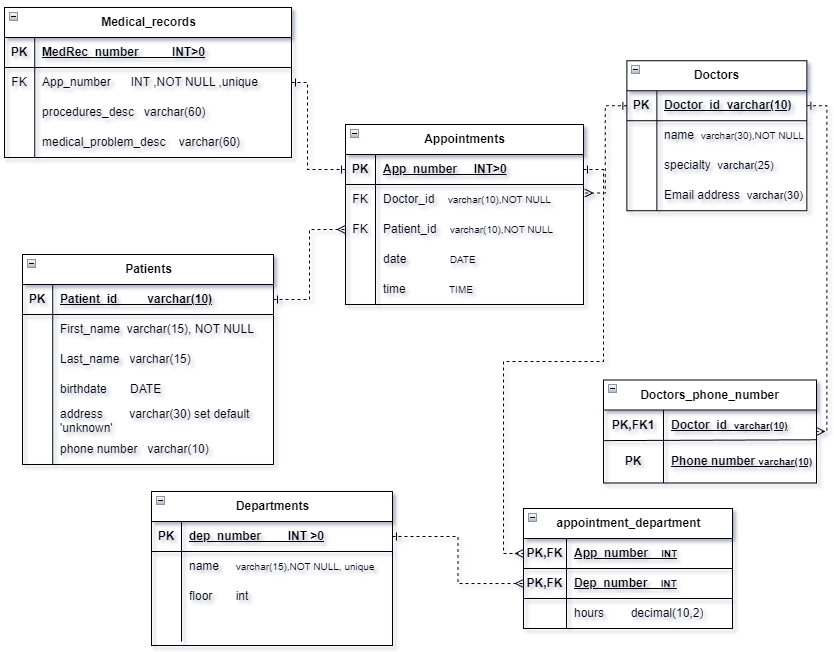
Technical Document

1. Introduction:

This document gives deep information and details on the design, implementation, structure, as well as the database system operations. It is a reference for the database developers, and technical employees that are involved in the maintenance of the database. Moreover, it describes the data stored in the database. It gives information considering the database schema, such as the structure (how tables are organized). It illustrates the implemented security methods, such as users’ roles, and their access to the database, and the information’s allowed to each one of them.

In addition, it includes illustrations of how different things work in the system. For example, it shows the grant and revoke security measured applied. Also, trying different attempts in views, procedures, schedules.

1. Physical Schema:



1. Database development:

3.1 Database Overview:

|  |  |  |
| --- | --- | --- |
| Table | Table name | Description |
|  | Doctors | This table contains information about the doctors in the hospital. The doctors id, name, specialty, email address. |
|  | Patients | This table includes information about the patients that visits the hospital for doctor appointments. It includes the patient id, first name, last name, birthdate, address, phone number. |
|  | Appointments | The table includes information about each appointment, such as the appointment number, date, time, doctors id, patient id. |
|  | departments | This table contains information about the departments available for patients to visit in the hospital, such as department number, name, and what floor the department is. |
|  | Medical\_record | This table contains information on the medial records for each patient, such as medical record number, appointment number, description of procedures made during or after that appointment, and description for medical problems that made the patient visit the doctor. |
|  | Appointment\_department | This table has information on the departments visited after a certain appointment, and the hours spent in that department. |
|  | Doctors\_phone\_number | This table has the phone number for each doctor, it contains the doctor’s id, and their phone number. It might contain different numbers for the same doctor. |

|  |  |  |
| --- | --- | --- |
| View | View name | Description |
|  | patients\_appointment\_view | This view is made for patients to view their appointments, and to get general information for it, such as the doctors name, the appointment date and time. |
|  | Nurse\_appointment\_view\_after2020 | This view is made for nurses to keep track of appointments after 2020, it includes patients and doctors name, the appointment date and time. |
|  | patients\_medical\_record | This view is made for patients to view their past information such as the description for procedures and medical problems they had. |
|  | doctors\_appointments | This view helps doctors to view their past or coming appointments. They get information such as date, time, patient name, and phone number to contact them if needed. |
|  | nurse\_medicalRecord\_view | This view is made for nurses to view the patient’s medical record. Depending on the appointment it will show the patient and doctor name and medical record information. |
|  | Number\_Of\_Appointments\_for\_Doctors | This view is made for managers to keep track of the number of appointments each doctor had. It helps them make decisions about the doctors, wither they are important or not. |
|  | medical\_record\_count | This view is made for managers to keep track of the number of medical records they have in the system. |

|  |  |  |
| --- | --- | --- |
| Procedure | Procedure name | Description |
|  | getPatientById | This procedure is used to get the patient information such as the name, birthdate, address, phone number. Using his id. |
|  | GetDoctorBySpeciality | This procedure is used to get the doctor information |
|  | getDepartmentByNumber | This procedure takes a department number and returns the name and floor of that department. |
|  | getAppointmentByDate | This procedure takes a specific date and returns the booked appointments on that day. It returns the doctor and patient names, the doctor’s specialty and the appointment time. |
|  | getMedicalRecordByPatientID | This procedure takes a specific patient id and returns the first name, last name, birthdate, address, and phone number of that patient. |
|  | NurseScheduleAppointment | This procedure is made especially for nurses because they usually make the appointments. It takes all information about the new appointment. |
|  | updatePatientMedicalRecord | This procedure is made to help doctors, nurses, or patients to update or fix a certain information in a specific medical record. |
|  | AddPatientMedicalRecord | This procedure is made to for doctors and nurses to add new medical record for patients after a certain appointment if needed. |
|  | AddPatients | This procedure is made for managers, doctors, nurses to add a patient to the database, it takes all the information related to that patient and add it to the patient’s table. |
|  | AddAppointmentDepartment | This procedure is adds a row to the appointment\_department table. It used after an appointment only if a patient needed to visit a department specified by the doctor. |
|  | deleteAppointment | This procedure is used by doctors, patients, nurses to delete a specific appointment from the appointments table. It only takes an appointment number. |
|  | deletePatient | This procedure is used to delete a patient from the database by deleting their information from the patients’ table. It only takes the patient id and removes their information. |
|  | deleteDoctor | This procedure can be used by the managers to remove a doctor from the database. It only takes the id of that doctor. |
|  | addDoctor | This procedure can be used by the managers to add a doctor to the database. It takes all the information of that doctor and adds it to the doctors’ table. |
|  | editDoctor | This procedure can be used by the managers to fix the doctor’s information if needed in the database. It eventually fixes the doctors information in the doctors’ table |
|  | editPatient | This procedure is made for doctors, nurses, and managers to fix the patient’s information in the patients’ table. |

