ASSIGNMENT GUIDANCE & FRONT SHEET

Student ID or IDs for group work	**1929183**

Module Title & Code	*WM264 Smart solutions development II*
Module Owner	*Jianhua Yang*
Module Tutor	
Module Marker	
Assesment type	*Project report*
Date Set	*28 September 2020*
Submission Date (excluding extensions)	*14th December 2020*
Marks return date (excluding extensions)	*14th January 2021*
Weighting of mark	*60%*

Assessment Detail	*The assessment is to designand implement a database system.*
Additional details	*Students are free to choose their own projects, ideally work-related. However, the project has to take into account of conceptual database design using modelling languages, database normalization, security, performance, etc. By doing the projects, they will also need to demonstrate their problem-solving skills.* *More details about the scenario and specific requirements can be found in the assignment brief.*
Module learning outcomes	*LO1: Know the role of data management systems in managing organisational data and information, and use a high-level language to create user interface for accessing data from DBMS.* *LO2: Describe the logical and conceptual data modelling, make entity relationship model for incorporating system and user requirements.* *LO3: Use SQL to perform simple queries including adding, updating and deleting queries.* *LO4: Identify the data redundancy problems and update anomalies, apply data normalization techniques to combat the data redundancy problem.* *LO5: Use analytical and critical thinking skills to technology solutions development, analyse and apply structured problem-solving techniques to complex systems and situations*
Learning outcomes assessed in this assessment	*LO1/2/4/5*

Marking guidelines

0 – 40%
Data Modelling: Little consideration of data modelling.
Database Design: Poor database design, little considerations of functional dependencies.
Problem-solving: Little evidence of problem-solving skills.
Report: The report contains lots of grammar mistakes and is difficult to follow.

41 – 60%
Data Modelling: Some good attempts at theoretical data modelling using ER diagrams. However, it contains some errors.
Database Design: Some considerations on normalization. However, design choices are not well justified.
Problem-solving: Some considerations of requirement gathering, functional/non-functional requirements. However, not able to link them with database design.
Report: The report makes use of illustrations and references, however, it contains some errors.

61 – 80%
Data Modelling: Good use of data modelling language and diagrams. Design choices are well justified.
Database Design: Some measures are in place for improving security and performance.
Problem-solving: Good project planning with well-defined deliverables and milestones.
Report: The report is clearly structured and well written with little mistakes.

81 – 100%
Data Modelling: Able to use object-oriented data modelling theories to guide modelling practice. Successfully fit data modelling with features in the datasets.
Database Design: Very Good use of normalization. Able to justify design choices using math notations. Advanced features for improved security and performance.
Problem-solving: Able to use test cases and scenarios. Able to learn and summarize for future improvements.
Report: Covers different aspects of the project, with thorough discussions and appealing visual presentation.

Submission guidance

You are limited to a maximum of 2,500 words in the report, excluding references or source code. The report should be either Word or PDF format. Also note that:

- *Your report should be in the format of academic writing, including sections such as introduction, method, results, discussions, and conclusions, etc.*
- *It should make proper use of illustrations such as tables and figures. Do not include large trunks of source code in your report body, as those need to be submitted separately online.*
- *Try to make good use of references to back up your design decision and/or support claims in your discussions.*
- *Any source code you wish to submit should be syntax highlighted and inserted as appendices. The source code should be commented on and follow conventions.*
- *There is no need to submit source code files separately.*

The submission link is on Tabula.

Academic Guidance

Follow the University of Warwick referencing guidelines.
https://warwick.ac.uk/services/library/students/referencing/referencing-styles/

Resubmission details

The University policy is that students should be given the opportunity to remedy any failure at the earliest opportunity. What that "earliest opportunity" means in terms of timing and other arrangements is different depending on Programme (i.e. Undergraduate, Full Time Masters, Part Time Postgraduate, or

	Overseas). Students are advised to consult your Programme Team or intranet for clarity.
Late submission details	If work is submitted late, penalties will be applied at the rate of **5 marks per University working day** after the due date, up to a **maximum of 10 working days** late. After this period the mark for the work will be reduced to 0 (which is the maximum penalty). "Late" means **after the submission deadline time as well as the date** – work submitted after the given time even on the same day is counted as 1 day late, as per University Regulation 36.3

Project Management SQL Database

Table of Contents

ANALYSIS OF THE PROBLEM

1.1. Problem Definition/Requirements

For the purpose of this assignment I have been tasked with designing and implementing a project management database to track and evaluate current tasks being completed in the

software team and also monitor resources allocated to each software release and project. I will be developing a database system and database management system (DBMS). Due to the nature of software work within Jaguar Land Rover, this system will follow an agile project management methodology. The system will be able to track issues raised by engineers in various departments and each ticket will be able to follow a set workflow with information regarding the issue attached to the ticket. Each ticket will be assigned to an engineer and will be raised by an engineer. They will also be linked to a certain release (software package) that will relate to a software project and sprint. A sprint is a short when a scrum team works to complete a set amount of work (Layton and Ostermiller, 2017).

1.2. Stakeholders

My database is aimed for use in the workplace or working from home. The users will be employees of Jaguar Land Rover or our suppliers. I will take the role of Database Designer and Database Administrator. My line manager and project owners will have the highest level of permissions after myself. Other users will be basic end users using a front end application.

1.3. Research The Problem

Currently, software teams in Jaguar Land Rover use a variety of systems to track software issues in the forms of tickets. The oldest system currently being used is ETracker, an internally developed project management and issue management software. The second system currently being used is Rational Team Concert an IBM software package part of their project and software/requirement management systems (IBM, 2020).

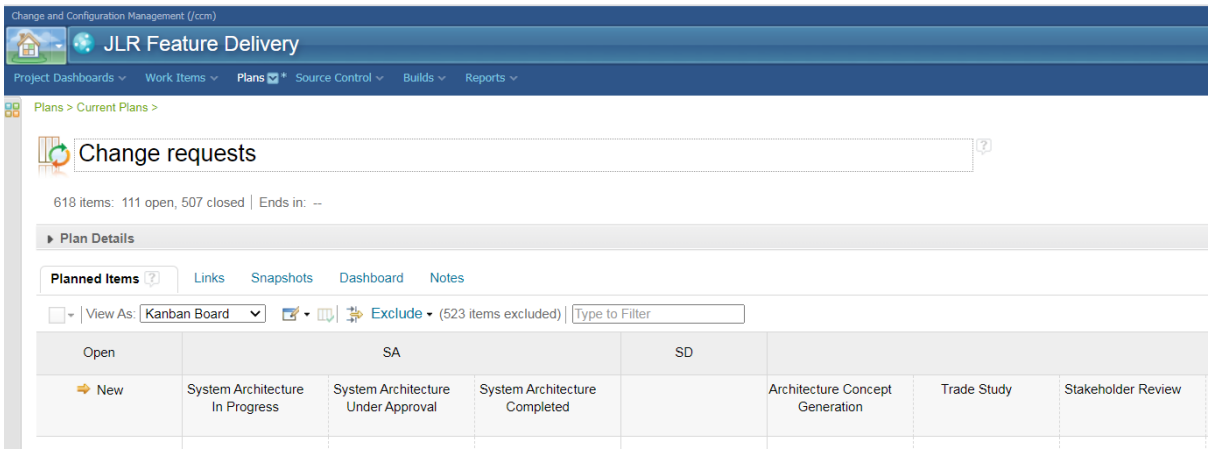


Figure 1: RTC Screenshot

Another system we currently use is JIRA developed by Atlassian that is tailored specifically to software and agile methodologies (Atlassian, 2020). I will be modelling my database with references and similarities to the JIRA platform as this system meets the requirements of my team and creating our own system gives us more freedom.

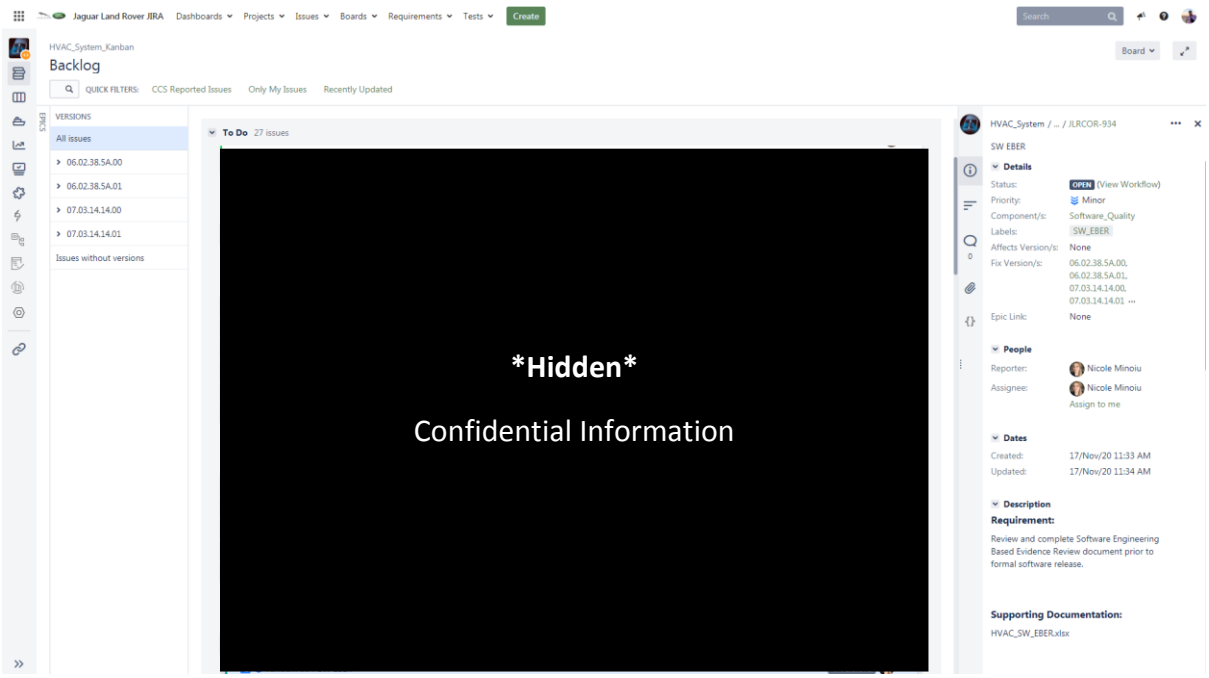


Figure 2: JIRA by Atlassian

It is clear there are many alternatives to developing a standalone, internal system. However, by doing this the company will reap the benefits of complete customization.

1.4. Specify The Proposed Solution

The database will be designed in a relational manner. A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database (Elmasri and Navathe, 2016). The following requirement will be marked against the final product by stakeholders; therefore each requirement is a defined deliverable. The main milestones are an implemented database and an API (Application Programmable Interface).

Non-Functional Requirements

The Database Management System (DBMS) shall be hosted on MySQL 8.0 server with legacy authentication. The application programming interface (API) will be based in Python 3.8 using QT Designer and PyQt 5.0. The API will be packaged into a distributable executable file and the database will be hosted either on a desktop PC or server. A keyboard and mouse will be needed at all times. Users will need hardware capable of connecting to the company intranet and running an executable application to access the database.

Functional Requirements

1. Database Requirements
    a. The database shall track all changes made to each ticket via an audit like system using an audit table and MySQL ‘AFTER UPDATE’ triggers.
    b. The database shall track all tickets from the BSM (Body Software Module), DZM (Door Zone Module), HVAC (Heating Ventilation Air Conditioning) and SZM (Seat Zone Module) systems.

c. The database shall have a standard user role for all engineers accessing the database which shall include insert and update SQL commands. They will not have the ability to use the DELETE SQL command.
d. The database admin accounts will have full privileges.
e. The tables will be as follows:
    i. Employee table – containing all account information and contact information
    ii. Tickets table – containing all issues raised in the project management software
    iii. Releases table – containing all releases, their id's and tickets linked to them.
    iv. Projects table – containing all software projects and their leads
f. Each table shall be linked to a minimum of one other table using a foreign key
g. Each table must have a primary key
h. Each table must have a minimum of one index
i. Users shall be able to view all tickets in the database
j. Only admin users shall be able to view the employee table

2. Front End Aesthetic Requirements
    a. All tabs shall use the same colour scheme
    b. The JLR logo shall be displayed at the top of the window and copyright info shall be displayed in the bottom left corner.
    c. All text shall be black unless background colour deems it necessary to use white
    d. Font will be the standard QT font
    e. The window will not be resizable
    f. There shall be a login tab
        i. The login tab will allow users to connect and disconnect from the database and input username and password to pass to the database authentication system.
        ii. Any saved passwords must be hashed
    g. There shall be a search tab
        i. The search tab shall contain all ticket, release or projects fields depending on a drop down selection and then allow users to search.
        ii. The search tab shall be read only.
    h. There shall be a ticket tab
        i. The tickets tab shall always contain all ticket fields and will allow editing of tickets
3. Inputs
    a. The user will control everything using a keyboard and mouse and will therefore be needed
    b. Screens are navigated using a mouse and on screen buttons
4. Processing
    a. All validation of SQL queries shall be done in python before being passed to the MySQL server.
    b. All processing of SQL queries shall be done internally via the MySQL server.

