BEN COXFORD

189006196

YAHUAS Online Management System

Databases & Networks – CB2104

Contents

[Database Design Strategy 0](#_Toc29380910)

[System Purpose 0](#_Toc29380911)

[System Requirements 0](#_Toc29380912)

[Priority 1 0](#_Toc29380913)

[Priority 2 0](#_Toc29380914)

[Business Assumptions 0](#_Toc29380915)

[System Outputs 2](#_Toc29380916)

[Possible and Rejected Entities 2](#_Toc29380917)

[Entity Relationship Matrix 3](#_Toc29380918)

[Metadata Schema 4](#_Toc29380919)

[GDPR & Data Protection Act (2018) 4](#_Toc29380920)

[Validations 5](#_Toc29380921)

[Normalisation Descriptions 5](#_Toc29380922)

[Normalisation 5](#_Toc29380923)

[Normalised to Third Normal Form 6](#_Toc29380924)

[Invoice Table 6](#_Toc29380925)

[Course Table 6](#_Toc29380926)

[Normalising Past Third Normal Form 6](#_Toc29380927)

[Report & Form Designs 7](#_Toc29380928)

[Form Design 7](#_Toc29380929)

[Report Design 7](#_Toc29380930)

[Page Wireframes 7](#_Toc29380931)

[Entity Relationship Diagram 8](#_Toc29380932)

[Forward Engineered 9](#_Toc29380933)

[SQL Scripts 10](#_Toc29380934)

[Login 10](#_Toc29380935)

[Change Password 10](#_Toc29380936)

[Search Person 11](#_Toc29380937)

[Search Apartment 19](#_Toc29380938)

[Search Hall of Residents 20](#_Toc29380939)

[Search Room 22](#_Toc29380940)

[Search University 24](#_Toc29380941)

[Search Lease 26](#_Toc29380942)

[Search Invoice 28](#_Toc29380943)

[Search Inspection 30](#_Toc29380944)

[Search Course 31](#_Toc29380945)

[Room Vacancy Report 33](#_Toc29380946)

[Waiting Student Report 34](#_Toc29380947)

[Unpaid Invoice Report 35](#_Toc29380948)

[Inspection Damage Report 35](#_Toc29380949)

[Planned Inspection Report 36](#_Toc29380950)

[Occupancy Report 37](#_Toc29380951)

[Income Report 38](#_Toc29380952)

[Unknown Leaving Date Report 38](#_Toc29380953)

[Create Lease 39](#_Toc29380954)

[Create Inspection 39](#_Toc29380955)

[Create Staff 40](#_Toc29380956)

[Create Apartment 41](#_Toc29380957)

[Create Hall of Residents 41](#_Toc29380958)

[Create University 42](#_Toc29380959)

[Create Course 42](#_Toc29380960)

[Create Room 43](#_Toc29380961)

[Create Invoice 43](#_Toc29380962)

[Create Adviser 44](#_Toc29380963)

[MariaDB Transactions (ACID) 45](#_Toc29380964)

[Search and Report Test Data 46](#_Toc29380965)

[How could the system be improved? 47](#_Toc29380966)

[Appendices 48](#_Toc29380967)

[Appendix A – Metadata Schema 48](#_Toc29380968)

[Appendix B - Wireframes 60](#_Toc29380969)

# Database Design Strategy

As described by Coronel and Morris (2016), there is two design strategies; a top-down design or a bottom-up design. Top-down requires the entities to be identified, followed by defining the attributes for each entity. Whereas, bottom-up requires the attributes to be identified and then grouped into entities. For this design, a top-down approach is selected due to the case study providing a more explicit view of each element. The analysis guide presented by Panesar (2019), shows a logical sequence of steps to create an entity-relationship diagram (ERD). This is used to develop an initial ERD model. The entities are then normalized to create a final, complete ER diagram.

# System Purpose

Todesign and implement a bespoke back-end prototype, for YAHUAS as a management solution for their business to optimize utilisation, and lower operating costs.

# System Requirements

## Priority 1

1. This single user must have login details to use the web page to carry out the administrative/manager functions of YAHUAS solution.
2. Data entry – administrators should be able to enter data into the system to achieve the business processes of YAHAUS.
3. Data validation – ensure the correct validations are considered for any database manipulations via the web interface.
4. All SQL reports should be presented in a tabular form via the web interface

## Priority 2

1. The manager can select which type of report they require from operational, tactical or strategic reports.
2. A search button.

# Business Assumptions

* The sole user is the residential office staff and the customer base is students.
* There is a dual role of one user for an administrator and manager.
* More than one university is inferred; hence, any relationships between an entity and a university should be treated as such.
* General details are held on students, staff, emergency contacts and advisers for the resident staff to contact regarding any issues. These include an address, email and mobile number.
* As an accommodation management system, any payments by a student are beyond the system.
* Students enter their details and emergency contact details via a link. Therefore, staff should not be able to create a new student and only be able to amend/delete their details when necessary.
* Test/previous data can be populated directly to the database via PHPMyAdmin/MySQLWorkbench/MySQL console.
* The system will have a standard, consistent design that allows the sole user of the system to navigate and produce the relevant reports.
* There are no design constraints.
* Accommodation maintenance logs are the same and inspection logs.
* Business reports are displayed in tabular form and can be printed separately from the interface.
* Logins are to be securely stored on the database.
* The business should already have documentation and a policy in place for GDPR. Their policy will need updating.
* There is a waiting list of students.
* A student enrols in a single course and can rent a hall or an apartment.
* Each student is assigned an adviser who monitors a student’s welfare during their time at university.
* Rooms are identified by a unique place number. Details are also stored on the monthly rate and room number.
* There are three semester periods 1 and 2 and the summer semester.
* Each lease is uniquely identified by a lease number and various data is stored on the lease.
* An invoice is attached to a lease through the lease number.
* Each invoice stores various details including payment method, payment due and first and second reminders.
* Student apartments are inspected by staff. Information is kept on the date, whether the inspection was satisfactory and any additional comments.
* A limited amount of information is kept on the courses offered by a university. Each student is associated with a single programme of study.

# System Outputs

Staff for YAHUAS must be able to query, update and produce strategical, tactical and operational reports on:

* Room Vacancies
* Student Waiting List
* Unpaid Invoices
* Planned Inspections
* Inspection Damage Report
* Room Occupancy Report
* Unknown Leaving Date Report
* Income report (all rooms, apartments, and hall-of-residents) between dates.

Additionally, they must be given the ability to search for an invoice, lease, university, course, student, staff, adviser, apartment, hall-of-resident, rooms, and any other potential entities. Staff members must be able to change their password. Staff must also be given the ability to delete records where necessary.

# Possible and Rejected Entities

* Student
* Resident Staff
* ~~Administrator~~: One dual-user user as resident staff.
* ~~Hall Manager~~: One dual-user as resident staff.
* General Details
* Address
* Adviser
* Apartment
* Hall of Residents (HOR)
* Course
* Invoice
* ~~Waiting List~~:A waiting list can be determined by whether a students’ lease has been assigned a room or not.
* University
* Emergency Contact
* Room
* Lease
* Inspection
* ~~Maintenance~~: Inspection is the only requirement.
* Login
* ~~Residential Office:~~  There is no indication there is more than one residential office.

*(Crossed Out=Rejected Entities)*

# Entity Relationship Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Student | Staff | Adviser | Address | Invoice | Lease | Apartment | Hall of Residents | General Details | University | Course | Inspection | Emergency Contact | Room | Login |
| Student | X | x | m:0/1 | x | X | 1:0/1 | x | X | 1:1 | X | m:0/1 | X | 1:m | X | X |
| Staff | X | X | x | X | X | X | X | 0/1:1 | 1:1 | X | X | 0/1:1 | X | X | 1:1 |
| Adviser | X | x | X | X | X | X | X | x | 1:1 | m:1 | X | X | X | X | X |
| Address | X | X | X | X | x | X | 1:1 | 1:1 | 1:1 | 1:1 | X | X | X | X | X |
| Invoice | X | X | X | X | X | 1:1 | X | X | X | X | X | X | X | X | X |
| Lease | X | X | X | X | X | X | x | X | X | X | X | X | X | 0/1:1 | X |
| Apartment | X | X | X | X | X | X | X | x | X | X | X | X | X | 0/1:m | X |
| Hall of Residents | X | X | X | X | X | X | X | X | X | X | X | X | X | 0/1:m | X |
| General Details | X | X | X | X | X | X | X | X | X | x | X | X | 1:1 | X | X |
| University | X | X | X | X | X | X | X | X | X | X | m:1 | X | X | X | X |
| Course | X | X | X | X | X | X | X | X | X | X | X | x | X | X | X |
| Inspection | X | X | X | X | X | X | X | X | X | X | X | X | X | 1:1 | X |
| Emergency Contact | X | X | X | X | X | X | X | X | X | X | X | X | X | x | X |
| Room | X | X | X | X | X | X | X | X | X | X | X | X | X | X | x |
| Login | X | x | X | x | x | x | x | x | x | x | x | x | X | X | x |

# Metadata Schema

See Appendix A.

# GDPR & Data Protection Act (2018)

The Data Protection Act 2018 (DPA) is the United Kingdom’s implementation of the General Data Protection Regulation (GDPR) and it controls how your personal information is used by organisations, businesses or the government.

As gov.uk states:

Data must be:

* used fairly, lawfully and transparently
* used for specified, explicit purposes
* used in a way that is adequate, relevant and limited to only what is necessary
* accurate and, where necessary, kept up to date
* kept for no longer than is necessary
* handled in a way that ensures appropriate security, including protection against unlawful or unauthorised processing, access, loss, destruction or damage

Furthermore, individuals have the right to be informed about how, why and what data is used and stored. If the data is outdated, the individual can have the incorrect data updated and even request the data be erased. In addition to these rights, an individual can restrict or stop the processing of their data and object to how data is processed in certain circumstances.

