



Medikal Estetik Kliniğinde Hasta Yönetim Sistemi

Yazılım Mühendisliği Ana Bilim Dalı

Yüksek Lisans Tezi

Berkay Eceoğlu

Tez Danışmanı: Prof. Dr. Ayтуğ Onan

Haziran 2023

İzmir Kâtip Çelebi Üniversitesi Fen Bilimleri Enstitüsü öğrencisi **Berkay Eceođlu** tarafından hazırlanan **Medikal Estetik Kliniđinde Hasta Yönetim Sistemi** başlıklı bu çalışma tarafımızca okunmuş olup, yapılan savunma sınavı sonucunda kapsam ve nitelik açısından başarılı bulunarak jürimiz tarafından YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

ONAYLAYANLAR:

Tez Danışmanı: **Prof. Dr. Aytuđ Onan**
İzmir Kâtip Çelebi Üniversitesi

Yazarlık Beyanı

Ben, **Berkay Eceođlu**, bařlıđı **Medikal Estetik Kliniđinde Hasta Yönetim Sistemi** olan bu tezimin ve tezin iinde sunulan bilgilerin řahsıma ait olduđunu beyan ederim.

Ayrıca:

- Bu alıřmanın bütünü veya esası bu üniversitede Yüksek Lisans derecesi elde etmek üzere alıřtıđım süre iinde gerçekleştirilmiştir.
- Daha önce bu tezin herhangi bir kısmı başka bir derece veya yeterlik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış alıřmalarına başvurduğum durumlarda bu alıřmalara açık biçimde atıfta buldum.
- Başkalarının alıřmalarından alıntıladıđımda kaynađı her zaman belirttim. Tezin bu alıntılar dıřında kalan kısmı tümüyle benim kendi alıřmamdır.
- Kayda deđer yardım aldıđım bütün kaynaklara teřekkür ettim.
- Tezde başkalarıyla birlikte gerçekleştirilen alıřmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

Tarih:

09.06.2023



Medikal Estetik Kliniğinde Hasta Yönetim Sistemi

Öz

Medikal Estetik Kliniğinde Hasta Yönetim Sistemi, estetik klinik merkezi faaliyetlerinin günlük operasyonları ve yönetimi ile ilgilenmek için tasarlanmış ve programlanmış organize bir bilgisayar sistemidir. Yoğun kliniklerde hasta takibini kolaylaştırmak, klinisyene basit bir arayüz sunmak ve yerel veritabanı kullanarak güvenli işlem yapılmasına olanak vermek önemlidir. Program klinisyenin hasta girişlerine, kayıtlarına, tanı ve tedavilerine, hekim bilgilerine, randevulara, kliniğin fatura verilerine bakmasını ve yerel veritabanından güvenli bir şekilde verileri aktarılmasını sağlar.

Anahtar Sözcükler: Medikal, hasta yönetimi, veritabanı, sistem

Patient Management System in Medical Aesthetics Clinic

Abstract

The Patient Management System in Medical Aesthetic Clinics is an organized computer system designed and programmed to handle the daily operations and management of aesthetic clinic centers. Facilitating patient tracking in busy clinics, providing a user-friendly interface to clinicians, and enabling secure transactions using a local database are crucial. The program allows clinicians to view and securely transfer data from the local database regarding patient entries, records, diagnoses and treatments, physician information, appointments, and clinic billing data.

Keywords: Medical, patient management, database, system.

Table of Contents

| | |
|--|----------|
| Authorship Statement (Yazarlık Beyanı) | ii |
| Öz | iii |
| Abstract | iv |
| List of Figures..... | vii |
| List of Tables | viii |
| List of Abbreviations | ix |
| Chapter 1:Introduction | 1 |
| 1.1 Introduction | 1 |
| 1.2 Problem Introduction..... | 2 |
| 1.3 Goals | 3 |
| 1.4 Objectives | 3 |
| 1.5 Scope of the Project | 4 |
| 1.6 Modules | 5 |
| 1.6.1 Login(Home) Page | 5 |
| 1.6.2 User Page | 5 |
| 1.6.3 Patient Page | 5 |
| 1.6.4 Treatment Page | 6 |
| 1.6.5 Finance Page | 6 |
| Chapter 2: Design | 7 |
| 2.1 System Design | 7 |
| 2.1.1 Introduction to UML | 7 |
| 2.2 UML Approach..... | 8 |
| 2.2.1 Use Case Diagram of the Project | 10 |
| 2.2.2 Class Diagram..... | 10 |
| 2.2.3 Sequence Diagram..... | 11 |

| | | |
|-------------------|------------------------------------|-----------|
| 2.2.3 | Deployment Diagram | 11 |
| 2.2.3 | ER Diagram | 12 |
| Chapter 3: | Analysis..... | 13 |
| 3.1 | Existing System..... | 13 |
| 3.2 | Proposed System..... | 13 |
| 3.3 | Feasibility Study | 13 |
| 3.3.1 | Economic Feasibility | 13 |
| 3.3.2 | Technical Feasibility | 14 |
| 3.3.3 | Operational Feasibility | 14 |
| 3.4 | Software Specification | 14 |
| Chapter 4: | Sample Screenshots..... | 18 |
| 4.1 | Login Page | 18 |
| 4.2 | Home Page..... | 19 |
| 4.3 | Users Page | 19 |
| 4.4 | Patients Page..... | 20 |
| 4.5 | Treatments Page..... | 20 |
| 4.6 | Financial Page..... | 21 |
| 4.7 | Databases | 21 |
| Chapter 5: | System Implementation | 22 |
| 5.1 | Login Page | 22 |
| 5.2 | Home Page..... | 24 |
| 5.3 | Users Page | 25 |
| 5.4 | Patients Page..... | 33 |
| 5.5 | Treatments Page..... | 36 |
| 5.6 | Financial Page..... | 38 |
| References | | 41 |

List of Figures

| | |
|--|----|
| Figure 2.1 Use case diagram of the project | 10 |
| Figure 2.2 Class diagram | 11 |
| Figure 2.3 Sequence diagram..... | 11 |
| Figure 2.4 Deployment diagram..... | 12 |
| Figure 2.5 ER diagram..... | 13 |
| Figure 4.1 Login page..... | 19 |
| Figure 4.2 Home page | 19 |
| Figure 4.3 Users page | 20 |
| Figure 4.4 Patients page..... | 20 |
| Figure 4.5 Treatments page..... | 21 |
| Figure 4.6 Financial page..... | 21 |

List of Tables

| | |
|--------------------------|----|
| Chart 1. Databases | 22 |
|--------------------------|----|

List of Abbreviations

| | |
|------|---------------------------------|
| PMS | Patient Management System |
| UML | The Unified Modeling Language |
| İKÇÜ | İzmir Kâtip Çelebi Üniversitesi |
| ID | Identification |
| ER | Entity-relationship |
| SQL | Structured Query Language |

CHAPTER 1: INTRODUCTION

1.1. Introduction

Medical aesthetic clinics play a significant role in the field of cosmetic medicine, providing a wide range of services to enhance patients' appearance and boost their self-confidence. As these clinics cater to a large number of patients on a daily basis, efficient management of patient information becomes crucial to ensure smooth operations and high-quality care. The advent of advanced computer systems has revolutionized the way healthcare facilities handle patient data, leading to the development of specialized software known as Patient Management Systems (PMS) for medical aesthetic clinics.

