|  |  |
| --- | --- |
| **Course Title** | Phase 2 Integration Project |
| **Project Code** | PP2E v1-0 Project 2005-0430 |
| **Project Title** | Northern Lights Hospital |
| **Institution** | CDI College |
| **Project done by** | Barsha Shaikh |
| **Student ID** | 635-201244 |
| **Date Completed** | 3rd May 2021 |

Report:

If the business plan is to have lots of small customers in this case it’s the number of patient records with huge volumes of data, then single DB for each client is advisable.

Single database can be easily Administered and Managed. In terms of security, single database for each client is highly secured and most importantly easy to operate. It can be quickly implemented and Performance is rather good. It is very easy to build new modules and add new features. No need to wonder, where to fetch data. And also have higher resistance against data crashes.

 In Single database each and every application remains unique or at least not dependent on others.If one application crashes, then database of other applications will not be impacted.

Multiple databases are necessary if the site/application needs to be highly scalable (e.g. internet scale). For example, host each database on a different physical server.

 If we think that an application might grow so much in little time. it is better to use different database for each client.

There are a lot of times where you are deploying your application in stages with each stage having a roll-out/tutoring/sign-off phases parallel to parts already working, and in these cases it is cost effective to have sample/test deployment databases where users are inputting. Where it is much easier to delete the old database and create a new one than cleaning up old data.

However, do this only when you think it is worth the effort. Writing an application to support multiple databases is a time consuming affair.

Good reasons to create separate databases would be to support different availability requirements or simplify administration. For example if the databases requires very different backup schedules or different recovery models. Another reason would be if you may want to run them on different instances.

**Advantages of Single Database for all clients**

* Less Cost
* Grouping of data into multiple databases each with a significantly fewer number of tables.
* It is easier to maintain only one database instead of several. It is also a bit easier to collect statistics about the usage of the application if the database is shared. Administrator application is easier to develop as well
* In terms of flexibility, it's much simpler to use a single database with a single copy of the tables.
* It's easier to add new features

In this database we define the seven tables that is used to store information.

Admission Records

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data Type** | **Relationships** |
| Admission ID | Varchar(5) | Primary Key |
| Patient ID | Varchar(5) | Not Null |
| Bed Number | Varchar(5) | Foreign Key |
| Surgery Schedule | Date/Time | Not Null |
| Admit Date | Date/Time | Not Null |
| Surgery Date | Date/Time | Not Null |
| Discharge Date | Date/Time | Not Null |

Beds

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data Type** | **Relationships** |
| Bed Number | Varchar(5) | Foreign Key |
| Bed Type | Varchar(5) | Not Null |
| Occupied | Varchar(5) | Not Null |
| Ward Name | Varchar(10) | Primary Key |

Extra\_Rates

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data Type** | **Relationships** |
| Amenity Name | Varchar(20) | Primary Key |
| Daily Cost | int | Not Null |

Doctors

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data Type** | **Relationships** |
| Doctor ID | Varchar(5) | Foreign Key |
| Last Name | Varchar(20) | Not Null |
| First Name | Varchar(20) | Primary Key |
| Specialty | Varchar(20) | Not Null |

Extras

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data Type** | **Relationships** |
| Patient ID | Varchar(5) | Foreign Key |
| Admission Record ID | Varchar(5) | Primary Key |
| TV | Varchar(5) | Not Null |
| Phone | int | Not Null |
| Semi-Private | Varchar(5) | Not Null |
| Private | Varchar(5) | Not Null |

Login

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data Type** | **Relationships** |
| Username | Varchar(10) | Primary Key |
| Password | Varchar(10) | Not Null |

Patient

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data Type** | **Relationships** |
| Health Number | Varchar(5) | Primary Key |
| Date of Birth | Date/Time | Not Null |
| Last Name | Varchar(20) | Not Null |
| First Name | Varchar(20) | Not Null |
| Address | Varchar(50) | Not Null |
| City | Varchar(20) | Not Null |
| Province | Varchar(10) | Not Null |
| Postal Code | Varchar(10) | Not Null |
| Phone | int | Not Null |
| Insurance Company | Varchar(20) | Not Null |
| Insurance Number | int | Not Null |
| Next of Kin | Varchar(10) | Not Null |
| Next of KinRelatonship | Varchar(10) | Not Null |
| Doctor | Varchar(20) | Foreign Key |

