EERD diagram.

User login password: hashed or encrypted

A new user registering in the Spartan QA tool will provide the password which will then be hashed into the User table. Bcrypt npm algorithm is the method used for password hashing in pwd column of the User table for which the steps are as follows:

1) Install bcrypt library and then include bcrypt module
2) Set a value for SaltRounds
3) Declare a password variable
4) Generate a salt
5) Hash the password




```
let getUserByIdQuery = `SELECT * FROM User WHERE e_id = ${e_id}`;
const getLastInerstedIdQuery = `SELECT LAST_INSERT_ID();`;

let sql_findEmail = `SELECT * FROM User WHERE email = '${email}'`;

async function hashPassword(password) {
  const salt = await genSalt(10);
  console.log(salt, password);
  const hash = await _hash(password, salt);
  return hash;
}
const checkEmailResponse = await connection.query(sql_findEmail);
if(parseRowDataPacket(checkEmailResponse).length > 0){
  return {
    success: false,
    message: 'Email Already exists'
  };
}

let finalObj;
finalObj = hashPassword(pwd).then(async (customerPassword) => {

  let sql_insert = `INSERT INTO User
  (e_id,
  email,
  firstName,
  lastName,
  pwd,
  type) VALUES (${null}, '${email}', '${firstName}', '${lastName}', '${customerPassword}', '${type}')`;


  const response = await connection.query(sql_insert);
  getUserByIdQuery = `SELECT * FROM User WHERE e_id = ${response.insertId}`;
  const insertedObject = await connection.query(getUserByIdQuery);
  const result = parseRowDataPacket(insertedObject);
```

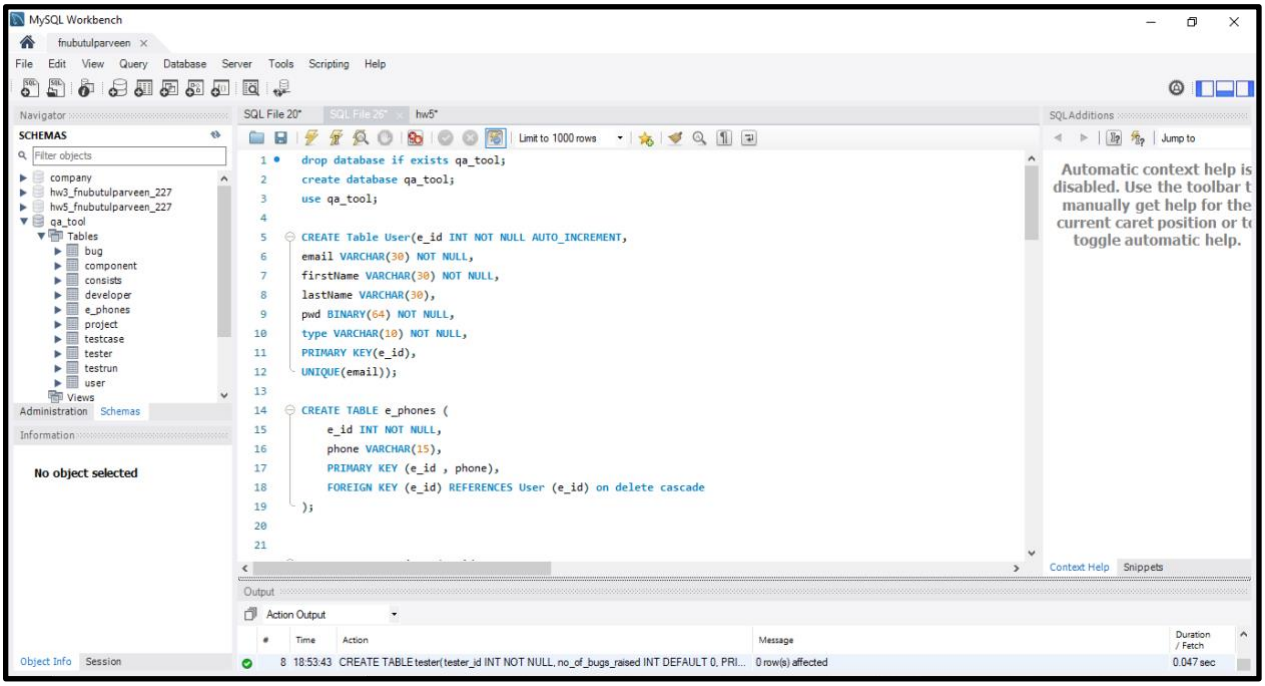
```
1
2  drop database if exists qa_tool;
3  create database qa_tool;
4  use qa_tool;
5
6  CREATE Table User(e_id INT NOT NULL AUTO_INCREMENT,
7  email VARCHAR(30) NOT NULL,
8  firstName VARCHAR(30) NOT NULL,
9  lastName VARCHAR(30),
10 pwd BINARY(64) NOT NULL,
11 type VARCHAR(10) NOT NULL,
12 PRIMARY KEY(e_id),
13 UNIQUE(email));
14
```

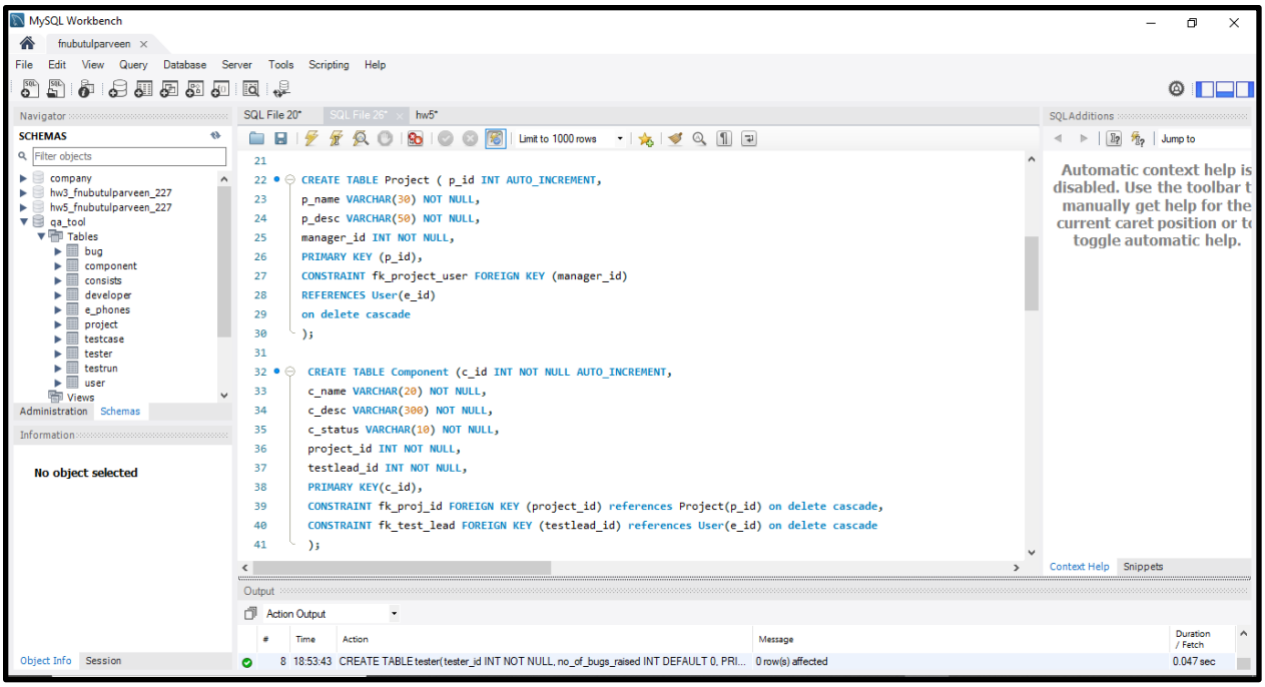
100% 1:1 4 errors found

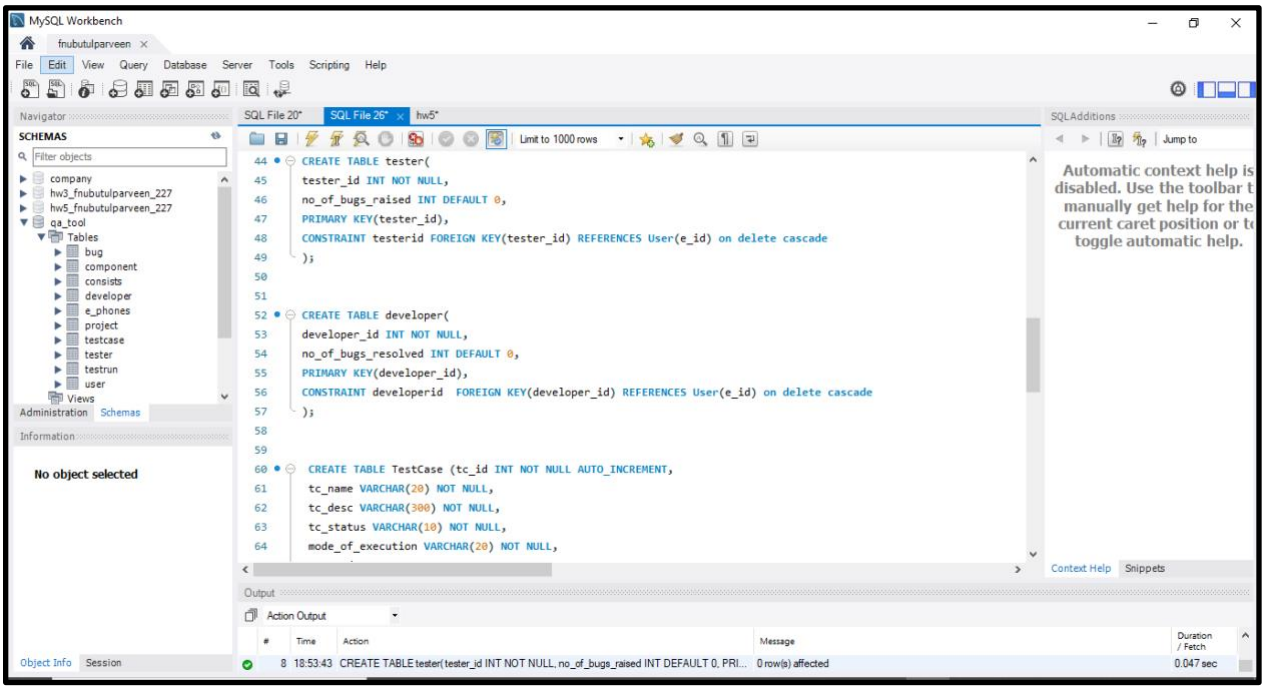
Result Grid Filter Rows: Search Edit: Export/Import:

