

## Data Dictionary

A data dictionary is a file that contains a database's metadata. In this database, 3 tables were used to store student information like their ages, gender, and grade levels. The following are data dictionaries used in the creation of this program.

Table: Students

Field	Data Type	Null	Key
<b>ID</b>	Auto Integer	No	Primary Key, Foreign Key
<b>FirstName</b>	Text	No	
<b>LastName</b>	Text	No	
<b>Grade</b>	Integer	No	
<b>Gender</b>	Boolean	No	
<b>Age</b>	Integer	No	
<b>HostelCode</b>	Text (2)	No	Foreign Key

Table: Hostel Table

Field	Data Type	Null	Key
<b>HostelCode</b>	Text (2)	No	Primary Key
<b>Hostel</b>	Text	No	

Table: TableGroups

Field	Data Type	Null	Key
<b>ID</b>	Integer	No	Primary Key
<b>GroupNumber</b>	Integer	No	

The database tables are related to each other and have referential integrity with the database management program. Below is an Entity Relationship Diagram (EDR) that shows the relationships between the tables.

## Database Diagrams

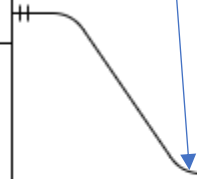
Figure 1. Logical Schema

The one-to-one relationship between Students and TableGroups will allow the program to take IDs from Students and place them in TableGroups with.

TableGroups	
PK	<u>ID</u>
	GroupNumber

Students	
PK, FK	<u>ID</u>
	FirstName
	LastName
	Grade
	Gender
	Age
	HostelCode

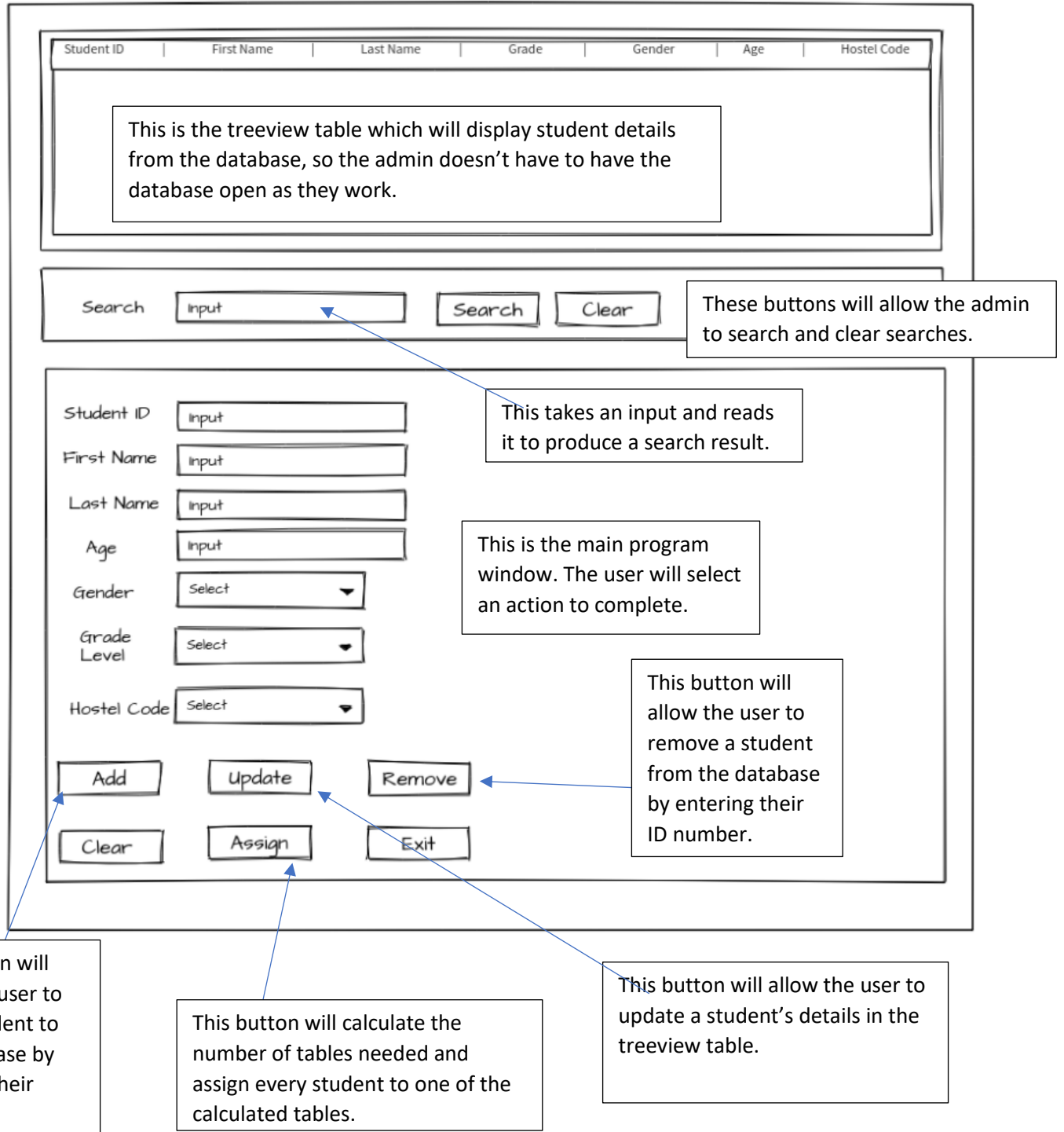
Hostel Table	
PK	<u>HostelCode</u>
	Hostel