This requires the system to allow residential staff to amend, delete and correct personal details as well as produce a copy of the personal data kept on an individual. Access rights must be given only to residential staff to protect the data. The metadata schema has been updated with two extra columns stating the purpose of the data and how long it should be kept for (See appendix A).

The business should already have training and their own data protection policies; however, additional training and policies should be provided on the system. This surrounds how to use the system lawfully when storing the personal details of a staff member, student, adviser, and emergency contact.

# Validations

Validation is required for each data type, to ensure the integrity of the database. For instance, from the table shown above, when updating or inserting a date for an inspection, it must be in the correct format YYYY-MM-DD. Additionally, validation is required to check the length of a 128-bit, MD5 hash value. The value entered or updated will be tested for 32 characters. Any integer or decimal values must be validated before updating/inserting into the database.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Format | Validation Checks | Example Pass | Example Fail |
| DATE | YYYY-MM-DD | Date Check | 2019-01-01 | 19-14-01 |
| INT | INTEGER | Is Integer Check | 1 | One |
| Mobile | INTEGER | Is Integer Check  Length Check | 07123456789 | 3456789 |
| Email | VARCHAR | Filter Email | Ben.coxford@example.com | bencoxford@email |
| Postcode | VARCHAR | Length Check | YO19TL | YO19T |
| Decimal/Integer | DECIMAL | Is Numerical | 200 | Two hundred/20.0.1 |

# Normalisation Descriptions

## Normalisation

The work of Coronel and Morris (2016a, p. 208), describes the steps to normalise database tables.

To conform to the first normal form (1NF), any repeating groups must be eliminated ensuring there is only atomic values for each attribute and to identify the primary key. Converting to the second normal form (2NF), there must be no partial dependencies and the table must be compliant to 1NF. To remove any partial dependencies, a new table should be created with the attribute which is causing the partial dependency. Next, to convert a table to the third normal form (3NF), it must be normalised to 2NF and have no transitive dependencies.

Given attributes A (primary key), B (non-key) and C (non-key), if both B and C depend on A, and B also depends on C, it means B relies on A through C; hence attribute B is transitively dependant on the primary key A. To remove this transitive dependency, a new table is created with C as a primary key and B as a non-key attribute. In the original table, there is a primary key A and a foreign key C.

## Normalised to Third Normal Form

**Tables:** Address, Student, Lease, Adviser, University, Apartment, Inspection, Login, ResidentStaff, HallOfResidents, GeneralDetails, and Room

There are no repeated groups within these tables and all attributes are atomic and cannot be broken down further; hence, it is normalised to the first normal form.

It is also in the second normal form where all the attributes are dependent on a single-column primary key. This also means that it is in the third normal form because all the non-key attributes do not depend on other non-key attributes.

## Invoice Table

The Invoice table has two attributes firstReminder and secondReminder, which both depend on the dateDue and not the primary key. To normalise this to second normal form (2NF) and third normal form (3NF), a new table ‘ReminderDates’ has been created with two attributes firstReminder and secondReminder; therefore, dateDue is only dependant on the primary key.

## Course Table

This table is normalised to the first normal form only. All the course’s details are dependent on the courseNumber and not the courseID. To put this into second and third normal forms, the primary key has been changed from courseID to the courseNumber; therefore, all attributes are only dependent on the primary key. The metadata schema has been updated (See appendix A).

## Normalising Past Third Normal Form

The tables and database could be normalised further. Once normalised to 3NF, the tables are compliant to Boyce-Codd Normal Form (BCNF) where there are no overlapping composite candidate keys.

A table is in the fourth normal form (4NF) where it conforms to BCNF and has no multi-valued dependencies. This prevents unnecessary redundancies and allows the tables to qualify for fifth normal form.

A table is in the fifth normal form (5NF) if it conforms to 4NF and there is no data-loss when tables are decomposed further and joined by a foreign key.

# Report & Form Designs

## Form Design

Field Name (e.g. First Name)

Placeholder (e.g. Required)

Submit Form

Tick the box to….

Each form will follow the same format. A subheading for each field name, a text field to enter the value with a placeholder to notify the user any additional information and a button to submit the form. In some cases, a checkbox may be required for a form.

This design benefits the user by indicating exactly what information is to be entered and the format required. Validation must be used to ensure data integrity.

## Report Design

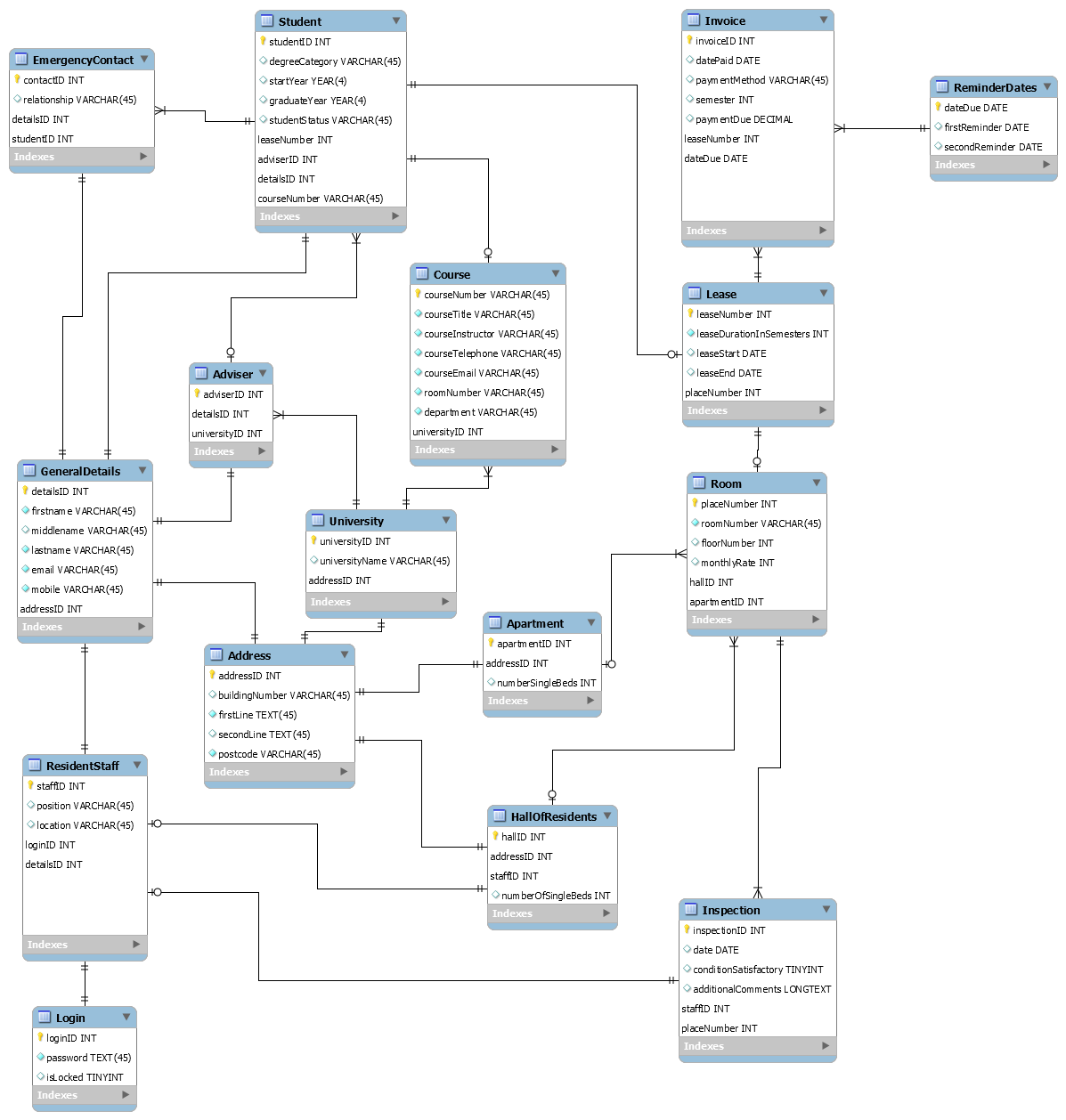
The reports will be produced in tabular form as specified by the requirements. An example is given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Inspection ID | Date | Place Number | Hall ID | Apartment ID |
| 1 | 2019-12-01 | 1 | 1 | N/A |
| 2 | 2019-11-03 | 2 | N/A | 1 |

## Page Wireframes

For each of the system outputs, a basic wireframe has been designed to aid the implementation of the system (See Appendix B).