The Patient Management System is a computerized tool designed to streamline and automate various aspects of clinic operations, enabling clinicians to effectively manage patient information, appointments, diagnoses, treatments, and billing data. This system serves as a centralized repository, storing and organizing crucial patient data, allowing for efficient tracking and retrieval of information as needed.

In this modern era, where technology plays a vital role in almost every industry, the integration of a PMS in medical aesthetic clinics offers numerous advantages. Firstly, it provides clinicians with a user-friendly interface, simplifying the process of accessing and updating patient records. By eliminating the need for manual record-keeping and paperwork, the system reduces the chances of errors, improves accuracy, and enhances overall efficiency in clinic operations.

Furthermore, a PMS facilitates seamless communication and data exchange between clinicians within the clinic network. With secure data transfer protocols, physicians can access patient information from the local database, enabling collaborative decision-making and ensuring continuity of care. This functionality enhances the quality of care provided to patients, as clinicians have access to comprehensive medical histories and treatment records, allowing for informed diagnoses and personalized treatment plans.

Another critical aspect of a PMS is its ability to handle clinic billing data securely. By integrating billing features into the system, medical aesthetic clinics can streamline

their financial processes, generate accurate invoices, and track payments efficiently. This not only reduces administrative burden but also helps clinics maintain transparent financial records and optimize revenue management.

Overall, the Patient Management System in medical aesthetic clinics revolutionizes the way patient data is managed, improving the efficiency, accuracy, and security of clinic operations. By providing a user-friendly interface, seamless data transfer, and comprehensive billing capabilities, the system enhances the overall quality of care delivered to patients. In this study, we aim to explore the implementation and impact of a PMS in a medical aesthetic clinic setting, examining its effectiveness in improving clinic workflows, patient outcomes, and overall clinic performance.

1.2. Problem Introduction:

Inefficient Patient Tracking: Manual patient tracking in busy aesthetic clinics can be time-consuming and prone to errors. Clinicians struggle to quickly access and update patient information, leading to delays in appointments and potential confusion in diagnosis and treatment.

Fragmented Data Management: With paper-based systems, patient records, diagnoses, treatments, physician information, appointments, and billing data are often stored in separate files or registers. This fragmentation makes it challenging to retrieve and consolidate comprehensive patient information efficiently.

Security Risks: Traditional paper-based systems pose security risks, as physical records can be lost, damaged, or accessed by unauthorized individuals.

Manual Billing Processes: Manual calculation and generation of patient bills based on various treatments can be error-prone and time-consuming. Inaccurate billing information may lead to disputes, financial losses, and decreased patient satisfaction.

Limited Accessibility and Collaboration: Without a centralized digital system, clinicians face difficulties accessing patient information in real-time, especially when working remotely or across multiple clinic locations.

Lack of Data Analysis and Insights: Manual systems provide limited opportunities for analyzing patient data and deriving valuable insights. Clinics may miss out on identifying trends, optimizing treatment plans, and enhancing patient outcomes.

1.3. Goals:

- 1-User Friendly
- 2- Fast and simple
- 3- Effective, low cost
- 4- Patient data security and management
- 5- Analyzeable

1.4. Objective:

- 1-) Developing a comprehensive Patient Management System (PMS) specifically tailored for medical aesthetic clinics, addressing the unique needs and challenges of these clinics in managing patient information, appointments, diagnoses, treatments, physician details, and billing data.
- 2-) Enhancing the efficiency of clinic operations by providing a user-friendly interface for clinicians to easily access, update, and retrieve patient information, reducing the time and effort required for administrative tasks and improving overall workflow.
- 3-) Ensuring the accuracy and integrity of patient data by implementing secure data transfer protocols and a robust local database, minimizing the risk of errors, loss, or unauthorized access to sensitive information.
- 4-) Enabling seamless communication and collaboration among clinicians within the clinic network, facilitating the exchange of patient data, supporting informed decision-making, and ensuring continuity of care.

5-) To streamline the billing processes in medical aesthetic clinics, making calculations based on treatments and generating accurate invoices from database, reducing manual errors, and improving financial management.

1.5. Scope of the Project:

1. The project will focus on developing a customized Patient Management System for medical aesthetic clinics, considering the specific requirements and workflows of these clinics.
2. The system will encompass modules for patient entry, record management, diagnoses, treatments, physician information, appointment scheduling, and clinic billing.
3. The scope includes designing a user-friendly interface that is intuitive and accessible to clinicians, ensuring ease of use and efficient navigation through various functionalities.
4. The project will involve implementing robust data security measures to protect patient information, including secure data transfer protocols and encryption.
5. The system will be developed to operate on a local database, ensuring data availability and minimizing reliance on external networks or cloud-based solutions.
6. The project will not cover integration with other clinical systems, such as electronic health records or inventory management, focusing solely on the Patient Management System functionality.
7. The project will include testing, validation, and refinement of the developed system, ensuring its effectiveness, reliability, and compatibility with existing clinic infrastructure.
8. The project does not include hardware procurement or installation; it will focus on the software development and implementation aspects of the Patient Management System.
9. The project will consider the ethical and legal implications of managing patient data, complying with relevant privacy regulations and guidelines.
10. The project will provide documentation and guidelines for the future maintenance, support, and scalability of the developed Patient Management System in medical aesthetic clinics.

1.6. Modules:

The entire project mainly consists of 7 modules, which are

- ❖ Home-Login module
- ❖ User module
- ❖ Patient module
- ❖ Treatment module
- ❖ Finance module

1.6.1. Home(Login) Module:

- Shows the login page.
- A defined clinician or staff member can log in.
- Username and password are required for login.
- Allows exiting the program.
- Provides clearing of defined user name and password entry.

1.6.2. User Module:

- It contains User ID, User Name, Work Field and Password fields.
- A user can be added.
- Users can be updated.
- Users can be deleted.
- Users information can be exported.

1.6.3. Patient Module:

By the relevant clinician;

- Patient ID entry
- Patient name entry
- Patient address and contact information entry
- Patient age entry

- Gender entry
- Blood group entry
- Diagnosis and treatment introduction

1.6.4. Treatment Module:

By the relevant clinician;

- Prescription ID
- Patient ID
- Patient name
- Area
- Operation
- Treatment
- Entry and updates

are organized.

1.6.5. Finance Module:

By defined user or accountant;

- Account ID
- Patient ID
- Patient name
- Payment Type
- Total Payment
- Payment Date

CHAPTER 2: DESIGN

2.1. System Design

2.1.1. Introduction to UML

UML Design

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the software system and its components. It is a graphical language, which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure, maintain, and control information about the systems.

The UML is a language for:

- Visualizing
- Specifying
- Constructing
- Documenting

Visualizing

Through UML we see or visualize an existing system and ultimately we visualize how the system is going to be after implementation. Unless we think, we cannot implement. UML helps to visualize, how the components of the system communicate and interact with each other.

Specifying

Specifying means building, models that are precise, unambiguous and complete UML addresses the specification of all the important analysis design, implementation decisions that must be made in developing and deploying a software system.

Constructing

UML models can be directly connected to a variety of programming language through mapping a model from UML to a programming language like JAVA or C++ or VB. Forward Engineering and Reverse Engineering is possible through UML.

Documenting

The Deliverables of a project apart from coding are some Artifacts, which are critical in controlling, measuring and communicating about a system during its developing requirements, architecture, design, source code, project plans, tests, prototypes releases, etc...