DESIGN OF THE SOLUTION

1.5. Decompose The Problem

Database Design

The database will be developed using SQL and MySQL as both are commonly used technologies and languages with plenty of support and external libraries. During the design, concepts like system architecture diagrams (SADs) and enhanced entity relationship (EER) diagrams shall be used. The system can be broken down simply into its components, The developer will use a SAD to decide how to decompose the database and break it into small manageable sections (Elmasri and Navathe, 2016).

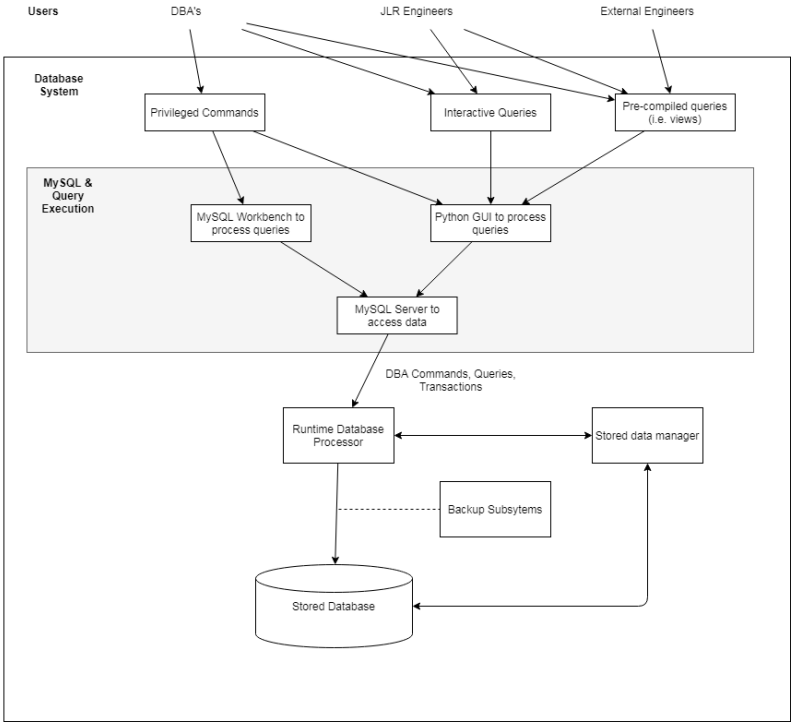


Figure 3: System Architecture Diagram

Figure 3 is a system architecture diagram detailing how the project management database will be designed and split into distinct components.

Each table will have one primary key except for the Audit table which has a composite key of TableID and TableName and TimeOccured. A primary key is a single column value used to identify a database record uniquely. A composite key is a primary key composed of multiple columns used to identify a record uniquely. Each table, except for the audit table, will have either a foreign key linked to it or its own primary key used as a foreign key in another table. This allows the database administrators to design the database is such a way that there is no data redundancy and allows for normalisation into 3rd normal form. This, in turn, will remove any many to many relationships leaving only one to many relations (Harrington, 2016b).

Below in figure 4 is a basic entity relationship diagram that is in 1st Normal Form. This shows many attributes could be repeated. Normalisation is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using

relationships. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically (Guru99, 2020).

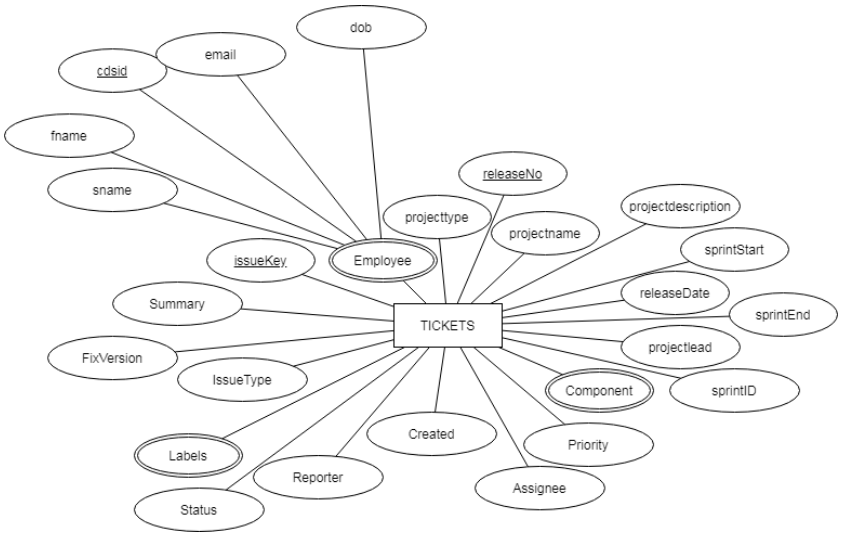


Figure 4: ER Diagram (Not Normalised)

Figure 5 has been normalised to 3rd normal form. Third normal form is where there is no transitive dependency for non-prime attributes and is also in second normal form (Harrington, 2016a). The tickets entity has been split into five distinct tables. All linked via a relationship and will be implemented as a foreign key. By normalising this data the database has removed all possibilities of data redundancy allowing for consistency in the database. 3rd normal form was chosen as there can be a lot of repeated data in these tables. For example, in the sprint table, there can be all the same dates but could be part of different sprints. The same applies to the releases table where dates can be the same, projects can be the same, sprints can be the same and so can the status. This also applies to employee table, as many names can be the same along with dates of birth.

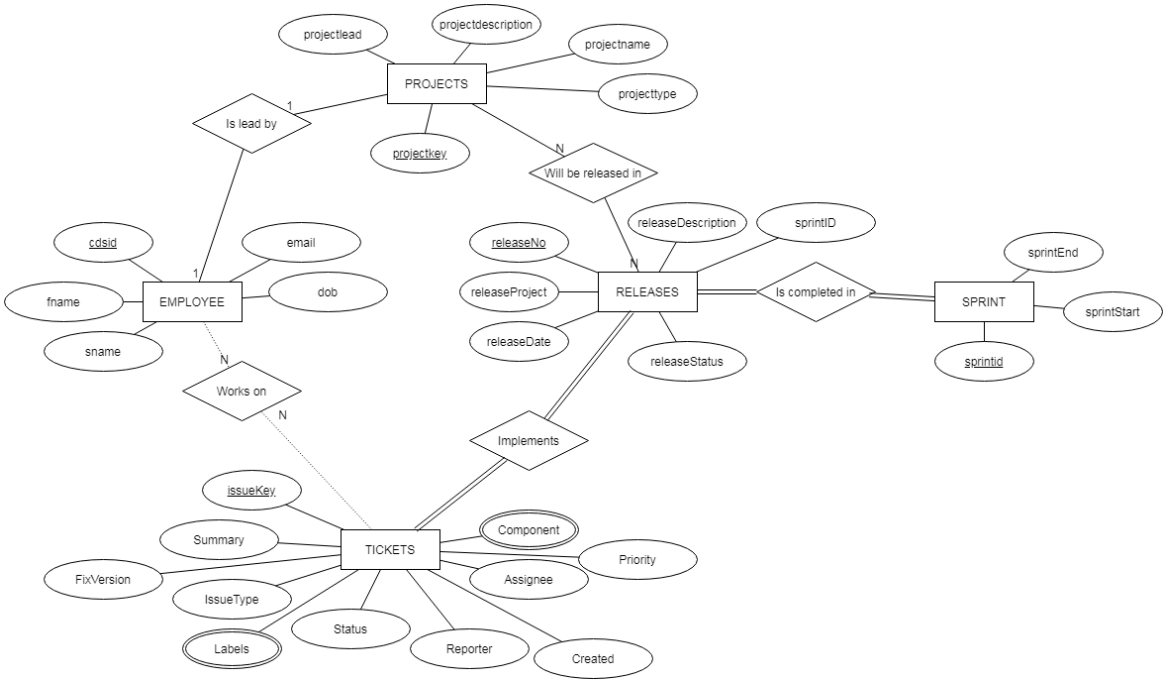


Figure 5: EER Diagram in 3rd Normal Form

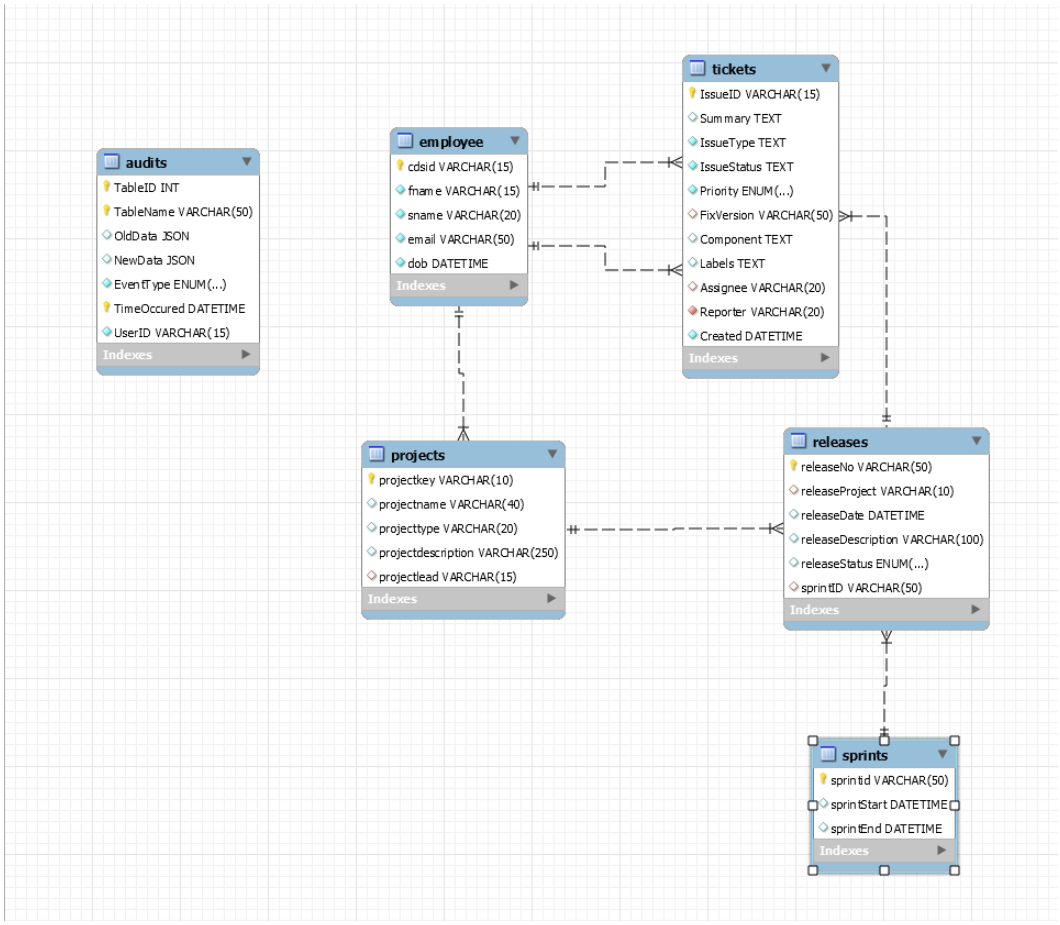


Figure 6: EER/UML Diagram

Figure 6 shows an EER diagram with field constraints, primary keys, indexes, foreign keys, and relationships. Each relationship is either a 1-1, 1-Many or Many-1. There a no Many-Many relationships. This allows for data integrity and adherence to ACID rules (Elmasri and Navathe, 2016).

Atomicity – this database is atomic. If one part of a transaction doesn't work like it is supposed it, the other will fail as a result. For example if a ticket is created and linked to a release, but the release is not in the release table then it will not create the ticket.

Consistency – By having foreign key constraints, character limits, data types and primary keys; the database can enforce referential integrity which in turn ensures consistency in data by meeting data validation rules. If a transaction occurs and the data inserted, deleted, or updated does not comply with rules then the transaction will be rolled back.

Isolation – Each database transaction will be executed in an isolated environment which allows for transactions to be executed simultaneously. Isolation ensures that tickets are only assigned to one engineer.

Durability – All failures are invisible to end users; all errors are caught in exceptions. Once a transaction is committed in cannot be rolled back. This ensures no data is lost during an outage.