3.2- Security:

|  |  |  |  |
| --- | --- | --- | --- |
| User Name | Privilege Command | Description | Screenshot |
| Nurse | CREATE USER Nurse identified BY '2003'; | This command creates a user with the name nurse, and gives it a password 2003 to be able to access using that password |  |
| Nurse | GRANT SELECT, UPDATE, DELETE ON medical\_record TO Nurse; | This gives the nurse permission to access the medical record table and be able to look at the table, update certain information, or delete an information. |  |
| Nurse | GRANT SELECT, UPDATE, DELETE ON appointments TO Nurse; | This gives the nurse permission to access the appointment table in order to read it, update it, or event delete information from it. |  |
| Nurse | GRANT SELECT ON Nurse\_appointment\_view\_after2020 TO Nurse; | This gives the nurse permission to view all appointments after the year 2020 |  |
| Nurse | GRANT SELECT ON nurse\_medicalrecord\_view TO Nurse; | This gives the nurse permission to the nurse\_medicalrecord\_view, to be able to know all the information related to a medical record. |  |
| Nurse | GRANT EXECUTE ON PROCEDURE AddPatientMedicalRecord TO Nurse; | This gives the nurse permission to be able to add new medical records to patients, using the procedure for easier addition. |  |
| Nurse | GRANT EXECUTE ON PROCEDURE updatePatientMedicalRecord TO Nurse; | This gives the nurse permission to be able to update certain information in the medical record table using the procedure. |  |
| Nurse | GRANT EXECUTE ON PROCEDURE NurseScheduleAppointment TO Nurse; | This allows the nurse to use the procedure in order to add new appointments easily. |  |
| patient | CREATE USER patient identified BY '2004'; | This command creates a user with the name patient and password 2004. |  |
| Patient | GRANT SELECT, UPDATE, DELETE ON patients\_appointment\_view TO patient; | This allows patients to view, update, or delete their appointment information. They get information such as their id, doctor name, date and time of appointment. |  |
| Patient | GRANT SELECT, UPDATE, DELETE ON patients\_medical\_record TO patient; | This allows patients to view their medical record and make changes to it. They get their id, first name, number of appointments, medical record information. |  |
| Patient | GRANT EXECUTE ON PROCEDURE getMedicalRecordByPatientID TO patient; | This allows the patients to use the procedure to be able to view their medical record information based on the patient id, they have to inter their id. |  |
| Doctor | CREATE USER Doctor identified BY '1962'; | This command creates a user with name doctor and password 1962 |  |
| Doctor | GRANT SELECT,INSERT, UPDATE, DELETE ON appointments TO Doctor; | This command allows the doctor to make any change on the appointment table. |  |
| Doctor | GRANT SELECT,INSERT, UPDATE, DELETE ON medical\_record TO Doctor; | This command allows doctors to make any changes on the medical record table. |  |
| Doctor | GRANT SELECT,INSERT, UPDATE, DELETE ON doctors\_phone\_number TO Doctor; | This command allows doctors to make any changes on the doctors\_phone\_number table, to add, edit, or delete their numbers. |  |
| Doctor | GRANT SELECT ON doctors\_appointments TO Doctor; | This command allows doctors to view the appointments by getting the important information. |  |
| Doctor | GRANT EXECUTE ON PROCEDURE AddPatientMedicalRecord TO Doctor; | This command allows doctors to use the AddPatientMedicalRecord procedure to add medical records. |  |
| Doctor | GRANT EXECUTE ON PROCEDURE updatePatientMedicalRecord TO Doctor; | This command allows doctors to use the updatePatientMedicalRecord procedure to update medical records. |  |
| Management | CREATE USER Management identified BY '121'; | This command creates a user management with password 121. |  |
| Management | GRANT SELECT, INSERT,UPDATE, DELETE ON doctors TO Management; | This command allows management to look, add, update, or delete elements from table doctors. |  |
| Management | GRANT SELECT, INSERT, UPDATE, DELETE ON patients TO Management; | This command allows management to look, add, update, or delete elements from table patients. |  |
| Management | GRANT SELECT, INSERT, UPDATE, DELETE ON departments TO Management; | This command allows management to look, add, update, or delete elements from table departments. |  |
| Management | GRANT SELECT ON Number\_Of\_Appointments\_for\_Doctors TO Management; | This command allows management to access the view in order to keep track of total number of appointments each doctor has. |  |
| Management | GRANT SELECT ON medical\_record\_count TO Management; | This command allows management to access the view in order to keep track of total number of medical records in the database. |  |
| Management | GRANT EXECUTE ON PROCEDURE GetDoctorBySpeciality TO Management; | This command allows management to use the procedure get the information about doctors based on a certain specialty. |  |
| Management | GRANT EXECUTE ON PROCEDURE AddPatients TO Management; | This command allows management to use the procedure AddPatients to be able to add new patients to the database. |  |
| Management | GRANT EXECUTE ON PROCEDURE editPatient TO Management; | This command allows management to use the procedure editPatient to be able to edit patients information in the database. |  |
| Management | GRANT EXECUTE ON PROCEDURE deletePatient TO Management; | This command allows management to use the procedure deletePatient to be able to remove patients from the database. |  |
| Management | GRANT EXECUTE ON PROCEDURE addDoctor TO Management; | This command allows management to use the procedure addDoctor to be able to add new doctors to the database. |  |
| Management | GRANT EXECUTE ON PROCEDURE editDoctor TO Management; | This command allows management to use the procedure editDoctor to be able to edit doctors’ information in the database. |  |
| Management | GRANT EXECUTE ON PROCEDURE deleteDoctor TO Management; | This command allows management to use the procedure deleteDoctor to be able to remove doctors from the database. |  |