2.2. UML Approach

UML Diagram

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices and arcs . you draw diagram to visualize a system from different perspective, so a diagram is a projection into a system. For all but most trivial systems, a diagram represents an elided view of the elements that make up a system. The same element may appear in all diagrams, only a few diagrams , or in no diagrams at all. In theory, a diagram may contain any combination of things and relationships. In practice, however, a small number of common combinations arise, which are consistent with the five most useful views that comprise the architecture of a software-intensive system. For this reason, the UML includes nine such diagrams:

1. Class diagram
2. Object diagram
3. Use case diagram

4. Sequence diagram
5. Collaboration diagram
6. State chart diagram
7. Component diagram
8. Deployment diagram

USE CASE DIAGRAM:

A usecase diagram in the Unified Modeling Language(UML) is atype of behavioral diagram defined by and created from a use-case analysis.its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals(represented as use cases),and any dependencies between those use cases.

2.2.1. Use Case Diagram of the Project

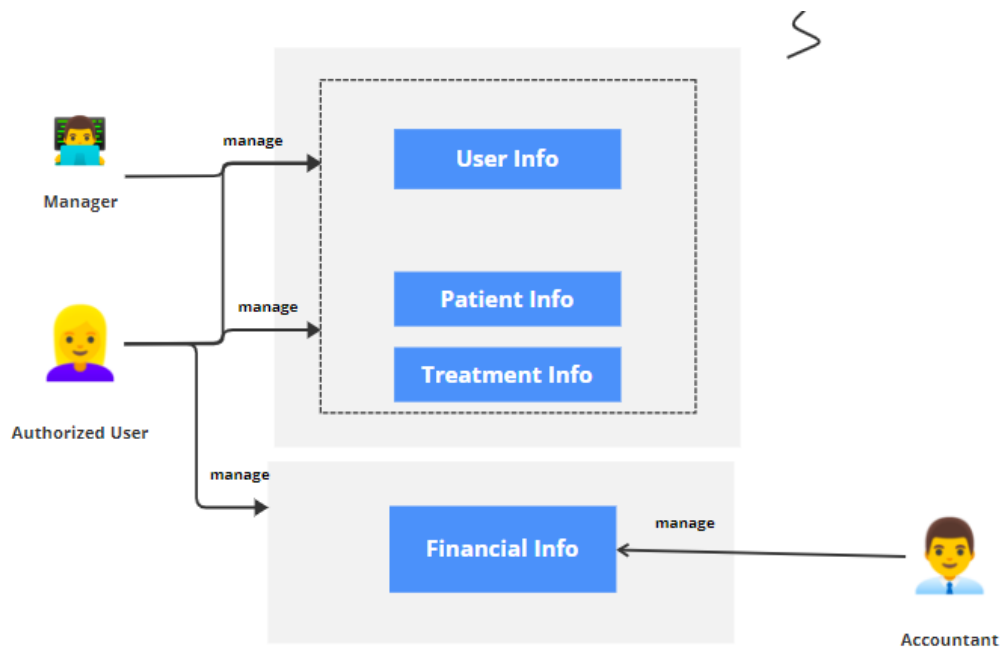


Figure 2.1

2.2.2. Class Diagram

A Class is a category or group of things that has similar attributes and common behavior. A Rectangle is the icon that represents the class it is divided into three areas. The upper most area contains the name, the middle; area contains the attributes and the lowest areas show the key. [4]

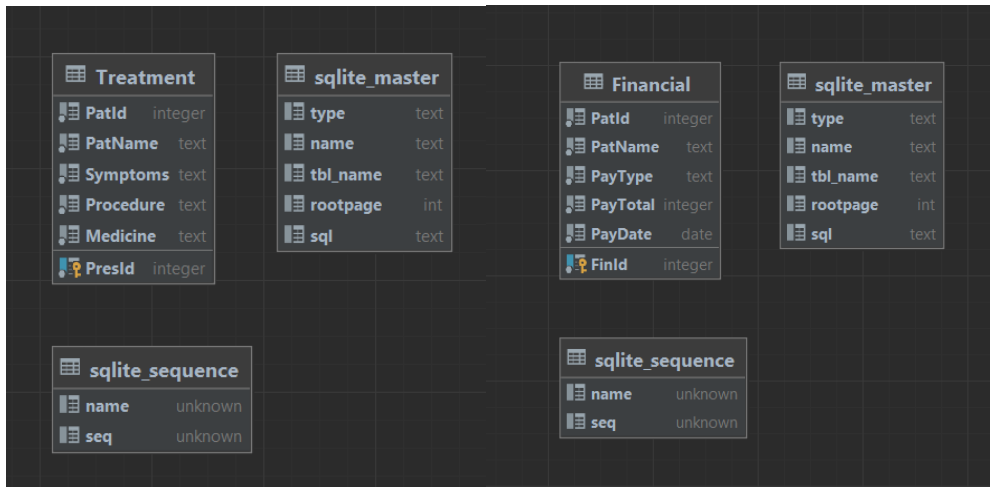
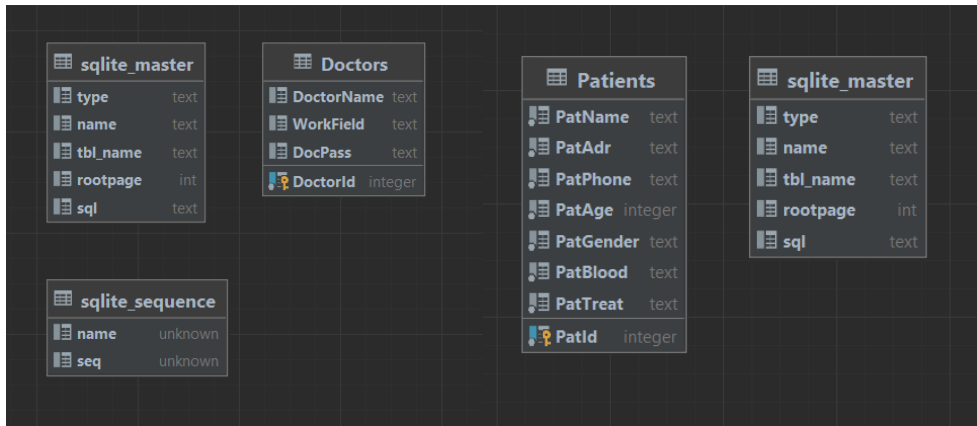


Figure 2.2

2.2.3. Sequence Diagram

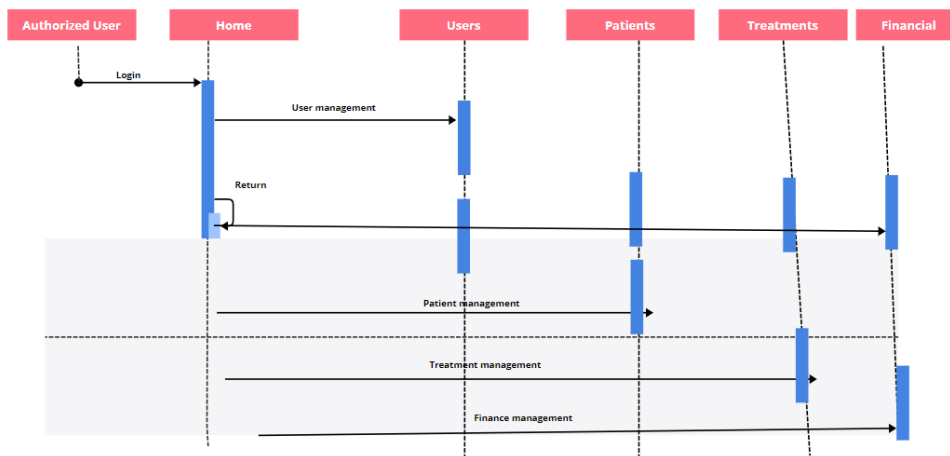


Figure 2.3

2.2.4. Deployment Diagram

A Deployment Diagram shows the configuration of run-time processing nodes and the components that live on them. Deployment diagrams address the static deployment view of architecture. They are related to component diagrams in that a node typically encloses one or more components.