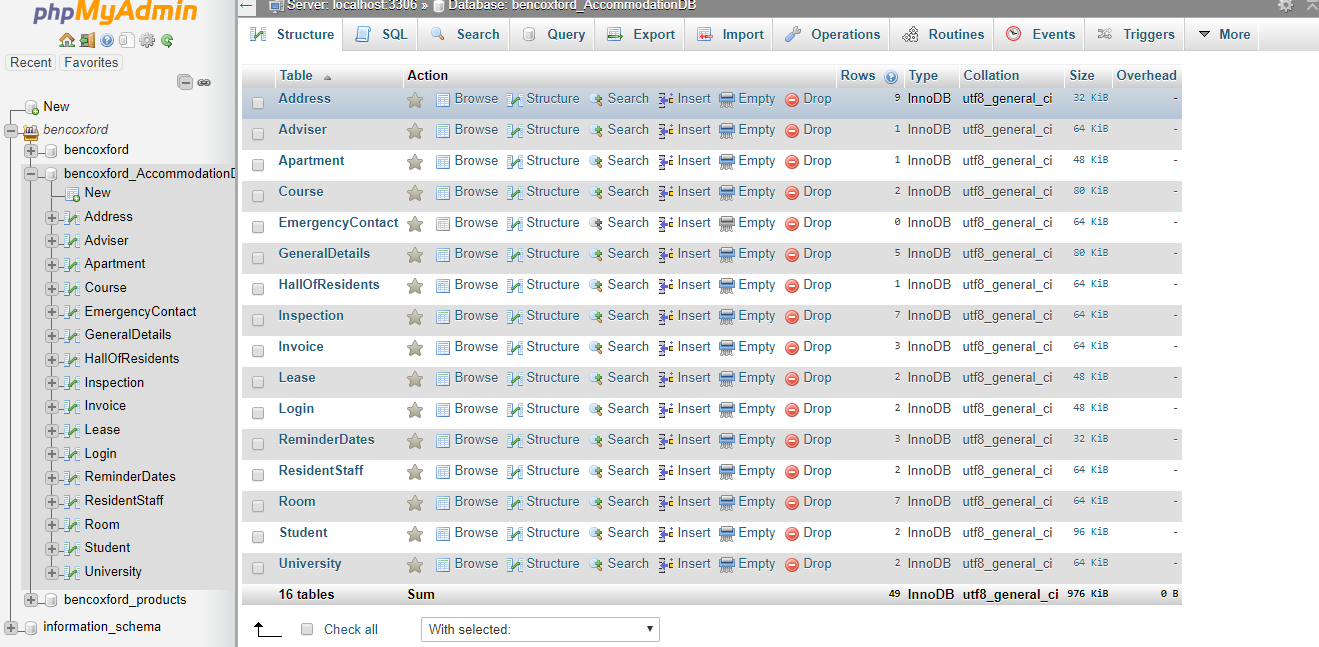
# Entity Relationship Diagram



Produced using Oracles’ (2019), MySQL Workbench software.

# Forward Engineered

The database was successfully forward engineered using MySQL Workbench.



# SQL Scripts

## Login

|  |  |
| --- | --- |
| Inputs | password, firstname, lastname |
| Validations | md5(password), lowercase name |
| Outputs | 1 (If one row, then login successful) |
| Tables | ResidentStaff, General Details, Login |
| SQL Code | SELECT 1 FROM ResidentStaff a INNER JOIN GeneralDetails b ON a.detailsID=b.detailsID INNER JOIN Login c ON a.loginID=c.loginID WHERE b.firstname=firstname AND b.lastname=lastname AND c.password=md5(password) |
| SQL Test |  |

## Change Password

|  |  |
| --- | --- |
| Inputs | old pass, new pass, confirm pass, firstname.lastname (session username) |
| Validations | Length Check (8 Characters), md5(password), check passwords match |
| Outputs | Update password |
| Tables | ResidentStaff, GeneralDetails, Login |
| SQL Code | UPDATE Login a INNER JOIN ResidentStaff b ON a.loginID=b.loginID INNER JOIN GeneralDetails c ON b.detailsID=c.detailsID SET a.password=md5(New Password) WHERE c.firstname=firstname AND c.lastname=lastname AND a.password=md5(Old Password) |
| SQL Test |  |

## Search Person

|  |  |
| --- | --- |
| Inputs | Firstname, Lastname, Email, Mobile |
| Validations | Email Validation, Mobile Validation (UK Format), Filter empty values, Lowercase name |
| Outputs | detailsID, firstname, lastname, email, mobile |
| Tables | GeneralDetails |
| SQL Code | Search Person  SELECT detailsID, firstname, lastname, email, mobile FROM GeneralDetails WHERE firstname=firstname OR lastname=lastname OR email=email OR mobile=mobile |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | detailsID |
| Validations | N/A |
| Outputs | detailsID |
| Tables | Student |
| SQL Code | Student Exist Check  SELECT detailsID FROM Student WHERE detailsID=detailsID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | detailsID |
| Validations | N/A |
| Outputs | firstname, middlename, lastname, email, mobile, buildingNumber, firstLine, secondLine, postcode, studentID, maxMonthly, degreeCategory, startYear, graduateYear, studentStatus, leaseNumber, adviserID, courseNumber |
| Tables | GeneralDetails, Address, Student |
| SQL Code | Return Student Values  SELECT a.firstname, a.middlename, a.lastname, a.email, a.mobile, b.buildingNumber, b.firstLine, b.secondLine, b.postcode, c.studentID, c.maxMonthly, c.degreeCategory, c.startYear, c.graduateYear, c.studentStatus, c.leaseNumber, c.adviserID, c.courseNumber FROM GeneralDetails a INNER JOIN Address b ON a.addressID=b.addressID INNER JOIN Student c ON a.detailsID=c.detailsID WHERE a.detailsID=detailsID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | detailsID |
| Validations | N/A |
| Outputs | detailsID |
| Tables | ResidentStaff |
| SQL Code | Staff Exist Check  SELECT detailsID FROM ResidentStaff WHERE detailsID=detailsID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | detailsID |
| Validations | N/A |
| Outputs | firstname, middlename, lastname, email, mobile, buildingNumber, firstLine, secondLine, postcode, staffID, position, location |
| Tables | GeneralDetails, Address, ResidentStaff |
| SQL Code | Return Staff Values  SELECT a.firstname, a.middlename, a.lastname, a.email, a.mobile, b.buildingNumber, b.firstLine, b.secondLine, b.postcode, c.staffID, c.position, c.location FROM GeneralDetails a INNER JOIN Address b ON a.addressID=b.addressID INNER JOIN ResidentStaff c ON a.detailsID=c.detailsID WHERE a.detailsID=detailsID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | detailsID |
| Validations | N/A |
| Outputs | detailsID |
| Tables | Adviser |
| SQL Code | Adviser Exist Check  SELECT detailsID FROM Adviser WHERE detailsID=detailsID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | detailsID |
| Validations | N/A |
| Outputs | firstname, middlename, lastname, email, mobile, buildingNumber, firstLine, secondLine, postcode, universityID, adviserID |
| Tables | Adviser, GeneralDetails, Address |
| SQL Code | Return Adviser Values  SELECT a.firstname, a.middlename, a.lastname, a.email, a.mobile, b.buildingNumber, b.firstLine, b.secondLine, b.postcode, c.universityID, c.adviserID FROM GeneralDetails a INNER JOIN Address b ON a.addressID=b.addressID INNER JOIN Adviser c ON a.detailsID=c.detailsID WHERE a.detailsID=detailsID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | detailsID |
| Validations | N/A |
| Outputs | detailsID |
| Tables | EmergencyContact |
| SQL Code | EmergencyContact Exist Check  SELECT detailsID FROM EmergencyContact WHERE detailsID=detailsID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | detailsID |
| Validations | N/A |
| Outputs | firstname, middlename, lastname, email, mobile, buildingNumber, firstLine, secondLine, postcode, studentID, relationship, contactID |
| Tables | EmergencyContact, GeneralDetails, Address |
| SQL Code | Return EmergencyContact Values  SELECT a.firstname, a.middlename, a.lastname, a.email, a.mobile, b.buildingNumber, b.firstLine, b.secondLine, b.postcode, c.studentID, c.relationship, c.contactID FROM GeneralDetails a INNER JOIN Address b ON a.addressID=b.addressID INNER JOIN EmergencyContact c ON a.detailsID=c.detailsID WHERE a.detailsID=detailsID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | firstname, middlename, lastname, email, mobile, buildingNumber, firstLine, secondLine, postcode |
| Validations | Email Validation, Mobile Validation (UK Format), Filter empty values, Lowercase name, Postcode Validation |
| Outputs | Update Record |
| Tables | GeneralDetails, Address |
| SQL Code | Update GeneralDetails Values  UPDATE GeneralDetails a INNER JOIN Address b ON a.addressID=b.addressID SET a.firstname=firstname, a.middlename=middlename, a.lastname=lastname, a.email=email, a.mobile=mobile, b.buildingNumber=buildingNumber, b.firstLine=firstLine, b.secondLine=secondLine, b.postcode=postcode WHERE a.detailsID=detailsID; |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | studentID, maxMonthly, degreeCategory, startYear, graduateYear, studentStatus, leaseNumber, adviserID, courseNumber |
| Validations | Integer Check, Date Check |
| Outputs | Update Record |
| Tables | Student |
| SQL Code | Update Student Values  UPDATE Student SET degreeCategory=degreeCategory, maxMonthly=maxMonthly, startYear=startYear, graduateYear=graduateYear, studentStatus=studentStatus, leaseNumber=leaseNumber, courseNumber=courseNumber, adviserID=adviserID WHERE detailsID=detailsID;  WHERE a.detailsID=detailsID; |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | staffID, position, location |
| Validations | N/A |
| Outputs | Update Record |
| Tables | ResidentStaff |
| SQL Code | Update ResidentStaff Values  UPDATE ResidentStaff SET position=position, location=location WHERE detailsID=detailsID; |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | studentID, relationship, contactID |
| Validations | N/A |
| Outputs | Update Record |
| Tables | EmergencyContact |
| SQL Code | Update EmergencyContact Values  UPDATE EmergencyContact SET studentID=studentID, relationship=relationship WHERE detailsID=detailsID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | universityID, adviserID |
| Validations | Integer Check |
| Outputs | Update Record |
| Tables | Adviser |
| SQL Code | Update Adviser Values  UPDATE Adviser SET universityID=universityID WHERE detailsID=detailsID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | studentID |
| Validations | N/A |
| Outputs | Delete Record |
| Tables | EmergencyContact, Student, GeneralDetails, Address |
| SQL Code | Delete Student (Transaction)  DELETE a, b, c FROM EmergencyContact a INNER JOIN GeneralDetails b ON a.detailsID=b.detailsID INNER JOIN Address c ON b.addressID=c.addressID WHERE a.studentID=$id  DELETE a, b, c FROM Student a INNER JOIN GeneralDetails b ON a.detailsID=b.detailsID INNER JOIN Address c ON b.addressID=c.addressID WHERE a.studentID=id |