3.3- User interface:

3.3.1 flowchart and data movement diagrams:

* A data flow diagram is a graphical representation that relates to the transmission of data along (inside) a specific system through certain processes. It illustrates the flow of the data that is considered as input or outputs into and from the system graphically. (https://www.conceptdraw.com, n.d.)

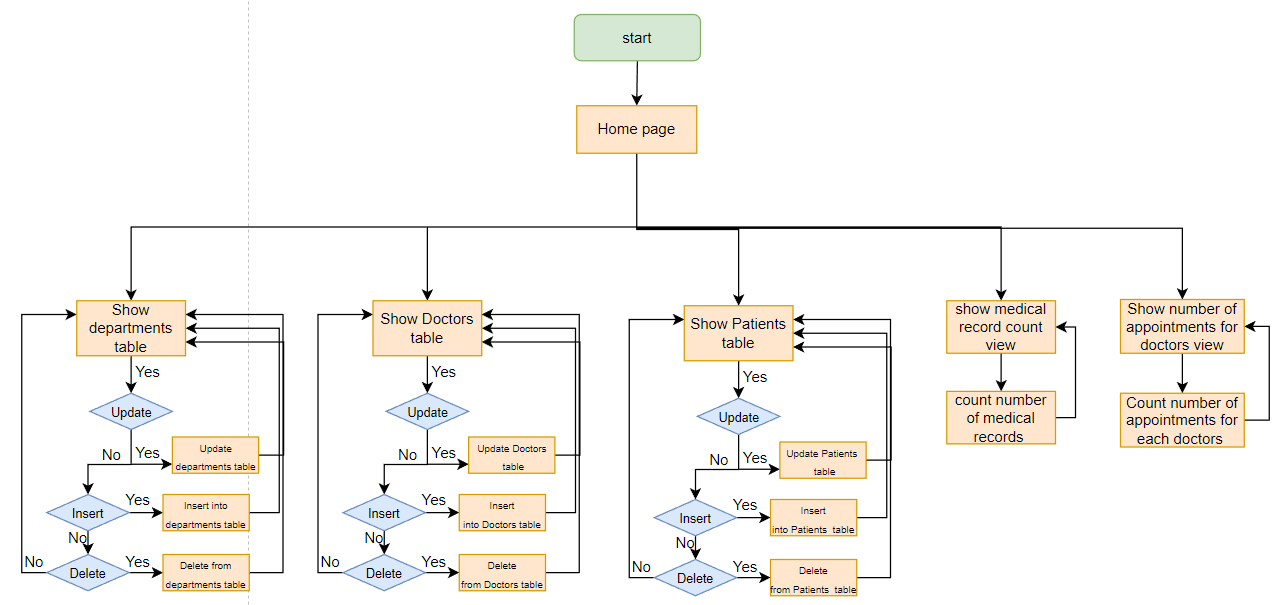
A picture containing text, diagram, screenshot, plan

Description automatically generated

This data flow diagram shows the processes that could be done by the management interface. The management are able to view the total number of appointments that every doctor had over the years. They are also able to view the medical record count, which counts the number of records that the database has.

Moreover, the management is able to access the department, doctors, and patients table, and be able to add, edit, or delete rows from it.

* A flowchart diagram is a visual representation that shows the ordered logic and arrangement of actions that are related to database activities. It involves multiple symbols and connections in order to show actions, flow of data, and decisions in the system for the database.



This figure shows the ordered sequence of events that could be done in the management user interface. Which means the steps to reach any desired process.

In conclusion, DFD’s are different from the flowcharts in the usage of arrows. Flowcharts use arrows to identify the sequence of the events. DFDs on the other hand, use arrows to represent the flow of the data along the system. DFDs are not concerned with the order of occurrence (flow) of data. in addition, DFDs including data modification tasks, such as computations, transformation, and data checking.