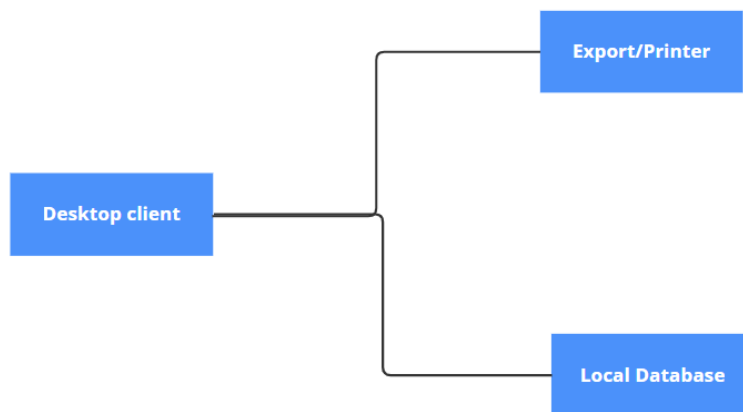


Figure 2.4

2.2.5. ER Diagram

Database is absolutely an integral part of software system. To fully utilize ER Diagram in database engineering guarantee you to produce high quality database design to use in database creation, management and maintenance. An ER model also provides a means for communication. [1]

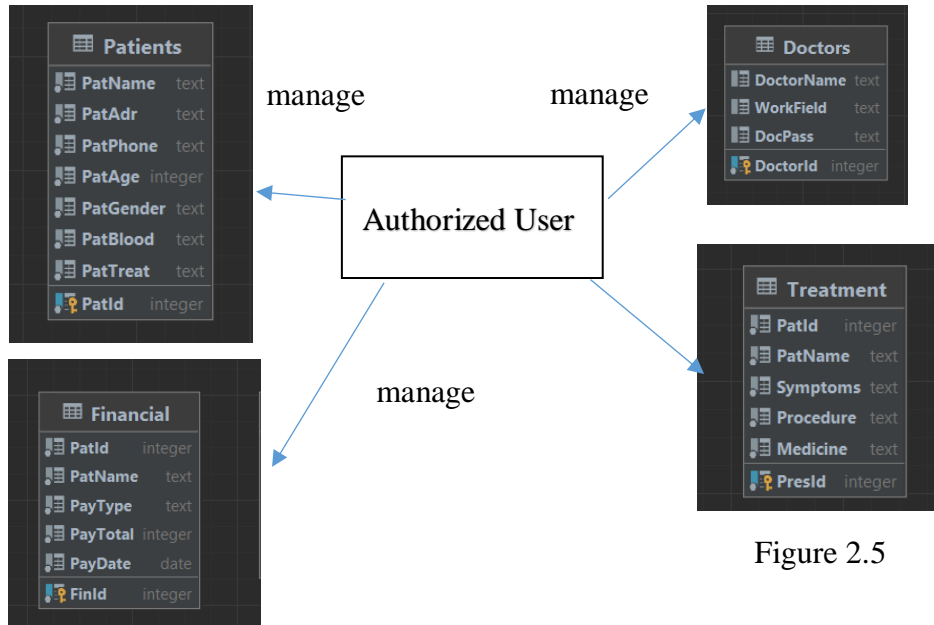


Figure 2.5

CHAPTER 3: ANALYSIS

3.1. Existing System

Clinics currently use a manual system for the management and maintenance of critical information. The current system requires numerous paper forms, with data stores spread through out the hospital management infrastructure. Often information is incomplete or does not follow management standards. Forms are often lost in transit between departments requiring a comprehensive auditing process to ensure that no vital information is lost. Multiple copies of the same information exist in the hospital and may lead to inconsistencies in data in various data stores.

3.2. Proposed System

The Patient Management System in Medical Aesthetics Clinic is designed for any clinic to replace their existing manual paper based system. The new system is to control the information of patients. Doctors, treatment information and patient invoices. These services are to be provided in an efficient, cost effective manner, with the goal of reducing the time and resources currently required for such tasks .

3.3. Feasibility Study

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

3.3.1. Economic Feasibility

This study is carried out to check the economic impact will have on the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customised products have to be purchased.

19

3.3.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes for the implementing this system.

3.3.3 Operational Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.4. Software Specification

C#:

C# (pronounced "C sharp") is a widely-used, object-oriented programming language developed by Microsoft. It was introduced as part of the .NET initiative and has gained popularity for its simplicity and versatility. C# is primarily used for developing applications on the Microsoft platform, including Windows desktop applications, web applications, and games. It offers a modern, type-safe programming environment with features such as automatic memory management through garbage collection. C# provides a rich set of libraries and frameworks that simplify common programming tasks, such as user interface design, database access, and networking. With its robust ecosystem and strong integration with the .NET framework, C# enables developers to create efficient, scalable, and reliable software solutions. [5]

.NET Framework:

The .NET Framework is a comprehensive software development platform created by Microsoft. It provides a programming model, a large set of class libraries, and a runtime environment for developing and running various types of applications. The .NET Framework supports multiple programming languages, with C# being one of the most commonly used languages. It offers a wide range of functionality, including user interface development, database connectivity, networking, and security. The framework promotes code reuse, scalability, and interoperability, allowing developers to build robust and cross-platform applications. The .NET Framework has a rich ecosystem of tools, libraries, and frameworks that facilitate software development and

enable developers to create efficient and reliable solutions for different domains, ranging from desktop applications to web services and cloud-based applications. [6]

MySQL:

MySQL is developed, distributed, and supported by Oracle Corporation. MySQL is a database system used on the web it runs on a server. MySQL is ideal for both small and large applications. It is very fast, reliable, and easy to use. It supports standard SQL. MySQL can be compiled on a number of platforms.

The data in MySQL is stored in tables. A table is a collection of related data, and it consists of columns and rows. Databases are useful when storing information categorically. [7]

FEATURES OF MySQL:

Internals and portability:

- Written in C and C++.
- Tested with a broad range of different compilers.
- Works on many different platforms.
- Tested with Purify (a commercial memory leakage detector) as well as with Valgrind, a GPL tool.
- Uses multi-layered server design with independent modules.

Security:

- A privilege and password system that is very flexible and secure, and that enables host-based verification.
- Password security by encryption of all password traffic when you connect to a server.

Scalability and Limits:

- Support for large databases. We use MySQL Server with databases that contain 50 million records. We also know of users who use MySQL Server with 200,000 tables and about 5,000,000,000 rows.

- Support for up to 64 indexes per table (32 before MySQL 4.1.2). Each index may consist of 1 to 16 columns or parts of columns. The maximum index width is 767 bytes for InnoDB tables, or 1000 for MyISAM; before MySQL 4.1.2, the limit is 500 bytes. An index may use a prefix of a column for CHAR, VARCHAR, BLOB, or TEXT column types.

CONNECTIVITY:

Clients can connect to MySQL Server using several protocols:

- Clients can connect using TCP/IP sockets on any platform.

