

YAHUAS Online Management system

Database Technical Document



July 10, 2021

nujcharee haswell

York St John’s University

# Executive Summary

## Project Overview

The technical document describes a prototype for Yorkshire & Humberside student accommodation and facilities centralised database solution (YAHUAS). It describes the process of client-server YAHUAS database application accessible via a web browser. The backend database is designed from scratch. It is proposed that the solution includes three different replicated databases: **development, testing and production** where production will be used for daily operation while the former two serve purposes as additional modular developments and respective user acceptance testing.

The report covers normalisation process, SQL commands used to create, read, update and delete (CRUD), testing scripts, ERD diagram, reports and forms and metadata schema.

## Service Description:

The database is to contain a full catalogue of student accommodations, lease and accommodation inspections. It is a operational, tactical and strategic tool for YAHUAS. The database is required to monitor and maintain student accommodations and inspections arrangements.

## Purpose and Scope of this Specification

The purpose of this project is to design and implement a bespoke back-end prototype. The following specification provides the basic requirements from which the project can be developed.

#### In scope

The project will include the creation of the analysis and core design documents such as ERD diagrams, and normalisation descriptions as well as wireframes which are to support the build phase. Logical design implementation including database, forms, reports, queries, and it may also include stored procedures, functions and triggers and anything else necessary to ensure the effective operation of the database system.

#### Outside scope

The following items are out of scope:

* Full detailed user documentation, a physical implementation specification
* Backup and maintenance schedules
* Role descriptors and functions
* Training documentation
* Operational procedures
* Finance and income management functionalities are outside the scope of this system

## Business Assumptions

**Students & Staff**

* Student data will be stored on the database including details of their course and their respective academic year
* A student can only be associated to a single programme of study
* A student can be either placed or on the waiting list
* A student may rent either a room in a hall of residence or student apartment
* A student is assigned to an Advisor
* A student will have one next of kin details used as an emergency contact
* A university staff can be a student advisor
* An advisor cannot be assigned to more than five students

**Residence Office**

* Residence office manages hall of residence / student apartments
* A place number identifies each room
* A flat identifies each apartment
* Residence office assigns students to a place
* Each hall of residence has contact details including a Hall Manager
* Each hall of residence provides single rooms, each has a room number, place number, and monthly rent rate
* Each apartment has apartment number, address, and the number of single bedrooms available in each apartment
* A single-room accommodation cannot be allocated to more than five students

**Lease**

* The duration of lease is a minimum of one semester and a maximum of one year; there are three semesters: first, second and summer semester
* The lease only offers a monthly rate
* A lease contract starts a week before semester begins and ends a week after semester ends

**Payments**

* An invoice is sent directly to students following rental period
* A reminder for payments is sent, up to two times if overdue

**Inspections**

* Inspections are carried out by university staff
* A university staff can inspect up to five properties in a day

**Courses**

* A course can be added into the system
* A course must have a lead instructor and on-campus instructor details

**System Access**

* The solution is designed for admin staff to use as one of their key business systems.
* The system is accessible by any of university’s accommodation office.
* Access to the system is granted to those with sufficient privilege which is stored on the database
* System is accessible via web browser (thin client) within the YAHUAS internal network.
* It is staff responsibility to keep record up-to-date and correct and comply to GDPR.
* Reports are viewable in tabular format via the web browser and can be printed using the built in browser print option

## Dependencies

There are no dependencies on any other systems

## Constraint

1. Whenever possible, information on a student’s next of kin is stored, which includes the name, relationship, address and contact telephone number. Only one next of kin is recorded per one student.
2. Security – SQL Injection
3. Accessibility
4. GDPR

## Requirements

Priority 1 requirements

1. The application must be accessible by Residence Office operators from internal YAHUAS network.
2. The application must be able to interact with the database to retrieve and edit data
3. The application will be able to create a new record on the database.
4. The application will be able to edit data in the database.
5. The application will be able to retrieve data from the database
6. The application will be able to create reports based on information stored on the database.
7. The administrator must have login details to use the web page to carry out the administrative/manager functions of YAHUAS solution.
8. The administrator will be able to run reports via the web interface.
9. Data entry – administrator should be able enter data into the system to achieve the business processes of YAHAUS.
10. Data validation – ensure the correct validations are considered for any database manipulations via the web interface.
11. All SQL reports should be presented in tabular form via the web interface.