3.3.2 – Interfaces Development:

This Interface was made for management in order to be able to make important functions related to the hospital. For example, making changes to the doctors, patients, and departments tables. The following table shows the available tables for the management to use.

|  |  |  |  |
| --- | --- | --- | --- |
| **Page ID** | **Title** | **Description** | **screenshot** |
|  | Departments | This page displays the available departments in the hospital, by showing their number, name, and floor. Managers could add, edit, or delete a department. |  |
|  | Doctors | This page displays information on the available doctors in the hospital, by showing their id, name, specialty, and email. Managers can add, edit, or delete a doctor from the database. |  |
|  | Patients | This page displays information on the patients that have visited the hospital over the years. It shows their id, name, birthdate, address, and phone number. |  |
|  | Medical Record number | This page only displays the number of medical records available in the database. |  |
|  | Number Of Appointments For Doctors | This page displays the total number of appointments that each doctor had over the years. |  |

A screenshot of a computer

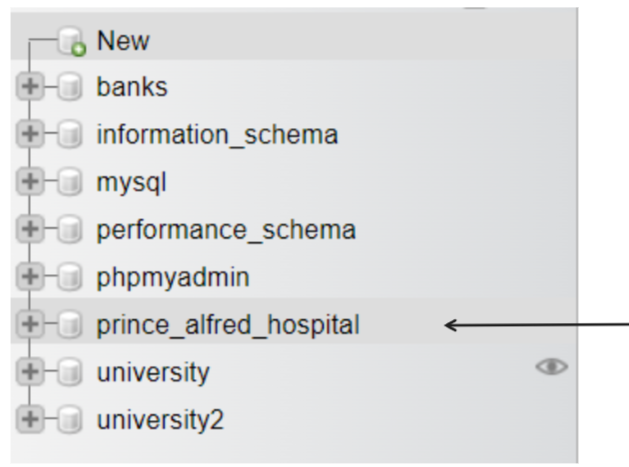
Description automatically generated with medium confidence

1. Maintenance:

4.1- Database recovery and backups:

Database recovery and backup in phpMyAdmin is achieved using import and export. The following Screenshots illustrates the steps that are followed to achieve this:

1. Click on the name of the database that wanted to be backed up, and here it’s the prince Alfred hospital database:

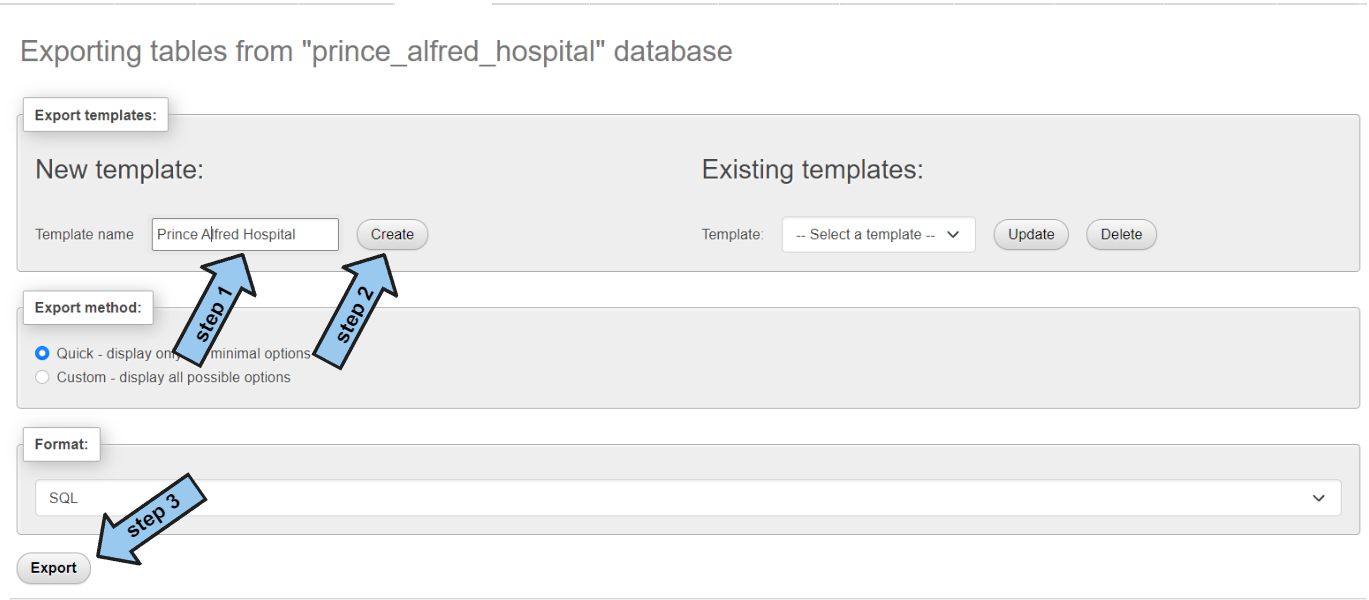


1. Click on the Export button to have a page that says: Exporting tables from "prince\_alfred\_hospital" database.

A screenshot of a computer

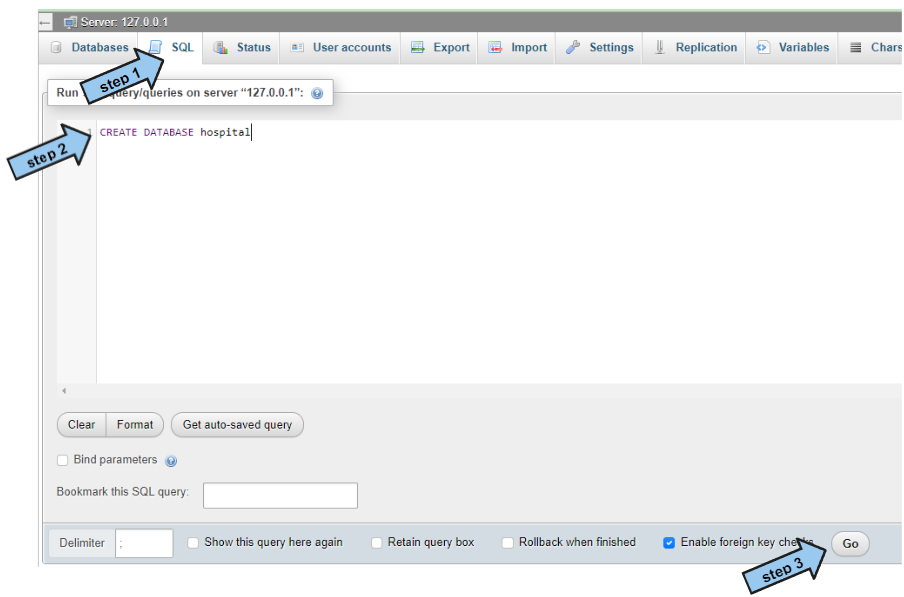
Description automatically generated with medium confidence