- On Windows systems in the NT family (NT, 2000, XP, 2003, or Vista), clients can connect using named pipes if the server is started with the `--enable-named-pipe` option. In MySQL 4.1 and higher, Windows servers also support shared-memory connections if started with the `--shared-memory` option. Clients can connect through shared memory by using the `--protocol=memory` option.

- On UNIX systems, clients can connect using Unix domain socket files.

LOCALIZATION:

- The server can provide error messages to clients in many languages.

- All data is saved in the chosen character set.

CLIENTS AND TOOLS:

MySQL includes several client and utility programs. These include both command-line programs such as `mysqldump` and `mysqladmin`, and graphical programs such as MySQL Workbench.

MySQL Server has built-in support for SQL statements to check, optimize, and repair tables. These statements are available from the command line through the `mysqlcheck` client.

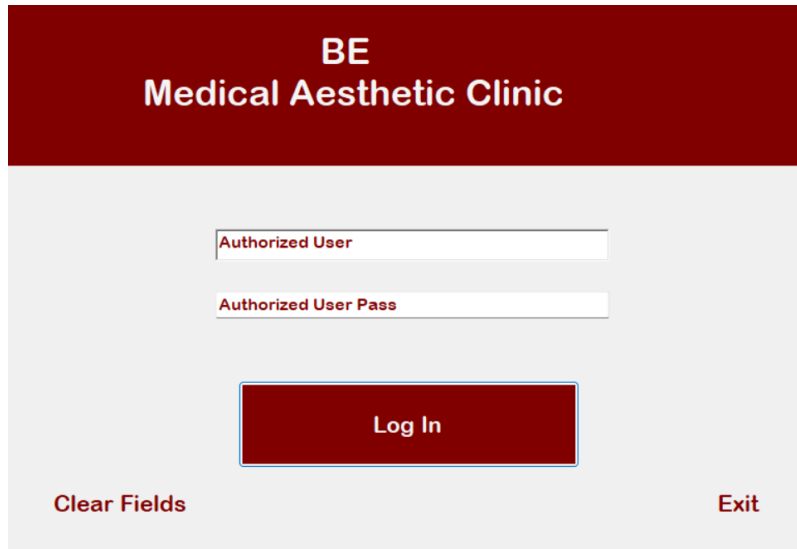
MySQL programs can be invoked with the `--help` or `-?` option to obtain online assistance.

DataGrip is a powerful and versatile integrated development environment (IDE) designed specifically for working with databases. Developed by JetBrains, DataGrip provides comprehensive functionality for database management, allowing developers and database administrators to efficiently work with various database systems. It supports a wide range of databases, including relational databases like MySQL, PostgreSQL, Oracle, SQL Server, as well as NoSQL databases such as MongoDB and Cassandra.

Microsoft Visual Studio is a popular integrated development environment (IDE) widely used by developers for creating a wide range of applications. It offers a comprehensive set of tools and features that streamline the development process across different platforms, including Windows, web, cloud, and mobile. Visual Studio supports multiple programming languages such as C#, C++, and JavaScript, and provides powerful code editing, debugging, and testing capabilities.

CHAPTER 4: SAMPLE SCREENSHOTS

4.1.Login Page



The screenshot shows a login interface for the BE Medical Aesthetic Clinic. At the top, there is a dark red header with the text "BE Medical Aesthetic Clinic" in white. Below the header, there are two input fields: "Authorized User" and "Authorized User Pass". A red "Log In" button is positioned below the password field. At the bottom left, there is a "Clear Fields" link, and at the bottom right, there is an "Exit" link.

Figure 4.1

4.2.Home Page



Figure 4.2

4.3.Users Page

Doctors

ADDUPDAT

DELETHome

EXPOR

| Doctor ID | Doctor ... | Work Fi... | Password |
|-----------|------------|------------|----------|
| 1321356 | Berkay ... | Clinics | 12345 |
| 3487324 | Ali Veli | Surgery | 42348727 |

Figure 4.3

4.4.Patients Page

Patients

ADDUPDAT

DELETHome

| Colu... | ColumnHea... | Column... | Column... | Column... | Column... | Column... |
|---------|--------------|-----------|-----------|-----------|-----------|-----------|
|---------|--------------|-----------|-----------|-----------|-----------|-----------|

Figure 4.4

4.5. Treatments Page

The screenshot shows a web page titled "Treatments" with a dark red header. On the left side, there is a form with the following fields: Prescription ID, Patient ID, Patient Name, Area, Operation, and Medicine. Below these fields are four buttons: ADD, UPDAT, DELET, and Home. On the right side, there is a table with five columns, each labeled "Column...".

Figure 4.5

4.6. Financial Page

The screenshot shows a web page titled "Financial" with a dark red header. On the left side, there is a form with the following fields: Account ID, Patient ID, Patient Name, Payment Type, Total Payment, and Payment Date. Below these fields are four buttons: ADD, UPDATE, DELETE, and Home. On the right side, there is a table with five columns, each labeled "Column...".

Figure 4.6

4.7. Databases

The screenshot displays a database management interface with four tables. Each table has a toolbar with navigation and action icons. The 'Doctors' table has 2 rows, 'Treatment' has 0 rows, 'Financial' has 0 rows, and 'Patients' has 1 row.

| DoctorId | DoctorName | WorkField | DocPass |
|----------|----------------|-----------|----------|
| 1321356 | Berkay Eceoglu | Clinics | 12345 |
| 3487324 | Ali Veli | Surgery | 42348727 |

| PresId | PatId | PatName | Symptoms | Procedure | Medicine |
|--------|-------|---------|----------|-----------|----------|
|--------|-------|---------|----------|-----------|----------|

| FinId | PatId | PatName | PayType | PayTotal | PayDate |
|-------|-------|---------|---------|----------|---------|
|-------|-------|---------|---------|----------|---------|

| PatId | PatName | PatAdr | PatPhone | PatAge | PatGender | Pa |
|--------|---------|----------|-------------|--------|-----------|----|
| 123654 | Efe Can | Istanbul | 05559655544 | 25 | Male | A- |

Chart 1.

CHAPTER 5: SYSTEM IMPLEMENTATION

5.1.Login Page

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MedicalOffice
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            doctorName.AllowDrop = true;
            doctorName.DragEnter += doctorName_DragEnter;
            doctorName.DragDrop += doctorName_DragDrop;
        }
    }
}
```

```

private void label1_Click(object sender, EventArgs e)
{
}

private void label2_Click(object sender, EventArgs e)
{
}

private void button1_Click(object sender, EventArgs e)
{
    string doctorNameInput = doctorName.Text;
    string doctorPassInput = doctorPass.Text;

    if (doctorNameInput == "Berkay Eceoglu" && doctorPassInput
== "12345")
    {
        Home home = new Home();
        home.Show();
        Hide();
    }
    else if (doctorNameInput == "Ali Veli" && doctorPassInput ==
"42348727")
    {
        Home home = new Home();
        home.Show();
        Hide();
    }

    else
    {
        MessageBox.Show("Invalid credentials. Please try
again.");
        doctorName.Clear();
        doctorPass.Clear();
        doctorName.Focus();
    }
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
}

private void clearF_Click(object sender, EventArgs e)
{
    doctorName.Text = string.Empty;
    doctorPass.Text = string.Empty;
}

private void btnExit_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Are you sure you want
to exit?", "Confirmation", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);

    if (result == DialogResult.Yes)
    {
        Application.Exit();
    }
}

```



```

private void doctorName_TextChanged(object sender, EventArgs e)
{
}

private void doctorName_DragEnter(object sender, DragEventArgs
e)
{
    if (e.Data.GetDataPresent(DataFormats.Text))
    {
        e.Effect = DragDropEffects.Copy;
    }
}

private void doctorName_DragDrop(object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent(DataFormats.Text))
    {
        string droppedText = e.Data.GetData(DataFormats.Text) as
string;
        doctorName.Text = droppedText;
    }
}

private void doctorPass_TextChanged(object sender, EventArgs e)
{
}

private void panell1_Paint(object sender, PaintEventArgs e)
{
}

private void Form1_Load(object sender, EventArgs e)
{
}
}
}

```

5.2.Home Page

```

namespace MedicalOffice
{
    public partial class Home : Form
    {
        public Home()
        {
            InitializeComponent();
        }

        private void label5_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form1 home = new Form1();
            home.Show();
        }

        private void label4_Click(object sender, EventArgs e)
        {
            DoctorF doctorForm = new DoctorF();
            doctorForm.Show();
        }
    }
}

```