Priority 2 requirements

1. A single page that allows any search
2. A facility that group reports by functions such as data quality reports, performance reports and tactical reports

## System outputs

Staff must be able to query and update information system and produce management information reports.

The following reports are included:

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

In addition, there are facilities to search for items such as:

* Invoices
* Leases
* Office
* Courses
* Students

## GDPR

## Implementations

The end user will need to log and has ability to change password

View UML Diagram – demonstrates user access and control

### Form Design

Each form will follow the same format. Minimise keystroke and take advantage from user controls such as drop down menu, check box, radio button and submit button. This will minimise operator error.

Not included: predictive text.

Default value:

Report Design

The reports output in tabular form as specified by the user requirements. An example is provided below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Invoice Number | Invoice Date | Student ID | Date Due | Status |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Page Wireframes

For each outputs, a basic wireframe has been designed to add the implementation of the system.

Access & Security

* Login
* Change Password
* Log-out
* Time-out

Search functionality

* Search Apartment
* Search Hall of Residence
* Search Room
* Search University
* Search Lease
* Search Invoice
* Search Inspection
* Search Course

Reports

* Room Vacancy Report
* Waiting Student Report
* Unpaid invoice report
* Inspection damage report
* Inspection damage report
* Planned inspection report
* Occupancy report
* Income report
* Unknown leaving date report

Add functionality

* Create Lease
* Create Inspection
* Create Staff
* Create Apartment
* Create Hall of Residence
* Create University
* Create Course
* Create Room
* Create Invoice

Future implementation

How can I improve?

* Use UPRN to look up for address
* Look & feel

## User Interface

* The login page consists of two text boxes: Username and Password.
* The login button allows users to log into the system.

Once successfully logs in, the user will be able to process student applications.

There are three types of users for the potential system – students, administrators and managers.

For the purpose of this prototype there will be one user with a dual role of administrator and manager.

## Possible and rejected entities

The ERD was created and part of the normalization process it became evident that there are several entities that can be redundant In the Normalisation section, the rejected entities will be included.

* Student
* Staff
* Administrator
* Hall Manager
* General Details
* ~~Next of Kin records~~ Only one next of kin for emergency
* Advisor
* Address
* Apartment
* ~~Lease term~~ this can happen prior to recording onto the system
* Instructors table – outside scope
* ~~Payment methods~~

System will use MYSQL as the server and you can assume that the front end will allow students to be able to enter their personal details and select an accommodation to rent - using a link. For this back-end prototype solution, student and rental booking data can be populated directly to the database via PHPMyAdmin/MYSQL Workbench/MYSQL console.

This prototype solution will have a standard and consistent look for the home page and other follow-up pages.

## Entity Relationship Matrix

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
| **Student** |  |  |  |  |  |  |  |  |  |
| **Staff** |  |  |  |  |  |  |  |  |  |
| **Adviser** |  |  |  |  |  |  |  |  |  |
| **Address** |  |  |  |  |  |  |  |  |  |
| **Invoice** |  |  |  |  |  |  |  |  |  |
| **Lease** |  |  |  |  |  |  |  |  |  |
| **Apartment** |  |  |  |  |  |  |  |  |  |

## Validations

The purpose of validation is to comply with business rules set identified in user requirements. This eliminates any possible data quality issue and ensure data integrity.

For example, the table ‘Student’ represents Student entity with follow attributes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Format** | **Validation Check** | **Example Pass** | **Example Fail** |
| DATE | YYYY-MM-DD |  |  |  |
|  |  |  |  |  |

Javascript will be used in pages with CREATE or UPDATE operations.