1. In the New template field, write a name for the template (step 1), press create (step 2) you will have a message that says template was created, then click on Export button (step 3) to download the file to the device:



In order to obtain the SQL database, do the following steps:

1. First when you access the phpMyAdmin page you will click on the SQL button from the server directly, then write the command create database followed by the name you want, and then press Go:

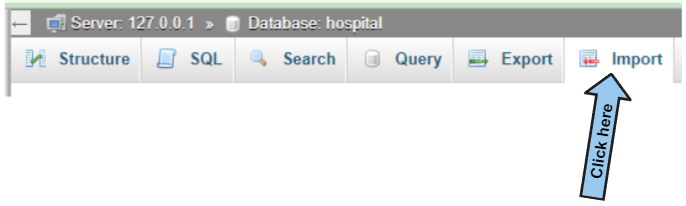
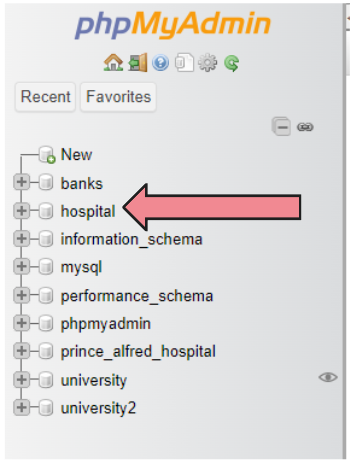


1. You will get a massage that assure you that the database was created:

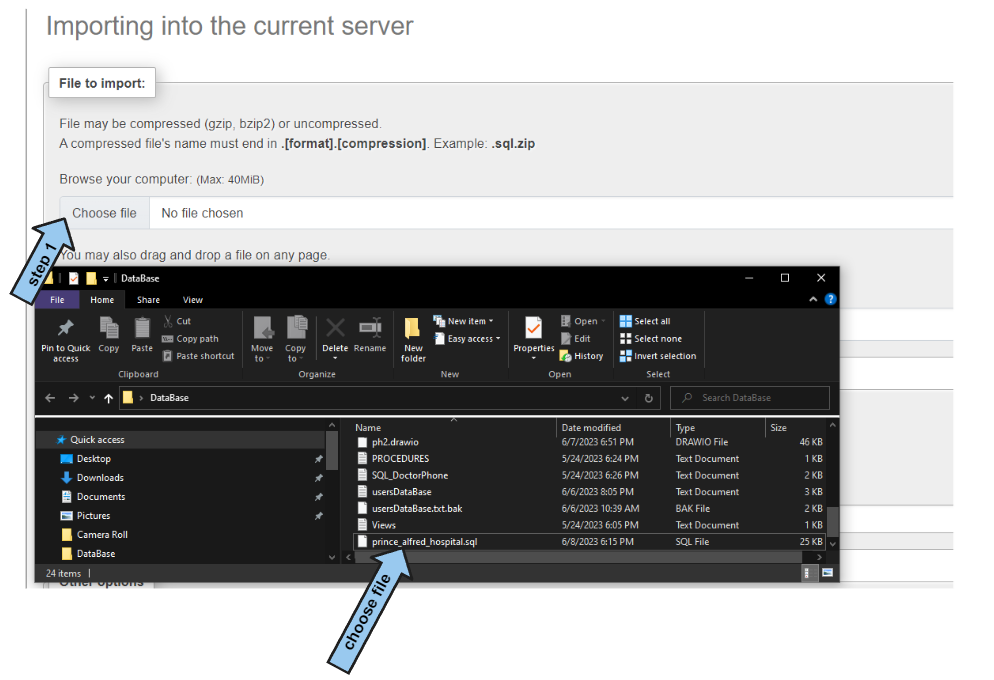
A picture containing text, screenshot, font, line

Description automatically generated

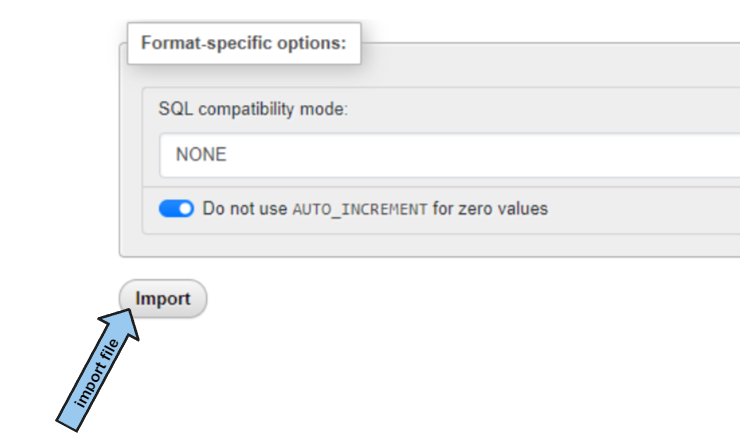
1. Click on the name of the database that you just created and then click on the import button:



1. You will have a page that contains a field that says File to import, click on choose file, then choose the file you want to download:



1. After you have chosen the file name, click on the Import button at the left end of the page:



Now the database is downloaded under the hospital name:

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

4.2- Database maintenance in general:

Database maintenance is considered to be a group of operations that could be used to make sure that the database is operating correctly. For example, database management systems, such as the SQL server and Oracle, requires frequent updates in order to keep up with the new technologies and to avoid new security threats.