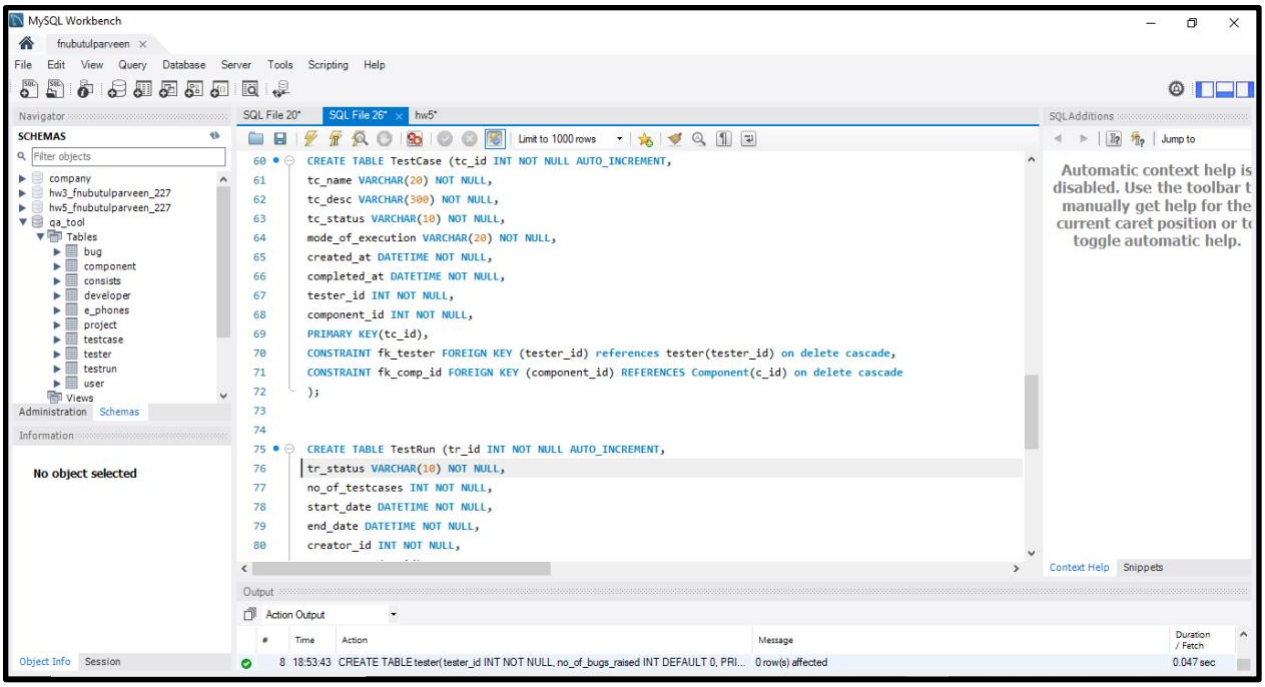
e_id	email	firstName	lastName	type	pwd
21	tester1@spartan.com	tester1	t1	tester	$2b$10$sY8pb3VCtWBcmTof/HRdBOsEjdzvt0NTmL.bxAStZVF7CVvFV8VKK
22	tester2@spartan.com	tester2	t2	tester	$2b$10$jcHIlYHhawpFMdbxCF1vLet9Lwq8SFNGZb4Z57Wte/TrRWNf6rlcq
23	tester3@spartan.com	tester3	t3	tester	$2b$10$EtHoOyeYzBzVCrQVp14xF.sKPAaxwVeDomjO4cVtfdai/wJvzTUBy
24	developer1@spartan.com	developer1	d1	developer	$2b$10$.lHKtdGiGnWvtWcw1nFKdOJlshc3R9jPpoINWbRrN.EkZ8ZzsVgPW
25	tester4@spartan.com	tester4	t4	tester	$2b$10$Pg0YT5r7jhJxE7cCqFwkBuoYO9s5S984OEmX4OH8duEm4tf3X.dMK
26	developer2@spartan.com	developer2	d2	developer	$2b$10$VAOkg6eQSvl74RnV3ly8LePuDUq7YsZT3qH8ZOfu3Kh4IB/tWzck.
27	tester5@spartan.com	tester5	t5	tester	$2b$10$hqV1X4yzBHMiWC9WGBhPQebDNy/y2LQrgiOUxn/HcZrbinE6PYKpq
28	developer3@spartan.com	developer3	d3	developer	$2b$10$2kR1ulyu4vyvuJwmq2cSR.JbNwPU48ltFJTTltS1AY7L4DyvVzlhC
29	manager1@spartan.com	manager1	m1	manager	$2b$10$AnCUy/dRc16lYfmA1WtF6OLshJbQCDfqHzb4fWe6QhkJCJcY5pixW
30	testlead1@spartan.com	testlead1	tl1	testlead	$2b$10$dngSG3IR/NC9zGkPbflwn.J.QewwD2iA06R2HnCrm/SugXlgbJpBu
31	manager2@spartan.com	manager2	m2	manager	$2b$10$O6UJlllXcT/siBVs9.qwueG3ivnOLYNtrIngKIS3u/7yc7XObnsMK
32	testlead2@spartan.com	testlead2	tl2	testlead	$2b$10$GLXmOyKOKCf8EamntYiObeb1BA1w9FQeVWNyXQSHPVO421sX3rnci
NULL	NULL	NULL	NULL	NULL	NULL

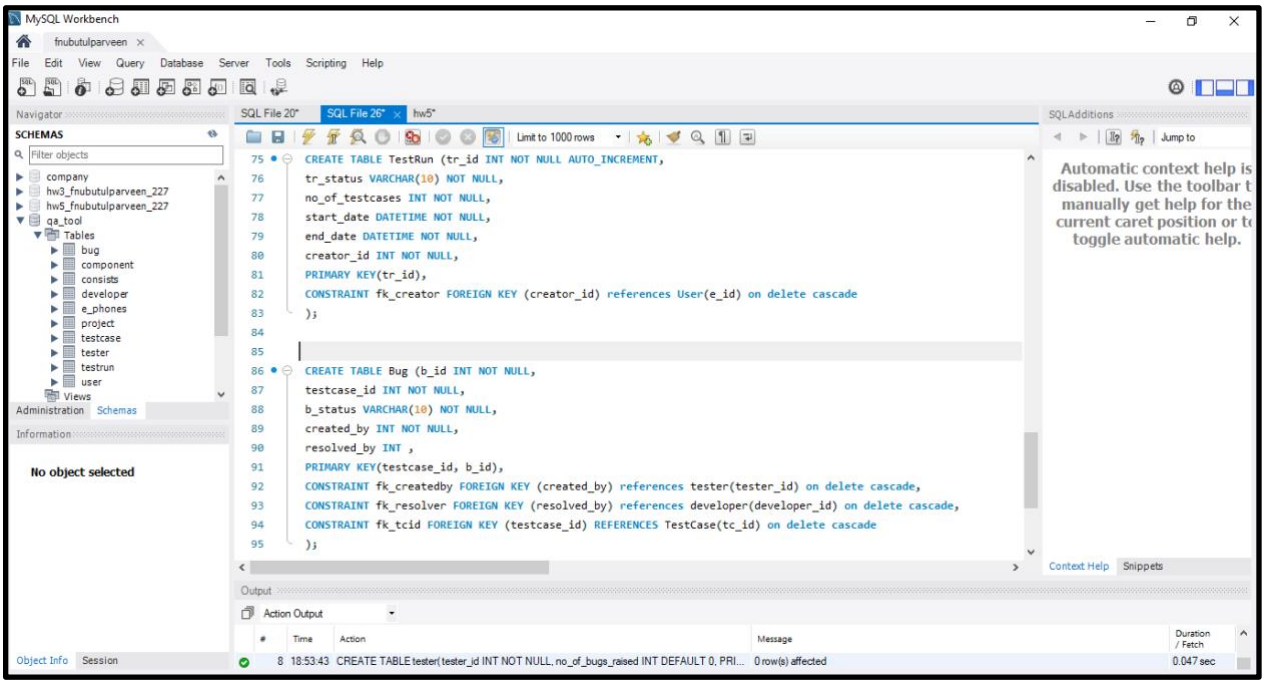
SQL Scripts:

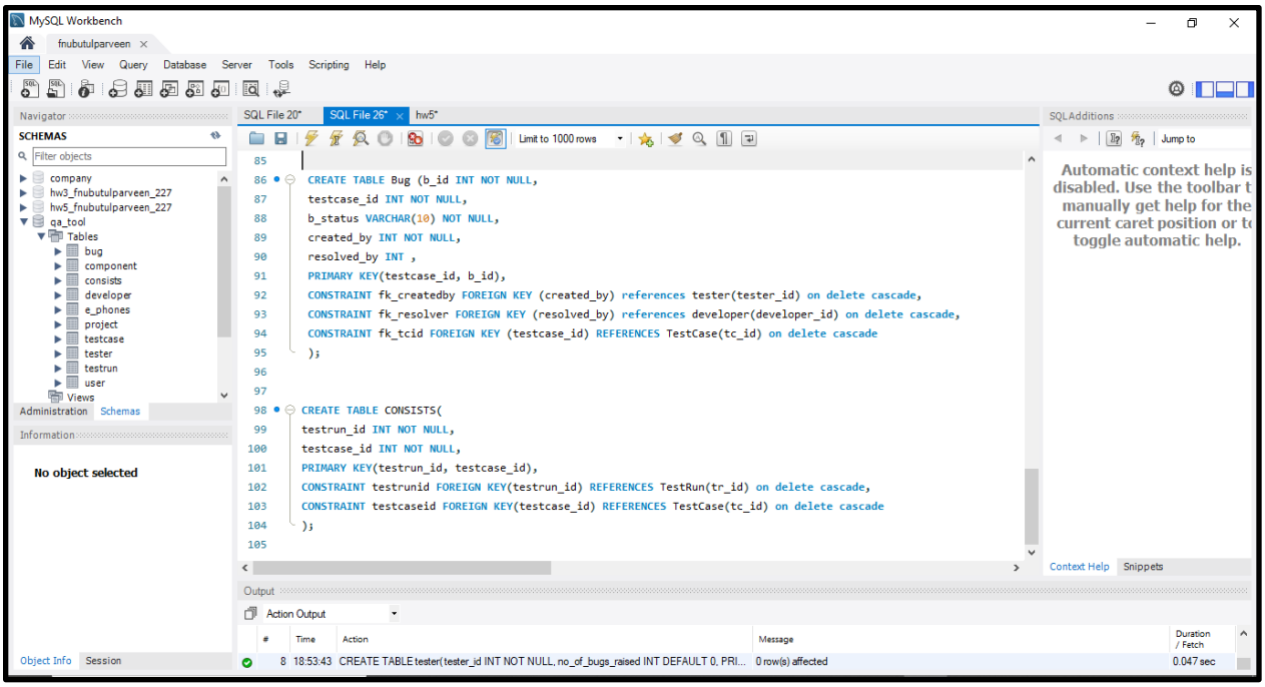



MySQL Workbench
fnubutulparveen
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
company
hw3_fnubutulparveen_227
hw5_fnubutulparveen_227
qa_tool
Tables
bug
component
consists
developer
e_phones
project
testcase
tester
testrun
user
Views
Administration Schemas
Information
No object selected
Object Info Session
SQL File 20* SQL File 26* hw5*
Limit to 1000 rows
21
22 CREATE TABLE Project (p_id INT AUTO_INCREMENT,
23 p_name VARCHAR(30) NOT NULL,
24 p_desc VARCHAR(50) NOT NULL,
25 manager_id INT NOT NULL,
26 PRIMARY KEY (p_id),
27 CONSTRAINT fk_project_user FOREIGN KEY (manager_id)
28 REFERENCES User(e_id)
29 on delete cascade
30);
31
32 CREATE TABLE Component (c_id INT NOT NULL AUTO_INCREMENT,
33 c_name VARCHAR(20) NOT NULL,
34 c_desc VARCHAR(300) NOT NULL,
35 c_status VARCHAR(10) NOT NULL,
36 project_id INT NOT NULL,
37 testlead_id INT NOT NULL,
38 PRIMARY KEY(c_id),
39 CONSTRAINT fk_proj_id FOREIGN KEY (project_id) references Project(p_id) on delete cascade,
40 CONSTRAINT fk_test_lead FOREIGN KEY (testlead_id) references User(e_id) on delete cascade
41);
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
8 18:53:43 CREATE TABLE tester(tester_id INT NOT NULL, no_of_bugs_raised INT DEFAULT 0, PRI... 0 row(s) affected 0.047 sec