The one to many relationship will allow for multiple students to belong to one hostel without causing any errors to the program.

# Software Interface

## Main User Interface



The diagram illustrates the Main User Interface for a student database system. It features a treeview table at the top for displaying student details, a search section with an input field and buttons, and a main program window with various input fields and action buttons. Annotations explain the functions of these elements.

**Student Details Table:**

Student ID	First Name	Last Name	Grade	Gender	Age	Hostel Code
------------	------------	-----------	-------	--------	-----	-------------

**Search Section:**

Search

**Main Program Window:**

Student ID   
First Name   
Last Name   
Age   
Gender   
Grade Level   
Hostel Code

**Action Buttons:**

**Annotations:**

- This is the treeview table which will display student details from the database, so the admin doesn't have to have the database open as they work.
- These buttons will allow the admin to search and clear searches.
- This takes an input and reads it to produce a search result.
- This is the main program window. The user will select an action to complete.
- This button will allow the user to remove a student from the database by entering their ID number.
- This button will allow the user to add a student to the database by entering their details.
- This button will calculate the number of tables needed and assign every student to one of the calculated tables.
- This button will allow the user to update a student's details in the treeview table.

## Section B: Algorithms and Flowcharts

### Main Program Algorithm in Pseudocode

// Calculating the number of tables needed

MaxTables = 0

// To count the number of rows. Each row contains a student.

Results = Select count (\*) from Students record.

loop row in results

Maximum = row[0]/7

MaxTables = round(Maximum, 1)

end loop

//Delete statement to prevent duplications from the TableGroup assignment

// TableGroups is a table that contains the Student ID and the Assigned group number they have been given.

Delete \* from TableGroups

//Loop to assign group numbers to all records.

genderIndex ["F", "M"]

tableNumber = 1

// this is to loop through all the possible grade levels of students in the school

loop grade from 7 to 12

//So that Students with gender F will be assigned first then students with gender M

loop genderIndex from "F" to "M"

sql = Select ID from Students Where Grade = str(grade), and Gender = gender

return sql

results = return sql

loop row in results

// Table number increases after a student has been given a table number and placed in the TableGroups table.