Some examples on database maintenance are Data export/import operations, security audit (follow certain security rules), upgrading database software, monitor database performance.

Database Export/Import operations are critical as they enable individuals to bounce back after having system failures or errors, which might lead to data corruption, or even hardware failures. Therefore, they allow individuals to back up or move parts of the database to a different system for backup purposes. (Odogwu, 2022)

1. Testing:
   1. Data Validation:

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Type** | **Description** | **screenshot** |
|  | All cases of PK | I will try inserting a primary key that already exists in the doctors table |  |
| I will try inserting a doctor without a primary key |  |
|  | All cases of FK | I will try inserting a phone number in the doctors phone number table for a doctor who doesn’t exists in the doctors table. |  |
| I will try updating a doctors id, and see wither the id will change in the doctors phone number table or not. |  |
| I will try deleting a doctor from the doctors table and see wither it get’s deleted in the doctors phone number or not. |  |
|  |  |
|  | Unique | I will try inserting a new department that has the same name as another department. |  |
|  | Default | I will try inserting a new patient without address, it is supposed to fill unknown automatically |  |
|  | Not null | I will try inserting an appointment information without a doctors id |  |
|  | Check | I will try inserting a department number less than or equal to 0 |  |
|  | INT | I will try inserting characters instead of INT in department number |  |
|  | Varchar(10) | I will try inserting a doctors phone more than 10. | it took 10 only |
|  | date | I will try to inter characters | It dosen’t take it |
|  | Time | I will try to inter characters | It dosen’t take it |
|  | Multi-valued | I will try to add another phone number for the same doctor |  |
|  | One to one | The tables, appointments and medical record are one to one relationship. I will try to insert multiple same appointment for multiple medical records |  |
|  | One to many | I will try to enter multiple patients for the same appointment.  And to check if a patient can have many appointments |  |
|  | Many to many | I want to check if the same appointment has many departments.  And if a department can have many appointment |  |

* 1. Output validation:

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Query Description** | **Screenshot (query + result)** | **Result validation** |
|  | Inserting a new doctor to the doctors table |  | The doctor was added to the doctors table |
|  | Updating a doctors information |  | Dr. lubna lenar specialty was changed to eye surgery. |
|  | Creating view for patient to view his appointments |  | The query was succefful and all the informations were available in the view for patients to look at their appointments |
|  | Creating view for nurse for appointments to make nurses view appointments after 2020 |  | the query was successful and the all the information the user nurse needs for appointments is in the table.  I selected a certain patient using his Id and all his appointment information were shown |
|  | Creating medical record view for patients |  | The medical record view table was made, and all medical record information where included.  I selected the patient information based on his Id from the table, and it was shown correctly |
|  | Creating Appointment view for doctors |  | The doctors appointment view was successfully made and the table showed all the information they need. |
|  | Creating medical record view for nurses |  | The medical record view table for nureses was sucesfuly made and all information was constructed clearly.  Using a patient id the nurse was able to view his medical record information. |
|  | Creating view that counts number of appointments for each doctor | Using the user managements | The view that shows the number of appointments for each doctor was succeasuly made.  I tried using this view using the managments user and the managments were able to use this view. |
|  | Creating view that shows the total number of medical record in the database | Using the user managements | The view that counts the number of medical records in the database was made successfully.  The user management was able to view the number of medical record. |
|  | Creating procedure getPatientById and try using it |  | the procedure was made successfully, and I was able to get the patient information by their id |
|  | Creating procedure GetDoctorBySpeciality and try using it |  | the procedure was made successfully, and I was able to get the doctor information, such as all their phone number with the email. Only by entiring the speciality. It prited out two doctors with the same speciality, and deffirent phone numbers |
|  | Creating procedure getDepartmentByNumber and try using it |  | the procedure was made successfully, I entered the number of a department and it printed it’s name and floor. |
|  | Creating procedure getAppointmentByDate and try using it |  | the procedure was made successfully, and I was able to get all the information related to appointments in a specific date |
|  | Creating procedure getMedicalRecordByPatientID and try using it |  | the procedure was made successfully, and I was able to get a patient medical record information simply by their id |
|  | Creating procedure NurseScheduleAppointment and try using it |  | the procedure was made successfully, I was able to add a new appointment to the appointment table |
|  | Creating procedure updatePatientMedicalRecord and try using it | Medical record table before:        After: | the procedure was made successfully, Iwas able to update an existing medical record and it was changed in the medical record table |
|  | Creating procedure AddPatientMedicalRecord and try using it | An appointment number that already exists: (one-to-one)    New medical record: | the procedure was made successfully, I was able to add a new medical record to the medical record table, simply by inderting the data needed |
|  | Creating procedure AddPatients and try using it |  | the procedure was made successfully, I was able to add a patient to the patient table simply by inserting the patient’s information. And it was added to the patients table |
|  | Creating procedure AddAppointmentDepartment and try using it |  | the procedure was made successfully, I added a new department to an appointment and it was added to the department appointment table. |
|  | Creating procedure deleteAppointment and try using it | I will delete appointment 21 is is supposed to be deleted from the appointment table and appointment department:  Before:      Procedure:      Tables after: | the procedure was made successfully, I tried deleting an appointment from the appointment table using the procedure, and it was removed from the main table and all the other table that has the appointment as foreign key |
|  | Creating procedure deletePatient and try using it |  | the procedure was made successfully, I tried deleting a patient from the patient table and it was remove. |
|  | Creating procedure deleteDoctor and try using it |  | the procedure was made successfully, I tried deleting a doctor from the doctors table, and it was removed. |
|  | Creating procedure addDoctor and use it |  | the procedure was made successfully, I tried adding a new doctor to the doctor table, I simply added the doctor’s information, and it was added to the doctors table. |
|  | Creating procedure editDoctor and use it |  | the procedure was made successfully, I tried changing a doctor information, such as speciality and email, and it was changed in the doctors table. |
|  | Creating procedure editPatient and use it |  | the procedure was made successfully, I tried changing a patient information, such as date of birth , and it was changed in the patient’s table. |