|  |  |
| --- | --- |
| Inputs | staffID |
| Validations | N/A |
| Outputs | Delete Record |
| Tables | HallOfResidents, Inspection, ResidentStaff, GeneralDetails, Address |
| SQL Code | Delete ResidentStaff (Transaction)  UPDATE HallOfResidents SET staffID = NULL WHERE staffID=id  UPDATE Inspection SET staffID = NULL WHERE staffID=id  DELETE a, b, c, d FROM Login a INNER JOIN ResidentStaff b ON a.loginID=b.loginID INNER JOIN GeneralDetails c ON b.detailsID=c.detailsID INNER JOIN Address d ON c.addressID=d.addressID WHERE b.staffID=id |

|  |  |
| --- | --- |
| Inputs | contactID |
| Validations | N/A |
| Outputs | Delete Record |
| Tables | EmergencyContact, GeneralDetails, Address |
| SQL Code | Delete EmergencyContact  DELETE a, b, c FROM EmergencyContact a INNER JOIN GeneralDetails b ON a.detailsID=b.detailsID INNER JOIN Address c ON b.addressID=c.addressID WHERE a.contactID=id |

|  |  |
| --- | --- |
| Inputs | adviserID |
| Validations | N/A |
| Outputs | Delete Record |
| Tables | Adviser, GeneralDetails, Address, Student |
| SQL Code | Delete Adviser (Transaction)  UPDATE Student SET adviserID = NULL WHERE adviserID=id  DELETE a, b, c FROM Adviser a INNER JOIN GeneralDetails b ON a.detailsID=b.detailsID INNER JOIN Address c ON b.addressID=c.addressID WHERE a.adviserID=id |

## Search Apartment

|  |  |
| --- | --- |
| Inputs | First Line, Postcode, Place Number, Building Number |
| Validations | Integer Checks. |
| Outputs | apartmentID, buildingNumber, firstLine, secondLine, postcode |
| Tables | Apartment, Address, Room |
| SQL Code | Search Apartment  SELECT DISTINCT a.apartmentID, c.buildingNumber, c.firstLine, c.secondLine, c.postcode FROM Apartment a INNER JOIN Room b ON a.apartmentID=b.apartmentID INNER JOIN Address c ON a.addressID=c.addressID WHERE b.placeNumber=placeNumber OR c.buildingNumber=buildingNumber OR c.postcode=postcode OR c.firstLine=firstLine  b.addressID=c.addressID WHERE a.adviserID=id |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | apartmentID |
| Validations | Integer Check. |
| Outputs | apartmentID, buildingNumber, firstLine, secondLine, postcode, numberOfSingleBeds |
| Tables | Apartment, Address |
| SQL Code | Return Apartment Values  SELECT a.apartmentID, a.numberSingleBeds, b.buildingNumber, b.firstLine, b.secondLine, b.postcode FROM Apartment a INNER JOIN Address b ON a.addressID=b.addressID WHERE a.apartmentID=apartmentID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | apartmentID |
| Validations | Integer Checks. |
| Outputs | Delete Apartment |
| Tables | Apartment, Room |
| SQL Code | Delete Apartment  UPDATE Room SET apartmentID = NULL WHERE apartmentID=id  DELETE FROM Apartment WHERE apartmentID=id |

## Search Hall of Residents

|  |  |
| --- | --- |
| Inputs | placeNumber, buildingNumber, firstLine, postcode |
| Validations | Postcode Validation, Integer Validation |
| Outputs | hallID, buildingNumber, firstLine, secondLine, postcode |
| Tables | Room, HallOfResidents, Address |
| SQL Code | Search Hall of Residents  SELECT DISTINCT b.hallID, c.buildingNumber, c.firstLine, c.secondLine, c.postcode FROM Room a INNER JOIN HallOfResidents b ON a.hallID=b.hallID INNER JOIN Address c ON b.addressID=c.addressID WHERE a.placeNumber=placeNumber OR c.buildingNumber=buildingNumber OR c.postcode=postcode OR c.firstLine=firstLine |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | hallID |
| Validations | N/A |
| Outputs | hallID, buildingNumber, firstLine, secondLine, postcode, numberOfSingleBeds |
| Tables | HallOfResidents, Address |
| SQL Code | Return HallOfResidents Values  SELECT a.hallID, a.numberSingleBeds, b.buildingNumber, b.firstLine, b.secondLine, b.postcode FROM HallOfResidents a INNER JOIN Address b ON a.addressID=b.addressID WHERE a.hallID=hallID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | buildingNumber, firstLine, secondLine, postcode, numberOfSingleBeds |
| Validations | Postcode Validation, Integer Validation |
| Outputs | Update Values |
| Tables | HallOfResidents, Address |
| SQL Code | Update HallOfResidents Values  UPDATE HallOfResidents a INNER JOIN Address b ON a.addressID=b.addressID SET b.buildingNumber=buildingNumber, b.firstLine=firstLine, b.secondLine=secondLine, b.postcode=postcode, a.numberSingleBeds=numberSingleBeds WHERE a.hallID=hallID; |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | hallID |
| Validations | N/A |
| Outputs | Delete Record |
| Tables | Room, HallOfResidents |
| SQL Code | Delete HallOfResidents (Transaction)  UPDATE Room SET hallID = NULL WHERE hallID=id  DELETE FROM HallOfResidents WHERE hallID=id |

## Search Room

|  |  |
| --- | --- |
| Inputs | placeNumber, firstname, lastname |
| Validations | Integer Check |
| Outputs | Room details (placeNumber, roomNumber, floorNumber, monthlyRate) and if it exists, the occupants’ details (firstname, lastname). |
| Tables | Room, Lease, Student, GeneralDetails |
| SQL Code | Search  IF EXISTS (SELECT 1 FROM Room a INNER JOIN Lease b ON a.placeNumber=b.placeNumber INNER JOIN Student c ON c.leaseNumber=b.placeNumber INNER JOIN GeneralDetails d ON d.detailsID=c.detailsID WHERE a.placeNumber=placeNumber)  THEN  SELECT a.placeNumber, a.roomNumber, a.floorNumber, a.monthlyRate, d.firstname, d.lastname FROM Room a INNER JOIN Lease b ON a.placeNumber=b.placeNumber INNER JOIN Student c ON c.leaseNumber=b.placeNumber INNER JOIN GeneralDetails d ON d.detailsID=c.detailsID WHERE a.placeNumber=placeNumber LIMIT 1;  ELSE  SELECT placeNumber, roomNumber, floorNumber, monthlyRate FROM Room WHERE placeNumber=placeNumber LIMIT 1;  END IF; |
| SQL Test | N/A |

|  |  |
| --- | --- |
| Inputs | placeNumber |
| Validations | Integer Check |
| Outputs | placeNumber, roomNumber, floorNumber, monthlyRate, hallID, apartmentID |
| Tables | Room |
| SQL Code | Return Rooms Values  SELECT placeNumber, roomNumber, floorNumber, monthlyRate, hallID, apartmentID FROM Room WHERE placeNumber=placeNumber |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | placeNumber, roomNumber, floorNumber, monthlyRate, hallID, apartmentID |
| Validations | Integer Check |
| Outputs | Update Record |
| Tables | Room |
| SQL Code | Update Room Values  UPDATE Room SET roomNumber=roomNumber, floorNumber=floorNumber, monthlyRate=monthlyRate, hallID=hallID, apartmentID=apartmentID WHERE placeNumber=placeNumber |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | placeNumber |
| Validations | Integer Check |
| Outputs | Delete Record |
| Tables | Room, Lease, Inspection |
| SQL Code | Delete Room (Transaction)  UPDATE Lease SET placeNumber = NULL WHERE placeNumber=id  DELETE FROM Inspection WHERE placeNumber=id  DELETE FROM Room WHERE placeNumber=id |

## Search University

|  |  |
| --- | --- |
| Inputs | universityID, buildingNumber, firstLine, postcode |
| Validations | Integer Check, Postcode Validation |
| Outputs | universityID, buildingNumber, secondLine, postcode |
| Tables | University, Address |
| SQL Code | Search  SELECT b.universityID, c.buildingNumber, c.firstLine, c.secondLine, c.postcode FROM University b INNER JOIN Address c ON b.addressID=c.addressID WHERE b.universityID=universityID OR c.buildingNumber=buildingNumber OR postcode=postcode OR firstLine=firstLine |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | universityID |
| Validations | N/A |
| Outputs | a.universityID, b.buildingNumber, b.firstLine, b.secondLine, b.postcode |
| Tables | University, Address |
| SQL Code | Return University Values  SELECT a.universityID, b.buildingNumber, b.firstLine, b.secondLine, b.postcode FROM University a INNER JOIN Address b ON a.addressID=b.addressID WHERE a.universityID=universityID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | universityID |
| Validations | Integer Check, Postcode Validation |
| Outputs | Update Records |
| Tables | University, Address |
| SQL Code | Update University Values  UPDATE University a INNER JOIN Address b ON a.addressID=b.addressID SET b.buildingNumber=buildingNumber, b.firstLine=firstLine, b.secondLine=secondLine, b.postcode=postcode WHERE a.universityID=universityID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | universityID |
| Validations | N/A |
| Outputs | Delete Record |
| Tables | University, Address, Course |
| SQL Code | Delete University Record (Transaction)  DELETE FROM Course WHERE universityID=id  DELETE a, b FROM University a INNER JOIN Address b ON a.addressID=b.addressID WHERE a.universityID=id |