Relational Algebra

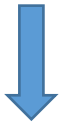
A query used in this database in SQL is as follows:

Equation 1: SQL command

SELECT IssueKey, IssueStatus, Summary, Assignee, fname, sname, email

FROM tickets, employee

WHERE Assignee = dtayl169 and assignee = cdsid

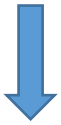


$DOMINIC\_TICKETS \leftarrow \sigma_{assignee='dtayl169'}(TICKETS)$

$DOMINIC\_EMPLOYEE \leftarrow (DOMINIC\_TICKETS \mid><\mid_{assignee = cdisd} EMPLOYEE)$

$RESULT \leftarrow \pi\ IssueKey, IssueStatus, Summar, Assignee, fname, sname, email\ (DOMINIC\_EMPLOYEE)$

Equation 2: Relational Algebra 1st Conversion



$\pi\ IssueKey, IssueStatus, Summar, Assignee, fname, sname, email$

$((\sigma_{assignee='dtayl169'}(TICKETS)) \mid><\mid_{assignee = cdisd} (EMPLOYEE))$

Security

The database will be used internally and externally, but only for approved users with Jaguar Land Rover intranet access. The database will use MySQL 8.0 legacy authenticated role based access. Each user will be assigned a role from the following list.

- DBA
  - The database administrator will have access to all tables and can create, delete, and update all fields including user accounts.
  - The DBA will be responsible for ensuring all data is handled correctly and by creating this role it stops users changing records they should not have access to.
  - They will have access to all database commands
- DB Manager/Project Lead
  - They will have access to all tables except the audit table.
  - They will be able to create new users and reset passwords
  - They will be able to insert, delete and update all records in all projects.

- Engineer/Standard Role
    - They will be able to insert and update records in projects specified by Project Leaders.

```
CREATE USER 'HR'@'%' IDENTIFIED BY 'HR101';
CREATE USER 'CEO'@'%' IDENTIFIED BY 'CEO';
CREATE USER 'dtayl169'@'%' IDENTIFIED BY 'Dom';
CREATE USER 'LaurenO'@'%' IDENTIFIED BY 'Lauren';

GRANT select, insert, DELETE ON jiradb.* TO 'dtayl169'@'localhost';
GRANT ALL PRIVILEGES ON *.* TO 'CEO'@'localhost';
GRANT ALL PRIVILEGES ON *.* TO 'HR'@'localhost';
GRANT ALL PRIVILEGES ON *.* TO 'LaurenO'@'localhost';
```

Figure 7: Roles and privileges

Figure 6 shows a sample of employees and accounts and their roles.

No passwords will be saved locally or through the front end application. Passwords will be passed to the database login system; once verified they will be deleted. No saving of passwords or logins will be allowed.

The database shall have triggers assigned to the employee table that will monitor any changes in the records and record these changes in the audit table (See appendix – SQL for all source code).

A procedure shall be coded to raise an exception and stop any unauthorised access. (See figure 7)

```
/*Creating a procedure which is called when each transaction is made and checks if user has access to the corresponding record
if not then an exception is raised, ending thetransaction*/
CREATE PROCEDURE errorCheckEmpPermissions ()
BEGIN
IF current_user() <> 'CEO@localhost' or current_user() <> 'HR@localhost' or current_user() <> 'LaurenO@localhost' then
signal sqlstate '45000' set message_text = 'My Error Message';
end if;
END
```

Figure 8: Procedure to prevent access

Front End Application

The front end application shall be developed using Python 3.8, PyQt5 and MySQL Connector 5.0. It shall be developed in an object oriented fashion using classes and objects. Each tab of the graphical user interface (GUI) shall have its own class (Lott and Fatouhi, 2014). Each class will have appropriately named functions it (Kapil, 2019). For the purpose of this assignment the report will not delve any deeper into the design of front end.

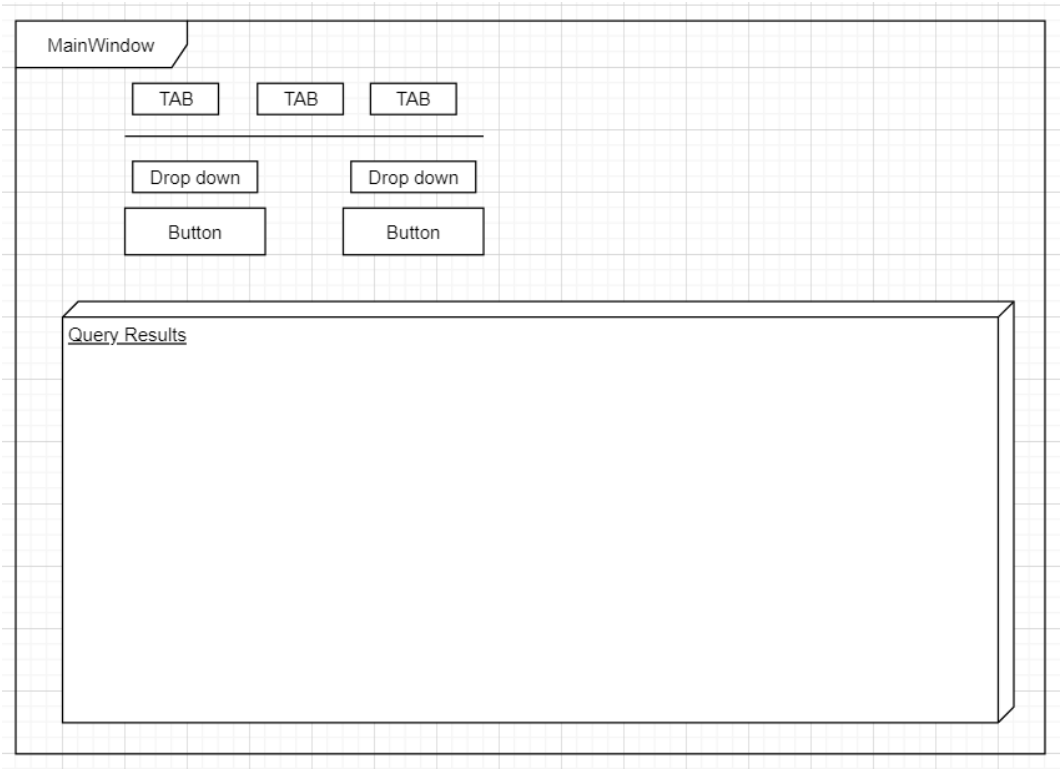


Figure 9: First Tab of GUI (Query)

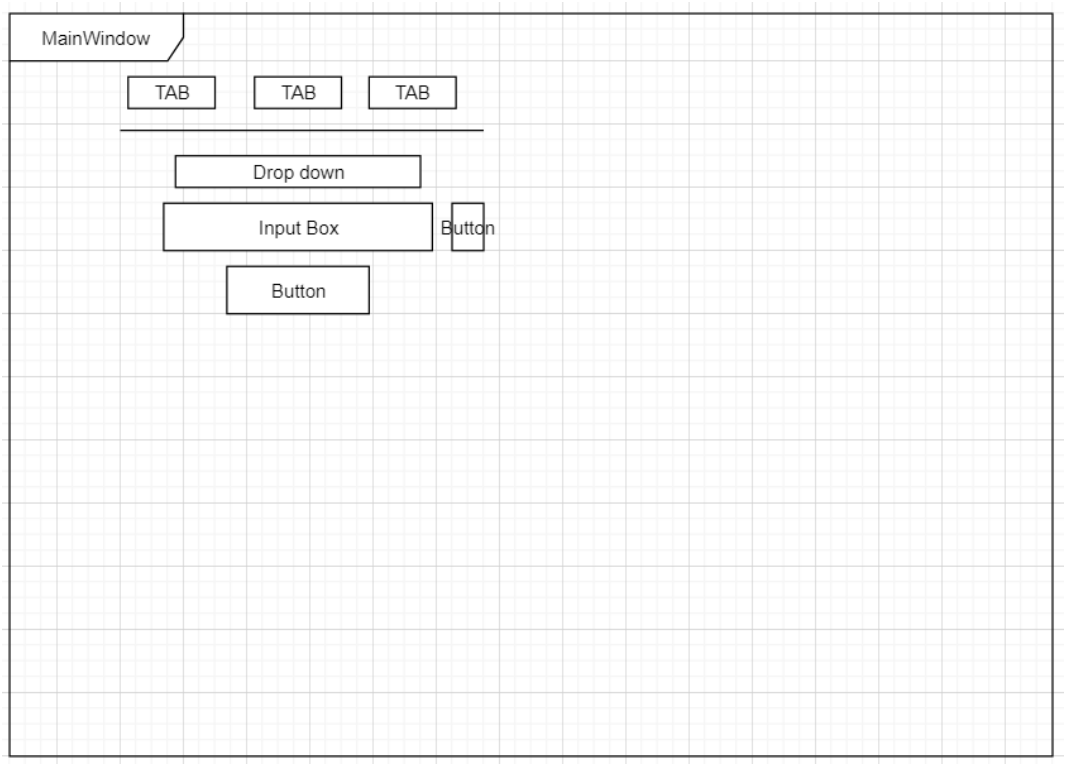


Figure 10: Second Tab of GUI (Export)

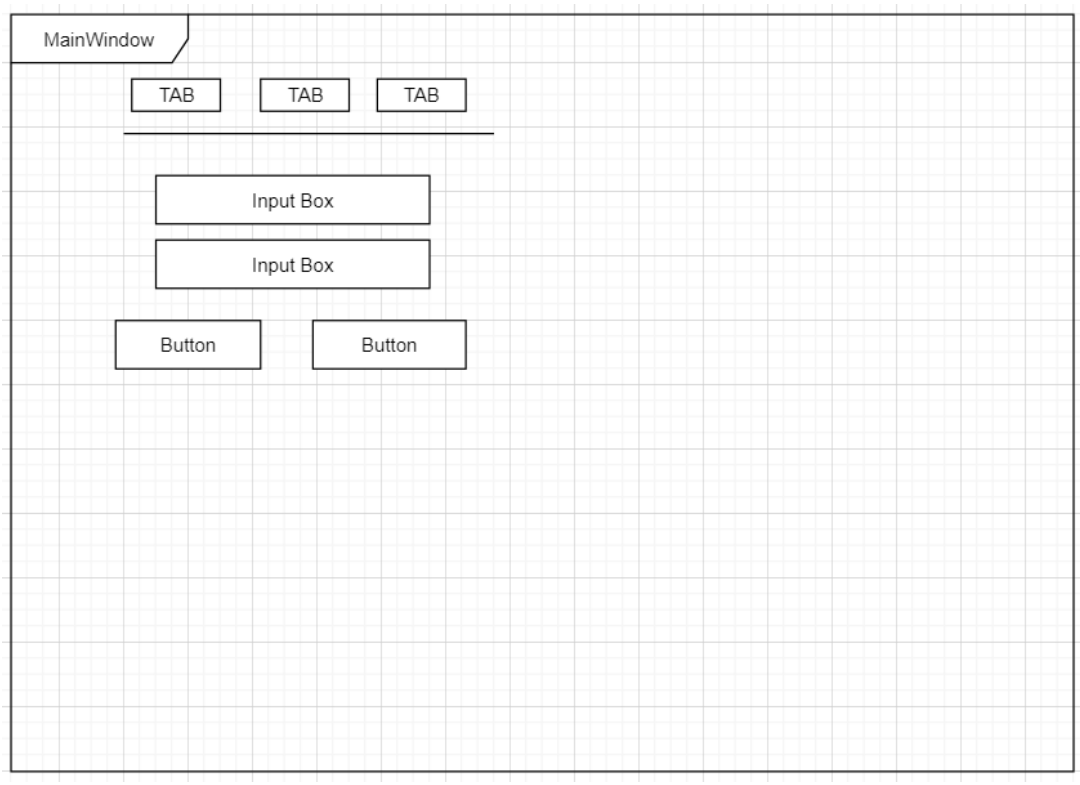


Figure 11: Final Tab of GUI (Login)

1.6. Approach To Testing

Firstly, the system shall be unit tested. It shall test each table/function independently of each other. Secondly it shall be integration testing, where all parts are put together and tested in the form of the final product.

Black box testing can be incorporated into the first deployment of the product into the workplace, where I gather information and error reports from users to further enhance and debug the product.

Using black box testing will be free from any bias that developers may have that could mean certain bugs might not be found. However, there will be other times where it is easier for developers to test the system in white box testing, as they know more what are the weaker parts of the system that can be tested more rigorously and stress test the variables to expose weaknesses.

I can get feedback from the black box testing by using a questionnaire to gather information from the users to see what they found useful or elements that were counterproductive.

EVALUATION

1.7. Database Maintenance & Development

As this is the first iteration of this application and database the stakeholders are satisfied all requirements are met. In the future, developers would like to add more versatility to the database and allow for comments to be attached to tickets along with supporting documents. The developers would enhance the user experience (UX) with regards to the front end application and allow for tickets to be edited and created in an easier way.

All programming has been done in a modular and functional fashion using object oriented methodologies. This allows for easy further development on the database and application. Furthermore, this means the product is very maintainable for the future and debugging is simple when compared to the size of the project. All changes can be made in classes and functions (procedures and tables for database) independently of each other. All code is commented which allows other developers or engineers to transfer seamlessly to the project with no issues.

