1. INTRODUCTION

1.1 Problem definition:

The software enables efficient cataloguing of prescriptions by a doctor by providing them a platform to choose from all the medicines, their doses, write personal advice, etc. The personalized prescription can then be printed and given to the patient. The patient details are also recorded in the prescription and stored for easy future access and monitoring.

1.2 Scope of the Project:

In this modern day and age the only thing done in the primitive way is writing prescriptions by practitioners. Most of the time the prescription will be written by them using pen and paper in a hurry, resulting in oftentimes the writing being illegible. The doctors are unable to access their patient's previous records too for future consultations. These nuances cause the patient or the pharmacist to misinterpret the prescription or the doctor to misdiagnose the patient. The prescriptions, by virtue of being a physical entity, are vulnerable to the forces of nature. The prescription may be damaged, lost, or in any other way be rendered useless. The practitioner is also required to remember the correct name and dosage of all the medicines without any spelling errors, with the overwhelming amount of medicines present in the modern world, this becomes increasingly impossible. Mundane and automatable tasks like writing prescriptions, remembering all the medicine's names and dosages should thus be delegated to software counterparts. This helps the practitioner to easily look up the medicines and procedures and prescribe the same. The patients too can easily re-issue prescriptions and keep a soft-copy backup with them. Thus the digitization and automation of prescription processing is the only logical way forward. MD.APP solves that requirement by computerizing the prescription composing and printing procedure, aided by a gigantic database of all the medicines and medical procedures present.

1.3 Modules in the project:

1. Authentication Module – The doctor needs to login to access his patient records, or can use the application in guest mode, where past records won't be visible.
2. Prescription Editor – Edit all the fields of the prescription, e.g. patient details, medicines prescribed, dosage, miscellaneous details, doctor's advice, etc.

3. Medicine and Procedure Search – Search from a wide database of medicines and medical procedures to be advised/prescribed to the patient.

4. Printing Module – Print the finished prescription through a connected printer, or save it as a Portable Document Format (pdf) file.

5. Patient History – Show previous prescriptions to patients, search through name, age, etc.

6. Admin Module- show all visits history and add remove medicine and procedure.

2.SYSTEM STUDY

Existing system:

In the existing system the doctors write the prescription manually. This may be hard to understand for the patients and also the doctor needs to remember the medicine's name and the dosages for each patient when prescribed. Any misinterpretation by the patient or pharmacy may lead to catastrophic effects on the patient's health. Some solutions exist but are highly specific and thus are restricted to their particular organizations and are not widespread outside the organization or the general norm.

Problems in existing system:

1. Hard to understand the manually written prescription.

2. The patients may unknowingly get some other dosage of the medicine due to misreading.

3. Doctors may face difficulty in remembering the names of the medicine and the dosage.

4. Manually written prescriptions may be lost but it can be saved in the software.

5. Preexisting specialized solutions are not applicable to the majority of people and thus cannot be used.

Proposed system:

The proposed solution digitizes the prescription composing and printing process, making it resilient to misinterpretation and physical damage. The prescription backup along with the patient details ensures the doctor can look up the previous medical history of their patients when required without any guesswork required. The software is generic and customizable enough so that any practitioner can use it without having to re-write the software for their organization.

Advantages over preexisting system:

1. Doctors can easily enter the details of the patient and the medicine prescribed to them by accessing the database in the software.

2. This prescription can be saved in the software and It is easy to access the details when required.

3. Reduces paperwork and saves time.

4. All the details are secure as there is a login module.

5. The software is usable by any doctor / organization.

3. SYSTEM DESIGN

In the design phase the architecture is established. This phase starts with the requirement document delivered by the requirement phase and maps the requirements into architecture. The architecture defines the components, their interfaces and behaviors. The deliverable design document is the architecture.

The design document describes a plan to implement the requirements. This phase represents the ``how'' phase. Details on computer programming languages and environments, machines, packages, application architecture, distributed architecture layering, memory size, platform, algorithms, data structures, global type definitions, interfaces, and many other engineering details are established. The design may include the usage of existing components. Analyzing the trade-offs of necessary complexity allows for many things to remain simple which, in turn, will eventually lead to a higher quality product. The architecture team also converts the typical scenarios into a test plan.