## Search Lease

|  |  |
| --- | --- |
| Inputs | leaseNumber, leaseStart, leaseEnd, leaseDurationInSemesters |
| Validations | Integer Check, Date Validation |
| Outputs | leaseNumber, leaseDurationInSemesters, leaseStart, leaseEnd, placeNumber |
| Tables | Lease |
| SQL Code | Search Lease  SELECT DISTINCT leaseNumber, leaseDurationInSemesters, leaseStart, leaseEnd, placeNumber FROM Lease WHERE leaseNumber=leaseNumber OR leaseDurationInSemesters=leaseDurationInSemesters OR (leaseStart BETWEEN leaseStart AND leaseEnd OR leaseEnd BETWEEN leaseStart AND leaseEnd) |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | leaseNumber |
| Validations | N/A |
| Outputs | leaseNumber, leaseDurationInSemesters, leaseStart, leaseEnd, placeNumber |
| Tables | Lease |
| SQL Code | Return Lease Details  SELECT leaseNumber, leaseDurationInSemesters, leaseStart, leaseEnd, placeNumber FROM Lease WHERE leaseNumber=leaseNumber |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | leaseNumber, leaseDurationInSemesters, leaseStart, leaseEnd, placeNumber |
| Validations | Integer Check, Date Validation |
| Outputs | Update Record |
| Tables | Lease |
| SQL Code | Update Lease Record  UPDATE Lease SET leaseDurationInSemesters=leaseDurationInSemesters, leaseStart=leaseStart, leaseEnd=leaseEnd, placeNumber=placeNumber WHERE leaseNumber=leaseNumber |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | leaseID |
| Validations | N/A |
| Outputs | Delete Record |
| Tables | Student, Invoice, Lease |
| SQL Code | Delete lease records  UPDATE Student SET leaseID = NULL WHERE leaseID=id  UPDATE Invoice SET leaseID = NULL WHERE leaseID=id  DELETE FROM Lease WHERE leaseID=id |

## Search Invoice

|  |  |
| --- | --- |
| Inputs | leaseNumber, semester, paymentMethod, dateDue, datePaid |
| Validations | Integer Check, Date Check |
| Outputs | InvoiceID, leaseNumber, semester, dateDue |
| Tables | Invoice |
| SQL Code | Search Invoice  SELECT DISTINCT invoiceID, leaseNumber, semester, dateDue FROM Invoice WHERE leaseNumber=leaseNumber OR semester=semester OR paymentMethod=paymentMethod OR dateDue=dateDue OR datePaid=datePaid |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | invoiceID |
| Validations | N/A |
| Outputs | invoiceID, semester, dateDue, datePaid, firstReminder, secondReminder, paymentMethod, paymentDue |
| Tables | Invoice, ReminderDates |
| SQL Code | Return Invoice Values  SELECT a.invoiceID, a.semester, a.dateDue, a.datePaid, b.firstReminder, b.secondReminder, a.paymentMethod, a.paymentDue FROM Invoice a INNER JOIN ReminderDates b ON a.dateDue=b.dateDue WHERE invoiceID=invoiceID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | dateDue, invoiceID, datePaid, paymentMethod, semester, paymentDue |
| Validations | N/A |
| Outputs | Update Records |
| Tables | Invoice, ReminderDates |
| SQL Code | Update Invoice  UPDATE Invoice SET datePaid=datePaid, dateDue=dateDue, paymentMethod=paymentMethod, semester=semester, paymentDue=paymentDue WHERE invoiceID=invoiceID  UPDATE ReminderDates SET firstReminder=firstReminder, secondReminder=secondReminder WHERE dateDue=dateDue |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | invoiceID |
| Validations | N/A |
| Outputs | Delete Record |
| Tables | Invoice |
| SQL Code | Delete Record  DELETE FROM Invoice WHERE invoiceID=id |

## Search Inspection

|  |  |
| --- | --- |
| Inputs | Date, placeNumber, conditionSatisfactory |
| Validations | Date Validation, Integer Check |
| Outputs | inspectionID, date, conditionSatisfactory, additionalComments, staffID, placeNumber |
| Tables | Inspection |
| SQL Code | Search Inspection  SELECT DISTINCT inspectionID, date, conditionSatisfactory, additionalComments, staffID, placeNumber FROM Inspection WHERE date=date OR placeNumber=placeNumber OR conditionSatisfactory IS NULL |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | conditionSatisfactory, additionalComments, staffID, inspectionID |
| Validations | Integer Check |
| Outputs | Update Record |
| Tables | Inspection |
| SQL Code | Update Inspection Values  UPDATE Inspection SET conditionSatisfactory=conditionSatisfactory, additionalComments=additionalComments, staffID=staffID WHERE inspectionID=logID |
| SQL Test |  |

## Search Course

|  |  |
| --- | --- |
| Inputs | universityID, courseNumber |
| Validations | Integer Check |
| Outputs | All Attributes |
| Tables | Course |
| SQL Code | Search & Return Course Values  SELECT DISTINCT \* FROM Course WHERE courseNumber=courseNumber OR universityID=universityID |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | courseNumber, courseTitle, courseInstructor, courseTelephone, ccourseEmail, roomNumber, department, universityID |
| Validations | Mobile Validation, Email Validation, Integer check |
| Outputs | Update Record |
| Tables | Course |
| SQL Code | Update Record  UPDATE Course SET courseTitle=courseTitle, courseInstructor=courseInstructor, courseTelephone=courseTelephone, courseEmail=courseEmail, roomNumber= roomNumber, department=department, universityID=universityID WHERE courseNumber=courseNumber |
| SQL Test |  |

|  |  |
| --- | --- |
| Inputs | courseNumber |
| Validations | N/A |
| Outputs | Delete Record |
| Tables | Student, Course |
| SQL Code | Delete Course  UPDATE Student SET courseNumber = NULL WHERE courseNumber=id  DELETE FROM Course WHERE courseNumber=id |

## Room Vacancy Report

|  |  |
| --- | --- |
| Inputs | N/A, hallID/apartmentID |
| Validations | Integer Check |
| Outputs | placeNumber, roomNumber, floorNumber, monthlyRate, hallID, apartmentID |
| Tables | Room, Lease |
| SQL Code | Search all vacant rooms.  SELECT placeNumber, roomNumber, floorNumber, monthlyRate, hallID, apartmentID FROM Room WHERE placeNumber NOT IN (SELECT a.placeNumber FROM Lease a)  SELECT placeNumber, roomNumber, floorNumber, monthlyRate, hallID, apartmentID FROM Room WHERE placeNumber NOT IN (SELECT a.placeNumber FROM Lease a) AND (hallID=hallID OR apartmentID=apartmentID) |
| SQL Test |  |

## Waiting Student Report

|  |  |
| --- | --- |
| Inputs | N/A |
| Validations | N/A |
| Outputs | studentID, studentStatus, maxMonthly, firstname, lastname, email, mobile |
| Tables | GeneralDetails, Student |
| SQL Code | Search all students with no lease.  SELECT a.studentID, a.studentStatus, a.maxMonthly, b.firstname, b.lastname, b.email, b.mobile FROM Student a INNER JOIN GeneralDetails b ON a.detailsID=b.detailsID WHERE a.leaseNumber IS NULL |
| SQL Test |  |

## Unpaid Invoice Report

|  |  |
| --- | --- |
| Inputs | Semester |
| Validations | Integer Check |
| Outputs | invoiceID, paymentDue, dateDue, datePaid, semester, leaseNumber, firstname, lastname, email, mobile |
| Tables | Lease, Invoice, Student, GeneralDetails |
| SQL Code | Produce report on all unpaid invoices.  SELECT a.invoiceID, a.paymentDue, a.dateDue, a.datePaid, a.semester, a.leaseNumber, d.firstname, d.lastname, d.email, d.mobile FROM Invoice a INNER JOIN Lease b ON a.leaseNumber=b.leaseNumber INNER JOIN Student c ON b.leaseNumber=c.leaseNumber INNER JOIN GeneralDetails d ON c.detailsID=d.detailsID WHERE a.datePaid IS NULL AND a.semester=semester |
| SQL Test |  |

## Inspection Damage Report

|  |  |
| --- | --- |
| Inputs | dateTo, dateFrom |
| Validations | Date Validation |
| Outputs | inspectionID, date, additionalComments, staffID and placeNumber |
| Tables | Inspection |
| SQL Code | Produce report on all damaged inspections.  SELECT a.inspectionID, a.date, a.additionalComments, a.staffID, a.placeNumber FROM Inspection a WHERE a.date < dateTo AND a.date > dateFrom AND a.conditionSatisfactory=0 |
| SQL Test |  |

## Planned Inspection Report

|  |  |
| --- | --- |
| Inputs | N/A |
| Validations | N/A |
| Outputs | inspectionID, placeNumber, apartmentID, hallID, date |
| Tables | Inspection, Room |
| SQL Code | Produce report on all planned inspections.  SELECT a.inspectionID, b.placeNumber, b.apartmentID, b.hallID, a.date FROM Inspection a INNER JOIN Room b ON a.placeNumber=b.placeNumber WHERE conditionSatisfactory IS NULL |
| SQL Test |  |

## Occupancy Report

|  |  |
| --- | --- |
| Inputs | N/A |
| Validations | N/A |
| Outputs | placeNumber, roomNumber, floorNumber, monthlyRate, hallID, apartmentID, firstname, lastname, email, mobile |
| Tables | Room, Lease, Student, GeneralDetails |
| SQL Code | Produce all rooms with an occupant.  SELECT a.placeNumber, a.roomNumber, a.floorNumber, a.monthlyRate, a.hallID, a.apartmentID, d.firstname, d.lastname, d.email, d.mobile FROM Room a INNER JOIN Lease b ON a.placeNumber=b.placeNumber INNER JOIN Student c ON b.leaseNumber=c.leaseNumber INNER JOIN GeneralDetails d ON c.detailsID=d.detailsID |
| SQL Test |  |