1.8. Front End Final Product

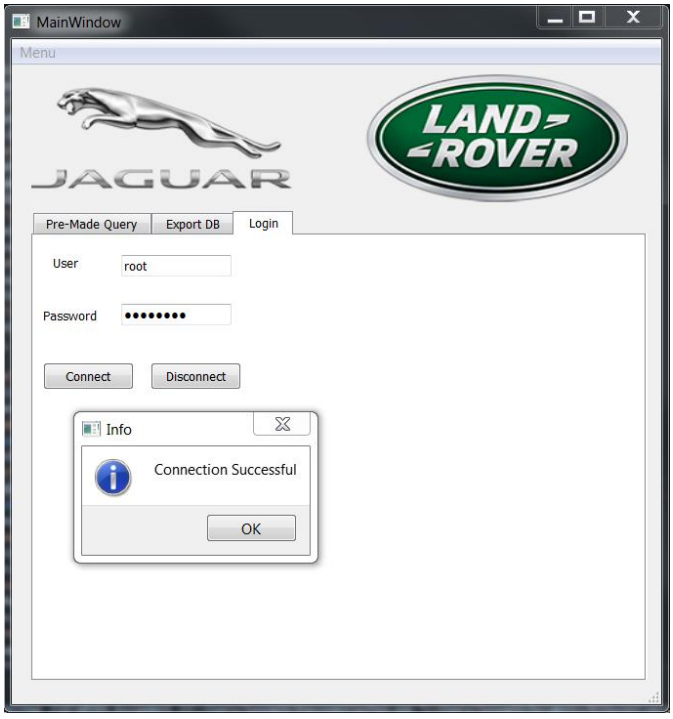


Figure 12: Connection to DB

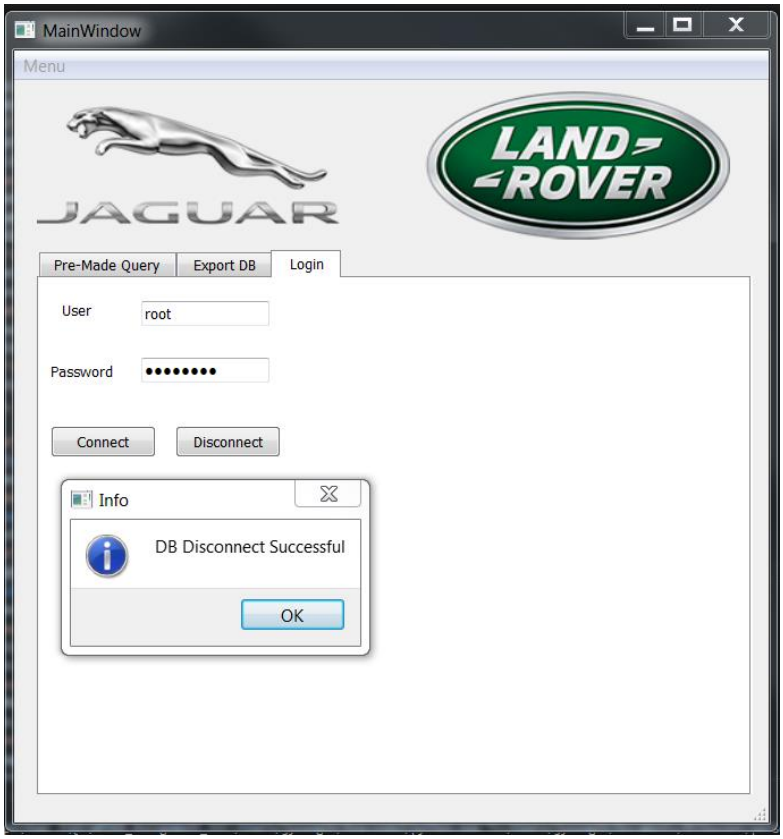


Figure 13: Disconnection from DB

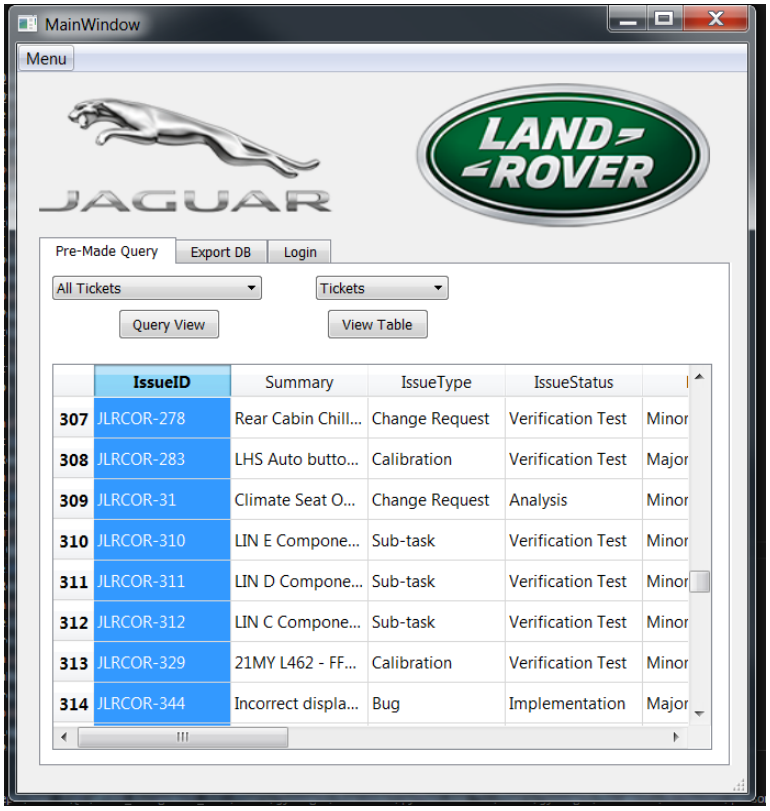


Figure 14: GUI showing results of query

Figure 15: GUI Error handling

Figure 16: GUI export

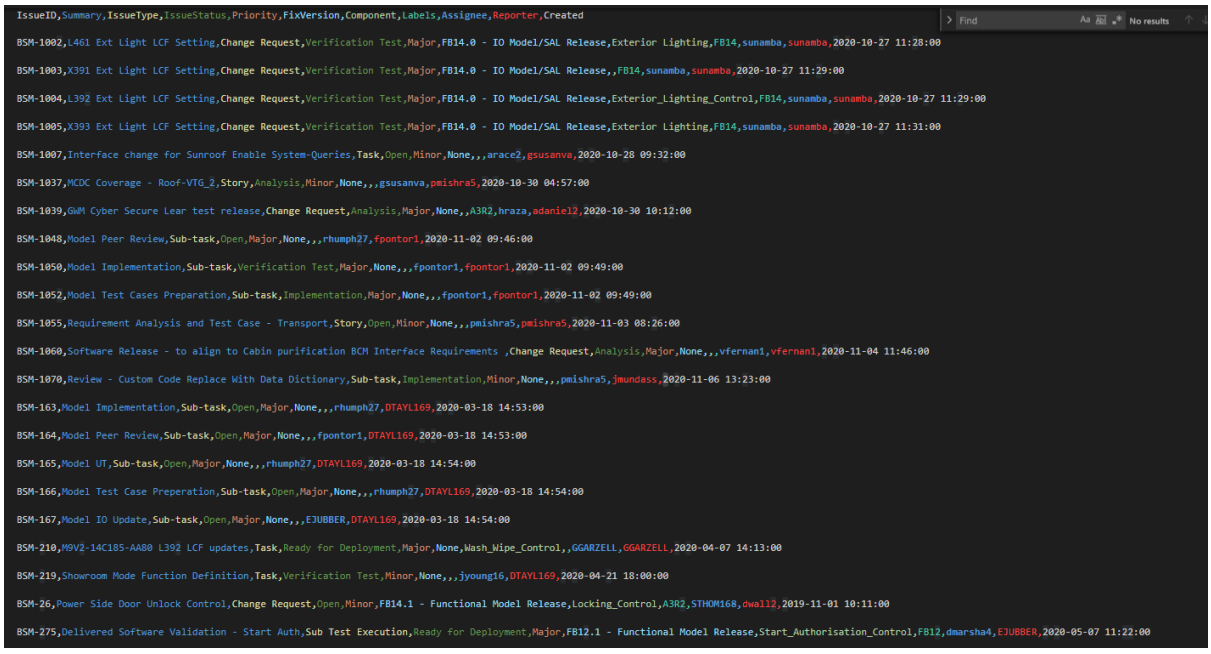


Figure 17: Proof of export into CSV

Appendix

1.9. SQL

```
/*Creating the database*/
CREATE database jiradb
use jiradb

/*Creating employee table*/
create table employee (
    cdsid VARCHAR(15) PRIMARY KEY NOT NULL UNIQUE,
    fname VARCHAR(15) NOT NULL,
    sname VARCHAR(20) NOT NULL,
    email VARCHAR(50) NOT NULL,
    dob DATETIME NOT NULL
)

/*Creating projects table with FOREIGN KEY that REFERENCES the employee table*/
create table projects (
    projectkey VARCHAR(10) PRIMARY KEY UNIQUE,
    projectname VARCHAR(20),
    projecttype VARCHAR(20),
    projectdescription VARCHAR(250),
    projectlead VARCHAR(15),
    FOREIGN KEY(projectlead) REFERENCES employee(cdsid)
)

/*Creating sprint table*/
create table sprints (
    sprintid VARCHAR(10) PRIMARY KEY UNIQUE,
    sprintStart DATETIME,
    sprintEnd DATETIME,
)

/*Creating release table with FOREIGN KEYS REFERENCING the sprints table and projects table*/
CREATE TABLE releases (
    releaseNo VARCHAR(20) PRIMARY KEY NOT NULL,
    releaseProject VARCHAR(10),
    startDate DATETIME,
    releaseDate DATETIME,
    releaseDescription VARCHAR(100),
    releaseStatus ENUM('Released','Unreleased'),
    sprintID VARCHAR(50),
    FOREIGN KEY (sprintID) REFERENCES sprints(sprintid),
    FOREIGN KEY (releaseProject) REFERENCES projects(projectkey)
)

/*Creating the audits TABLE with a composite key*/
CREATE TABLE audits(
    TableID int NOT NULL,
```

```
    TableName VARCHAR(50) NOT NULL,
    OldData JSON,
    NewData JSON,
    EventType ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL,
    TimeOccured DATETIME NOT NULL,
    UserID VARCHAR(15) NOT NULL,
    PRIMARY KEY (TableID, TableName, TimeOccured)
)

/*Creating the ticket table with three FOREIGN KEYS*/
CREATE TABLE tickets(
    IssueID VARCHAR(15),
    Summary VARCHAR(200),
    IssueType VARCHAR(25),
    IssueStatus VARCHAR(25),
    IssuePriority ENUM('Major', 'Minor', 'Enhancement', 'Blocker', 'Trivial', 'Critical'),
    FixVersion VARCHAR(50),
    Component VARCHAR(50),
    Labels VARCHAR(50),
    Assignee VARCHAR(15),
    Reporter VARCHAR(15),
    Created DATETIME,
    FOREIGN KEY (FixVersion) REFERENCES releases(releaseNo),
    FOREIGN KEY (Assignee) REFERENCES employee(cdsid),
    FOREIGN KEY (Reporter) REFERENCES employee(cdsid)
)

/*Inserting employee records*/
INSERT INTO employee VALUES
    ('dtayl169', 'Dominic', 'Taylor', 'dtayl169@jaguarlandrover.com', '1968-03-21'),
    ('dstockd1', 'Daniel', 'Stockdale', 'dstockd1@jaguarlandrover.com', '1972-05-30'),
    ('jyoung16', 'Jack', 'Young', 'jyoung16@jaguarlandrover.com', '2000-11-18'),
    ('loverto1', 'Lauren', 'Overton', 'loverto1@jaguarlandrover.com', '1990-09-06'),
    ('aabu', 'Abin', 'Abu', 'aabu@jaguarlandrover.com', '1990-09-06'),
    ('aarmst35', 'Alan', 'Armstrong', 'aarmst35@jaguarlandrover.com', '1970-02-26'),
    ('achauha6', 'Ashvin', 'Chauhan', 'achauha6@jaguarlandrover.com', '1972-07-21'),
    ('adaniel2', 'Alex', 'Daniels', 'adaniel2@jaguarlandrover.com', '1989-09-17'),
    ('afowler', 'Andy', 'Fowler', 'afowler@jaguarlandrover.com', '1975-01-06'),
    ('aganesan', 'Ajay', 'Ganesan', 'aganesan@jaguarlandrover.com', '1985-11-06'),
    ('ahayer3', 'Amandeep', 'Hayer', 'ahayer3@jaguarlandrover.com', '1977-10-29'),
    ('ahudaalf', 'Ahmad', 'Al Firdaus', 'ahudaalf@jaguarlandrover.com', '1982-04-22'),
    ('akipling', 'Alistair', 'Kipling', 'akipling@jaguarlandrover.com', '1972-12-12'),
    ('akrishn6', 'Athul', 'Krishan', 'akrishn6@jaguarlandrover.com', '1991-09-27'),
    ('arace2', 'Andrew', 'Race', 'arace2@jaguarlandrover.com', '1988-07-02'),
    ('arajend1', 'ArunKumar', 'Rajendren', 'arajend1@jaguarlandrover.com', '1986-07-20'),
    ('araducan', 'Aran', 'Ducana', 'araducan@jaguarlandrover.com', '1972-10-05'),
    ('aretnam', 'Arun', 'Retnam', 'aretnam@jaguarlandrover.com', '1977-08-12'),
    ('arober55', 'Adrian', 'Roberts', 'arober55@jaguarlandrover.com', '1977-08-12'),
```