```

    }

    private void label3_Click(object sender, EventArgs e)
    {
        PatientF patientForm = new PatientF();
        patientForm.Show();
    }

    private void label2_Click(object sender, EventArgs e)
    {
        Prescription prescriptionForm = new Prescription();
        prescriptionForm.Show();
    }

    private void label1_Click(object sender, EventArgs e)
    {
        Prescription prescriptionForm = new Prescription();
        prescriptionForm.Show();
    }

    private void pictureBox1_Click(object sender, EventArgs e)
    {
    }

    private void Home_Load(object sender, EventArgs e)
    {
    }

    private void label6_Click(object sender, EventArgs e)
    {
        Financial financialForm = new Financial();
        financialForm.Show();
    }
}
}

```

5.3.Users Page

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SQLite;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Drawing;
using System.Drawing.Printing;

```

```

namespace MedicalOffice
{
    public partial class DoctorF : Form
    {
        private string connectionString = "Data
Source=C:\\Users\\Berkay\\Desktop\\MedicalOffice-
main\\MedicalOffice\\bin\\Debug\\doctordb.db;Version=3";
        private DataSet dataSet;

        public class Doctor
        {
            public int DoctorId { get; set; }
            public string DoctorName { get; set; }
            public string WorkField { get; set; }
            public string DocPass { get; set; }
        }

        public DoctorF()
        {
            InitializeComponent();

            KeyPreview = true;
            KeyDown += DoctorF_KeyDown;

            ToolStripMenuItem fileToolStripMenuItem = new ToolStripMenuItem();
            ToolStripMenuItem printPreviewToolStripMenuItem = new
ToolStripMenuItem();
            ToolStripMenuItem pageSetupToolStripMenuItem = new
ToolStripMenuItem();

            fileToolStripMenuItem.Text = "Print";
            printPreviewToolStripMenuItem.Text = "Print Preview";
            pageSetupToolStripMenuItem.Text = "Page Setup";

            fileToolStripMenuItem.DropDownItems.AddRange(new ToolStripItem[] {
printPreviewToolStripMenuItem, pageSetupToolStripMenuItem });
            menuStrip1.Items.Add(fileToolStripMenuItem);

        }

        private void DoctorF_Load(object sender, EventArgs e)
        {
            lstvDoctors.View = View.Details;
            lstvDoctors.Columns.Add("Doctor ID");
            lstvDoctors.Columns.Add("Doctor Name");
            lstvDoctors.Columns.Add("Work Field");
        }
    }
}

```

```

        lstvDoctors.Columns.Add("Password");

        using (SQLiteConnection connection = new
SQLiteConnection(connectionString))
        {
            connection.Open();

            using (SQLiteDataAdapter adapter = new SQLiteDataAdapter("SELECT *
FROM Doctors", connection))
            {
                dataSet = new DataSet();
                adapter.Fill(dataSet, "Doctors");
            }

            connection.Close();
        }

        foreach (DataRow row in dataSet.Tables["Doctors"].Rows)
        {
            ListViewItem item = new ListViewItem(row["DoctorId"].ToString());
            item.SubItems.Add(row["DoctorName"].ToString());
            item.SubItems.Add(row["WorkField"].ToString());
            item.SubItems.Add(row["DocPass"].ToString());
            lstvDoctors.Items.Add(item);
        }
    }

private void button4_Click(object sender, EventArgs e)
{
    Home home = new Home();
    home.Show();
    this.Hide();
}

private void btnAddDc_Click(object sender, EventArgs e)
{
    int doctorId = Convert.ToInt32(textBox1DC.Text);
    string doctorName = textBox2DC.Text;
    string workField = textBox3DC.Text;
    string password = textBox4DC.Text;

    if (dataSet != null && dataSet.Tables.Contains("Doctors"))
    {
        DataRow newRow = dataSet.Tables["Doctors"].NewRow();
        newRow["DoctorId"] = doctorId;
        newRow["DoctorName"] = doctorName;
        newRow["WorkField"] = workField;
        newRow["DocPass"] = password;
    }
}

```

```

        using (SQLiteConnection connection = new
SQLiteConnection(connectionString))
        {
            connection.Open();

            string query = "INSERT INTO Doctors (DoctorId, DoctorName,
WorkField, DocPass) " +
                "VALUES (@doctorId, @doctorName, @workField,
@password)";

            SQLiteCommand command = new SQLiteCommand(query, connection);
            command.Parameters.AddWithValue("@doctorId", doctorId);
            command.Parameters.AddWithValue("@doctorName", doctorName);
            command.Parameters.AddWithValue("@workField", workField);
            command.Parameters.AddWithValue("@password", password);

            command.ExecuteNonQuery();

            connection.Close();
        }

        using (SQLiteConnection connection = new
SQLiteConnection(connectionString))
        {
            connection.Open();

            using (SQLiteDataAdapter adapter = new SQLiteDataAdapter("SELECT
* FROM Doctors", connection))
            {
                dataSet.Tables["Doctors"].Clear();

                adapter.Fill(dataSet, "Doctors");
            }

            connection.Close();
        }

        textBox1DC.Clear();
        textBox2DC.Clear();
        textBox3DC.Clear();
        textBox4DC.Clear();
    }
}

private void btnUpdDc_Click(object sender, EventArgs e)
{
    if (lstvDoctors.SelectedItems.Count > 0)
    {

```

```

ListViewItem selectedDoctor = lstvDoctors.SelectedItems[0];

int doctorId = Convert.ToInt32(selectedDoctor.SubItems[0].Text);
string doctorName = textBox2DC.Text;
string workField = textBox3DC.Text;
string password = textBox4DC.Text;

selectedDoctor.SubItems[1].Text = doctorName;
selectedDoctor.SubItems[2].Text = workField;
selectedDoctor.SubItems[3].Text = password;

using (SQLiteConnection connection = new
SQLiteConnection(connectionString))
{
    connection.Open();

    using (SQLiteDataAdapter adapter = new SQLiteDataAdapter("SELECT
* FROM Doctors", connection))
    {
        SQLiteCommandBuilder commandBuilder = new
SQLiteCommandBuilder(adapter);
        adapter.Update(dataSet, "Doctors");
    }

    connection.Close();
}

MessageBox.Show("Data updated successfully.");
}
else
{
    MessageBox.Show("Please select a doctor to update.");
}
}

private void btnDltDc_Click(object sender, EventArgs e)
{
    if (lstvDoctors.SelectedItems.Count > 0)
    {
        ListViewItem selectedDoctor = lstvDoctors.SelectedItems[0];
        int doctorId = Convert.ToInt32(selectedDoctor.SubItems[0].Text);

        lstvDoctors.Items.Remove(selectedDoctor);

        using (SQLiteConnection connection = new
SQLiteConnection(connectionString))
        {
            connection.Open();