## Income Report

|  |  |
| --- | --- |
| Inputs | dateTo, dateFrom |
| Validations | Date Validation |
| Outputs | leaseNumber, paymentDue, datePaid, paymentMethod, firstname, lastname |
| Tables | Invoice, Lease, Student, GeneralDetails |
| SQL Code | Produce income report of all paid invoices between two dates.  SELECT a.leaseNumber, a.paymentDue, a.datePaid, a.paymentMethod, d.firstname, d.lastname FROM Invoice a INNER JOIN Lease b ON a.leaseNumber=b.leaseNumber INNER JOIN Student c ON b.leaseNumber=c.leaseNumber INNER JOIN GeneralDetails d ON c.detailsID=d.detailsID WHERE a.datePaid < dateTo AND a.datePaid > dateFrom |
| SQL Test |  |

## Unknown Leaving Date Report

|  |  |
| --- | --- |
| Inputs | N/A |
| Validations | N/A |
| Outputs | placeNumber, firstname, lastname, email, mobile |
| Tables | Student, Lease, GeneralDetails |
| SQL Code | Produce report on all records with a NULL leaseEnd.  SELECT c.placeNumber, b.firstname, b.lastname, b.email, b.mobile FROM Student a INNER JOIN GeneralDetails b ON a.detailsID=b.detailsID INNER JOIN Lease c ON a.leaseNumber=c.leaseNumber WHERE c.leaseEnd IS NULL |
| SQL Test |  |

## Create Lease

|  |  |
| --- | --- |
| Inputs | leaseDurationInSemesters, leaseStart, leaseEnd, placeNumber |
| Validations | Integer Check, Date Validation |
| Outputs | Inserts a new record into the table |
| Tables | Lease |
| SQL Code | Create a new lease  INSERT INTO Lease (leaseNumber, leaseDurationInSemesters, leaseStart, leaseEnd, placeNumber) VALUES (NULL, leaseDurationInSemesters,leaseStart,leaseEnd,placeNumber) |
| SQL Test |  |

## Create Inspection

|  |  |
| --- | --- |
| Inputs | Date, NumberOfDays |
| Validations | Integer Check, Date Validation |
| Outputs | Creates a new inspection for each room. |
| Tables | Room, Inspection |
| SQL Code | For each room, create inspection for each split between the duration of inspection.  SELECT placeNumber FROM Room  INSERT INTO Inspection (inspectionID, date, placeNumber) VALUES (NULL, dates[i], placeNumbers[z]) |

## Create Staff

|  |  |
| --- | --- |
| Inputs | Firstname, middlename, lastname, email, mobile, buildingNumber, firstLine, secondLine, postcode, buildingNumber, firstLine, secondLine, password, passwordConfirm, position, location. |
| Validations | Md5(password), Postcode Validation, Mobile Validation, Integer Check, Length Check (8 Characters) |
| Outputs | Creates a new staff member. |
| Tables | Address, GeneralDetails, Login, ResidentStaff |
| SQL Code | Create new staff member (Transaction)  INSERT INTO Address (addressID, buildingNumber, firstLine, secondLine, postcode) VALUES (NULL, buildingNumber, firstLine, secondLine, postcode)  INSERT INTO GeneralDetails (detailsID, firstname, middlename, lastname, email, mobile, addressID) VALUES (NULL, firstname, middlename, lastname, email, mobile, (SELECT addressID FROM Address WHERE buildingNumber=buildingNumber AND firstLine=firstLine AND postcode=postcode))  INSERT INTO Login (loginID, password, isLocked) VALUES (NULL, pass, 0)    INSERT INTO ResidentStaff (staffID, position, location, loginID, detailsID) VALUES (NULL, position, location, (SELECT loginID FROM Login WHERE password=pass), (SELECT detailsID FROM GeneralDetails WHERE email=email AND mobile=mobile)) |

## Create Apartment

|  |  |
| --- | --- |
| Inputs | buildingNumber, firstLine, secondLine, postcode, addressID, numberSingleBeds |
| Validations | Postcode Validation, Integer check |
| Outputs | Creates a new apartment record |
| Tables | Apartment, Address |
| SQL Code | Create new apartment (Transaction)  INSERT INTO Address (addressID, buildingNumber, firstLine, secondLine, postcode) VALUES (NULL, buildingNumber, firstLine, secondLine, postcode)    INSERT INTO Apartment (apartmentID, addressID, numberSingleBeds) VALUES (NULL, (SELECT addressID FROM Address WHERE firstLine=firstLine AND buildingNumber=buildingNumber), numberSingleBeds) |

## Create Hall of Residents

|  |  |
| --- | --- |
| Inputs | buildingNumber, firstLine, secondLine, postcode, numberSingleBeds, staffID |
| Validations | Postcode Validation |
| Outputs | Creates a new hall of residents |
| Tables | HallOfResidents, Address |
| SQL Code | Create a new hall of residents (Transaction)  INSERT INTO Address (addressID, buildingNumber, firstLine, secondLine, postcode) VALUES (NULL, buildingNumber, firstLine, secondLine, postcode)    INSERT INTO HallOfResidents (hallID, addressID, numberSingleBeds, staffID) VALUES (NULL, (SELECT addressID FROM Address WHERE firstLine=firstLine AND buildingNumber=buildingNumber), numberSingleBeds, staffID) |

## Create University

|  |  |
| --- | --- |
| Inputs | buildingNumber, firstLine, secondLine, postcode, universityName |
| Validations | Integer Check, Postcode Validation |
| Outputs | Creates a new university |
| Tables | University, Address |
| SQL Code | Creates a new university (Transaction)  INSERT INTO Address (addressID, buildingNumber, firstLine, secondLine, postcode) VALUES (NULL, buildingNumber, firstLine, secondLine, postcode)    INSERT INTO University (universityID, universityName, addressID) VALUES (NULL, universityName, (SELECT addressID FROM Address WHERE buildingNumber=buildingNumber AND firstLine=firstLine AND postcode=postcode)) |

## Create Course

|  |  |
| --- | --- |
| Inputs | courseNumber, courseTitle, courseInstructor, ccourseTelephone, ccourseEmail, roomNumber, department, universityID |
| Validations | Mobile Validation, Email Validation, Integer Check |
| Outputs | Creates a new course |
| Tables | Course |
| SQL Code | Creates a new course  INSERT INTO Course (courseNumber, courseTitle, courseInstructor, courseTelephone, courseEmail, roomNumber, department, universityID) VALUES (courseNumber, courseTitle, courseInstructor, courseTelephone, courseEmail, roomNumber, department, universityID) |

## Create Room

|  |  |
| --- | --- |
| Inputs | roomNumber, floorNumber, monthlyRate, apartmentID, hallID |
| Validations | Integer Check |
| Outputs | Creates a new room |
| Tables | Room |
| SQL Code | Creates a new room  INSERT INTO Room (placeNumber, roomNumber, floorNumber, monthlyRate, apartmentID, hallID) VALUES (NULL, roomNumber, floorNumber, monthlyRate, apartmentID, hallID); |

## Create Invoice

|  |  |
| --- | --- |
| Inputs | dateDue, semester, paymentDue, leaseNumber, firstReminder, secondReminder |
| Validations | Date Validation, Integer Check |
| Outputs | Creates a new invoice |
| Tables | Invoice, ReminderDates |
| SQL Code | Creates a new invoice  INSERT INTO Invoice (invoiceID, dateDue, semester, paymentDue, leaseNumber) VALUES (NULL, dateDue, semester, paymentDue, leaseNumber)    INSERT INTO ReminderDates (dateDue, firstReminder, secondReminder) VALUES (dateDue, firstReminder, secondReminder) |

## Create Adviser

|  |  |
| --- | --- |
| Inputs | buildingNumber, firstLine, secondLine, postcode, firstname, middlename, lastname, email, mobile, universityID |
| Validations | Postcode Validation, Email Validation, Mobile Validation, Integer Check |
| Outputs | Creates a new adviser |
| Tables | Address, General Details, Adviser |
| SQL Code | Creates a new adviser  INSERT INTO Address (addressID, buildingNumber, firstLine, secondLine, postcode) VALUES (NULL, buildingNumber, firstLine, secondLine, postcode)    INSERT INTO GeneralDetails (detailsID, firstname, middlename, lastname, email, mobile, addressID) VALUES (NULL, firstname, middlename, lastname, email, mobile, (SELECT addressID FROM Address WHERE buildingNumber=buildingNumber AND firstLine=firstLine AND postcode=postcode))    INSERT INTO Adviser (adviserID, detailsID, universityID) VALUES (NULL, (SELECT detailsID FROM GeneralDetails WHERE email=email AND mobile=mobile), universityID); |

# MariaDB Transactions (ACID)

Multiple form designs require a transaction where there is a sequence of SQL statements that must be committed. As MariaDB (2018) explains, it supports ACID-compliant transactions to ensure data integrity. Transactions follow four principles:

* **Atomicity** – the entire transaction has been completed in full, otherwise, the entire transaction is aborted. This is a vital principle in the systems development. For instance, if a record is deleted from one table and then from an adjoining table (through an identifying-relationship) but one statement fails, we would want to roll back to the previous state and abort the transaction. Therefore, this ensures the referential integrity of the database.
* **Consistency** – only valid data can be written to the database. Again, if the transaction violates the consistency of a database, the database is rolled back to its original state.
* **Isolation** – any transactions occurring at the same time cannot use the data being used by another transaction. The second transaction must wait for the first to be completed.
* **Durability** – any transactions committed will not be lost, even after a system crash. A backup will restore the database to a consistent state before the transaction executes.