In our approach, given a complete requirement document, must also indicate critical priorities for the implementation. A critical implementation priority leads to a task that has to be done right. If it fails, the product fails. If it succeeds, the product might succeed. At the very least, the confidence level producing a successful product will increase. The information conveyed is a skill based on experience more than a science based on fundamental foundations.

System design is the process of defining the architecture components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the Process of defining and developing systems to satisfy specified requirements of the user.

Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design. The UML has become the standard language in object oriented analysis and design. It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.

Architectural design:

The architectural design of a system emphasizes the design of the system architecture that describes the structure, behavior and more views of that system and analysis.

Logical design:

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems, designs are included. Logical design includes entity-relationship diagrams (ER diagrams).

Physical design:

The physical design relates to the actual input and output processes of the system. This is explained in terms of how data is input into a system, how it is verified /authenticated, how it is processed, and how it is displayed.

In physical design, the following requirements about the system are decided.

1. Input requirement,

2. Output requirements,

3. Storage requirements,

4. Processing requirements,

5. System control and backup or recovery.

Put another way, the physical portion of system design can generally be broken down into three sub-tasks:

1. User Interface Design

2. Data Design

3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system.

At the end of the system design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase. Physical design, in this context, does not refer to the tangible physical design of an information system.

To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc.

3.1 ER Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

At first glance an entity relationship diagram looks very much like a flowchart. It is the specialized symbols, and the meanings of those symbols, that make it unique.
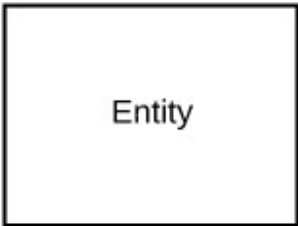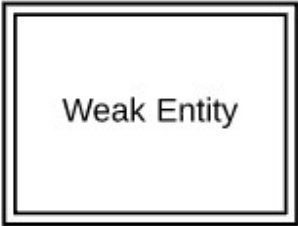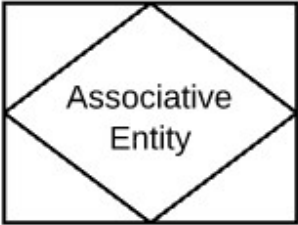
Because this ER tutorial focuses on beginners below are some tips that will help you build effective ER diagrams:

• Identify all the relevant entities in a given system and determine the relationships among these entities.

• An entity should appear only once in a particular diagram.

• Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram. Terms that are simple and familiar always beats vague, technical-sounding words. In naming entities, remember to use singular nouns. However, adjectives may be used to distinguish entities belonging to the same class (part-time employee and full-time employee, for example). Meanwhile attribute names must be meaningful, unique, system-independent, and easily understandable.

• Remove vague, redundant or unnecessary relationships between entities.

• Never connect a relationship to another relationship.

• Make effective use of colors. You can use colors to classify similar entities or to highlight key areas in your diagrams.

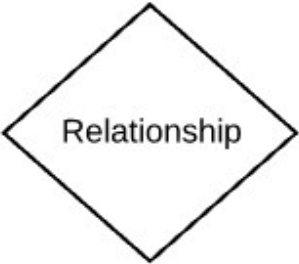## Structure of an Entity Relationship Diagram with Common ERD Notations
ERD entity symbols

Entities are objects or concepts that represent important data. Entities are typically nouns such as product, customer, location, or promotion. There are three types of entities commonly used in entity relationship diagrams.

Entity Symbol	Name	Description
Entity	Strong entity	These shapes are independent from other entities, and are often called parent entities, since they will often have weak entities that depend on them. They will also have a primary key, distinguishing each occurrence of the entity.
Weak Entity	Weak entity	Weak entities depend on some other entity type. They don't have primary keys, and have no meaning in the diagram without their parent entity.
Associative Entity	Associative entity	Associative entities relate the instances of several entity types. They also contain attributes specific to the relationship between those entity instances.

ERD relationship symbols

Within entity-relationship diagrams, relationships are used to document the interaction between two entities. Relationships are usually verbs such as assign, associate, or track and provide useful information that could not be discerned with just the entity types.