MySQL Workbench
fnubutulparveen
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
company
hw3_fnubutulparveen_227
hw5_fnubutulparveen_227
qa_tool
Tables
bug
component
consists
developer
e_phones
project
testcase
tester
testrun
user
Views
Administration Schemas
Information
No object selected
Object Info Session
SQL File 20* SQL File 26* hw5*
Limit to 1000 rows
44 CREATE TABLE tester(
45 tester_id INT NOT NULL,
46 no_of_bugs_raised INT DEFAULT 0,
47 PRIMARY KEY(tester_id),
48 CONSTRAINT testerid FOREIGN KEY(tester_id) REFERENCES User(e_id) on delete cascade
49);
50
51
52 CREATE TABLE developer(
53 developer_id INT NOT NULL,
54 no_of_bugs_resolved INT DEFAULT 0,
55 PRIMARY KEY(developer_id),
56 CONSTRAINT developerid FOREIGN KEY(developer_id) REFERENCES User(e_id) on delete cascade
57);
58
59
60 CREATE TABLE TestCase (tc_id INT NOT NULL AUTO_INCREMENT,
61 tc_name VARCHAR(20) NOT NULL,
62 tc_desc VARCHAR(300) NOT NULL,
63 tc_status VARCHAR(10) NOT NULL,
64 mode_of_execution VARCHAR(20) NOT NULL,
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
8 18:53:43 CREATE TABLE tester(tester_id INT NOT NULL, no_of_bugs_raised INT DEFAULT 0, PRI... 0 row(s) affected 0.047 sec


MySQL Workbench
fnubutulparveen
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
company
hw3_fnubutulparveen_227
hw5_fnubutulparveen_227
qa_tool
Tables
bug
component
consists
developer
e_phones
project
testcase
tester
testrun
user
Views
Administration Schemas
Information
No object selected
Object Info Session
SQL File 20* SQL File 26* hw5*
Limit to 1000 rows
60 CREATE TABLE TestCase (tc_id INT NOT NULL AUTO_INCREMENT,
61 tc_name VARCHAR(20) NOT NULL,
62 tc_desc VARCHAR(300) NOT NULL,
63 tc_status VARCHAR(10) NOT NULL,
64 mode_of_execution VARCHAR(20) NOT NULL,
65 created_at DATETIME NOT NULL,
66 completed_at DATETIME NOT NULL,
67 tester_id INT NOT NULL,
68 component_id INT NOT NULL,
69 PRIMARY KEY(tc_id),
70 CONSTRAINT fk_tester FOREIGN KEY (tester_id) references tester(tester_id) on delete cascade,
71 CONSTRAINT fk_comp_id FOREIGN KEY (component_id) REFERENCES Component(c_id) on delete cascade
72);
73
74
75 CREATE TABLE TestRun (tr_id INT NOT NULL AUTO_INCREMENT,
76 tr_status VARCHAR(10) NOT NULL,
77 no_of_testcases INT NOT NULL,
78 start_date DATETIME NOT NULL,
79 end_date DATETIME NOT NULL,
80 creator_id INT NOT NULL,
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
8 18:53:43 CREATE TABLE tester(tester_id INT NOT NULL, no_of_bugs_raised INT DEFAULT 0, PRI... 0 row(s) affected 0.047 sec


MySQL Workbench
fnubutulparveen
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
company
hw3_fnubutulparveen_227
hw5_fnubutulparveen_227
qa_tool
Tables
bug
component
consists
developer
e_phones
project
testcase
tester
testrun
user
Views
Administration Schemas
Information
No object selected
Object Info Session
SQL File 20* SQL File 26* hw5*
Limit to 1000 rows
75 CREATE TABLE TestRun (tr_id INT NOT NULL AUTO_INCREMENT,
76 tr_status VARCHAR(10) NOT NULL,
77 no_of_testcases INT NOT NULL,
78 start_date DATETIME NOT NULL,
79 end_date DATETIME NOT NULL,
80 creator_id INT NOT NULL,
81 PRIMARY KEY(tr_id),
82 CONSTRAINT fk_creator FOREIGN KEY (creator_id) references User(e_id) on delete cascade
83);
84
85
86 CREATE TABLE Bug (b_id INT NOT NULL,
87 testcase_id INT NOT NULL,
88 b_status VARCHAR(10) NOT NULL,
89 created_by INT NOT NULL,
90 resolved_by INT ,
91 PRIMARY KEY(testcase_id, b_id),
92 CONSTRAINT fk_createdby FOREIGN KEY (created_by) references tester(tester_id) on delete cascade,
93 CONSTRAINT fk_resolver FOREIGN KEY (resolved_by) references developer(developer_id) on delete cascade,
94 CONSTRAINT fk_tcid FOREIGN KEY (testcase_id) REFERENCES TestCase(tc_id) on delete cascade
95);
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
8 18:53:43 CREATE TABLE tester(tester_id INT NOT NULL, no_of_bugs_raised INT DEFAULT 0, PRI... 0 row(s) affected 0.047 sec



Querry1.Based on the Project status,the Component is TestReady or Completed

```sql
select Project.p_id,Project.p_name,Project.p_desc,Project.manager_id,project_status_table.TestReady,project_status_table.Completed
from(
    select intertable.p_id,COALESCE(intertable.TestReady,0) as TestReady,COALESCE(intertable.Completed,0) as Completed
    from (
        select a1.p_id, TestReady, Completed
                from (select tab1.p_id as p_id, tab1.cnt as TestReady
                from ((select p_id,c_status,count(c_id) as cnt
                        from Project Left Join Component on p_id=project_id
                        group by Project.p_id,c_status) )as tab1
                where tab1.c_status='TestReady' or tab1.c_status is Null)as a1 Left JOIN
                (select tab2.p_id as p_id, tab2.cnt as Completed
                from ((select p_id,c_status,count(c_id) as cnt
                        from Project Left Join Component on p_id=project_id
                        group by Project.p_id,c_status) )as tab2
                where tab2.c_status='Completed' or tab2.c_status is Null)as a2 on a1.p_id=a2.p_id
            UNION
            select a2.p_id, TestReady, Completed
                from (select tab1.p_id as p_id, tab1.cnt as TestReady
                from ((select p_id,c_status,count(c_id) as cnt
                        from Project Left Join Component on p_id=project_id
                        group by Project.p_id,c_status) )as tab1
                where tab1.c_status='TestReady' or tab1.c_status is Null)as a1 RIGHT JOIN
                (select tab2.p_id as p_id, tab2.cnt as Completed
                from ((select p_id,c_status,count(c_id) as cnt
                        from Project Left Join Component on p_id=project_id
                        group by Project.p_id,c_status) )as tab2
                where tab2.c_status='Completed' or tab2.c_status is Null)as a2 on a1.p_id=a2.p_id) as intertable)as project_status_table, Project
        where Project.p_id=project_status_table.p_id;
```

p_id	p_name	p_desc	manager_id	TestReady	Completed
4	Project1	This is the description for project 1.....	29	2	0
5	Project2	This is the description for project 2	29	1	2
6	Project3	This is the description for project 3	29	2	1
8	Project4	This is the description for project 4	31	2	0
9	Project5	This is the description for project 5	31	2	0
10	Project6	This is the description for project 6	31	1	2
12	Project7	This is the description for project 7	29	0	1

Query 2.Based on component the test case is passed,Failed,Blocked,InProgress and Todo

```sql
SELECT finaltable.c_id,finaltable.c_name,finaltable.c_desc ,COALESCE(finaltable.Passed,0) as Passed,COALESCE(finaltable.Failed,0)
as Failed,COALESCE(finaltable.Blocked,0) as Blocked,COALESCE(finaltable.InProgress,0) as InProgress,COALESCE(finaltable.ToDo,0) as ToDo
from (SELECT Component.c_id,Component.c_name,Component.c_desc,Passed, Failed, Blocked, InProgress, ToDo
        FROM Component
                        Left Join
                                (select table1.c_id,table1.c_name,table1.c_desc,table1.cnt as Passed
                                from (
                                select c_id,c_name,c_desc,c_status,tc_status,count(tc_status) as cnt
                                from Component Left Join TestCase on c_id=component_id
                                group by c_id,c_name,c_desc,tc_status) as table1
                                where tc_status='Passed' or tc_status is Null) as passtable on Component.c_id=passtable.c_id
                        Left Join
                                (select table2.c_id,table2.c_name,table2.c_desc,table2.cnt as Failed
                                from (
                                select c_id,c_name,c_desc,c_status,tc_status,count(tc_status) as cnt
                                from Component Left Join TestCase on c_id=component_id
                                group by c_id,c_name,c_desc,tc_status) as table2
```

```
152                         group by c_id,c_name,c_desc,tc_status) as table2
153                         where tc_status='Failed' or tc_status is Null) as failtable on Component.c_id=failtable.c_id
154                 Left Join
155                         (select table3.c_id,table3.c_name,table3.c_desc,table3.cnt as Blocked
156                         from (
157                         select c_id,c_name,c_desc,c_status,tc_status,count(tc_status) as cnt
158                         from Component Left Join TestCase on c_id=component_id
159                         group by c_id,c_name,c_desc,tc_status) as table3
160                         where tc_status='Blocked' or tc_status is Null) as blocktable on Component.c_id=blocktable.c_id
161                 Left Join
162                         (select table4.c_id,table4.c_name,table4.c_desc,table4.cnt as InProgress
163                         from (
164                         select c_id,c_name,c_desc,c_status,tc_status,count(tc_status) as cnt
165                         from Component Left Join TestCase on c_id=component_id
166                         group by c_id,c_name,c_desc,tc_status) as table4
167                         where tc_status='InProgress' or tc_status is Null) as inprogresstable on Component.c_id=inprogresstable.c_id
```

```
168                 Left Join
169                         (select table5.c_id,table5.c_name,table5.c_desc,table5.cnt as ToDo
170                         from (
171                         select c_id,c_name,c_desc,c_status,tc_status,count(tc_status) as cnt
172                         from Component Left Join TestCase on c_id=component_id
173                         group by c_id,c_name,c_desc,tc_status) as table5
174                         where tc_status='ToDo' or tc_status is Null) as todotable on Component.c_id=todotable.c_id)as finaltable;
```