tableNumber = tableNumber + 1

```

    if tableNumber > maxTables

        // This causes the loop to iterate so each student is given a group number
        tableNumber = 1
    end if

    //Students are placed in a database table with the group number they were assigned.
    return INSERT INTO TableGroups (ID, GroupNumber, Grade) VALUES(str(row[0]),
    str(tableNumber), str(grade))

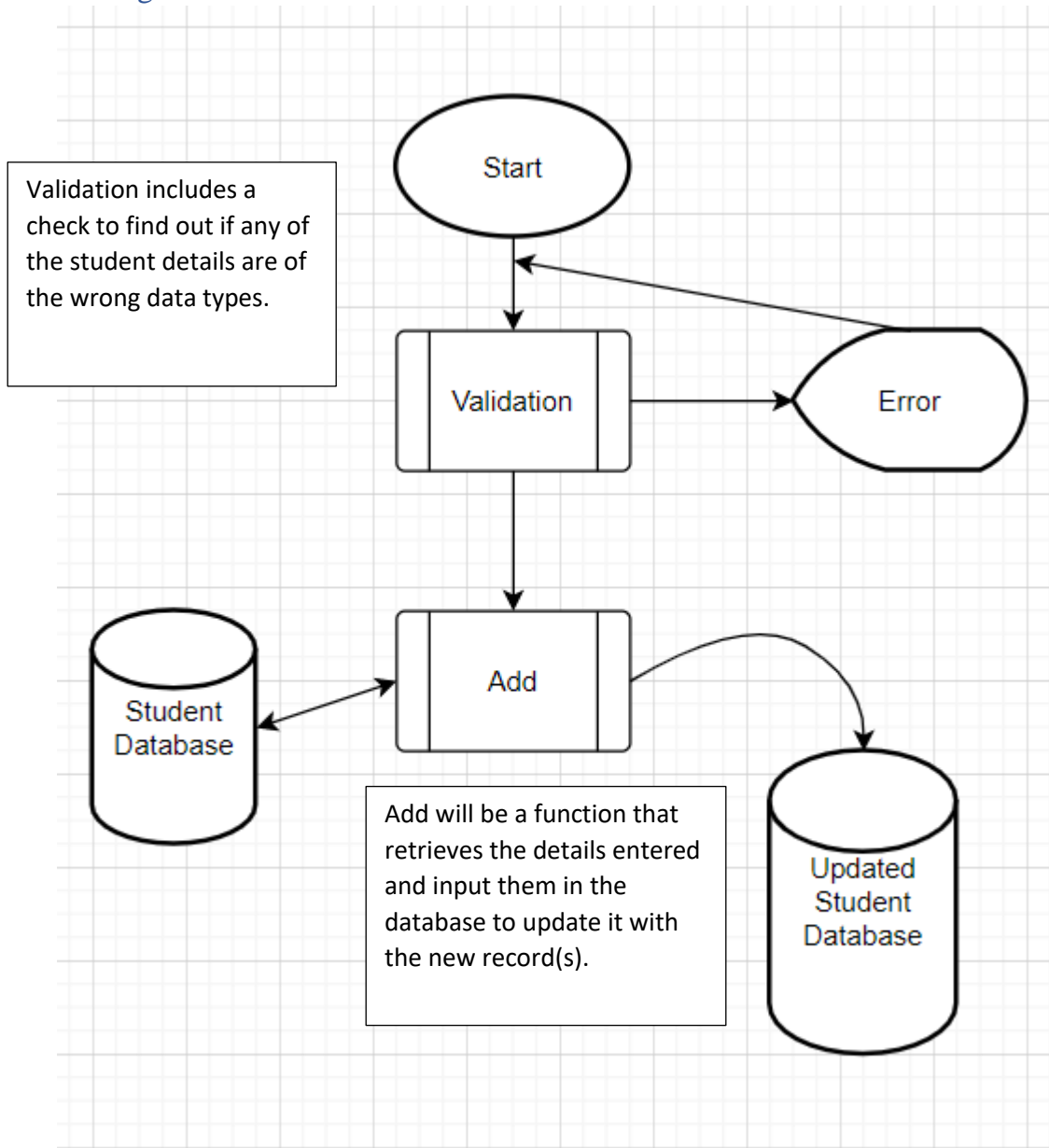
    end loop
end loop
end loop

//Writing the database results in a file.
Filewrite = Open 'groups.txt'
Loop GNumber from 1 to maxTables+1
    Filewrite.write ("\n")
    Filewritr.write(Table Number + GNumber + '\n')
    sql = SELECT S.FirstName, S.LastName, S.Grade From Students S INNER JOIN TableGroups
T on
    S.ID=T.ID
    sql = sql + WHERE T.GroupNumber = GNumber
    return sql
    results = return sql
end loop

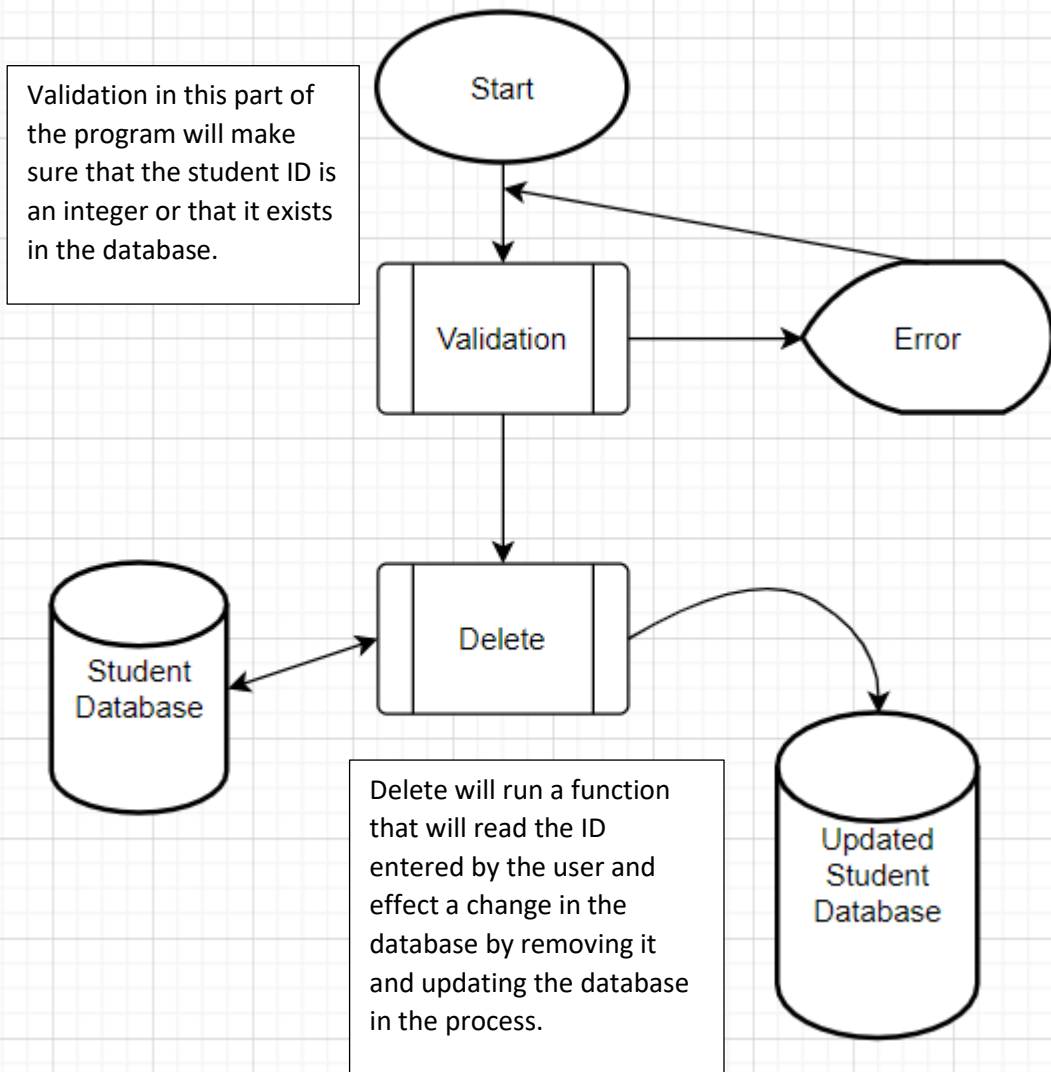
//Displaying Student Names as well as Table numbers in a file.
Count=1
For row in results
    Filewrite.write(count)
    Filewitre.write(row[0] + row[1] + row [2]
    Count = count+1s
End loop