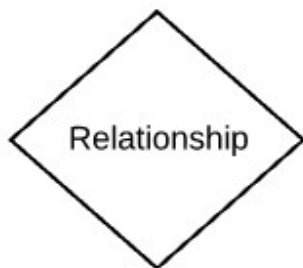| Relationship Symbol	Name	Description
	Relationship	Relationships are associations between or among entities.
	Weak relationship	Weak Relationships are connections between a weak entity and its owner.

ERD attribute symbols

ERD attributes are characteristics of the entity that help users to better understand the database. Attributes are included to include details of the various entities that are highlighted in a conceptual ER diagram.

Attribute Symbol	Name	Description
	Attribute	Attributes are characteristics of an entity, a many-to-many relationship, or a one-to-one relationship.
	Multivalued attribute	Multivalued attributes are those that are can take on more than one value.
	Derived attribute	Derived attributes are attributes whose value can be calculated from related attribute values.

| Attribute Symbol	Name	Description
Relationship (diamond shape)	Relationship	Relationships are associations between or among entities.

Cardinality

specifies how many instances of an entity relate to one instance of another entity. Ordinarily is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinarily describes the relationship as either mandatory or optional. In other words, cardinality

Modality

As cardinality is the maximum number of connections between table rows (either one or many), modality is the least number of row connections! Modality also only has two options, 0 being the least or 1 being the least.

ER DIAGRAM OF MEDICINE DATABASE AND AUTOMATED PRISCRIPTION PRINTING
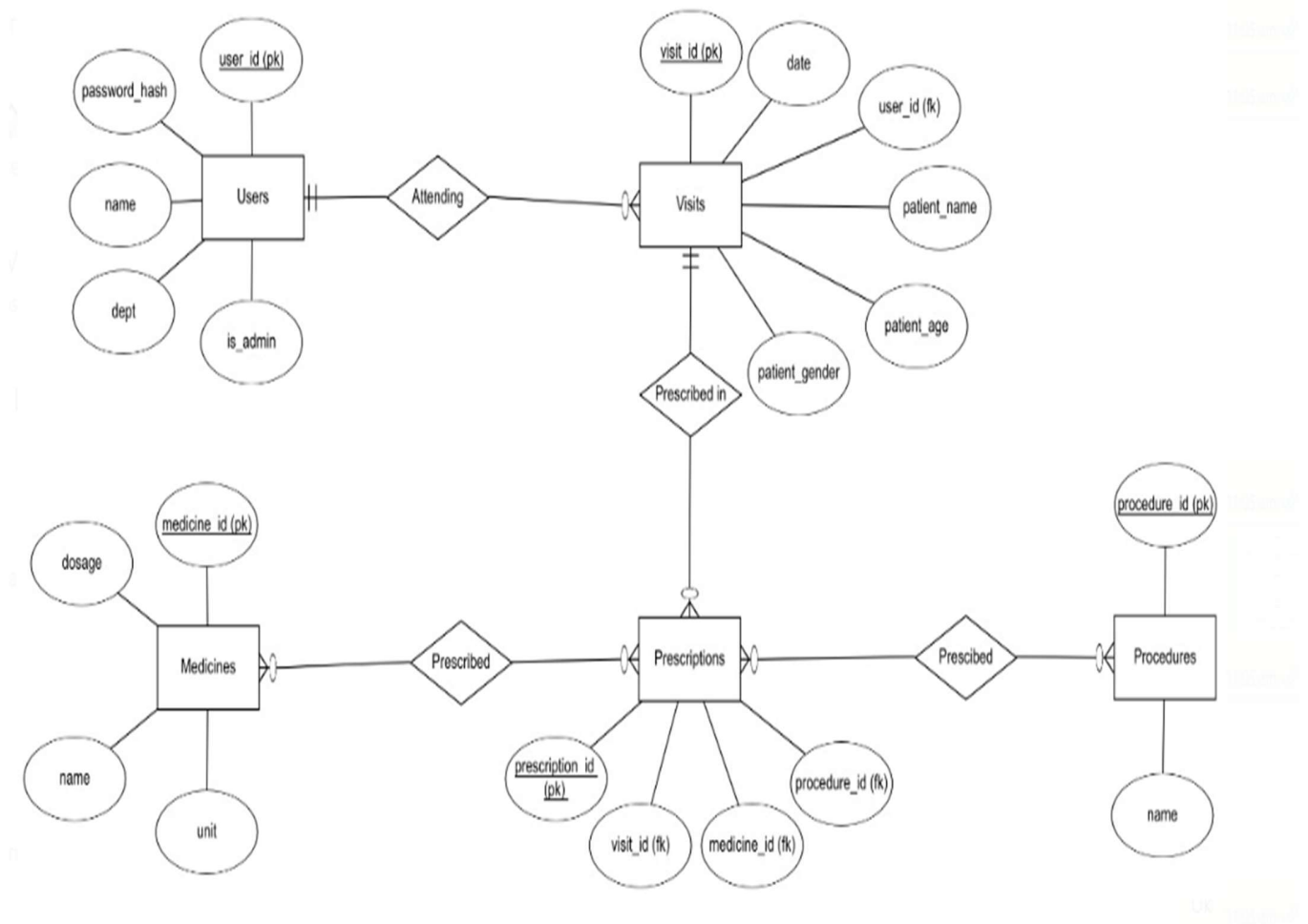
3.2 Data flow diagram (level 0 and level 1)