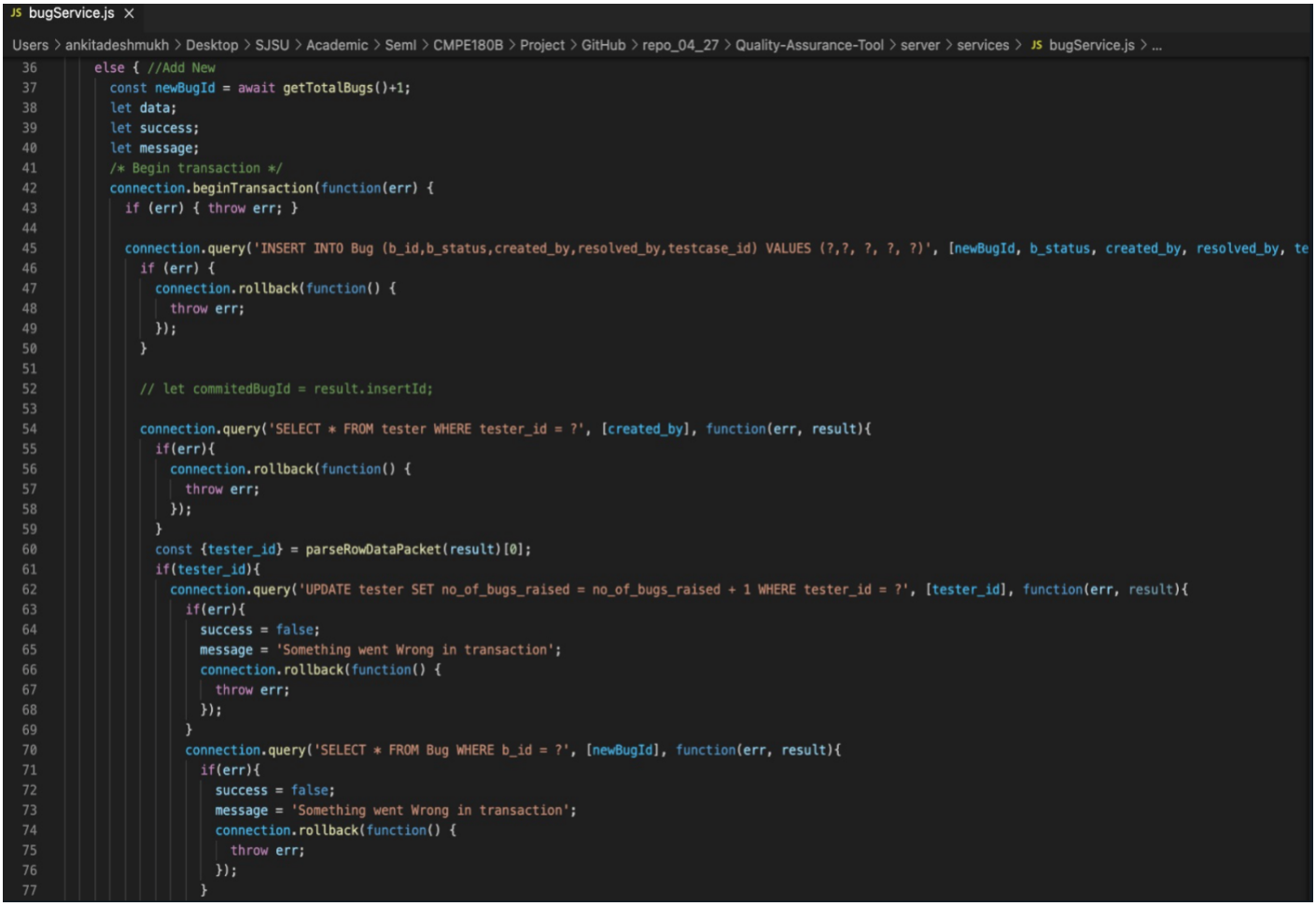
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

c_id	c_name	c_desc	Passed	Failed	Blocked	InProgress	ToDo
6	Component9	This is the description for Component 9	1	1	0	0	0
7	Component1	This is the description for Component 1	1	0	1	0	0
8	Component10	This is the description for Component 10	0	1	1	0	0
9	Component11	This is the description for Component 11	1	0	0	1	0
10	Component12	This is the description for Component 12	1	0	0	2	0
11	Component2	This is the description for Component 2	0	1	0	1	0
12	Component3	This is the description for Component 3	0	0	0	2	0
13	Component13	This is the description for Component 13	0	0	0	2	0
14	Component4	This is the description for Component 4	3	0	0	0	0
15	Component14	This is the description for Component 14	2	0	0	0	0
16	Component15	This is the description for Component 15	2	0	0	0	0
17	Component5	This is the description for Component 5	2	0	0	0	0
18	Component6	This is the description for Component 6	0	0	0	0	2
19	Component7	This is the description for Component 7	0	0	0	0	2

Final design of DB application portion

MULTI-SQL-STATEMENT DB TRANSACTIONS :

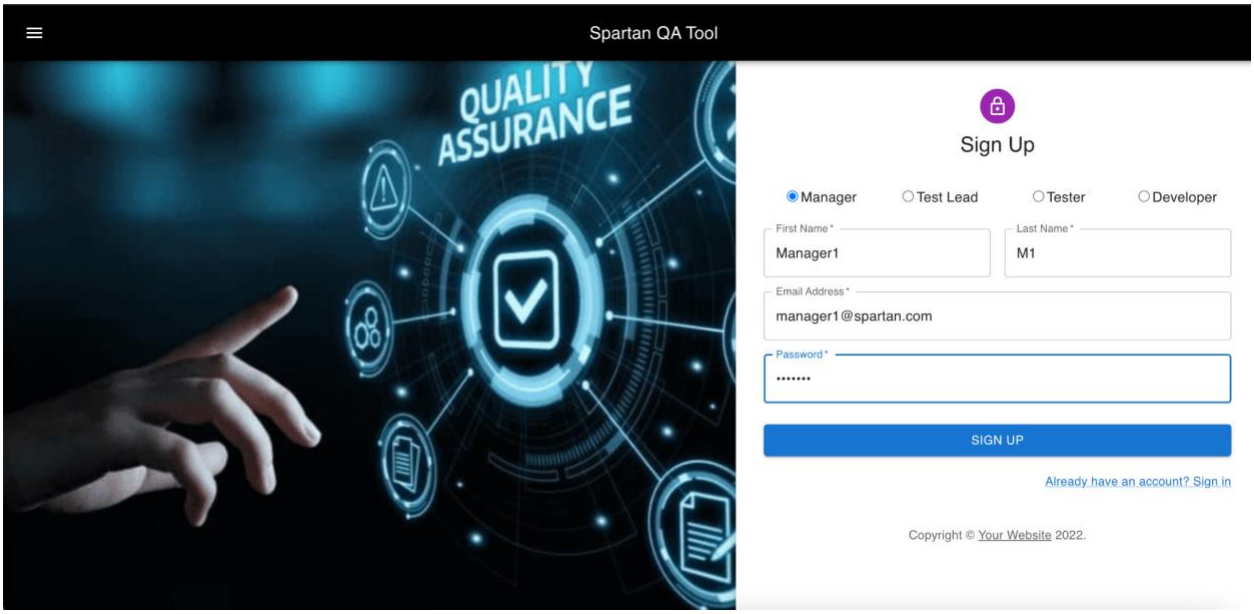
- When a tester raises a bug, a new entry is to be created in the ‘Bug’ table. At the same time the count of the ‘num_of_bugs_raised’ needs to be increased in the ‘Tester’ table. We have combined these 2 SQL statements as a Transaction and implemented them from the application side.



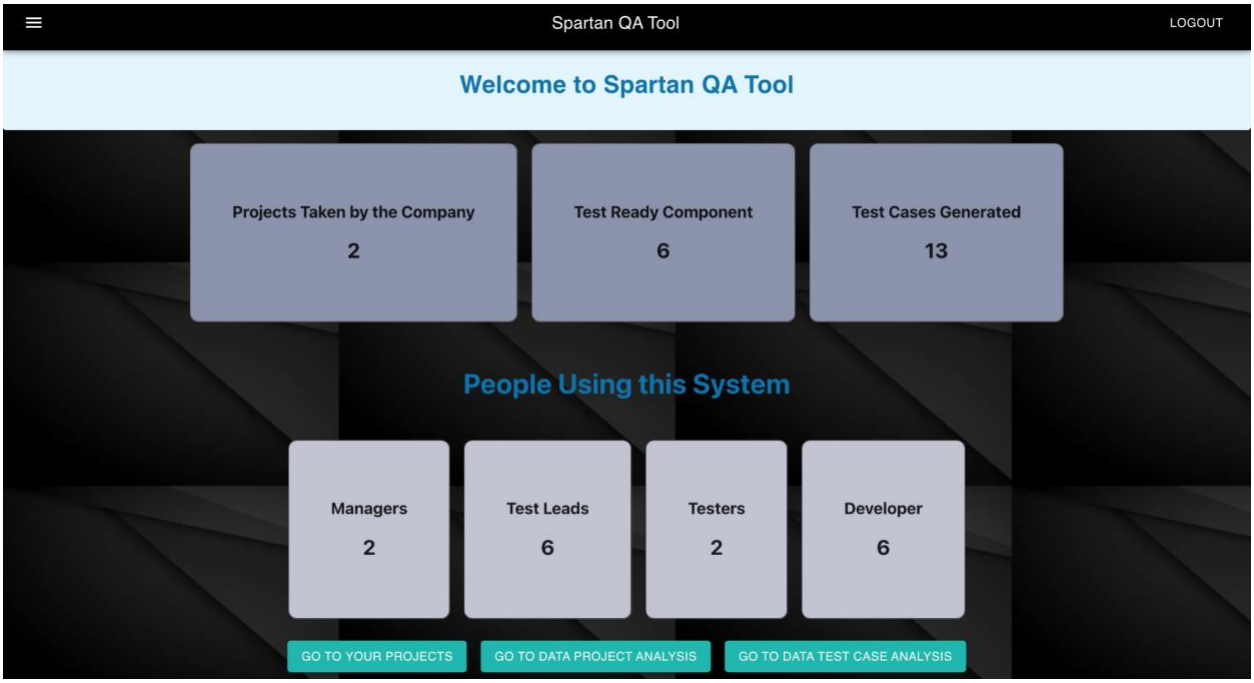
```
                    data = parseRowDataPacket(result);
                    success = true;
                  });
                });
              }
              else{
                connection.query('INSERT INTO tester (tester_id, no_of_bugs_raised) VALUES (?, ?)', [created_by, 0], function(err, result){
                  if(err){
                    success = false;
                    message = 'Something went Wrong in transaction';
                    connection.rollback(function() {
                      throw err;
                    });
                  }
                  connection.query('SELECT * FROM Bug WHERE b_id = ?', [commitedBugId], function(err, result){
                    data = parseRowDataPacket(result);
                    success = true;
                  })
                })
              }
              connection.commit(function(err) {
                if (err) {
                  connection.rollback(function() {
                    throw err;
                  });
                }
                connection.end();
              });
            });
          });
          // return{
          //   success,
          //   data: data[0],
          //   message
          // }
        });

        /* End transaction */
```