```

```

        string query = "DELETE FROM Doctors WHERE DoctorId =
@doctorId";

        SQLiteCommand command = new SQLiteCommand(query, connection);
        command.Parameters.AddWithValue("@doctorId", doctorId);
        command.ExecuteNonQuery();

        connection.Close();
    }

    MessageBox.Show("Data deleted successfully.");
}
else
{
    MessageBox.Show("Please select a doctor to delete.");
}
}

private void DoctorF_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Escape)
    {
        Home home = new Home();
        home.Show();
        Hide();
    }
}

private void textBox1DC_DragEnter(object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent(DataFormats.Text))
    {
        e.Effect = DragDropEffects.Copy;
    }
}

private void textBox1DC_DragDrop(object sender, DragEventArgs e)
{
    string droppedText = e.Data.GetData(DataFormats.Text) as string;

    textBox1DC.Text = droppedText;
}

private void btnExportReport_Click(object sender, EventArgs e)
{
    List<Doctor> doctors = new List<Doctor>();

    foreach (ListViewItem item in lstvDoctors.Items)
    {

```

```

        Doctor doctor = new Doctor
        {
            DoctorId = Convert.ToInt32(item.SubItems[0].Text),
            DoctorName = item.SubItems[1].Text,
            WorkField = item.SubItems[2].Text,
            DocPass = item.SubItems[3].Text
        };

        doctors.Add(doctor);
    }

    ExportReportToFile(doctors);
}

private void ExportReportToFile(List<Doctor> doctors)
{
    StringBuilder reportContent = new StringBuilder();

    reportContent.AppendLine("Doctor          ID\tDoctor          Name\tWork
Field\tPassword");

    foreach (Doctor doctor in doctors)
    {

reportContent.AppendLine($"{ doctor.DoctorId }\t{ doctor.DoctorName }\t{ doctor.Wo
rkField }\t{ doctor.DocPass }");
    }

    try
    {
        SaveFileDialog saveFileDialog = new SaveFileDialog();
        saveFileDialog.Filter = "Text files (*.txt)|*.txt";
        saveFileDialog.Title = "Export Report";
        saveFileDialog.ShowDialog();

        if (saveFileDialog.FileName != "")
        {
            File.WriteAllText(saveFileDialog.FileName, reportContent.ToString());
            MessageBox.Show("Report exported successfully.");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("An error occurred while exporting the report: " +
ex.Message);
    }
}

```



```

private void printDocument_PrintPage(object sender, PrintPageEventArgs e)
{
    Font font = new Font("Microsoft Sans Serif", 10);

    var pageSettings = e.PageSettings;

    var printAreaHeight = e.MarginBounds.Height;
    var printAreaWidth = e.MarginBounds.Width;

    var marginLeft = e.MarginBounds.Left;
    var marginTop = e.MarginBounds.Top;

    const int rowHeight = 20;
    var columnWidth = printAreaWidth / 4;

    StringFormat fmt = new StringFormat(StringFormatFlags.LineLimit);
    fmt.Trimming = StringTrimming.EllipsisCharacter;

    var currentY = marginTop;
    var currentX = marginLeft;

    e.Graphics.DrawString("Doctor ID", font, Brushes.Black, currentX,
currentY);
    currentX += columnWidth;

    e.Graphics.DrawString("Doctor Name", font, Brushes.Black, currentX,
currentY);
    currentX += columnWidth;

    e.Graphics.DrawString("Work Field", font, Brushes.Black, currentX,
currentY);
    currentX += columnWidth;

    e.Graphics.DrawString("Password", font, Brushes.Black, currentX, currentY);

    currentY += rowHeight;

    foreach (ListViewItem item in lstvDoctors.Items)
    {
        currentX = marginLeft;

        e.Graphics.DrawString(item.SubItems[0].Text, font, Brushes.Black,
currentX, currentY);
        currentX += columnWidth;

        e.Graphics.DrawString(item.SubItems[1].Text, font, Brushes.Black,
currentX, currentY);
        currentX += columnWidth;
    }
}

```

```

        e.Graphics.DrawString(item.SubItems[2].Text, font, Brushes.Black,
currentX, currentY);
        currentX += columnWidth;

        e.Graphics.DrawString(item.SubItems[3].Text, font, Brushes.Black,
currentX, currentY);

        currentY += rowHeight;

        if (currentY + rowHeight > printAreaHeight)
        {
            e.HasMorePages = true;
            return;
        }
    }
}

private void label1_Click(object sender, EventArgs e)
{
}

private void panel1_Paint(object sender, PaintEventArgs e)
{
}

private void textBox2DC_TextChanged(object sender, EventArgs e)
{
}

private void textBox4DC_TextChanged(object sender, EventArgs e)
{
}

private void lstvDoctors_SelectedIndexChanged(object sender, EventArgs e)
{
}
}
}

```

5.4. Patients Page

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SQLite;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MedicalOffice
{
    public partial class PatientF : Form
    {
        public PatientF()
        {
            InitializeComponent();

            //public ListView PatientsListView
            //{
            //    get { return lstvPatients; }
            //}

            private void PatientF_Load(object sender, EventArgs e)
            {
                btnAddPt.Click += btnAddPt_Click;
                btnUpdPt.Click += btnUpdPt_Click;
                btnDltPt.Click += btnDltPt_Click;
            }

            private void button4_Click(object sender, EventArgs e)
            {
                Home home = new Home();
                home.Show();
                this.Hide();
            }

            private void btnAddPt_Click(object sender, EventArgs e)
            {
                string value1 = textBox1.Text;
                string value2 = textBox2.Text;
                string value3 = textBox3.Text;
                string value4 = textBox4.Text;
                string value5 = textBox5.Text;
                string value6 = textBox6.Text;
                string value7 = comboBox1.Text;
                string value8 = comboBox2.Text;

                ListViewItem newItem = new ListViewItem(value1);
                newItem.SubItems.Add(value2);
                newItem.SubItems.Add(value3);
                newItem.SubItems.Add(value4);
                newItem.SubItems.Add(value5);
                newItem.SubItems.Add(value6);
                newItem.SubItems.Add(value7);
                newItem.SubItems.Add(value8);

                lstvPatients.Items.Add(newItem);
            }
        }
    }
}
```

```

        ClearInputFields();
    }

private void ClearInputFields ()
{
    textBox1.Clear();
    textBox2.Clear();
    textBox3.Clear();
    textBox4.Clear();
    textBox5.Clear();
    textBox6.Clear();
    comboBox1.SelectedIndex = -1;
    comboBox2.SelectedIndex = -1;
}

private void btnUpdPt_Click(object sender, EventArgs e)
{
    if (lstvPatients.SelectedItems.Count > 0)
    {
        ListViewItem selectedPatient =
lstvPatients.SelectedItems[0];

        string patientId = textBox1.Text;
        string patientName = textBox2.Text;
        string gender = comboBox1.SelectedItem.ToString();
        string age = textBox3.Text;
        string procedure = textBox4.Text;
        string medication = textBox5.Text;
        string doctor = comboBox2.SelectedItem.ToString();

        selectedPatient.SubItems[0].Text = patientId;
        selectedPatient.SubItems[1].Text = patientName;
        selectedPatient.SubItems[2].Text = gender;
        selectedPatient.SubItems[3].Text = age;
        selectedPatient.SubItems[4].Text = procedure;
        selectedPatient.SubItems[5].Text = medication;
        selectedPatient.SubItems[6].Text = doctor;