The Data Flow Diagrams (DFDs) are used for structure analysis and design. DFDs show the flow of data from external entities into the system. DFDs also show how the data moves and are transformed from one process to another, as well as its logical storage. The following symbols are used within DFDs.

For clarity, a key has been provided at the bottom of this page.

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).
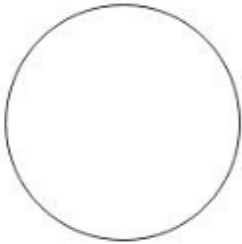
A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

Physical vs. logical DFD

A logical DFD captures the data flows that are necessary for a system to operate. It describes the processes that are undertaken, the data required and produced by each process, and the stores needed to hold the data. On the other hand, a physical DFD shows how the system is actually implemented, either at the moment (Current Physical DFD), or how the designer intends it to be in the future (Required Physical DFD).

Thus, a Physical DFD may be used to describe the set of data items that appear on each piece of paper that move around an office, and the fact that a particular set of pieces of paper are stored together in a filing cabinet. It is quite possible that a Physical DFD will include references to data that are duplicated, or redundant, and that the data stores, if implemented as a set of database tables, would constitute an un-normalized (or de-normalized) relational database. In contrast, a Logical DFD attempts to capture the data flow aspects of a system in a form that has neither redundancy nor duplication.

DATA FLOW SYMBOLS AND THEIR MEANINGS

Symbol	Name	Description
	Entity	It is represented by a rectangle and simply depicts a source or termination of the diagram by mapping real-world entities.
	Process	It is represented by a circle and depicts how the data is handled and processed in the system.
	Data Store	It is represented by two parallel lines and depicts a location where data is stored in the system.
	Data Flow	It is represented by directional lines and depicts the flow of data from one location to another.

A level-0 DFD is the most basic form of DFD. It aims to show how the entire system works at a glance. There is only one process in the system and all the data flows either into or out of this process. Level-0 DFD's demonstrates the interactions between the process and external entities. They do not contain Data Stores.

When drawing Level-0 DFD's, we must first identify the process, all the external entities and all the data flows. We must also state any assumptions we make about the system. It is advised that we draw the process in the middle of the page. We then draw our external entities in the corners and finally connect our entities to our process with the data flows