# Search and Report Test Data

Website Link: <http://ysjcs.net/~ben.coxford/2CB104/Assesment/Website/Login.php>

Test Username: james.east

Test Password: thisismypassword

Test data is provided to test the search and report pages.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Page |  |  |  |  |
| Person Search | First Name: James | Last Name: East | Email: may.last@example.com | Mobile: 07123456787 |
| Apartment Search | Place Number: 2 | Building Number: 4 |  |  |
| Hall of Resident Search | Place Number: 1 | Building Number: 3 |  |  |
| Room Search | Place Number: 5 | Full name: louise southgate |  |  |
| University Search | University Id: 1 | Building Number: 3 | Postcode: YOTKA6 |  |
| Lease Search | Lease Number: 1 | Duration: 3 | Start Date:  2019-03-01 | End Date:  2028-09-09 |
| Invoice Search | Lease Number: 1 | Payment Method: visa debit | Semester: 3 | Date Due: 2019-09-30 |
| Inspection Search | Place Number: 1 | Date:  2020-01-01 |  |  |
| Course Search | University Id: 1 | Course Number: 2CB304 |  |  |
| Inspection Damage Report | Date From: 2014-01-01 | Date To:  2022-01-01 |  |  |
| Income Report | Date From: 2014-01-01 | Date To:  2022-01-01 |  |  |

# How could the system be improved?

The interface could have been improved by using object-oriented programming, creating each page as its own object. There are three types of pages – report, search and view. Each could have been programmed as a class, creating objects for each page.

The security of the system could be improved further adding extra password protection such as requiring an alphanumerical password with more than one capitalised letter. The password for the database should be encrypted in a config file.

Emails could be sent automatically on the reminder dates. This could be set up using phpMyAdmin, apache server and PHP.

Furthermore, a NoSQL database could be implemented instead of an SQL database. NoSQL stores the data in a JSON format with a key-value pair without structure. The advantage of this is storing large amounts of data which doesn’t limit the data types. This is advantageous for businesses with changing needs. More importantly, it can be stored on cloud-based computers which have the advantage of being accessible anywhere an using internet connection and saves money as cloud-storage services are based on demand. The disadvantage of cloud storage is if the bandwidth allowance is breached, it could cost more. This is an alternative solution, however, for a small city-based accommodation team, it would not be necessary.

# Appendices

## Appendix A – Metadata Schema

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Field Name | Table | Field Type | Data Type | Can be Null | Is Unique? | Notes | Purpose | Storage Duration |
| studentID | Student | PK | INT | FALSE | TRUE |  | Uniquely identify a student. | The length of time the resident staff give the students to renew their lease. Otherwise this data should be deleted once the students lease has ended. |
| Degree-Category | Student | Data | VARCHAR(45) | TRUE | FALSE | i.e. Undergraduate | Type of degree |  |
| startYear | Student | Data | YEAR(4) | TRUE | FALSE |  | Students starting year |  |
| maxMonthly | Student | Data | DECIMAL (10,0) | FALSE | FALSE |  | Identify the maximum amount a student is willing to pay monthly. Aids staff when assigning rooms/leases to students if they are on the waiting list. |  |
| graduateYear | Student | Data | YEAR(4) | TRUE | FALSE |  | Students graduating year (final year of accommodation) |  |
| Student-Status | Student | Data | VARCHAR(45) | TRUE | FALSE | i.e. Accepted/Pending/Denied |  |  |
| Lease-Number | Student | FK | INT | TRUE | TRUE |  |  |  |
| adviserID | Student | FK | INT | TRUE | TRUE |  |  |  |
| detailsID | Student | FK | INT | FALSE | TRUE |  |  |  |
| courseID | Student | FK | INT | TRUE | TRUE |  |  |  |
| StaffID | ResidentStaff | PK | INT | FALSE | TRUE |  | Uniquely identifies resident staff | Kept until a staff member is no longer within the business. |
| position | ResidentStaff | Data | VARCHAR(45) | TRUE | FALSE | i.e. Hall Manager | Identifies the position at the residential office. |
| location | ResidentStaff | Data | VARCHAR(45) | TRUE | FALSE | i.e. Residential Office | Identifies the location of the staff member. |
| loginID | ResidentStaff | FK | INT | FALSE | TRUE |  |  |  |
| detailsID | ResidentStaff | FK | INT | FALSE | TRUE |  |  |  |
| adviserID | Adviser | PK | INT | FALSE | TRUE |  | Uniquely identifies an adviser. |  |
| detailsID | Adviser | FK | INT | FALSE | TRUE |  |  |  |
| universityID | Adviser | FK | INT | FALSE | TRUE |  |  |  |
| addressID | Address | PK | INT | FALSE | TRUE |  | Uniquely identifies an address. | Kept until the individual’s details are deleted. i.e. Student leaves university, staff member contract terminated, adviser is no longer with the university, emergency contact details are no longer needed. |
| buildingNumber | Address | Data | VARCHAR(45) | FALSE | FALSE |  | Allows residential staff the ability to send letters. This could be a requested invoice or a personal data request for example. |
| First-Line | Address | Data | VARCHAR(100) | FALSE | FALSE |  |
| Second-Line | Address | Data | VARCHAR(100) | TRUE | FALSE |  |
| postcode | Address | Data | VARCHAR(8) | FALSE | TRUE |  |
| invoiceID | Invoice | PK | INT | FALSE | TRUE |  | Uniquely identifies an invoice | Can be stored indefinitely as financial records for the business. Additionally, gov.uk states that invoice records must be stored for 6 years from the end of the last financial year. |
| datePaid | Invoice | Data | DATE | TRUE | FALSE |  | States the date the invoice was paid |
| paymentDue | Invoice | Data | DECIMAL | FALSE | FALSE |  | States the amount due |
| dateDue | Invoice | Data | DATE | TRUE | FALSE |  | States the date the payment is due. |
| paymentMethod | Invoice | Data | VARCHAR(45) | TRUE | FALSE | i.e. VISA Debit | States the payment type made. |
| First-Reminder | Invoice | Data | DATE | TRUE | FALSE |  | States the date the first reminder was sent |
| Second-Reminder | Invoice | Data | DATE | TRUE | FALSE |  | States the date the second reminder was sent |
| Current-Semester | Invoice | Data | INT | TRUE | FALSE | i.e. 1/2/3 (Summer) | States the semester numbers for the invoice |
| Lease-Number | Invoice | FK | INT | FALSE | TRUE |  | Identifies the lease the invoice belongs to. |  |
| Lease-Number | Lease | PK | INT | FALSE | TRUE |  | Uniquely identifies a lease. | As the invoice belongs to a lease, this requires the lease to be stored for as long as the invoices. |
| Lease-Duration-In-Semester | Lease | Data | INT | FALSE | FALSE | i.e. 1/2/3 (Summer) | Duration of the lease i.e. 2 semesters |
| Lease-Start | Lease | Data | DATE | FALSE | FALSE |  | Lease end date |
| leaseEnd | Lease | Data | DATE | TRUE | FALSE |  | Lease start date |
| Place-Number | Lease | FK | INT | FALSE | TRUE |  | Place number the lease attaches to. |
| Apartment-ID | Apartment | PK | INT | FALSE | TRUE |  | Uniquely identifes an apartment. |  |
| Number-Of-Single-Beds | Apartment | Data | INT | FALSE | FALSE |  | States amount of single beds in an apartment. |  |
| addressID | Apartment | FK | INT | FALSE | TRUE |  |  |  |
| hallID | HallOfResidents | PK | INT | FALSE | TRUE |  | Uniquely identifies a hall of residents |  |
| Number-of-Single-Beds | HallOfResidents | Data | INT | TRUE | FALSE |  | States amount of single beds in an apartment. |  |
| addressID | HallOfResidents | FK | INT | FALSE | TRUE |  |  |  |
| staffID | HallOfResidents | FK | INT | TRUE | TRUE |  |  |  |
| detailsID | GeneralDetails | PK | INT | FALSE | TRUE |  | Uniquely identifies the general detail record | Kept until the individual’s details are deleted. i.e. Student leaves university, staff member contract terminated, adviser is no longer with the university, emergency contact details are no longer needed. |
| firstname | GeneralDetails | Data | VARCHAR(45) | FALSE | FALSE |  | First name of individual |
| Middle-name | GeneralDetails | Data | VARCHAR(45) | TRUE | FALSE |  | Middle name of the individual |
| lastname | GeneralDetails | Data | VARCHAR(45) | FALSE | FALSE |  | Last name of the individual |
| email | GeneralDetails | Data | VARCHAR(45) | FALSE | TRUE |  | Email used by resident staff to email and contact an individual. |
| mobile | GeneralDetails | Data | VARCHAR(45) | FALSE | TRUE |  | Allows resident staff to phone an individual and contact them. Critical for emergency contacts/next of kin. |
| addressID | GeneralDetails | FK | INT | FALSE | TRUE |  |  |  |
| universityID | University | PK | INT | FALSE | TRUE |  | Uniquely identifies a university. | N/A |
| universityName | University | Data | VARCHAR(45) | TRUE | TRUE |  | Universities name |  |
| addressID | University | FK | INT | FALSE | TRUE |  | Identifies which address belongs to the university. |  |
| Course-Number | Course | Data | VARCHAR(45) | FALSE | TRUE |  | Uniquely identifies a course | Until a course doesn’t exist. |
| Course-Title | Course | Data | VARCHAR(45) | FALSE | FALSE |  | States the courses title. |
| Course-Instructor | Course | Data | VARCHAR(45) | FALSE | FALSE |  | States the courses instructors name. |
| Course-Telephone | Course | Data | VARCHAR(45) | FALSE | TRUE |  | States the telephone for the campus department |
| Course-Email | Course | Data | VARCHAR(45) | FALSE | TRUE |  | States the email for the campus department |
| room-Number | Course | Data | VARCHAR(45) | FALSE | FALSE |  | States the room number for the campus department |
| department | Course | Data | VARCHAR(45) | FALSE | FALSE |  | States the campus department name. |
| universityID | Course | FK | INT | FALSE | TRUE |  | Identifies which university the coruse belongs to. |  |
| Inspection-ID | Inspection | PK | INT | FALSE | TRUE |  | Uniquely identifies each inspection record. | Kept until student leaves and the inspection report has been resolved. |
| date | Inspection | Data | DATE | FALSE | FALSE |  | States the date of an inspection |
| Condition-Satisfactory | Inspection | Data | TINYINT | TRUE | FALSE |  | States the condition of an inspection 0=bad, 1=good |
| additionalComments | Inspection | Data | TEXT(45) | TRUE | FALSE |  | States any additional comments by a staff member. |
| staffID | Inspection | FK | INT | TRUE | FALSE |  |  |  |
| Place-Number | Inspection | FK | INT | FALSE | FALSE |  |  |  |
| contactID | EmergencyContact | PK | INT | FALSE | TRUE |  | Uniquely identifies an emergency contact. | Kept stored until the students details have been deleted. |
| relationship | EmergencyContact | Data | VARCHAR(45) | FALSE | FALSE |  | Decribes the relationship between student and emergency contact |
| detailsID | EmergencyContact | FK | INT | FALSE | TRUE |  |  |
| studentID | EmergencyContact | FK | INT | FALSE | TRUE |  |  |
| Place-Number | Room | PK | INT | FALSE | TRUE |  | Uniquely identifies a room |  |
| Room-Number | Room | Data | VARCHAR(45) | FALSE | FALSE |  | States the rooms physical room number |  |
| Floor-Number | Room | Data | VARCHAR(45) | TRUE | FALSE |  | Identifies the rooms floor number |  |
| Monthly-Rate | Room | Data | DECIMAL | FALSE | FALSE |  | States the cost to rent the room for a month. |  |
| hallID | Room | FK | INT | TRUE | TRUE |  |  |  |
| apartmentID | Room | FK | INT | TRUE | TRUE |  |  |  |
| loginID | Login | PK | INT | FALSE | FALSE |  | Uniquely identifies the login |  |
| password | Login | Data | CHAR(32) | FALSE | FALSE | Stored as a hash value (MD5) | Stores the password as a protected hash value. |  |
| isLocked | Login | Data | TINYINT | TRUE | FALSE |  | States whether the account is locked or not. |  |