Screenshots - Sample App execution

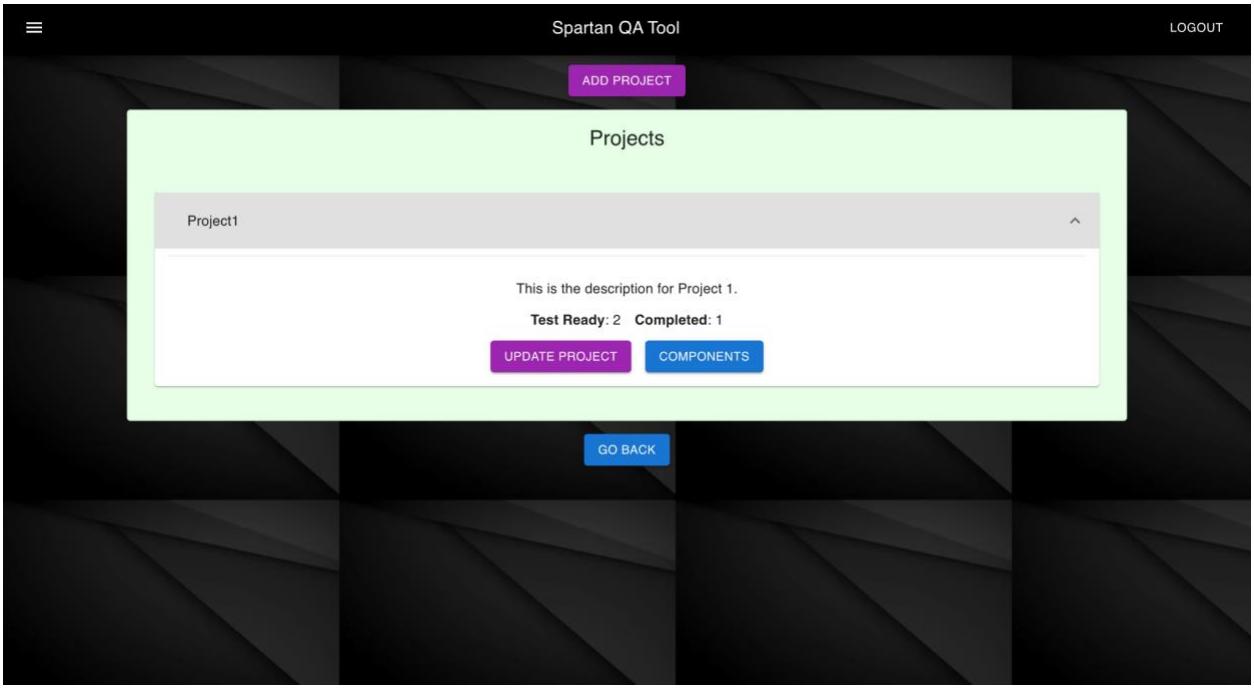
Signup



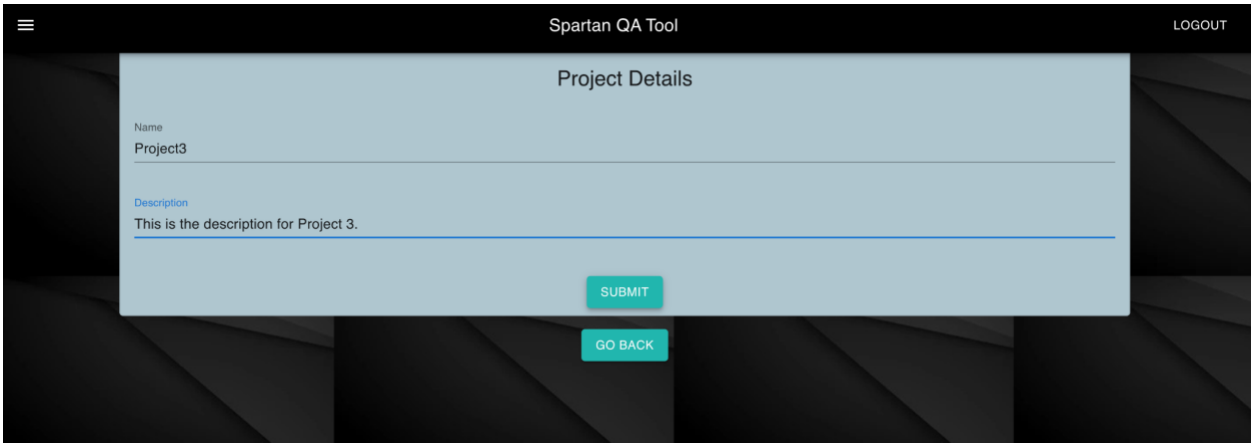
Home page for manager



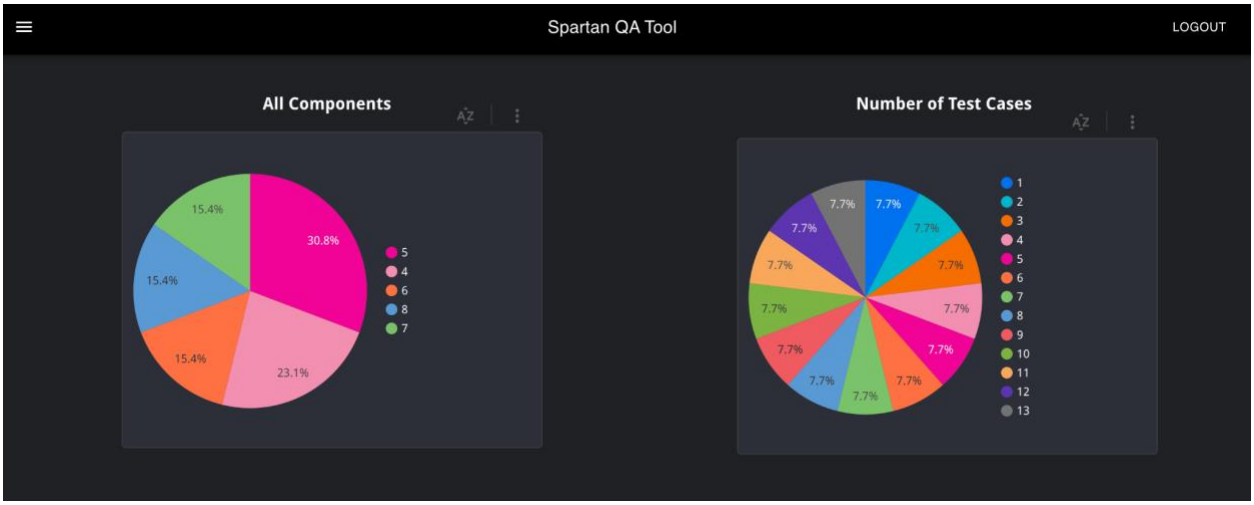
Project View for manager

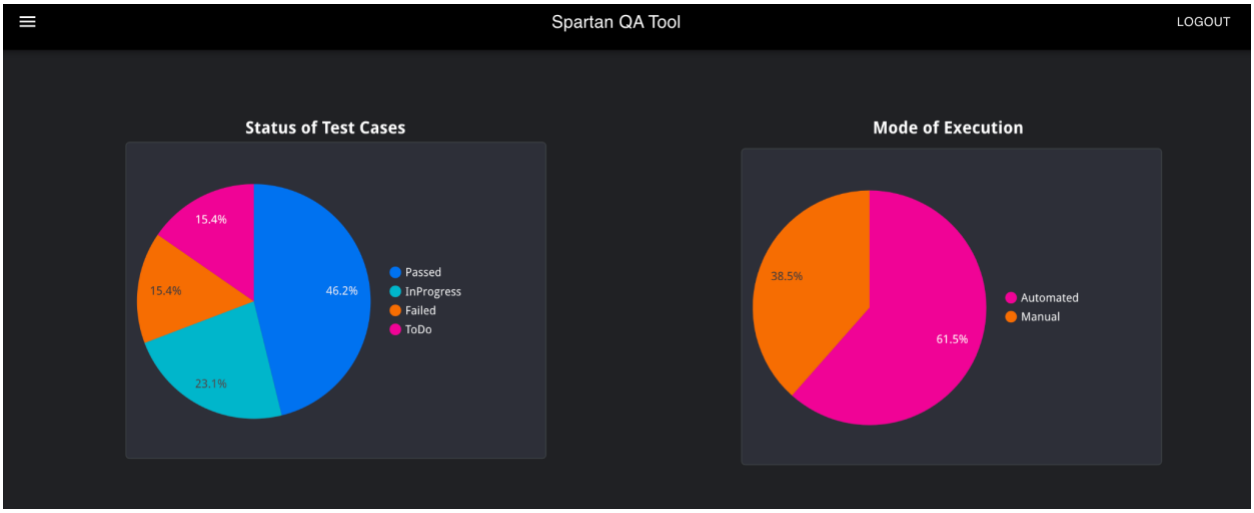


Create Project for Manager

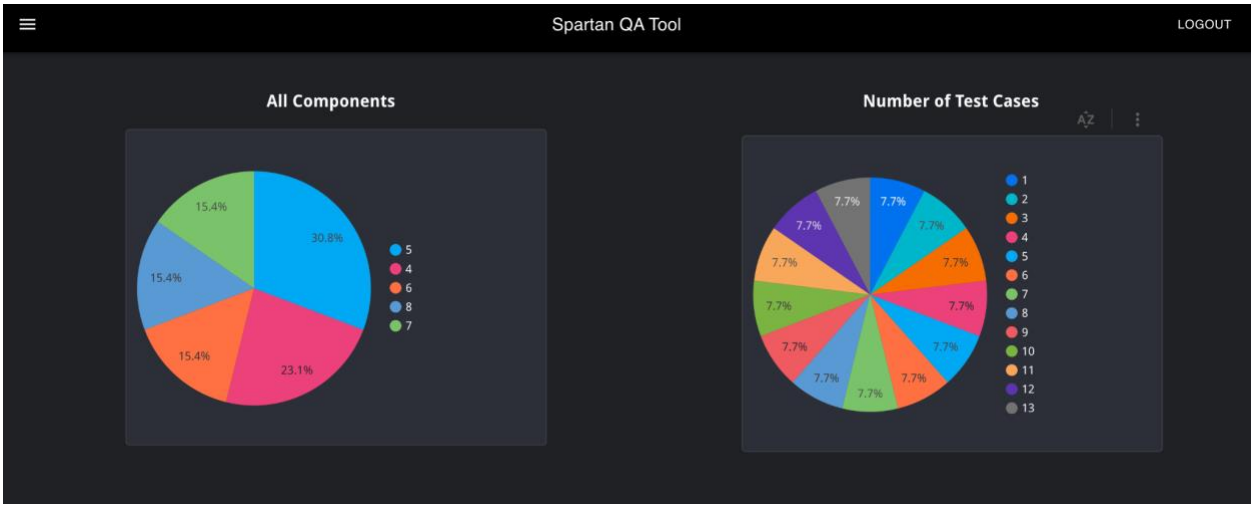


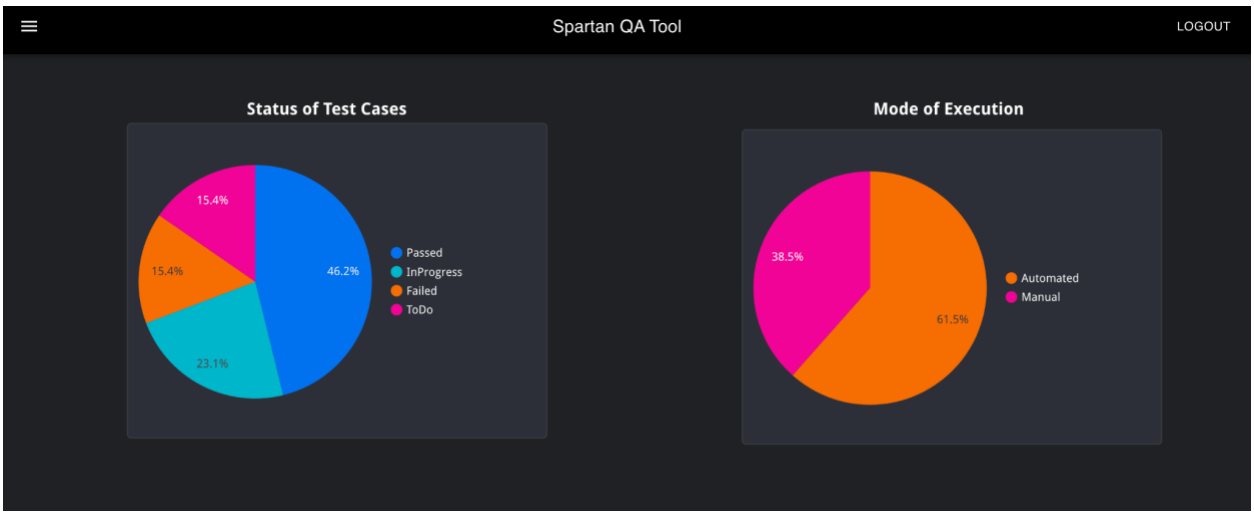
Project-Component Analysis view for manager