String validation

HTML form text max length

## Normalisation Process

The following are examples of normalisation to achieve third form.

Below is a set of data on Staff captured in the database as a list of attributes associated to individual staff. The table has no form of normalisation.

**1 Staff table - No normalisation**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Staff Name** | **Position** | **Role** | **Location** | **Assigned Students** | **Assigned Accommodation** |
| John Doe | Hall Manager | Admin | Yorkshire Office | Liz Hurley,  Hugh Grant,  Julia Roberts | Student Flat 1 |
| Jane Doe | Student Flat Manager | Assistant | Hull Office | Damien Hurley,  Drew Barrymore,  Benjamin F | Student Flat 2 |
| Jason B | Hall Manager | Admin | Yorkshire Office | Jason St,  Laura St,  Simone J | Apartment 3 |
| Nora J | Residence Office Manager | Operator | Central Office | David Lee,  Lee David,  Leeson S | Flat 2 |
| Jane Doe | Student Flat Manager | Assistant | Hull Office | Jon Bon Jovi | Hall 1 |
| Jane Doe | Student Flat Manager | Assistant | Hull Office | Richie Sambura | Hall 2 |
| Jane Doe | Student Flat Manager | Assistant | Hull Office | Will Smith | Hall 3 |

The above problem raises a data redundancy issue however this is can be addressed by normalisation process. The normalisation process makes data more meaningful and reusable.

### Steps in normalisation process

First Normal Form (1NF) normalisation has four basic rules.

* Rule 1: Each Colum will contain atomic value
* Rule 2: Each column will of the same type
* Rule 3: Each column will have a unique name
* Rule 4: Order of columns stored on the database does not matter

The table above satisfies rule 3 because all columns have unique name. However, it violates rule1 because ‘**Assigned Students’** column does not have atomic value.

**First Normal Form (1NF)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Staff Name | Position | Role | Location | Assigned Students | Assigned Accommodation |
| John Doe | Hall Manager | Admin | Yorkshire Office | Liz Hurley | Student Flat 1 |
| John Doe | Hall Manager | Admin | Yorkshire Office | Hugh Grant | Student Flat 1 |
| John Doe | Hall Manager | Admin | Yorkshire Office | Julia Roberts | Student Flat 1 |
| Jane Doe | Student Flat Manager | Assistant | Hull Office | Damien Hurley | Student Flat 2 |
| Jane Doe | Student Flat Manager | Assistant | Hull Office | Drew Barrymore | Student Flat 2 |
| Jane Doe | Student Flat Manager | Assistant | Hull Office | Benjamin F | Student Flat 2 |
| Jason B | Hall Manager | Admin | Yorkshire Office | Jason St | Apartment 3 |
| Jason B | Hall Manager | Admin | Yorkshire Office | Laura St | Apartment 3 |
| Jason B | Hall Manager | Admin | Yorkshire Office | Simone J | Apartment 3 |
| Nora J | Residence Office Manager | Operator | Central Office | David Lee | Flat 2 |
| Nora J | Residence Office Manager | Operator | Central Office | Lee David | Flat 2 |
| Nora J | Residence Office Manager | Operator | Central Office | Leeson S | Flat 2 |
| Jane Doe | Student Flat Manager | Assistant | Hull Office | Jon Bon Jovi | Hall 1 |
| Jane Doe | Student Flat Manager | Assistant | Hull Office | Richie Sambura | Hall 2 |
| Jane Doe | Student Flat Manager | Assistant | Hull Office | Will Smith | Hall 3 |

In addition, one must observe at each table, one by one and answer the following questions:

* Does the combination of all columns make a unique row every single time?
* Which column can be used to uniquely identify the row?