## Appendix B - Wireframes

### Login

**firstname.lastname**

**password**

**title**

### Dashboard

**Logout**

**Dashboard**

**Search Tool**

**Report Tool**

**Create Tool**

**Change Password**

**Username**

### Change Password

**New password**

**Confirm password**

**Old password**

**Update password**

**Return**

### Search Dashboard

**Logout**

**Dashboard**

**People Search**

**Username**

**Apartment Search**

**Hall of Residents Search**

**Room Search**

**University Search**

**Lease Search**

**Invoice Search**

**Inspection Search**

**Course Search**

**Return**

### Create Dashboard

**Logout**

**Dashboard**

**New Lease**

**Username**

**New Inspection Log**

**New Staff**

**New Apartment**

**New Hall of Residents**

**New University**

**New Course**

**New Room**

**New Adviser**

**New Invoice**

**Return**

### Report Dashboard

**Logout**

**Dashboard**

**Room Vacancy Report**

**Username**

**Waiting Student Report**

**Unpaid Invoice Report**

**Inspection Damage Report**

**Planned Inspection Report**

**Occupancy Report**

**Income Report**

**Unknown Leaving Report**

**Return**

### Search People

**Username**

**Results**

**Logout**

**Dashboard**

**First name**

**Search**

**Last name**

**Email**

**Mobile**

### Search Apartment

**Username**

**Results**

**Logout**

**Dashboard**

**Place Number**

**Search**

**Building Number**

**First Line**

**Postcode**

### Search Hall of Residents

**Username**

**Results**

**Logout**

**Dashboard**

**Place Number**

**Search**

**Building Number**

**First Line**

**Postcode**

### Search Room

**Username**

**Results**

**Logout**

**Dashboard**

**Place Number**

**Search**

**Lastname**

**Firstname**

### Search University

**Results**

**Username**

**Logout**

**Dashboard**

**University Id**

**Search**

**Building Number**

**First Line**

**Postcode**

### Search Lease

**Username**

**Results**

**Logout**

**Dashboard**

**Lease Number**

**Search**

**Duration**

**Start Date**

**End Date**

### Search Invoice

**Username**

**Results**

**Search**

**Logout**

**Dashboard**

**Lease Number**

**Date Paid**

**Payment Method**

**Semester**

**Date Due**

### Search Inspection

**Results**

**Username**

**Logout**

**Dashboard**

**Place Number**

**Search**

**Date**

**Tick box for uncompleted logs**

### Search Course

**Username**

**Results**

**Logout**

**Dashboard**

**University Id**

**Search**

**Course Number**

### Room Vacancy Report

**Logout**

**Dashboard**

**Print Report**

**Username**

**Place Number**

**Floor Number**

**Room Number**

**Results**

**Monthly Rate**

**Hall/Apartment ID**

### Waiting Student Report

**Logout**

**Dashboard**

**Print Report**

**Username**

**Student ID**

**Student Status**

**maxMonthly**

**FirstName**

**LastName**

**Mobile**

**Results**

### Unpaid Invoice Report

**Logout**

**Dashboard**

**Print Report**

**Username**

**Invoice ID**

**Date Due**

**Date Paid**

**Semester**

**Payment Due**

**First and LastName**

**Results**

### Inspection Damage Report

**Logout**

**Dashboard**

**Print Report**

**Username**

**Inspection Id**

**Comments**

**Date**

**Results**

**Staff ID**

**Place Number**

### Planned Inspection Report

**Logout**

**Dashboard**

**Print Report**

**Username**

**InspectionID**

**PlaceNumber**

**Date**

**Results**

**Hall ID**

**ApartmentID**

### Occupancy Report

**Results**

**Logout**

**Dashboard**

**Print Report**

**Username**

**Place Number**

**Room Number**

**Floor Number**

**Monthly Rate**

**Occupant Name**

**Occupant email and mobile**

### Income Report

**Results**

**Total Income Between Date1 and Date2: £0000**

**Logout**

**Dashboard**

**Print Report**

**Username**

**Lease Number**

**Amount Paid**

**Date Paid**

**Payment Method**

**First Name**

**Last name**

### Unknown Leaving Date Report

**Logout**

**Dashboard**

**Print Report**

**Username**

**Place Number**

**Last name**

**First name**

**Results**

**Email**

**Mobile**

### Create Lease

**Logout**

**Dashboard**

**Place Number**

**Create Lease**

**Username**

**Duration (Semesters)**

**Start Date**

**End Date**

### Create Inspection

**Logout**

**Dashboard**

**Start Date**

**Create Inspection Logs**

**Username**

**Duration (Days)**

### Create Staff

**Logout**

**Dashboard**

**First Name**

**Email**

**Create Staff**

**Mobile**

**Last Name**

**Position**

**Location**

**Building Number**

**First Line**

**Second Line (Optional)**

**Post Code**

**Middle Name**

**Username**

### Create Apartment

**Logout**

**Dashboard**

**Number of Beds**

**Create Apartment**

**Username**

**Building Number**

**First Line**

**Second Line (Optional)**

### Create Hall of Residents

**Staff ID**

**Logout**

**Dashboard**

**Number of Beds**

**Create Apartment**

**Username**

**Building Number**

**First Line**

**Second Line (Optional)**

### Create University

**Postcode**

**Logout**

**Dashboard**

**University Name**

**Create University**

**Username**

**Building Number**

**First Line**

**Second Line (Optional)**

### Create Course

**Logout**

**Dashboard**

**Course Number**

**Course Telephone**

**Create Course**

**Course Email**

**Course Instructor**

**Room Number**

**Department**

**University ID**

**Course Title**

**Username**

### Create Room

**Hall ID**

**Logout**

**Dashboard**

**Room Number**

**Create Room**

**Username**

**Floor Number**

**Monthly Rate**

**Apartment ID**

### Create Invoice

**Logout**

**Dashboard**

**Date Due**

**Create Invoice**

**Username**

**Semester**

**Payment Due**

**Lease Number**

### Create Adviser

**Logout**

**Dashboard**

**First Name**

**Email**

**Create Adviser**

**Mobile**

**Last Name**

**University Id**

**Building Number**

**First Line**

**Second Line (Optional)**

**Post Code**

**Middle Name**

**Username**

# Bibliography

gov.uk (no date) *Data protection and your business* [Internet] Available from https://www.gov.uk/data-protection-your-business [Accessed 4th December 2019]

TutorialPoint (no date) *SQL Tutorial* [Internet] Available from https://www.tutorialspoint.com/sql/index.htm [Accessed 12th November 2019]

# Reference List

Coronel, C., Morris S. (2016) Database Design. *Database Systems Design, Implementation and Management.* 12th ed. Boston, Cengage Learning, pp. 440-477

Coronel, C., Morris S. (2016) Database Design. *Database Systems Design, Implementation and Management.* 12th ed. Boston, Cengage Learning, pp. 207-225

gov.uk (no date) *Data Protection* [Internet] Available from https://www.gov.uk/data-protection [Accessed 4th December 2019]

gov.uk (no date) *Running a limited company* [Internet] Available from https://www.gov.uk/running-a-limited-company/company-and-accounting-records [Accessed 4th December 2019]

MariaDB (2018) *ACID: Concurrency Control with Transactions* [Internet] Available from https://mariadb.com/kb/en/acid-concurrency-control-with-transactions/ [Accessed 10th December 2019]

Oracle Corporation (2019) *MySQL Workbench* [Microsoft Windows application]. Vers. 8.0.18. Available from https://dev.mysql.com/downloads/workbench/

Panesar, K. (2019) *Analysis required for an Entity Relationship Diagram* Presented at Week 2 - Introduction to database lifefcycle, Entity Relationship Modelling and Interacting with Databases. York St John University, York [1st October 2019]