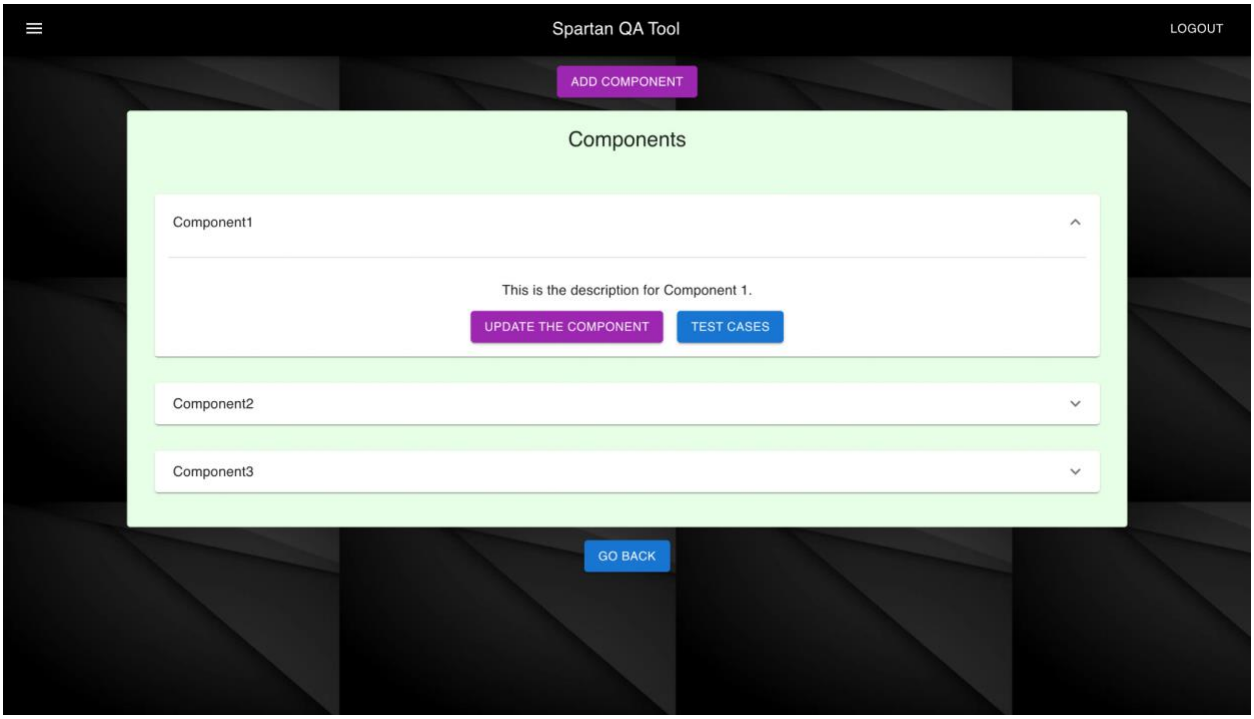


Component-TestCase Analysis for manager

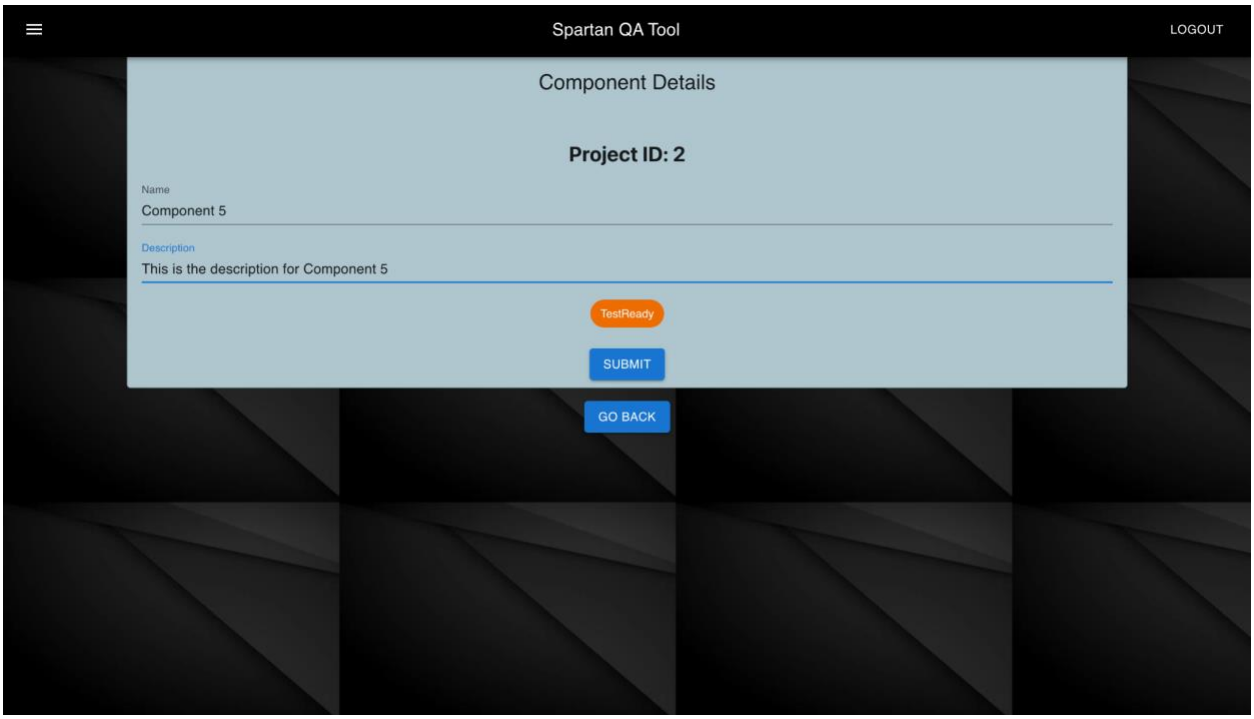




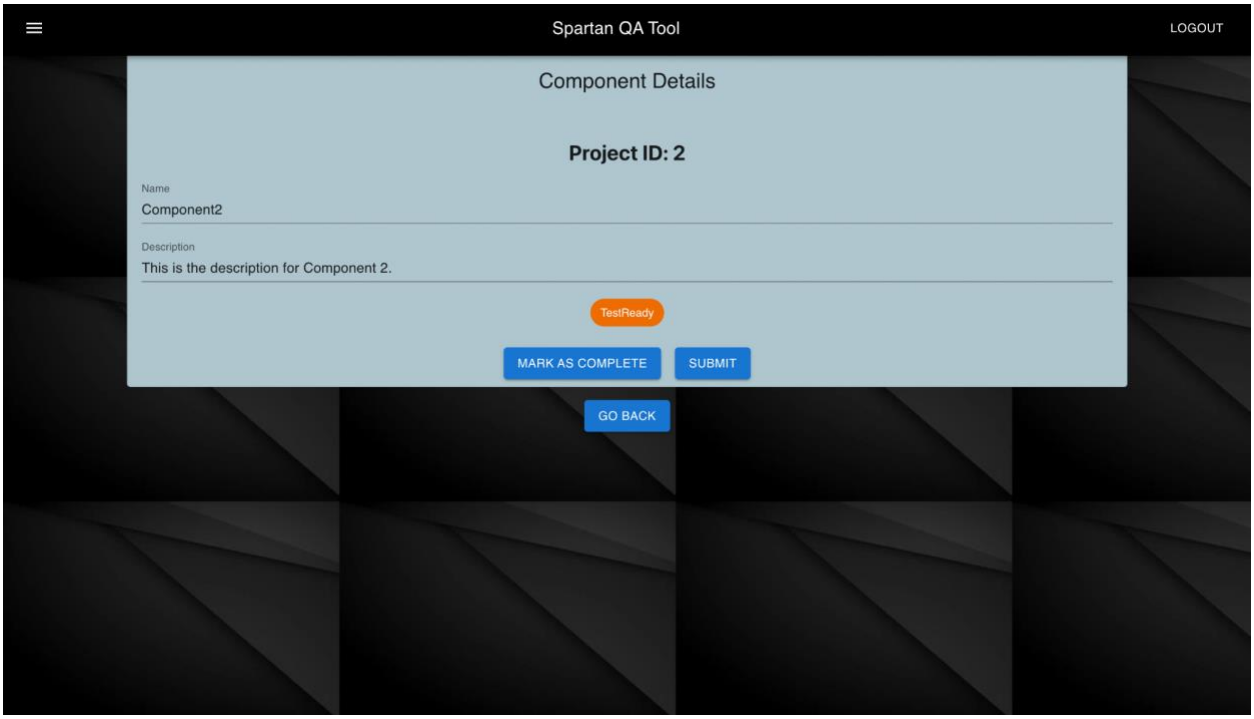
Test Lead’s view for adding and updating the Components