An Activity Diagram:

**Admission Records**

Admission ID

Patient ID  
Bed Number

Surgery Schedule

Admit Date

Surgery Date

Discharge Date

**Extras**

Patient ID

Admission Record ID

TV

Phone

Semi-Private

Private

**Patients**

Health Number

Date of Birth

Last Name

First Name

Address

City

Province

Postal Code

Phone

Insurance Company

Insurance Number

Next of Kin

Next of KinRelatonship

Doctor

**Beds**

Bed Number

Bed Type

Occupied

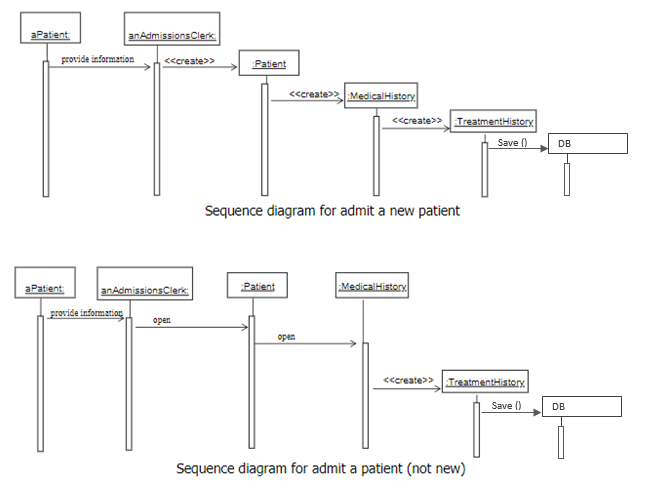
WardName

**Extra\_Rates**

Amenity Name

Daily Cost

Sequence Diagram:



Package Diagram:

Northern Lights Hospital

Database

(Consultation)

Reports

Human Resources

Accounts

Availability

Work Roaster

Leave Roaster

Remote Consulting

Booking Resources

Leave Roaster

Booking

Payment Billing

Registration

Management

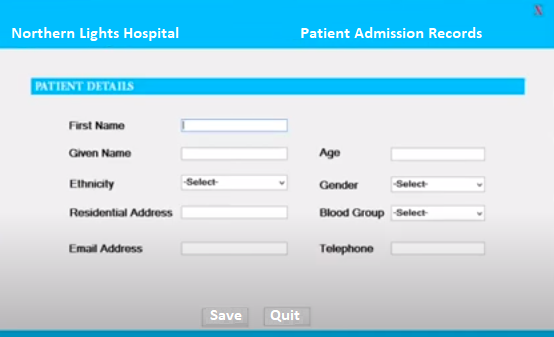
Staff

Web

Resources

Patient

Windows Form:



#includecstudio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

Void clrscr(void)

{

System(\*cls()\*);

}

void emp(void);

void menu(void);

void pat(void);

void inv(void);

struct address

{

Int nlh;

char street(40);

char city(40);

char state(40);

};

struct admrecords

{

char admid(5);

char patid(5);

char bedno(5);

char admitdate(10);

char surdate(10);

char disdate(10);

};

struct patient

{

char hlthno(40);

char dob(40);

char fname(40);

char lname(40);

int age;

char gender;

char city(10);

char province(10);

struct address a;

char ph(10);

char doc\_name(40);

};

struct doctors

char docid(5);

char fname(40);

char lname(40);

char spclty(40);

};

struct inventory

{

Int snc;

char name(30);

char dop(20);

int qnty;

float price;

float amount;

};//check

void menu()

{

char choice;

printf(“\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*”);

printf(“\n\t\t \*\*\* WELCOME TO HOSPITAL MANAGEMENT SYSTEM \*\*\*”);

printf(“\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*”);

printf(“\n\n\n\n\n\n\n”);

printf(“\t\t1.PATIENT”);

printf(“\n\t\t2.DOCTORS”);

printf(“\n\t\t3.INVENTORY”);

printf(“\n\t\t0.EXIT”);

printf(\n\n\tEnter your choice “);

fflush(stdIn);

switch(choice)

{

Case ‘1’;