        ClearInputFields();

        MessageBox.Show("Data updated successfully.");
    }
    else
    {
        MessageBox.Show("Please select a patient to update.");
    }
}

private void btnDltPt_Click(object sender, EventArgs e)
{
    if (lstvPatients.SelectedItems.Count > 0)
    {
        ListViewItem selectedPatient =
lstvPatients.SelectedItems[0];

        lstvPatients.Items.Remove(selectedPatient);

        MessageBox.Show("Data deleted successfully.");
    }
    else
    {
        MessageBox.Show("Please select a patient to delete.");
    }
}

```

```

    }

    private void label1_Click(object sender, EventArgs e)
    {
    }

    private void lstvPatients_SelectedIndexChanged(object sender,
EventArgs e)
    {
    }

    private void textBox6_TextChanged(object sender, EventArgs e)
    {
    }

    private void comboBox2_SelectedIndexChanged(object sender,
EventArgs e)
    {
    }

    private void panell1_Paint(object sender, PaintEventArgs e)
    {
    }

    private void PatientF_Load_1(object sender, EventArgs e)
    {
    }
}
}

```

5.5.Treatmens Page

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SQLite;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

```

```

namespace MedicalOffice
{
    public partial class Prescription : Form
    {
        private string connectionString = "Data
Source=C:\\Users\\Berkay\\Desktop\\MedicalOffice-
main\\MedicalOffice\\bin\\Debug\\prescriptiondb.db;Version=3;";

        public Prescription()
        {
            InitializeComponent();
        }

        private void Prescription_Load(object sender, EventArgs e)
        {
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Home homeForm = new Home();
            homeForm.Show();
            this.Hide();
        }

        private void ClearInputFields()
        {
            textBox1.Clear();
            textBox3.Clear();
            textBox4.Clear();
            textBox5.Clear();
            textBox6.Clear();
            textBox2.Clear();
        }

        private void btnAddPr_Click(object sender, EventArgs e)
        {
            string value1 = textBox1.Text;
            string value2 = textBox3.Text;
            string value3 = textBox4.Text;
            string value4 = textBox5.Text;
            string value5 = textBox6.Text;
            string value6 = textBox2.Text;

            ListViewItem newItem = new ListViewItem(value1);
            newItem.SubItems.Add(value2);
            newItem.SubItems.Add(value3);
            newItem.SubItems.Add(value4);
            newItem.SubItems.Add(value5);
            newItem.SubItems.Add(value6);

            lstvPrescriptions.Items.Add(newItem);

            ClearInputFields();
        }

        private void btnUpdPr_Click(object sender, EventArgs e)
        {
            if (lstvPrescriptions.SelectedItems.Count > 0)
            {

```

```

        ListViewItem selectedItem =
lstvPrescriptions.SelectedItems[0];

        selectedItem.SubItems[1].Text = textBox3.Text;
        selectedItem.SubItems[2].Text = textBox4.Text;
        selectedItem.SubItems[3].Text = textBox5.Text;
        selectedItem.SubItems[4].Text = textBox6.Text;
        selectedItem.SubItems[4].Text = textBox2.Text;

        ClearInputFields();
    }
}

private void btnDltPr_Click(object sender, EventArgs e)
{
    if (lstvPrescriptions.SelectedItems.Count > 0)
    {
lstvPrescriptions.Items.Remove(lstvPrescriptions.SelectedItems[0]);

        ClearInputFields();
    }
}

private void textBox4_TextChanged(object sender, EventArgs e)
{
}

private void textBox5_TextChanged(object sender, EventArgs e)
{
}

private void label1_Click(object sender, EventArgs e)
{
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
}

private void panell1_Paint(object sender, PaintEventArgs e)
{
}
}
}

```

5.6.Financial Page

```
using System;
```

```

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SQLite;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MedicalOffice
{
    public partial class Financial : Form
    {
        private string connectionString = "Data
Source=C:\\Users\\Berkay\\Desktop\\MedicalOffice-
main\\MedicalOffice\\bin\\Debug\\prescriptiondb.db;Version=3;";

        public Financial()
        {
            InitializeComponent();
        }

        private void Financial_Load(object sender, EventArgs e)
        {
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Home homeForm = new Home();
            homeForm.Show();
            this.Hide();
        }

        private void ClearInputFields()
        {
            textBox1.Clear();
            textBox3.Clear();
            textBox4.Clear();
            textBox5.Clear();
            textBox6.Clear();
            textBox2.Clear();
        }

        private void btnAddPr_Click(object sender, EventArgs e)
        {
            string value1 = textBox1.Text;
            string value2 = textBox3.Text;
            string value3 = textBox4.Text;
            string value4 = textBox5.Text;
            string value5 = textBox6.Text;
            string value6 = textBox2.Text;

            ListViewItem newItem = new ListViewItem(value1);
            newItem.SubItems.Add(value2);
            newItem.SubItems.Add(value3);
            newItem.SubItems.Add(value4);
            newItem.SubItems.Add(value5);
            newItem.SubItems.Add(value6);

            lstvPrescriptions.Items.Add(newItem);
        }
    }
}

```



```

        ClearInputFields();
    }

    private void btnUpdPr_Click(object sender, EventArgs e)
    {
        if (lstvPrescriptions.SelectedItems.Count > 0)
        {
            ListViewItem selectedItem =
lstvPrescriptions.SelectedItems[0];

            selectedItem.SubItems[1].Text = textBox3.Text;
            selectedItem.SubItems[2].Text = textBox4.Text;
            selectedItem.SubItems[3].Text = textBox5.Text;
            selectedItem.SubItems[4].Text = textBox6.Text;
            selectedItem.SubItems[4].Text = textBox2.Text;

            ClearInputFields();
        }
    }

    private void btnDltPr_Click(object sender, EventArgs e)
    {
        if (lstvPrescriptions.SelectedItems.Count > 0)
        {
lstvPrescriptions.Items.Remove(lstvPrescriptions.SelectedItems[0]);

            ClearInputFields();
        }
    }

    private void textBox4_TextChanged(object sender, EventArgs e)
    {
    }

    private void textBox5_TextChanged(object sender, EventArgs e)

```

```
{  
}  
private void labell1_Click(object sender, EventArgs e)  
{  
}  
private void textBox1_TextChanged(object sender, EventArgs e)  
{  
}  
private void panell1_Paint(object sender, PaintEventArgs e)  
{  
}  
}  
}
```

REFERENCES

- [1] Connolly, T., & Begg, C. (2014). "Database Systems: A Practical Approach to Design, Implementation, and Management." Pearson Education
- [2] Ambler, S. W. (2017). "Use Case Modeling." Agile Modeling.
- [3] Jacobson, I., Booch, G., & Rumbaugh, J. (1999). "The Unified Modeling Language Reference Manual." Addison-Wesley Professional.
- [4] Ambler, S. W. (2017). "Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process." John Wiley & Sons.
- [5] Skeet, J. (2020). "C# in Depth." Manning Publications
- [6] Troelsen, A. (2017). "Pro C# 7: With .NET and .NET Core." Apress.
- [7] DuBois, P. (2019). "MySQL Cookbook: Solutions for Database Developers and Administrators." O'Reilly Media