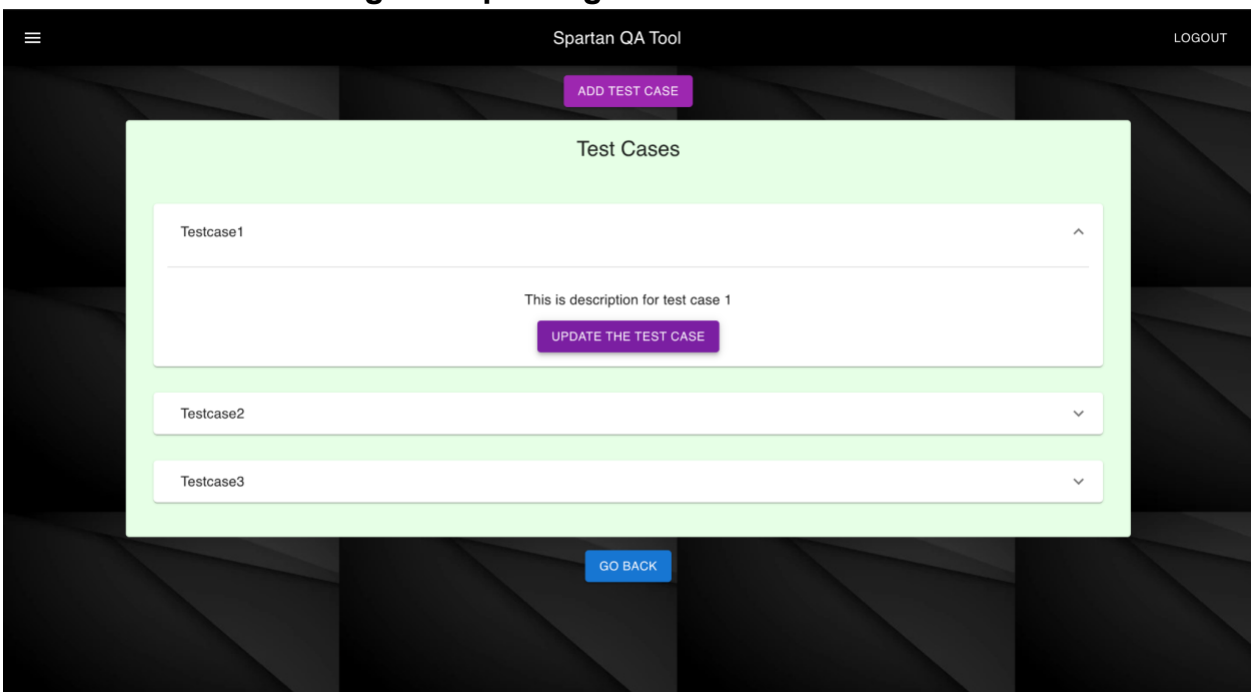
Testlead's view to a create Component



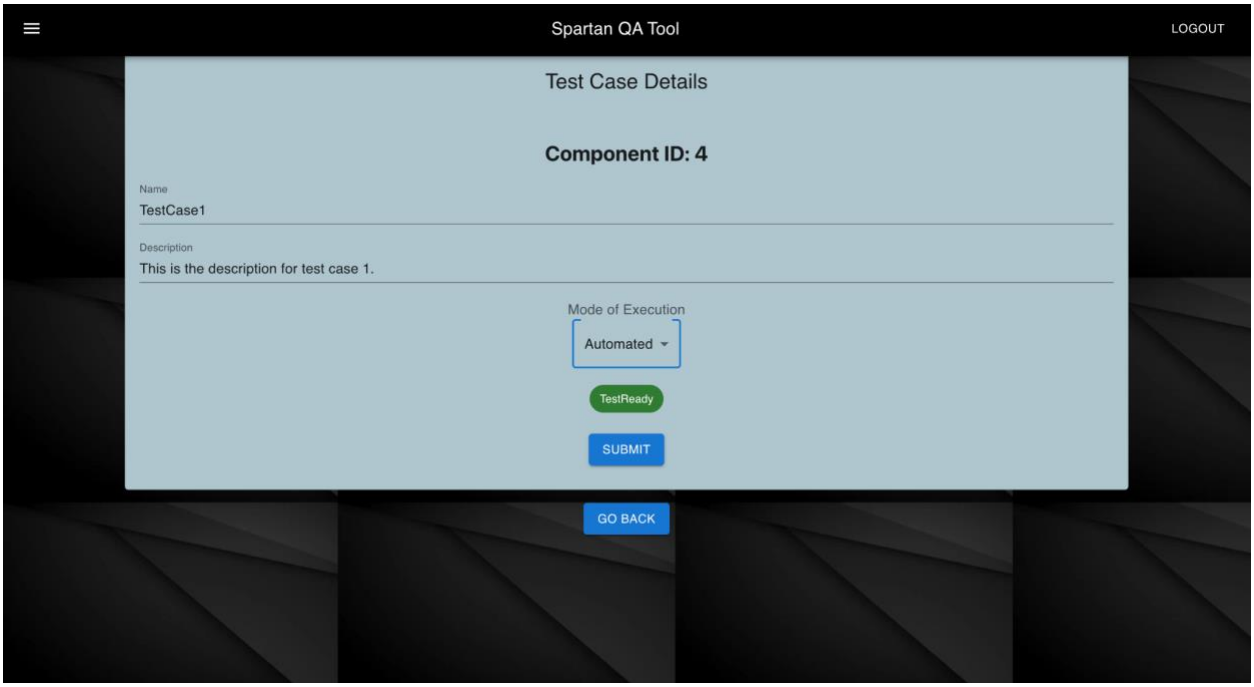
Testlead's view for updating a Component



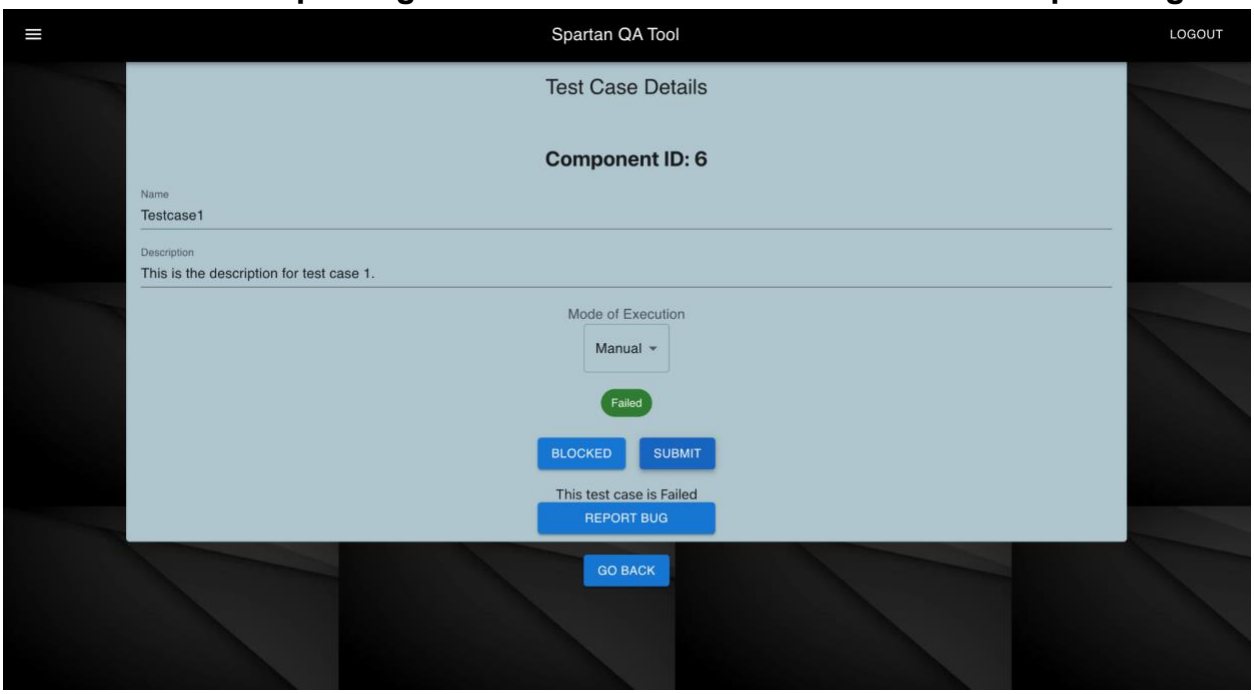
Tester's view for adding and updating test cases



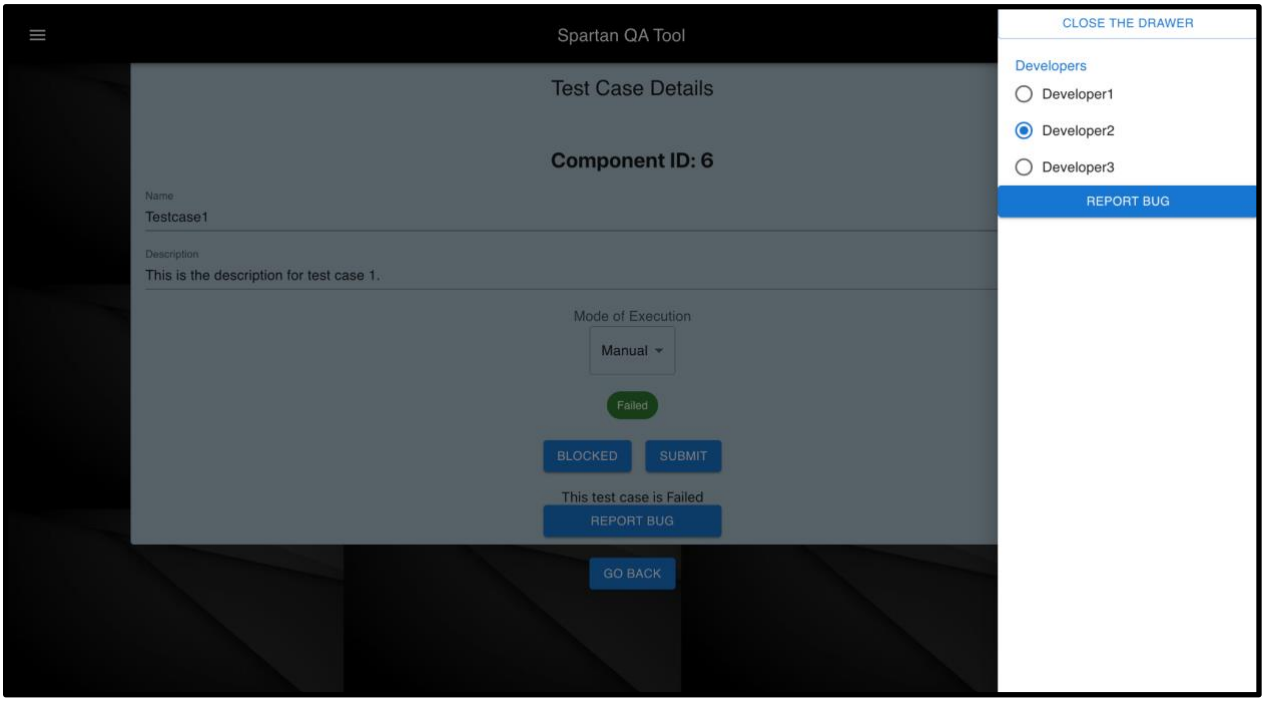
Tester's view for creating a new test case for a particular component



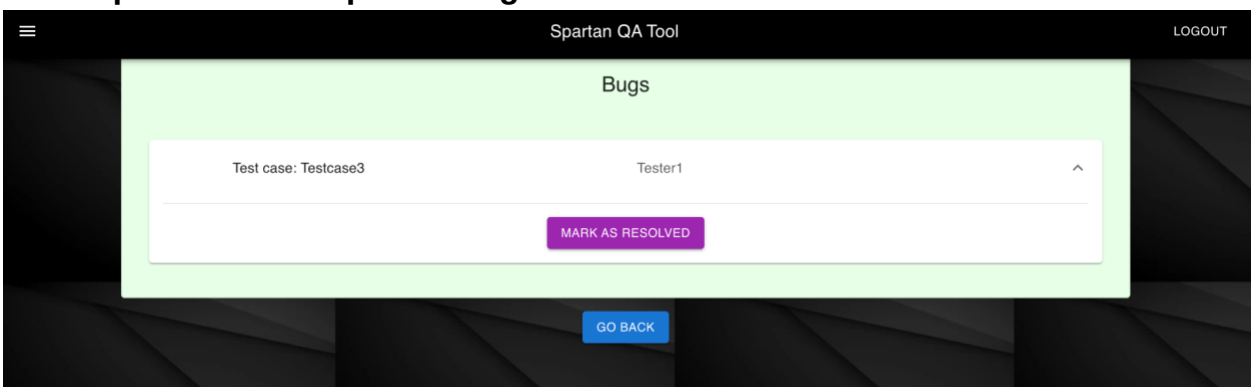
Tester's view for updating the status of a current test case or to report bug.