* 1. Security Validation: given and not given privileges

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **User Name** | **Description of privilege/no privilege** | **Screenshot (query + result)** |
|  | Nurse | Able to select, and update, a medical record. But not able to delete a medical record. |  |
|  | Nurse | Able select Nurse\_appointment\_view\_after2020 |  |
|  | Nurse | Able to select, delete and update, an appointment. |  |
|  | Patient | Able to select, update, and delete on the patients\_appointment\_view |  |
|  | Patient | Able to select, update, and delete on patients\_medical\_record |  |
|  | Doctor | Able to insert, update, delete, select appointments table |  |
|  | Managment | Able to SELECT,INSERT, UPDATE, DELETE ON doctors table |  |

* 1. GUI Validation

|  |  |  |
| --- | --- | --- |
| **Number** | **Description** | **screenshot** |
|  | This screen shows the tables and views available for the management to use. |  |
|  | This is the department table and the users are able to insert, update, delete, and select information on it |  |
|  | Doctors table: on this table managers are able to insert, update, delete, and select information on it |  |
|  | Patients table: on this table managers are able to insert, update, delete, and select patients |  |
|  | Medical record count page, shows the number of total medical record in the data base |  |
|  | Number Of Appointments For Doctors page shows the total number of appointments that each doctor had over the years |  |

* 1. Assess whether meaningful data has been extracted:

First of all, the data that was inserted in each table was considered as a real-life data.

for example, the doctor’s table’s information was life-like, the id for the doctors were the same number of characters, and each doctor had their unique id number, which make sense because there can’t be two doctors with the same id. The doctor’s name was full, clear and was made from real world names. Moreover, the doctor’s specialty column contains real specialty name. There were multiple doctors with the same specialty, and this is reasonable because multiple doctors could have the same specialty in a hospital. The doctor’s email address was made like a life-like email address.

The appointment table contained unique appointment numbers, it contained patients’ id and doctors’ id that exists in the table. It also contained real dates and time for each appointment.

The doctors phone number table contained the phone number for each doctor, and it was clear that a doctor could have multiple phone numbers.

The department table contained unique numbers for each department. The name of the departments was filled with real-life department names.

From that data validation table, we got information about the way the tables interact with each other.

for example, we have known that the relationship between medical record and appointment table is a one-to-one relationship, which means that each appointment can only have one medical record, and each medical record can have only one appointment related to it.

Also, we know that each appointment can be related to only one patient. However, each patient could have multiple appointments.

Moreover, it was clear that each appointment could have many departments to be visited, and that each department can be visited by multiple appointments.

* 1. Assess the effectiveness of testing.

Testing is a very important step in database development. Testing helps us evaluate the data quality as well as the effectiveness of the database. Testing is essential to help developers find errors, missing data, as well as saving time of revisiting the database multiple times in the future. Before I mention my own work in testing, there are many benefits of database testing:

* Ensuring the accuracy of the dataset by identifying errors, making sure that there could be no violation of constraints, and that there are no difference between the information in a certain column for example.
* Testing with real world data (life-like) will give as an overall evaluation for our dataset. For example, we could measure the time for performing data, we could Identify certain errors when trying to insert new data.
* Testing ensures that our dataset doesn’t have vulnerabilities, or misconfiguration when trying to access methods, users authentication.

To start with my own work in testing:

Data validation has helped me to make sure that all tables of the database are working in the way they are intended to.

First of all, I made sure that all primary keys in the table are unique and that they all contain values, by trying to insert a primary key that already exists, or by trying to insert information without a primary key. And both of these had given me an error message showing that it’s working.

The foreign key cases were all working correctly. I tried to insert data that isn’t available in the main table, which gave me an error message, which means that it’s working correctly. And when I tried changing the doctor’s id, the foreign key in the doctor’s phone number table(doctor Id) was also changed. Same when I tried deleting a doctor from the doctors table, it was removed from the doctor’s phone number table.

When I tried inserting the same value for a column that is considered unique, such as the name in the department table, an error message occurred indicating that is working.

I tried not inserting information in the column address in the patients table, which has a default value(unknown) it worked, and the address was filled with unknown.

I tried inserting information for a new appointment to see wither it will be inserted or not, an error message occurred indicating that it cannot be null.

When I tried inserting a negative or zero value for the department number, an error message occurred indicating that there shouldn’t be a negative value. therefore the (check) constraint worked.

When I tried inserting a character in an integer value an error message occurred. Also, when I tried inserting a phone number for a doctor that contains more than 10 characters, it only saves the first 10.

I examined the one-to-one relationship between the medical record and appointment table by inserting multiple medical records for the same appointment, as well as inserting multiple appointments for the same medical record, an error message occurred in both of them. which protects the one-to-one relationship rule.

For the one-to-many relationship I tried inserting multiple patients for the same appointment, and an error message occurred. Which protects the one-to-many relationship.

For the many-to-many relationship I tried checking if I was able to add multiple appointments for a department, as well as adding multiple departments for an appointment.