```

## Adding a Student



## Removing a Student



## Test Plan

Test Number	Description	Related success criteria	Expected outcome
1	Check if Student details are valid	The program displays an error if Student details are not valid.	Program stops the process and reminds the user of the correct data types allowed.
2	Removing a student record	Admin is able to remove students from the database.	The student record is removed from the database and is no longer available
3	Searching for student record	Admin is able to search for records based on names or grades.	The display table should update to show the relevant records based on the user's input in the search entry.
4	Adding a student record	Admin is able to add students to the database.	The student record is added to the database and is now available for grouping.
5	Updating a Student record.	Admin is able to update student data.	The display table should update itself in the relevant column based on what the admin changes about the record.
6	Final Table groups	The program can output a document with the table groups and students.	A document is sent to a predetermined folder which contains every table group and all the students that have been assigned there.
7	Number of tables needed	The program can calculate the number of tables needed to seat all students.	The program calculates how many tables are needed to seat all the students and is displayed on the document that is produced.
8	Equal number of students on each table	It can place an equal number of students on each table.	On the document, there will be an equal number of students on most tables to cater for a possible odd number of students.
9	All grade level inclusive	The program can place at least 1 person from each grade on a table.	There should be at least one member of every grade level on a table, accounting for the number of people in that grade.



## Data Structures

**Data Tables:** This is any display in any tabular form with rows and columns that have names. This is like the physical schema of the database. Each row is an individual record, and the columns are the fields names. If we run an output query it will select the data we need and put it in a data table.

Select FirstName, LastName, Grade

From Students

Where Age < 11

We get a Data table like this:

FirstName	LastName	Grade
Maame	Johnson	8

**Arrays:** This is a way of storing values or strings in adjoining blocks of data in a contiguous manner. This is a set number of objects that when called will only have the values available in it for operation in the program. There should be about 3 arrays in the program.

GenderArray= ["F", "M"]

GradeArray = [7, 8, 9, 10, 11, 12]

HostelArray = ["AN", "FR", "CE", "CA"]