The combination of all columns does not make a unique row every single time nor there is a column that can be used to uniquely identify the row. Therefore, it requires a primary key to be created. For **Staff** **table**, a staff\_id will be created and used as a primary key.

|  |  |  |
| --- | --- | --- |
| Staff\_Id | Staff Name | Position |
| A1001 | John Doe | Hall Manager |
| A1002 | Jane Doe | Student Flat Manager |
| A1003 | Jason B | Hall Manager |
| A1004 | Nora J | Residence Office Manager |
| A1005 | Jane Doe | Student Flat Manager |

**2 Second Normal Form (2NF)**

* Fulfils the requirements of second normal form
* Has no transitive functional dependency

The functional dependency refers to the concept that fields in the table are determined by the primary key.

Table 1: Identify functional dependency

|  |  |  |
| --- | --- | --- |
| Column name | Functional dependency | Reason |
| Staff\_ID | No | This is the primary key which uniquely identifies staff |
| Staff Name | Yes | This is dependent on the primary key. Each staff id has different name |
| Position | Yes | This is dependent on the primary key. Each staff id has different position |
| Location | Yes | This is dependent on the primary key. Each staff id is located in different office |
| Assigned Students | No | This is not dependent on staff. One staff can be assigned to multiple students |
| Assigned Accommodation | No | This is not dependent on staff. One staff can assigned many accommodations to students. |

The above table confirms there are a combination of functional dependent and functional independent columns.

**Students**

This column is not dependent on staff as there can be many students assigned to one staff. **Student** is not part of definition of **Staff**.

**AUDIT RECORD**

**3 Third Normal Form (3NF)**

The transitive functional dependency refers to attributes that depend on the primary key only. For Staff table, this can be normalised into 3NF as seen below:

## Modelling

UML (unified modelling language) will be used as a tool to analyse user requirements. It models the activities of the application and identify a list of tasks to the users. UML helps break down the development into list of process which makes it easier to implement.

## Database design strategy

## High-level System Requirements

## Implementation

Report design

Use SQL Views

Possible and Rejected Entities

Entity Relationship Matrix

Metadata Schema

GDPR & Data Protection Act (2018)

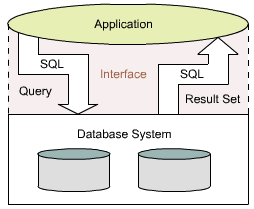
Validations

Normalisation Descriptions

* Normilisation
* Normalised to Third Normal Form
* Invoice Table
* Course Table
* Normalising Past third Normal form
* Report & Form Designs
* Form Design
* Report Design
* Page Wireframes
* ERD
* Forward Engineered

SQL Script

* Login
* Change password
* Search Person
* Search Apartment
* Search Hall of Residents



<http://www.gitta.info/DBSysConcept/en/html/DBLanguages_learningObject3.html>

Client-Server Architecture

Advantages:

Easier to maintain – only has to be updated once, on the server rather than each client machine

Communication is quicker than one-tier – a more powerful server handles the requests and is able to complete them quicker than a client machine.

CustomerNumber in the enrollment table relates to customerNumber in the customer table.

And the courseNumber relates to the courseNumber in the course table.

Using this we can see that Customer 1 is Ariel Johnson and they are enrolled on the Advanced Pastels and Intermediate Pastels courses.

We can change data within each table e.g. course date and it won’t cause any problems in the database.

We can insert new data into any of the tables easily without any errors in other tables.

And we can delete enrollment rows as needed e.g. if a customer drops out they can easily be deleted from the enrollment table and it won’t affect any data within the course or customer tables.

Table

Description automatically generated

Create Student

|  |  |
| --- | --- |
| Inputs | firstName, surname, |
| Validations |  |
| Outputs |  |
| Tables |  |
| SQL Code |  |

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

Chart

Description automatically generated

# Glossary

ERD: Entity Relationship Diagram

Database design is a sub domain of information processing. In information processing there are two types of strategies: top down – bottom up

Purpose

## Entity Relationship Diagram (ERD)

The diagram below shows entities, entity attributes, cardinality of relationships, key constraints, and relevant assumptions made in modelling the data. Must not have any M:N relations. All primary / foreign keys are listed along with M:N relations. Drawn in Visio Paradigm Online tool.

Diagram

Description automatically generated