Level-0 DFD
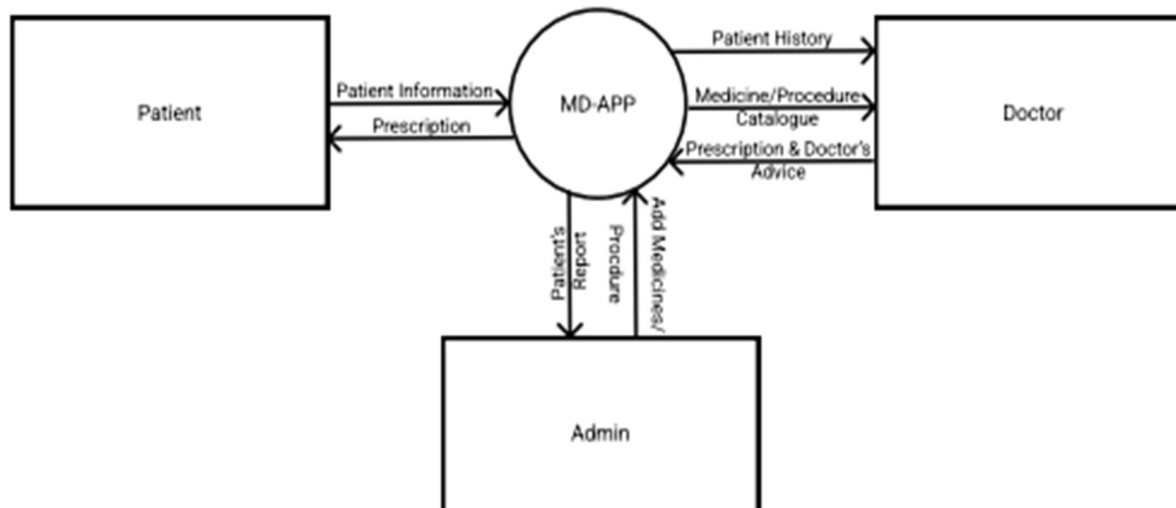
Level 1 DFD

Level 1 DFD's aim is to give an overview of the full system. They look at the system in more detail. Major processes are broken down into sub-processes. Level 1 DFD's also identifies data stores that are used by the major processes. When constructing a Level 1 DFD; we must start by examining the Context Level DFD. We must break up the single process into its subprocesses. We must then pick out the data stores from the text we are given and include them in our DFD. Like the Context Level DFD's, all entities, data stores and processes must be labeled. We must also state any assumptions made from the text