('atitus', 'Anand', 'Titus', 'atitus@jaguarlandrover.com', '1977-08-12'),
('avaxevan', 'Athanasios', 'Vaxevanos', 'avaxevan@jaguarlandrover.com', '1977-08-12'),
('BEDMON12', 'Brian', 'Edmonds', 'BEDMON12@jaguarlandrover.com', '1972-12-12'),
('bmcauli1', 'Brian', 'McAuliffe', 'bmcauli1@jaguarlandrover.com', '1977-08-12'),
('canders5', 'Craig', 'Anderson', 'canders5@jaguarlandrover.com', '1977-08-12'),
('CCOTTRIL', 'Chris', 'Cottril', 'CCOTTRIL@jaguarlandrover.com', '1977-08-12'),
('chorne2', 'Charlotte', 'Horne', 'chorne2@jaguarlandrover.com', '1977-08-12'),
('dasokan3', 'Dhillip', 'Asokan', 'dasokan3@jaguarlandrover.com', '1977-08-12'),
('dclark23', 'Daniel', 'Clark', 'dclark23@jaguarlandrover.com', '1977-08-12'),
('dcolema6', 'Darren', 'Coleman', 'dcolema6@jaguarlandrover.com', '1977-08-12'),
('dmarsha4', 'Daniel', 'Marhsall', 'dmarsha4@jaguarlandrover.com', '1977-08-12'),
('dwall2', 'David', 'Wall', 'dwall2@jaguarlandrover.com', '1977-08-12'),
('ejubber', 'Edmond', 'Jubber', 'ejubber@jaguarlandrover.com', '1977-08-12'),
('ekeepad2', 'Eldho', 'Keepadas', 'ekeepad2@jaguarlandrover.com', '1977-08-12'),
('fpontor1', 'Francesco', 'Pontorieiro', 'fpontor1@jaguarlandrover.com', '1977-08-12'),
('GGARZELL', 'Giovanni', 'Garzelli', 'GGARZELL@jaguarlandrover.com', '1977-08-12'),
('gmill193', 'Gordon', 'Miller', 'gmill193@jaguarlandrover.com', '1977-08-12'),
('gshaw1', 'Gareth', 'Shaw', 'gshaw1@jaguarlandrover.com', '1977-08-12'),
('gsusanva', 'Gisina', 'Susan Varghese', 'gsusanva@jaguarlandrover.com', '1977-08-12'),
('hgyurgya', 'Harrison', 'Gyurgyak', 'hgyurgya@jaguarlandrover.com', '1977-08-12'),
('hramesh', 'Hemanth', 'Ramesh', 'hramesh@jaguarlandrover.com', '1977-08-12'),
('jjoy1', 'James', 'Joy', 'jjoy1@jaguarlandrover.com', '1977-08-12'),
('jmundass', 'James', 'Mundass', 'jmundass@jaguarlandrover.com', '1977-08-12'),
('kmutton', 'Kate', 'Mutton', 'kmutton@jaguarlandrover.com', '1977-08-12'),
('KRAJANIK', 'RAJANIKANTH', 'KANDI', 'KRAJANIK@jaguarlandrover.com', '1977-08-12'),
('mahmed6', 'Mohammed', 'Ahmed', 'mahmed6@jaguarlandrover.com', '1977-08-12'),
('mcalahor', 'Manuel', 'Calahorra', 'mcalahor@jaguarlandrover.com', '1977-08-12'),
('mdomini2', 'Milu', 'Dominic', 'mdomini2@jaguarlandrover.com', '1977-08-12'),
('mhirons', 'Michael', 'Hirons', 'mhirons@jaguarlandrover.com', '1977-08-12'),
('MPARAMBA', 'Mujeeb', 'Parambath', 'MPARAMBA@jaguarlandrover.com', '1977-08-12'),
('mwhite16', 'Mollie', 'White', 'mwhite16@jaguarlandrover.com', '1977-08-12'),
('NKLEMOU', 'Neraida', 'Klemou', 'NKLEMOU@jaguarlandrover.com', '1977-08-12'),
('nminoiu1', 'Nicole', 'Minoiu', 'jfaulkn3@jaguarlandrover.com', '1977-08-12'),
('NROSE1', 'Nigel', 'Rose', 'NROSE1@jaguarlandrover.com', '1977-08-12'),
('nsilcox', 'Nick', 'Silcox', 'nsilcox@jaguarlandrover.com', '1977-08-12'),
('PDEVLIN4', 'Paddy', 'Devlin', 'PDEVLIN4@jaguarlandrover.com', '1977-08-12'),
('pmishra5', 'Priyanka', 'Mishra', 'pmishra5@jaguarlandrover.com', '1977-08-12'),
('randers6', 'Richard', 'Anderson', 'randers6@jaguarlandrover.com', '1977-08-12'),
('rbirch', 'Roger', 'Birch', 'rbirch@jaguarlandrover.com', '1977-08-12'),
('rjone157', 'Roy', 'Jones', 'rjone157@jaguarlandrover.com', '1977-08-12'),
('rkirkma1', 'Richard', 'Kirkman', 'rkirkma1@jaguarlandrover.com', '1977-08-12'),
('roakes4', 'Ross', 'Oakes', 'roakes4@jaguarlandrover.com', '1977-08-12'),
('rsmith15', 'Richard', 'Smith', 'rsmith15@jaguarlandrover.com', '1977-08-12'),
('RSPACKMA', 'Ryan', 'Spackman', 'RSPACKMA@jaguarlandrover.com', '1977-08-12'),
('sgoodchi', 'Sam', 'Goodchild', 'sgoodchi@jaguarlandrover.com', '1977-08-12'),
('shall', 'Simon', 'Hall', 'shall@jaguarlandrover.com', '1977-08-12'),
('ssoar2', 'Sam', 'Soar', 'ssoar2@jaguarlandrover.com', '1977-08-12'),
('STHOM168', 'Steve', 'Thomas', 'STHOM168@jaguarlandrover.com', '1977-08-12'),