For the output validation, I tried inserting, updating, selecting, deleting methods on tables, views, procedure, and they all worked as expected.

For the security validation, I tried performing allowed and unallowed methods on different tables, and views for the users in the database. And as expected the allowed methods worked successfully, and the denied privileges gave an error message indicating that this method shouldn’t be performed.

For the GUI validation, I tried using all the features of each page in the user website. And I was able to perform all the operations allowed for the user. And I made sure that I was able to access all the pages in the website.

1. Evaluation of database solution:
   1. Effectiveness of the database solution based on user and system requirement

The effectiveness of the implementation of a database can be measured by its ability to correctly store and present the information in order to meet the user's needs and the requirements’ demands.

To start from the first step, which is creating the tables:

Almost every table in the database contains columns with some related constraints. Some of the columns have the constraint NOT NULL. This constraint ensures that certain columns must be filled, as not filling it would result in some kind of misunderstanding or incorrectness. For example, the table appointment has the columns doctor id and patient id with a constraint that it shouldn’t be null, and it makes sense because the table appointment would be useless without this information.

In addition, some columns contain a constraint that checks wither an integer number is bigger than 0. For example, the table departments have this constraint for the department number column. And it makes sense because there shouldn’t be a department number less than or equal to 0.

Another constraint is the unique value. I have this constraint for the department name in the department table. I added this constraint because we should only have one department that does a specific function in the hospital, and adding two departments with the same name is misleading and unreasonable.

Moving on to the primary key constraint. This constraint sets a specific column as a primary key. This helps in insuring that this table has a specific important value that we can deal directly with, and to ensure that values in this column should be filled and that it cannot be duplicated (it should be unique). For example, the table medical record has the medical record number as the primary key, which makes each medical record number unique and can’t be repeated, which benefits the users when they want to look for specific details. Also, it ensures that this column is not null, also for the previous reasons.

Moving on to the primary key constraint, the primary key constraint helps us when we take a specific column from a table and make a copy for it in another table, to make sure that any modifications on the same table will result in the other table. For example, I used the foreign key (constraint on update cascade on delete cascade) on the doctor’s phone number table for the doctor id column, therefore if a doctor gets deleted, all his phone numbers will be deleted. Or if a doctor’s id numbers were modified, it will be modified in the doctor’s phone number table.

Moving on to the database filling:

I made sure to insert valuable and understandable information that is similar to a real-life database. This helps in making the database easier to detect problems or errors. As well as it’s easier when trying to make changes to the database. For example, I filled the doctors table with information that are very lifelike, which helped me in creating views, and procedures especially when it comes to joining multiple tables, as it would be hard if the data didn’t make any sense.

Moving on to the privileges:

I gave every user the privileges that makes sense for them, and not in a random way. For example, I gave the patients the ability to select, update, or delete information on the patients\_appointment\_view and patients\_medical\_record to only be able to modify, delete, and view their appointments and medical records. But I didn’t make them able to make appointments as this is the nurse job.

The management interface:

The management interface shows the tables that management are able to look at or perform operations on it. For example, gave them the ability to perform the important functions such as selecting, updating, deleting, or inserting a doctor to the database, as normally this is the management’s normal functionalities.

* 1. Suggested improvements

I think that there are some requirements that I can add to the database to make it better. Because the hospital could get bigger in the future and might require some additional functionalities.

For example, I have the patient phone number as simple and I might be asked in the feature to turn it to a multi-valued, as the system could require multiple phone numbers for a patient. Or for example, I might have some doctors with multiple specialties.

Also, I might be asked to add the doctors and patients’ gender to the tables, which is currently unavailable. I might be asked to add more information on the appointments, such as what office was the appointment performed in.

I might need to add deep information on the patient’s previous health issues.

The hospital in the future might have different buildings and each building would have different functionalities, for example building that have appointments only for treating cancer patient, and another building for appointments related to surgeries and that has special doctors. And a department that contains doctors for normal appointments.

* 1. Evaluation based on improvements needed:

For the multiple patients phone numbers, and multiple specialties, I will need to drop the patient’s phone number, and the doctor specialty features from the dataset. And then I would have to create to tables; one for the patient’s phone numbers, which will include the patients id and the phone numbers. And another for the doctor’s specialty, which will include the doctor’s id and the doctor's phone number.

For the doctors and patients gender information, I will only have to add a column to the doctors and patients table with the name (gender). It can contain the value male as a default, and then I have to edit each patient individually to add their gender. And for future insertion it will be added the moment a patient or doctor is added.

To add more information related to the appointments I could simply add a column if I had to add information or multiple pieces of information. But if I had to add a lot of information to the appointment, I would prefer to create a new table that only contains the appointments information, with the appointment number as the primary key.

Too add more information about patients’ previous health issues for example, I could add another table that contains the patient's id and the previous health issues as multi-valued. Therefore, it would be easier to fix certain information.

If the hospital get’s bigger and I had to have a department for each doctor, I would keep the doctors table as is, and I would create a new table that contains the doctor’s id and the name and number of the department.

References:

1. https://www.conceptdraw.com. (n.d.). Database Flowchart Symbols | ERD Symbols and Meanings | Data Flow Diagram Symbols. DFD Library | Symbols In Database Design. [online] Available at: <https://www.conceptdraw.com/examples/symbols-in-database-design#:~:text=A%20Data%20Flow%20Diagram%20%28DFD%29%20is%20a%20graphical,model%20and%20analyze%20the%20information%20processes%20and%20systems>.
2. Odogwu, C. (2022). What Is Database Maintenance and Why Is It Important? [online] MUO. Available at: <https://www.makeuseof.com/what-is-database-maintenance/>.