Tester reporting a bug for a test case to a developer



Developers view of reported bugs



Any major design decisions, trade-offs(and why)

We had initially planned that the manager assigns projects to the test leads, and the test lead assigns components to testers and developers. But while implementing we have not provided the 'assign' functionality for different users. We gave the flexibility to the test leads to take up a project and create components for it. Similarly, testers can view different components and create test cases for those components which he selects. If the tester finds a bug, he assigns the bug to the developer instead of the test lead.

We went ahead with this design because it was inline with our EERD design. The trade off is that in a real world corporate environment, tasks are assigned by superiors to subordinates and in our application this responsibility is on individual users.

Any major modifications from the proposal , EERD

There is a slight modification in EERD from the approved one. Originally, the relationship between Project and Component was of total participation which is now changed to partial participation. This is because a newly created project may not have the components.

Any unique designs you are proud of

We are proud of our EERD design and implementation of the EERD in Spartan QA Tool. We have implemented the pie charts in the front end by using a database.

Functionality test cases, and test plan execution

TC1	**Title**	Manager login to portal and project **view**
	Description	Manager is able to login to the portal with manager role and view the project list
	Steps	1. Login to Spartan QA tool with user login and select Manager radio button. 2. After successful login, select the project from drop down.
	Expected result:	Manager should be able to view only his/her own projects Manager should be unable to view the projects for which is not the manager
	Actual result:	Manager is able to view only his/her own projects Manager is unable to view the projects for which is not the manager
	Test result	Pass

TC2	**Title**	Manager login to portal and **add** project
	Description	Manager is able to login to the portal with manager role and add a new project
	Steps	1. Login to Spartan QA tool with user login and select Manager radio button. 2. After successful login, click on add project button.
	Expected result	Manager should be able to add a new project.
	Actual result:	Manager is able to add a new project.
	Test result	Pass

TC3	**Title**	Manager login to portal and **update** project
	Description	Manager is able to login to the portal with manager role and add a new project
	Steps	1. Login to Spartan QA tool with user login and select Manager radio button. 2. After successful login, click on update project button.
	Expected result	Manager should be able to update a project.
	Actual result:	Manager is able to update a project.
	Test result	Pass

TC4	**Title**	Test lead login to portal and **component view**
	Description	Test lead is able to login to the portal with lead role and view the component list
	Steps	1. Login to Spartan QA tool with user login and select lead radio button. 2. After successful login, select the component from drop down.
	Expected result	Test lead should be able to view the components.
	Actual result:	Test lead is able to view the components.
	Test result	Pass

TC5	**Title**	Test lead login to portal and **add** component
	Description	Test lead is able to login to the portal with lead role and add a new component.
	Steps	1. Login to Spartan QA tool with user login and select lead radio button. 2. After successful login, click on add component button.
	Expected result	Test lead should be able to add a component.
	Actual result:	Test lead is able to add components.
	Test result	Pass

TC6	**Title**	Test lead login to portal and **update** component
	Description	Test lead is able to login to the portal with lead role and update a new component.
	Steps	1. Login to Spartan QA tool with user login and select lead radio button. 2. After successful login, click on update component button.
	Expected result	Test lead should be able to update a component.
	Actual result:	Test lead is able to update components.
	Test result	Pass

TC7	**Title**	Tester login to portal and **test case view**
	Description	Tester is able to login to the portal with tester role and view the test cases list
	Steps	1. Login to Spartan QA tool with user login and select tester radio button. 2. After successful login, select the testcases from drop down.
	Expected result	Tester should be able to view the testcases.
	Actual result:	Tester is able to view the testcases.
	Test result	Pass

TC8	**Title**	Tester login to portal and **add** test case
	Description	Tester is able to login to the portal with tester role and add a new test case.
	Steps	1. Login to Spartan QA tool with user login and select tester radio button. 2. After successful login, click on add test case button.
	Expected result	Tester should be able to add a testcase.
	Actual result:	Tester is able to add a testcase.
	Test result	Pass

TC9	**Title**	Tester login to portal and **update** test case
	Description	Tester is able to login to the portal with tester role and update a test case.
	Steps	1. Login to Spartan QA tool with user login and select tester radio button. 2. After successful login, click on update test case button.
	Expected result	Tester should be able to update a testcase.
	Actual result:	Tester is able to update a testcase.
	Test result	Pass

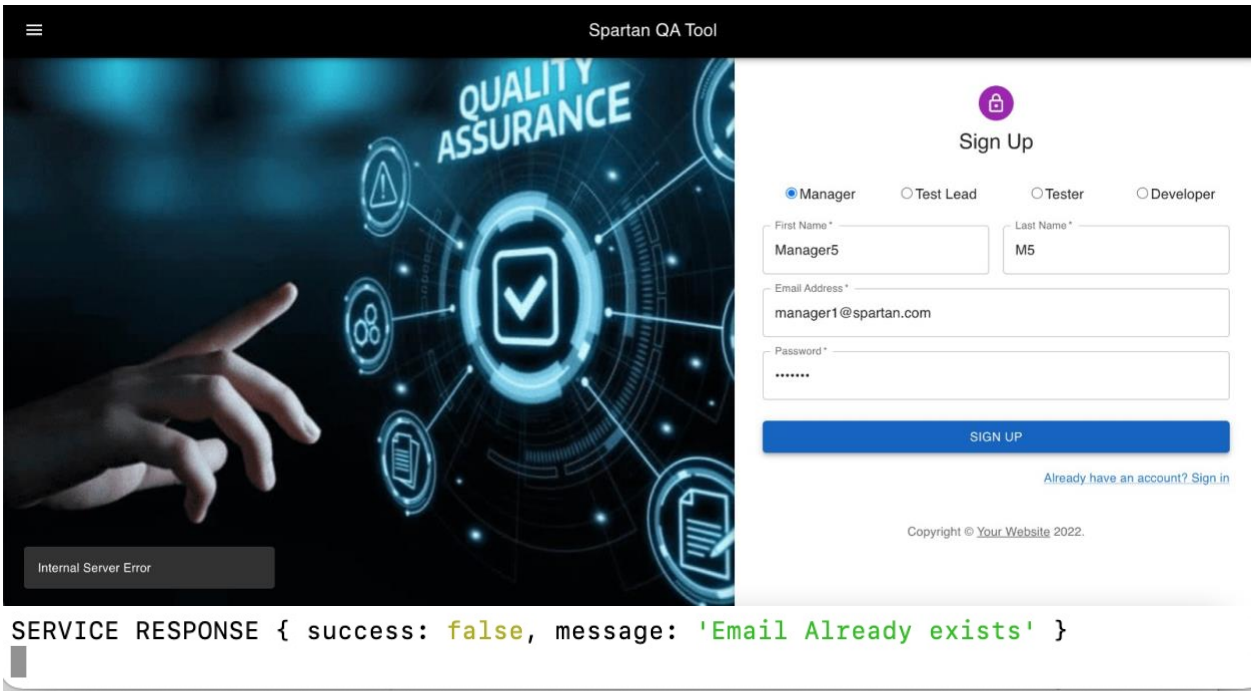
TC10	**Title**	Update test case execution status to in progress
	Description	Tester is able to update test cases execution status to in progress
	Steps	1. Login to Spartan QA tool with user login and select tester radio button. 2. After successful login, select a test cases with To Do status. 3. Click on in progress button to show start of execution of the test case.
	Expected result	Tester should be able to update testcase status to In progress from To Do.
	Actual result:	Tester is able to update testcase status to In progress from To Do.
	Test result	Pass

TC11	**Title**	Update test case execution status to Pass
	Description	Tester is able to update test cases execution status to in progress
	Steps	1. Login to Spartan QA tool with user login and select tester radio button. 2. After successful login, select a test cases with To Do status. 3. Click on in progress button to show start of execution of the test case. 4. Pass/Fail button should be displayed. 5. Click on Pass button.
	Expected result	Tester should be able to update testcase status to Pass.
	Actual result:	Tester is be able to update testcase status to Pass.
	Test result	Pass

TC12	**Title**	Update test case execution status to Fail
	Description	Tester is able to update test cases execution status to in progress
	Steps	1. Login to Spartan QA tool with user login and select tester radio button. 2. After successful login, select a test cases with To Do status. 3. Click on in progress button to show start of execution of the test case. 4. Pass/Fail button should be displayed. 5. Click on Fail button.
	Expected result	Tester should be able to update testcase status to Fail.
	Actual result:	Tester is be able to update testcase status to Fail.
	Test result	Pass

TC13	**Title**	Report bug for failed test case
	Description	For a failed test case, tester should be able to report a bug.
	Steps	1. Login to Spartan QA tool with user login and select tester radio button. 2. After successful login, select test cases with Failed status. 3. Click on report bug button. 4. Enter bug details. 5. Click on submit.
	Expected result	Tester should be able to create a bug for failed testcase.
	Actual result:	Tester is able to create a bug for failed testcase.
	Test result	Pass

TC14	**Title**	Restrict duplicated registration
	Description	New user cannot use an existing email id to Signup.
	Steps	1. Signup to Spartan QA tool using existing user email id. 2. Click on submit.
	Expected result:	User should not be able to signup.
	Actual result:	User is not able to signup.
	Test result	Pass



Project post mortem

- **Issues uncovered :**The issues uncovered for the relationship between the developer and Test lead that is assigned relationship. We thought to assign the bugs to Developer by test lead. But we made an assumption that the tester knowns the original developer for that particular component.The tester assigns the bugs for the developer.

- **Implement something differently:**
    - **Async/Await :** Nodejs is single threaded. To make use of Promise based services and function calls, we need a system to make the application synchronous yet at the same time make sure that user flow is not hindered. Generally, callback functions are used to handle that. The problem with

callback functions is the code becomes messy and has a lot of nested functions. To avoid this we have used async and await functionality to make the code clean, concise and achieve the same asynchronous functionality.

- **Potential improvements :**
    - We can create functionalities wherein the manager assigns the projects to the Test Leads and Test Lead assigns the components to the Testers and Developers, instead of each user choosing their own projects and components to work on.
    - We can implement transactions at some other places as well. When a new user signs up as a tester or developer , an entry needs to be created in the ‘User’ table. At the same time we need to add this entry in the ‘Tester’ or ‘Developer’ table as well.