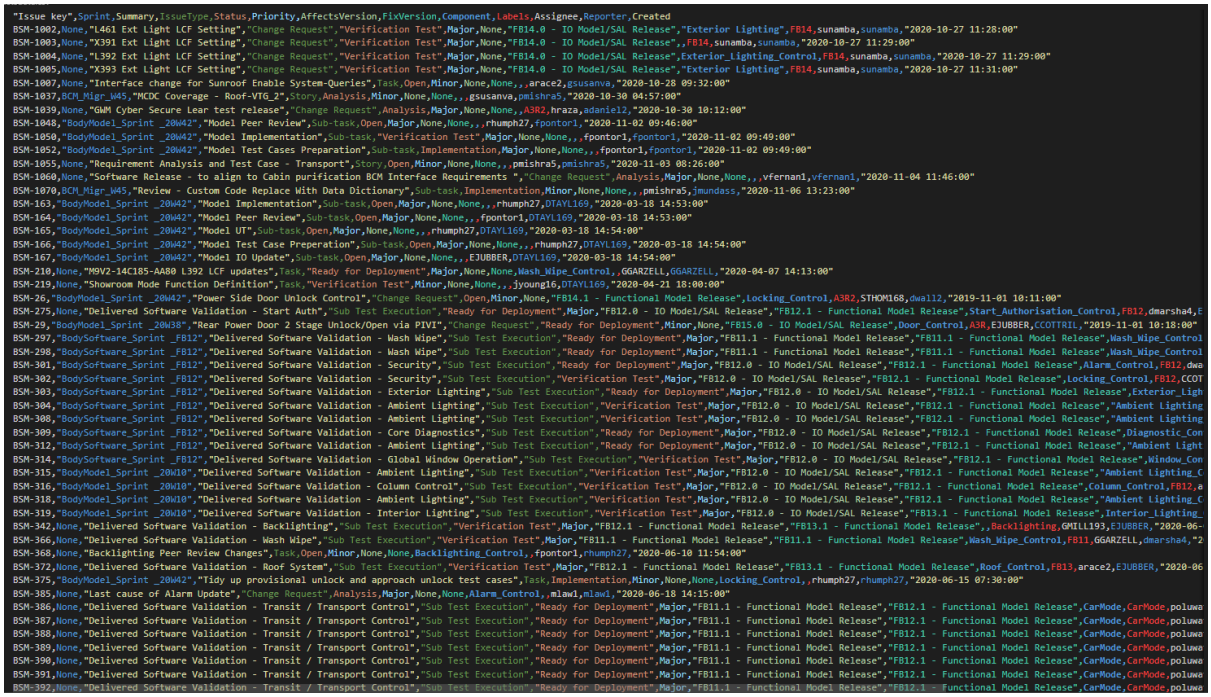
```
    ('sunamba', 'Simon', 'Unamba', 'sunamba@jaguarlandrover.com', '1977-08-12'),
    ('svaradar', 'Suresh', 'Varadarajan', 'svaradar@jaguarlandrover.com', '1977-08-12'),
    ('tgreasl1', 'Tim', 'Greasley', 'tgreasl1@jaguarlandrover.com', '1977-08-12'),
    ('vfernan1', 'Valentina', 'Fernandes', 'vfernan1@jaguarlandrover.com', '1977-08-12'),
    ('SWATKI14', 'Steve', 'Watkins', 'SWATKI14@jaguarlandrover.com', '1977-08-12'),
    ('rduffy1', 'Ryan', 'Duffy', 'rduffy1@jaguarlandrover.com', '1977-08-12'),
    ('poluwatu', 'Peace', 'Oluwatuji', 'poluwatu@jaguarlandrover.com', '1977-08-12'),
    ('l.seikowsky', 'Linus', 'Seikowsky', 'l.seikowsky@kostal.com', '1977-08-12'),
    ('h.zobel', 'Henning', 'Zobel', 'h.zobel@kostal.com', '1977-08-12'),
    ('dwaddel1', 'Dave', 'Waddell', 'dwaddel1@jaguarlandrover.com', '1977-08-12'),
    ('draju', 'Deepa', 'Raju', 'draju@jaguarlandrover.com', '1977-08-12'),
    ('asengut1', 'Arul', 'Senguttavan', 'asengut1@jaguarlandrover.com', '1977-08-12')

/*inserting project records and in each records there is a cdsid that links back to the employee table*/
INSERT INTO projects VALUES
    ('JLRCOR', 'HVAC_System', 'software', 'JIRA project to track all climate system, sub-
system requirements/defects and calibration', 'dtayl169'),
    ('SEAT', 'Seat_System', 'software', 'JIRA project to track all seat system, sub-
system requirements/defects and calibration', 'dtayl169'),
    ('BSM', 'Body Controller Software', 'software', 'JIRA project to track all body controller system software requi
rements, defects and calibration.', 'dtayl169'),
    ('DOOR', 'Door_System', 'software', 'JIRA project to track all door system, sub-
system and supplier requirements, defects and calibration.', 'dstockd1');

/*inserting release records for details of each sw release*/
INSERT INTO releases (
    releaseNo,
    releaseProject,
    startDate,
    releaseDate,
    releaseDescription,
    releaseStatus,
    sprintID
  )
VALUES
    ('LPLA-14C184-AE', 'BSM','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'BodyModel_Sprint _20W42'),
    ('L8B2-14C184-AK', 'BSM','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'BodyModel_Sprint _20W42'),
    ('JK62-18D619-AD', 'JLRCOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'BodyModel_Sprint _20W42'),
    ('GX73-14C184-AAC', 'BSM','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'ClimSys_Sprint_20W47'),
    ('FB15.1 - Functional Model Release', 'BSM','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'BodyModel_Sprint _20W38'),
    ('FB15.0 - IO Model/SAL Release', 'BSM','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'BodyModel_Sprint _20W34'),
```

```
    ('FB14.1 - Functional Model Release', 'BSM','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'BodyModel_Sprint _20W30'),
    ('FB14.0 - IO Model/SAL Release', 'BSM','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'BCM_Migr_W45'),
    ('FB13.1 - Functional Model Release', 'BSM','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'BCM_Migr_W45'),
    ('FB12.1 - Functional Model Release', 'BSM','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'BodyModel_Sprint _20W26'),
    ('FB11.1 - Functional Model Release', 'BSM','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'BodyModel_Sprint _20W10'),
    ('DZM2_21_D1', 'DOOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'DZM2_20_ D4'),
    ('DZM2_20_D6_Eng2', 'DOOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'DZM2_20_ D4'),
    ('DZM2_20_D6_Eng1', 'DOOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'DZM2_20_ D6_Eng2'),
    ('DZM2_20_D5_Eng2', 'DOOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'DZM2_20_ D6_Eng1'),
    ('DZM2_20_D5_Eng1', 'DOOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'DZM2_20_ D6_Eng1'),
    ('DZM2_20_D5', 'DOOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'DZM2_20_ D5_Eng2'),
    ('DZM2_20_D4', 'DOOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'DZM2_20_ D5_Eng2'),
    ('DZM2_20_D3Update', 'DOOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'DZM2_20_ D5_Eng2'),
    ('DZM2_20_D3', 'DOOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'DZM2_20_ D4'),
    ('DZM2_20_D2', 'DOOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'DZM2_20_ D4'),
    ('07.03.13.13.00', 'JLRCOR','2020-10-11', '2020-12-
01', 'MLA Engineering Release', 'Unreleased', 'ClimSys_Sprint_20W47'),
    ('07.03.12.12.00', 'JLRCOR','2020-10-11', '2020-12-
01', 'MLA Engineering Release', 'Released', 'ClimSys_Sprint_20W47'),
    ('07.03.11.11.00', 'JLRCOR','2020-10-11', '2020-12-
01', 'MLA Production Release', 'Released', 'ClimSys_Sprint_20W47'),
    ('07.03.0F.10.00', 'JLRCOR','2020-10-11', '2020-12-
01', 'MLA Production Release', 'Released', 'ClimSys_Sprint_20W44'),
    ('07.03.0C.0D.00', 'JLRCOR','2020-10-11', '2020-12-
01', 'MLA Production Release', 'Released', 'ClimSys_Sprint_20W44'),
    ('06.02.36.59.00', 'JLRCOR','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'ClimSys_Sprint_20W41'),
    ('05.02.67.56.00', 'JLRCOR','2020-10-11', '2020-12-
01', 'EVA1 Production Release', 'Released', 'ClimSys_Sprint_20W35'),
    ('02.03.34.38', 'SEAT','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Unreleased', 'SeatSys_Sprint_20W44'),
    ('02.02.27.2B', 'SEAT','2020-10-11', '2020-12-
01', 'EVA2 Engineering Release', 'Unreleased', 'SeatSys_Sprint_20W44'),
```

```
   ('02.02.28.2C', 'SEAT','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Released', 'SeatSys_Sprint_20W44'),
   ('02.02.2E.32', 'SEAT','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Released', 'SeatSys_Sprint_20W29'),
   ('02.02.2F.33', 'SEAT','2020-10-11', '2020-12-
01', 'EVA2 Engineering Release', 'Unreleased', 'SeatSys_Sprint_20W29'),
   ('02.02.31.35', 'SEAT','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Released', 'SeatSys_Sprint_20W29'),
   ('02.02.33.37', 'SEAT','2020-10-11', '2020-12-
01', 'EVA2 Production Release', 'Released', 'SeatSys_Sprint_20W29'),
   ('02.02.32.36', 'SEAT','2020-10-11', '2020-12-
01', 'EVA2 Engineering Release', 'Released', 'SeatSys_Sprint_20W29'),
   ('None', Null, Null, Null, Null, Null, 'None')

/*Creating a procedure which is called when each transaction is made and checks if user has access to the corres
ponding record - if not then an exception is raised, ending the
transaction*/
CREATE PROCEDURE errorCheckEmpPermissions ()
BEGIN
IF current_user() <> 'CEO@localhost' or current_user() <> 'HR@localhost' or current_user() <> 'JackYoung@l
ocalhost' then
signal sqlstate '45000' set message_text = 'My Error Message';
end if;
END

/*Creating a trigger to monitor database transactions*/
CREATE TRIGGER AfterDeleteEmployee
   AFTER DELETE
   ON employee FOR EACH ROW
BEGIN
   DECLARE currentdate DATETIME;
   DECLARE currentuser VARCHAR(15);
   select sysdate(), CURRENT_USER() into currentdate, currentuser;
   INSERT INTO audits (TableID, TableName, OldData, NewData, EventType, TimeOccured, UserID)
   VALUES (
      OLD.cdsid,
      'Employee',
      JSON_OBJECT(
         "Emp ID", OLD.cdsid,
         "First Name", OLD.fname,
         "Last Name", OLD.fname,
         "Email", OLD.email,
         "DOB", OLD.DOB,
      ),
      null,
      'DELETE',
      currentdate,
      currentuser
```

```
    );
END

/*Creating a trigger to monitor database transactions*/
CREATE TRIGGER AfterInsertEmployee
    AFTER INSERT
    ON employee FOR EACH ROW
BEGIN
    DECLARE currentdate DATETIME;
    DECLARE currentuser VARCHAR(15);
    select sysdate(), CURRENT_USER() into currentdate, currentuser;
    INSERT INTO audits (TableID, TableName, OldData, NewData, EventType, TimeOccured, UserID)
    VALUES (
        NEW.EmpNo,
        'Employee',
        null,
        JSON_OBJECT(
            "Emp ID", NEW.cdsid,
            "First Name", NEW.fname,
            "Last Name", NEW.fname,
            "Email", NEW.email,
            "DOB", NEW.DOB,
        ),
        'INSERT',
        currentdate,
        currentuser
    );
END

/*Creating a trigger to monitor database transactions*/
CREATE TRIGGER AfterUpdateEmployee
    AFTER UPDATE
    ON employee FOR EACH ROW
BEGIN
    DECLARE currentdate DATETIME;
    DECLARE currentuser VARCHAR(15);
    select sysdate(), CURRENT_USER() into currentdate, currentuser;
    INSERT INTO audits (TableID, TableName, OldData, NewData, EventType, TimeOccured, UserID)
    VALUES (
        NEW.EmpNo,
        'Employee',
        JSON_OBJECT(
            "Emp ID", OLD.cdsid,
            "First Name", OLD.fname,
            "Last Name", OLD.fname,
            "Email", OLD.email,
            "DOB", OLD.DOB,
        ),
```

```
        JSON_OBJECT(
            "Emp ID", NEW.cdsid,
            "First Name", NEW.fname,
            "Last Name", NEW.fname,
            "Email", NEW.email,
            "DOB", NEW.DOB,
        ),
        'UPDATE',
        currentdate,
        currentuser
    );
END

CREATE USER 'HR'@'%' IDENTIFIED BY 'HR101';
CREATE USER 'CEO'@'%' IDENTIFIED BY 'CEO';
CREATE USER 'dtayl169'@'%' IDENTIFIED BY 'Dom';
CREATE USER 'jyoung16'@'%' IDENTIFIED BY 'Young';

GRANT select, insert, DELETE ON jiradb.* TO 'dtayl169'@'localhost';
GRANT ALL PRIVILEGES ON *.* TO 'CEO'@'localhost';
GRANT ALL PRIVILEGES ON *.* TO 'HR'@'localhost';
GRANT ALL PRIVILEGES ON *.* TO 'jyoung16'@'localhost';
```



```
"Issue key",Sprint,Summary,IssueType,Status,Priority,AffectsVersion,FixVersion,Component,Labels,Assignee,Reporter,Created
BSM-1002,None,"L461 Ext Light LCF Setting","Change Request","Verification Test",Major,None,"FB14.0 - IO Model/SAL Release","Exterior Lighting",FB14,sunamba,sunamba,"2020-10-27 11:28:00"
BSM-1003,None,"X391 Ext Light LCF Setting","Change Request","Verification Test",Major,None,"FB14.0 - IO Model/SAL Release",,FB14,sunamba,sunamba,"2020-10-27 11:29:00"
BSM-1004,None,"L392 Ext Light LCF Setting","Change Request","Verification Test",Major,None,"FB14.0 - IO Model/SAL Release",Exterior_Lighting_Control,FB14,sunamba,sunamba,"2020-10-27 11:29:00"
BSM-1005,None,"X393 Ext Light LCF Setting","Change Request","Verification Test",Major,None,"FB14.0 - IO Model/SAL Release","Exterior Lighting",FB14,sunamba,sunamba,"2020-10-27 11:31:00"
BSM-1007,None,"Interface change for Sunroof Enable System-Queries",Task,Open,Minor,None,None,,,arace2,gsusanva,"2020-10-28 09:32:00"
BSM-1037,BCM_Migr_W45,"MCDC Coverage - Roof-VTG_2",Story,Analysis,Minor,None,None,,,gsusanva,pmishra5,"2020-10-30 04:57:00"
BSM-1039,None,"GWM Cyber Secure Lear test release","Change Request",Analysis,Major,None,None,,A3R2,hraza,adaniel2,"2020-10-30 10:12:00"
BSM-1048,"BodyModel_Sprint _20W42","Model Peer Review",Sub-task,Open,Major,None,None,,,rhumph27,fpontor1,"2020-11-02 09:46:00"
BSM-1050,"BodyModel_Sprint _20W42","Model Implementation",Sub-task,"Verification Test",Major,None,None,,,fpontor1,fpontor1,"2020-11-02 09:49:00"
BSM-1052,"BodyModel_Sprint _20W42","Model Test Cases Preparation",Sub-task,Implementation,Major,None,None,,,fpontor1,fpontor1,"2020-11-02 09:49:00"
BSM-1055,None,"Requirement Analysis and Test Case - Transport",Story,Open,Minor,None,None,,,pmishra5,pmishra5,"2020-11-03 08:26:00"
BSM-1060,None,"Software Release - to align to Cabin purification BCM Interface Requirements ","Change Request",Analysis,Major,None,None,,,vfernan1,vfernan1,"2020-11-04 11:46:00"
BSM-1070,BCM_Migr_W45,"Review - Custom Code Replace With Data Dictionary",Sub-task,Implementation,Minor,None,None,,,pmishra5,jmundass,"2020-11-06 13:23:00"
BSM-163,"BodyModel_Sprint _20W42","Model Implementation",Sub-task,Open,Major,None,None,,,rhumph27,DTAYL169,"2020-03-18 14:53:00"
BSM-164,"BodyModel_Sprint _20W42","Model Peer Review",Sub-task,Open,Major,None,None,,,fpontor1,DTAYL169,"2020-03-18 14:53:00"
BSM-165,"BodyModel_Sprint _20W42","Model UT",Sub-task,Open,Major,None,None,,,rhumph27,DTAYL169,"2020-03-18 14:54:00"
BSM-166,"BodyModel_Sprint _20W42","Model Test Case Preperation",Sub-task,Open,Major,None,None,,,rhumph27,DTAYL169,"2020-03-18 14:54:00"
BSM-167,"BodyModel_Sprint _20W42","Model IO Update",Sub-task,Open,Major,None,None,,,EJUBBER,DTAYL169,"2020-03-18 14:54:00"
BSM-210,None,"M9V2-14C185-AA80 L392 LCF updates",Task,"Ready for Deployment",Major,None,None,Wash_Wipe_Control,,GGARZELL,GGARZELL,"2020-04-07 14:13:00"
BSM-219,None,"Showroom Mode Function Definition",Task,"Verification Test",Minor,None,None,,,jyoung16,DTAYL169,"2020-04-21 18:00:00"
BSM-26,"BodyModel_Sprint _20W42","Power Side Door Unlock Control","Change Request",Open,Minor,None,"FB14.1 - Functional Model Release",Locking_Control,A3R2,STHOM168,dwal12,"2019-11-01 10:11:00"
BSM-275,None,"Delivered Software Validation - Start Auth","Sub Test Execution","Ready for Deployment",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release",Start_Authorisation_Control,FB12,dmarsha4,E
BSM-29,"BodyModel_Sprint _20W38","Rear Power Door 2 Stage Unlock/Open via PIVI","Change Request","Ready for Deployment",Minor,None,"FB15.0 - IO Model/SAL Release",Door_Control,A3R,EJUBBER,CCOTTRIL,"2019-11-01 10:18:00"
BSM-297,"BodySoftware_Sprint _FB12","Delivered Software Validation - Wash Wipe","Sub Test Execution","Ready for Deployment",Major,"FB11.1 - Functional Model Release","FB11.1 - Functional Model Release",Wash_Wipe_Control
BSM-298,"BodySoftware_Sprint _FB12","Delivered Software Validation - Wash Wipe","Sub Test Execution","Ready for Deployment",Major,"FB11.1 - Functional Model Release","FB11.1 - Functional Model Release",Wash_Wipe_Control
BSM-301,"BodySoftware_Sprint _FB12","Delivered Software Validation - Security","Sub Test Execution","Ready for Deployment",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release",Alarm_Control,FB12,dwa
BSM-302,"BodySoftware_Sprint _FB12","Delivered Software Validation - Security","Sub Test Execution","Verification Test",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release",Locking_Control,FB12,CCOT
BSM-303,"BodySoftware_Sprint _FB12","Delivered Software Validation - Exterior Lighting","Sub Test Execution","Ready for Deployment",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release",Exterior_Ligh
BSM-304,"BodySoftware_Sprint _FB12","Delivered Software Validation - Ambient Lighting","Sub Test Execution","Verification Test",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release","Ambient Lighting
BSM-308,"BodySoftware_Sprint _FB12","Delivered Software Validation - Ambient Lighting","Sub Test Execution","Verification Test",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release","Ambient Lighting
BSM-309,"BodySoftware_Sprint _FB12","Delivered Software Validation - Core Diagnostics","Sub Test Execution","Ready for Deployment",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release",Diagnostic_Con
BSM-312,"BodySoftware_Sprint _FB12","Delivered Software Validation - Ambient Lighting","Sub Test Execution","Ready for Deployment",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release","Ambient Light
BSM-314,"BodySoftware_Sprint _FB12","Delivered Software Validation - Global Window Operation","Sub Test Execution","Verification Test",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release",Window_Con
BSM-315,"BodyModel_Sprint _20W10","Delivered Software Validation - Ambient Lighting","Sub Test Execution","Verification Test",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release","Ambient Lighting_C
BSM-316,"BodyModel_Sprint _20W10","Delivered Software Validation - Column Control","Sub Test Execution","Verification Test",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release",Column_Control,FB12,a
BSM-318,"BodyModel_Sprint _20W10","Delivered Software Validation - Ambient Lighting","Sub Test Execution","Verification Test",Major,"FB12.0 - IO Model/SAL Release","FB12.1 - Functional Model Release","Ambient Lighting_C
BSM-319,"BodyModel_Sprint _20W10","Delivered Software Validation - Interior Lighting","Sub Test Execution","Verification Test",Major,"FB12.0 - IO Model/SAL Release","FB13.1 - Functional Model Release",Interior_Lighting_
BSM-342,None,"Delivered Software Validation - Backlighting","Sub Test Execution","Verification Test",Major,"FB12.1 - Functional Model Release","FB13.1 - Functional Model Release",,Backlighting,GMILL193,EJUBBER,"2020-06-
BSM-366,None,"Delivered Software Validation - Wash Wipe","Sub Test Execution","Verification Test",Major,"FB11.1 - Functional Model Release","FB11.1 - Functional Model Release",Wash_Wipe_Control,FB11,GGARZELL,dmarsha4,"2
BSM-368,None,"Backlighting Peer Review Changes",Task,Open,Minor,None,None,Backlighting_Control,,fpontor1,rhumph27,"2020-06-10 11:54:00"
BSM-372,None,"Delivered Software Validation - Roof System","Sub Test Execution","Verification Test",Major,"FB12.1 - Functional Model Release","FB13.1 - Functional Model Release",Roof_Control,FB13,arace2,EJUBBER,"2020-06
BSM-375,"BodyModel_Sprint _20W42","Tidy up provisional unlock and approach unlock test cases",Task,Implementation,Minor,None,None,Locking_Control,,rhumph27,rhumph27,"2020-06-15 07:30:00"
BSM-385,None,"Last cause of Alarm Update","Change Request",Analysis,Major,None,None,Alarm_Control,,mlaw1,mlaw1,"2020-06-18 14:15:00"
BSM-386,None,"Delivered Software Validation - Transit / Transport Control","Sub Test Execution","Ready for Deployment",Major,"FB11.1 - Functional Model Release","FB12.1 - Functional Model Release",CarMode,CarMode,poluwa
BSM-387,None,"Delivered Software Validation - Transit / Transport Control","Sub Test Execution","Ready for Deployment",Major,"FB11.1 - Functional Model Release","FB12.1 - Functional Model Release",CarMode,CarMode,poluwa
BSM-388,None,"Delivered Software Validation - Transit / Transport Control","Sub Test Execution","Ready for Deployment",Major,"FB11.1 - Functional Model Release","FB12.1 - Functional Model Release",CarMode,CarMode,poluwa
BSM-389,None,"Delivered Software Validation - Transit / Transport Control","Sub Test Execution","Ready for Deployment",Major,"FB11.1 - Functional Model Release","FB12.1 - Functional Model Release",CarMode,CarMode,poluwa
BSM-390,None,"Delivered Software Validation - Transit / Transport Control","Sub Test Execution","Ready for Deployment",Major,"FB11.1 - Functional Model Release","FB12.1 - Functional Model Release",CarMode,CarMode,poluwa
BSM-391,None,"Delivered Software Validation - Transit / Transport Control","Sub Test Execution","Ready for Deployment",Major,"FB11.1 - Functional Model Release","FB12.1 - Functional Model Release",CarMode,CarMode,poluwa
BSM-392,None,"Delivered Software Validation - Transit / Transport Control","Sub Test Execution","Ready for Deployment",Major,"FB11.1 - Functional Model Release","FB12.1 - Functional Model Release",CarMode,CarMode,poluwa
```

Figure 18: Data for ticket table that will be batch input

1.10. Python

```
'''
This program is created in Python 3.8 and is dependant on PyQt5 being installed via pip.
```

```
Requirements:
    PyQt5
    Python 3.8
    MySQL Server with legacy authentication hosted on localhost
    Server must be configured as per SQL Queries in Appendix above marked 'SQL'

Author: 1921983
'''


# importing GUI library
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QTableWidget, QTableWidgetItem, QFileDialog
# library to connect to mysql database
import mysql.connector
# library to export to csv
import csv

# global variables so all functions can connect to database
global conn
global cur
conn = 0
cur = 0

# this class contains all functions which directly affect the appearance of my application and the setup code for the Py Qt gui.
class Ui_DatabaseGUI(object):
    def setupUi(self, DatabaseGUI):
        DatabaseGUI.setObjectName("DatabaseGUI")
        DatabaseGUI.resize(665, 681)
        # setting widgets on tab 1 to their location and type
        self.centralwidget = QtWidgets.QWidget(DatabaseGUI)
        self.centralwidget.setObjectName("centralwidget")
        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
        self.tabWidget.setGeometry(QtCore.QRect(20, 150, 631, 481))
        self.tabWidget.setObjectName("tabWidget")
        self.queryTab = QtWidgets.QWidget()
        self.queryTab.setObjectName("queryTab")
        self.queryViewButton = QtWidgets.QPushButton(self.queryTab)
        self.queryViewButton.setGeometry(QtCore.QRect(70, 40, 93, 28))
        self.queryViewButton.setObjectName("queryViewButton")
        self.queryViewButton.clicked.connect(preMadeQueryTab.queryViewButtonPressed)
        self.viewsComboBox = QtWidgets.QComboBox(self.queryTab)
        self.viewsComboBox.setGeometry(QtCore.QRect(10, 10, 191, 22))
        self.viewsComboBox.setObjectName("viewsComboBox")
        self.viewsComboBox.addItem("")
        self.viewsComboBox.addItem("")
        self.viewsComboBox.addItem("")
```

```
        self.viewsComboBox.addItem("")
        self.viewsComboBox.addItem("")
        self.viewsComboBox.addItem("")
        self.queryTableButton = QtWidgets.QPushButton(self.queryTab)
        self.queryTableButton.setGeometry(QtCore.QRect(260, 40, 93, 28))
        self.queryTableButton.setObjectName("queryTableButton")
        self.queryTableButton.clicked.connect(preMadeQueryTab.queryTableButtonPressed)
        self.viewTableComboBox = QtWidgets.QComboBox(self.queryTab)
        self.viewTableComboBox.setGeometry(QtCore.QRect(250, 10, 121, 22))
        self.viewTableComboBox.setObjectName("viewTableComboBox")
        self.viewTableComboBox.addItem("")
        self.viewTableComboBox.addItem("")
        self.viewTableComboBox.addItem("")
        self.viewTableComboBox.addItem("")
        self.viewTableComboBox.addItem("")
        self.viewTableComboBox.addItem("")
        self.tableWidget = QtWidgets.QTableWidget(self.queryTab)
        self.tableWidget.setGeometry(QtCore.QRect(10, 90, 601, 351))
        self.tableWidget.setObjectName("tableWidget")
        self.tableWidget.setColumnCount(0)
        self.tableWidget.setRowCount(0)
        self.tabWidget.addTab(self.queryTab, "")
        self.exportTab = QtWidgets.QWidget()
        self.exportTab.setObjectName("exportTab")
        self.exportTableCombo = QtWidgets.QComboBox(self.exportTab)
        self.exportTableCombo.setGeometry(QtCore.QRect(140, 30, 121, 22))
        self.exportTableCombo.setObjectName("exportTableCombo")
        self.exportTableCombo.addItem("")
        self.exportTableCombo.addItem("")
        self.exportTableCombo.addItem("")
        self.exportTableCombo.addItem("")
        self.exportTableCombo.addItem("")
        self.exportTableCombo.addItem("")
        self.exportTableCombo.addItem("")
        self.pathToSaveEditBox = QtWidgets.QLineEdit(self.exportTab)
        self.pathToSaveEditBox.setGeometry(QtCore.QRect(120, 77, 181, 22))
        self.pathToSaveEditBox.setObjectName("pathToSaveEditBox")
        self.exportButton = QtWidgets.QPushButton(self.exportTab)
        self.exportButton.setGeometry(QtCore.QRect(150, 120, 93, 28))
        self.exportButton.setObjectName("exportButton")
        self.exportButton = QtWidgets.QPushButton(self.exportTab)
        self.selectTableLabel = QtWidgets.QLabel(self.exportTab)
        self.selectTableLabel.setGeometry(QtCore.QRect(20, 30, 81, 16))
        self.selectTableLabel.setObjectName("selectTableLabel")
        self.pathsSaveLabel = QtWidgets.QLabel(self.exportTab)
        self.pathsSaveLabel.setGeometry(QtCore.QRect(20, 80, 81, 16))
        self.pathsSaveLabel.setObjectName("pathsSaveLabel")
        self.selectPathButton = QtWidgets.QPushButton(self.exportTab)
```

```
        self.selectPathButton.setGeometry(QtCore.QRect(300, 77, 21, 21))
        self.selectPathButton.setObjectName("selectPathButton")
        self.selectPathButton.clicked.connect(exportTab.selectPathButtonPressed)
        self.tabWidget.addTab(self.exportTab, "")
        self.loginTab = QtWidgets.QWidget()
        self.loginTab.setObjectName("loginTab")
        self.userInput = QtWidgets.QLineEdit(self.loginTab)
        self.userInput.setGeometry(QtCore.QRect(90, 20, 113, 22))
        self.userInput.setObjectName("userInput")
        self.pwdInput = QtWidgets.QLineEdit(self.loginTab)
        self.pwdInput.setGeometry(QtCore.QRect(90, 70, 113, 22))
        self.pwdInput.setObjectName("pwdInput")
        self.userLabel = QtWidgets.QLabel(self.loginTab)
        self.userLabel.setGeometry(QtCore.QRect(20, 20, 55, 16))
        self.userLabel.setObjectName("userLabel")
        self.passwordLabel = QtWidgets.QLabel(self.loginTab)
        self.passwordLabel.setGeometry(QtCore.QRect(10, 74, 55, 16))
        self.passwordLabel.setObjectName("passwordLabel")
        self.connectButton = QtWidgets.QPushButton(self.loginTab)
        self.connectButton.setGeometry(QtCore.QRect(10, 130, 93, 28))
        self.connectButton.setObjectName("connectButton")
        self.connectButton.clicked.connect(loginTab.connectDbFunc)
        self.disconnectButton = QtWidgets.QPushButton(self.loginTab)
        self.disconnectButton.setGeometry(QtCore.QRect(120, 130, 93, 28))
        self.disconnectButton.setObjectName("disconnectButton")
        self.disconnectButton.clicked.connect(loginTab.disconnectDbFunc)
        self.tabWidget.addTab(self.loginTab, "")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(20, 10, 611, 131))
        self.label.setText("")
        self.label.setPixmap(QtGui.QPixmap("Screenshots/corporate.png"))
        self.label.setScaledContents(True)
        self.label.setObjectName("label")
        DatabaseGUI.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(DatabaseGUI)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 665, 26))
        self.menubar.setObjectName("menubar")
        self.menuMenu = QtWidgets.QMenu(self.menubar)
        self.menuMenu.setObjectName("menuMenu")
        DatabaseGUI.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(DatabaseGUI)
        self.statusbar.setObjectName("statusbar")
        DatabaseGUI.setStatusBar(self.statusbar)
        self.actionAbout = QtWidgets.QAction(DatabaseGUI)
        self.actionAbout.setObjectName("actionAbout")
        self.menuMenu.addAction(self.actionAbout)
        self.menubar.addAction(self.menuMenu.menuAction())
```

```
        self.retranslateUi(DatabaseGUI)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(DatabaseGUI)

    def retranslateUi(self, DatabaseGUI):
        _translate = QtCore.QCoreApplication.translate
        DatabaseGUI.setWindowTitle(_translate("DatabaseGUI", "MainWindow"))
        self.queryViewButton.setText(_translate("DatabaseGUI", "Query View"))
        self.viewsComboBox.setItemText(0, _translate("DatabaseGUI", "All Tickets"))
        self.viewsComboBox.setItemText(1, _translate("DatabaseGUI", "Tickets & Corresponding Sprint"))
        self.viewsComboBox.setItemText(2, _translate("DatabaseGUI", "How Many Tickets Are In Sprint"))
        self.viewsComboBox.setItemText(3, _translate("DatabaseGUI", "How Many Tickets Are In Release"))
        self.viewsComboBox.setItemText(4, _translate("DatabaseGUI", "How Many Tickets Are In Project"))
        self.viewsComboBox.setItemText(5, _translate("DatabaseGUI", "All Employees"))
        self.queryTableButton.setText(_translate("DatabaseGUI", "View Table"))
        self.viewTableComboBox.setItemText(0, _translate("DatabaseGUI", "Tickets"))
        self.viewTableComboBox.setItemText(1, _translate("DatabaseGUI", "Employee"))
        self.viewTableComboBox.setItemText(2, _translate("DatabaseGUI", "Release"))
        self.viewTableComboBox.setItemText(3, _translate("DatabaseGUI", "Sprints"))
        self.viewTableComboBox.setItemText(4, _translate("DatabaseGUI", "Project"))
        self.viewTableComboBox.setItemText(5, _translate("DatabaseGUI", "Audit"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.queryTab), _translate("DatabaseGUI", "Pre-
Made Query"))
        self.exportTableCombo.setItemText(0, _translate("DatabaseGUI", "Tickets"))
        self.exportTableCombo.setItemText(1, _translate("DatabaseGUI", "Employee"))
        self.exportTableCombo.setItemText(2, _translate("DatabaseGUI", "Project"))
        self.exportTableCombo.setItemText(3, _translate("DatabaseGUI", "Release"))
        self.exportTableCombo.setItemText(4, _translate("DatabaseGUI", "Sprints"))
        self.exportTableCombo.setItemText(5, _translate("DatabaseGUI", "Audit"))
        self.exportTableCombo.setItemText(6, _translate("DatabaseGUI", "All Tables"))
        self.exportButton.setText(_translate("DatabaseGUI", "Export"))
        self.selectTableLabel.setText(_translate("DatabaseGUI", "Select Table"))
        self.pathsSaveLabel.setText(_translate("DatabaseGUI", "Path to save"))
        self.selectPathButton.setText(_translate("DatabaseGUI", "..."))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.exportTab), _translate("DatabaseGUI", "Export DB"))
        self.userLabel.setText(_translate("DatabaseGUI", "User"))
        self.passwordLabel.setText(_translate("DatabaseGUI", "Password"))
        self.connectButton.setText(_translate("DatabaseGUI", "Connect"))
        self.disconnectButton.setText(_translate("DatabaseGUI", "Disconnect"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.loginTab), _translate("DatabaseGUI", "Login"))
        self.menuMenu.setTitle(_translate("DatabaseGUI", "Menu"))
        self.actionAbout.setText(_translate("DatabaseGUI", "About"))

# this class contains all functions which contain logic and output code for the 1st tab shown in the gui
class preMadeQueryTab_Logic():

    # function to query tables in database and return in form of table on gui
    def queryTableButtonPressed(self):
```

```
        # getting table and username from gui
        table = ui.viewTableComboBox.currentText()
        username = ui.userInput.text()
        # try except to catch error of logging in
        try:
        # setting columns to corresponding table selected
            if table == 'Employee':
                # column query gets info from the hidden information_schema table
                # query to select all from selected table
                query = "SELECT * from employee"
                columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'employee' and table_schema = 'jiradb' order by ordinal_position"
            elif table ==  'Project':
                query = "SELECT * from projects"
                columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'projects' and table_schema = 'jiradb' order by ordinal_position"
            elif table ==  'Release':
                query = "SELECT * from releases"
                columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'releases' and table_schema = 'jiradb' order by ordinal_position"
            elif table ==  'Sprints':
                query = "SELECT * from sprints"
                columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'sprints' and table_schema = 'jiradb' order by ordinal_position"
            elif table ==  'Tickets':
                query = "SELECT * from tickets"
                columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'tickets' and table_schema = 'jiradb' order by ordinal_position"
            elif table ==  'Audit':
                query = "SELECT * from audits"
                columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'audits' and table_schema = 'jiradb' order by ordinal_position"
            elif table == 'Audits' and (username == 'root' or username == 'DBA' or username == 'JackYoung'):
                query = "SELECT * from audits"
            else:
                QtWidgets.QMessageBox.warning(None, "Error", "You do not have access to this table")

            # array for columns to be stored in
            columns = []
            # cursor executing chosen column query
            cur.execute(columnQuery)
            for row in cur:
                row = row[0]
                # appending columns from query into the columns array
                columns.append(row)
            # 2d array to store records and attributes in them
            array = []
            # selecting all from chosen table
```

```
            cur.execute(query)
            for row in cur:
                # appending records to 2d array
                array.append(row)
            # setting gui table columns names and row and column lengths to the length of array
            ui.tableWidget.setColumnCount(len(array[0]))
            ui.tableWidget.setRowCount(len(array))
            ui.tableWidget.setHorizontalHeaderLabels(columns)
            for row in range(len(array)):
                for column in range(len(array[0])):
                    # inserting data into table
                    ui.tableWidget.setItem(row,column,QTableWidgetItem(str((array[row][column]))))
        except:
            # if user is not logged in, then send them to login tab
            QtWidgets.QMessageBox.warning(None, "Error", "You are not logged in")
            ui.tabWidget.setCurrentIndex(3)
            QtWidgets.QMessageBox.information(None, "Info", "Please log in")

# this class contains all functions which contain logic and output code for the 2nd tab shown in the gui
class exportDBTab_Logic():
    # function is called when export button is pressed on export tab
    def exportButtonPressed(self):
        # getting username and table selected from the gui
        username = ui.userInput.text()
        table = ui.exportTableCombo.currentText()
        # try and except to catch error from login missing
        try:
            # setting columns to corresponding table selected
            if table == 'Employee':
                # column query gets info from the hidden information_schema table
                # query to select all from selected table
                query = "SELECT * from employee"
                columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'employee' and table_schema = 'jiradb' order by ordinal_position"
            elif table ==  'Project':
                query = "SELECT * from projects"
                columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'projects' and table_schema = 'jiradb' order by ordinal_position"
            elif table ==  'Release':
                query = "SELECT * from releases"
                columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'releases' and table_schema = 'jiradb' order by ordinal_position"
            elif table ==  'Sprints':
                query = "SELECT * from sprints"
                columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'sprints' and table_schema = 'jiradb' order by ordinal_position"
            elif table ==  'Tickets':
                query = "SELECT * from tickets"
```

```
            columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'tickets' and table_schema = 'jiradb' order by ordinal_position"
        elif table ==  'Audit':
            query = "SELECT * from audits"
            columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'audits' and table_schema = 'jiradb' order by ordinal_position"
        elif table == 'All Tables':
            query = "SELECT * FROM exportTable"
            columnQuery = "SELECT distinct column_name from information_schema.columns where table_name = N
'exporttable' and table_schema = 'jiradb' order by ordinal_position"
        elif table == 'Audits' and (username == 'root' or username == 'DBA' or username == 'JackYoung'):
            query = "SELECT * from audits"
        else:
            QtWidgets.QMessageBox.warning(None, "Error", "You do not have access to this table")

        # array to save columns and set table column names
        columns = []
        cur.execute(columnQuery)
        for row in cur:
            row = row[0]
            columns.append(row)

        # 2d array to store records and attributes
        array = []
        cur.execute(query)
        for row in cur:
            array.append(row)

        # if file path selected is nothing
        if self.filePath == '':
            # ask user if they are sure they want to save to current working directory
            ans = QtWidgets.QMessageBox.question(None, "Warning", "File Path is empty - file will be saved in curren
t working directory\n Do you want to continue?")
            # checks question answer and acts accordingly
            if ans == QtWidgets.QMessageBox.No:
                # if answer is no then abort export
                QtWidgets.QMessageBox.warning(None, "Aborted", "Export Aborted")
            else:
                # if answer is yes then export to cwd
                with open("export.csv","w+") as my_csv:
                    csvWriter = csv.writer(my_csv,delimiter=',')
                    # write column names
                    csvWriter.writerow(columns)
                    # write records
                    csvWriter.writerows(array)
                # message to show where export is saved
                QtWidgets.QMessageBox.information(None, "Success", "Export complete in CWD/export.csv")
        else:
```

```
            # if file path is not empty then save to file path
            with open(self.filePath + "/export.csv","w+") as my_csv:
                csvWriter = csv.writer(my_csv,delimiter=',')
                # write column names
                csvWriter.writerow(columns)
                # write records
                csvWriter.writerows(array)
            # message to show where file is saved
            QtWidgets.QMessageBox.information(None, "Success", 'Export complete in "' + self.filePath + '/export.csv
")
        except:
            # if error then user is not logged in - show error message and send to login page
            QtWidgets.QMessageBox.warning(None, "Error", "You are not logged in")
            ui.tabWidget.setCurrentIndex(3)
            QtWidgets.QMessageBox.information(None, "Info", "Please log in")

    # function that is called when select path button is activated
    def selectPathButtonPressed(self):
        # file dialog to choose path
        self.filePath = QtWidgets.QFileDialog.getExistingDirectory(None, 'Select Folder To Save To')
        # set text box to path
        ui.pathToSaveEditBox.setText(self.filePath)

# this class contains all functions which contain logic and output code for the 3rd tab shown in the gui
class loginTab_Logic():

    # function to connect to database -  called when connect button is pressed
    def connectDbFunc(self):
        # global variables to use cursor in other functions
        global conn
        global cur
        # get username and password from gui
        username = ui.userInput.text()
        pwd = ui.pwdInput.text()
        # try to login
        try:
            # db connection via local host using input password and username
            conn = mysql.connector.connect(
                host="localhost",
                user=username,
                password=pwd,
                database="jiradb"
            )
            cur = conn.cursor()
            # when connected then show connection successful
            QtWidgets.QMessageBox.information(None, "Info", "Connection Successful")
        except:
            # if error then password incorrect or user incorrect
```

```
            QtWidgets.QMessageBox.warning(None, "Error", "User/Password Incorrect")

    # function to disconnect from database -  called when disconnect button is pressed
    def disconnectDbFunc(self):
        # global variables
        global cur
        global conn
        try:
            # disconnecting from db
            cur.close()
            conn.close()
            QtWidgets.QMessageBox.information(None, "Info", "DB Disconnect Successful")
        # catching error if db not connected
        except:
            QtWidgets.QMessageBox.warning(None, "Error", "DB Not connected - disconnection impossible")

if __name__ == "__main__":
    import sys
    # initialising classes with objects
    app = QtWidgets.QApplication(sys.argv)
    DatabaseGUI = QtWidgets.QMainWindow()
    loginTab = loginTab_Logic()
    preMadeQueryTab = preMadeQueryTab_Logic()
    exportTab = exportDBTab_Logic()
    exportTab.filePath = ''
    ui = Ui_DatabaseGUI()
    ui.setupUi(DatabaseGUI)
    # showing gui
    DatabaseGUI.show()
    sys.exit(app.exec_())
```

References

Atlassian. 2020. *JIRA by Atlassian* [Online]. Available: https://www.atlassian.com/software/jira?&aceid=&adposition=&adgroup=95003673729&campaign=9124878942&creative=415523160951&device=c&keyword=atlassian%20jira&matchtype=e&network=g&placement=&ds_kids=p51241609044&ds_e=GOOGLE&ds_eid=700000001558501&ds_e1=GOOGLE&gclid=CjwKCAiA-_L9BRBQEiwA-bm5fqpn6WxjFRuY9ScQhbBb10z51RpY3iYqXzfS-zjmORZpVlc_keheLxoCzrgQAvD_BwE&gclsrc=aw.ds [Accessed 2020].

Elmasri, R. & Navathe, S. B. 2016. *Fundamentals of Database Systems, Global Edition,* Harlow, United Kingdom, UNITED KINGDOM, Pearson Education Limited.

Guru99. 2020. *What is Normalization? 1NF, 2NF, 3NF, BCNF Database Example* [Online]. Available: https://www.guru99.com/database-normalization.html [Accessed 8th December 2020].

Harrington, J. L. 2016a. Chapter 3 - Why Good Design Matters. *In:* HARRINGTON, J. L. (ed.) *Relational Database Design and Implementation (Fourth Edition).* Boston: Morgan Kaufmann.

Harrington, J. L. 2016b. Chapter 5 - The Relational Data Model. *In:* HARRINGTON, J. L. (ed.) *Relational Database Design and Implementation (Fourth Edition).* Boston: Morgan Kaufmann.

Ibm. 2020. *Overview of Rational Team Concert* [Online]. Available: https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.4/com.ibm.team.concert.doc/topics/c_product-overview.html [Accessed 2020].

Kapil, S. 2019. Clean Python Elegant Coding in Python.

Layton, M. C. & Ostermiller, S. J. 2017. *Agile Project Management for Dummies,* Newark, UNITED STATES, John Wiley & Sons, Incorporated.

Lott, S. F. & Fatouhi, D. 2014. Mastering Object-oriented Python.