LEVEL 1 DFD OF MEDICINE DATABASE AND AUTOMATED PRISCRIPTION PRINTING

Level 1 DFD:
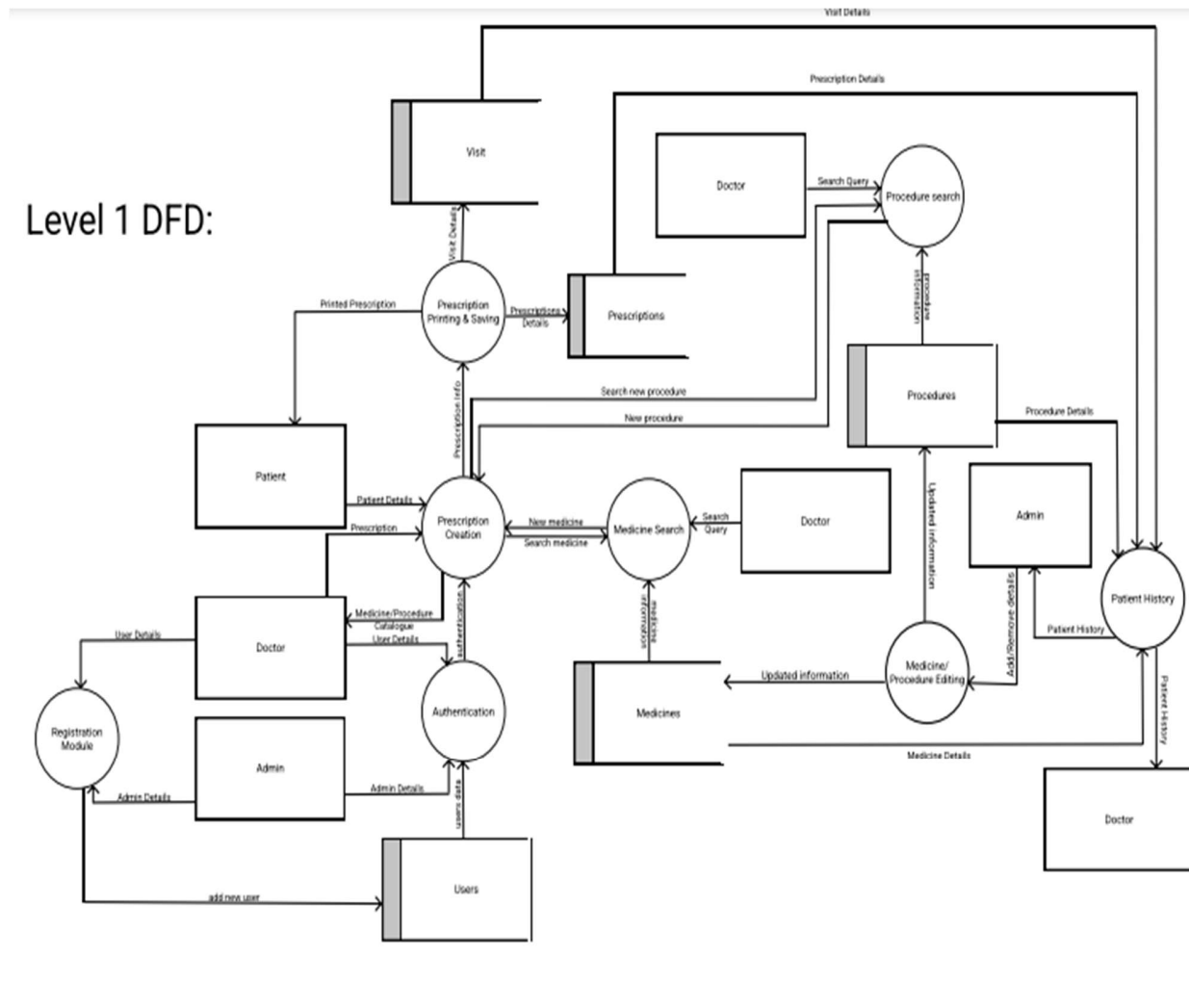
3.3 Gantt Chart

A Gantt chart is a type of bar chart, devised by Henry Gantt in the 1910s, that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e., precedence network) relationships between activities. Gantt charts can be used to show current schedule status using percent-complete shadings and a vertical "TODAY" line as shown here. Although now regarded as a common charting technique, Gantt charts were considered revolutionary when first introduced. This chart is also used in information technology to represent data that has been collected.

GANTT CHART BENEFITS:

Clarity:

One of the biggest benefits of a Gantt chart is the tool's ability to boil down multiple tasks and timelines into a single document. Stakeholders throughout an organization can easily understand where teams are in a process while grasping the ways in which independent elements come together toward project completion.

Communication:

Teams can use Gantt charts to replace meetings and enhance other status updates. Simply clarifying chart positions offers an easy, visual method to help team members understand task progress.

Motivation:

Some teams or team members become more effective when faced with a form of external motivation. Gantt charts offer teams the ability to focus work at the front of a task timeline, or at the tail end of a chart segment. Both types of team members can find Gantt charts meaningful as they plug their own work habits into the overall project schedule.

Coordination:

For project managers and resource schedulers, the benefits of a Gantt chart include the ability to sequence events and reduce the potential for overburdening team members. Some project managers even use combinations of charts to break down projects into more manageable sets of tasks.

Creativity:

Sometimes, a lack of time or resources forces project managers and teams to find creative solutions. Seeing how individual tasks intertwine on Gantt charts often encourages new partnerships and collaborations that might not have evolved under traditional task assignment systems.

Time Management:

Most managers regard scheduling as one of the major benefits of Gantt charts in a creative environment. Helping teams understand the overall impact of project delays can foster stronger collaboration while encouraging better task organization

Flexibility:

Whether you use Excel to generate Gantt charts or you load tasks into a more precise chart generator, the ability to issue new charts as your project evolves lets you react to unexpected changes in project scope or timeline. While revising your project schedule too frequently can eliminate some of the other benefits of Gantt charts, offering a realistic view of a project can help team members recover from setbacks or adjust to other changes.

Manageability:

For project managers handling complex assignments, like software publishing or event planning, the benefits of Gantt charts include externalizing assignments. By visualizing all of the pieces of a project puzzle, managers can make more focused, effective decisions about resources and timetables.

Efficiency:

Another one of the benefits of Gantt charts is the ability for teams members to leverage each other's deadlines for maximum efficiency. For instance, while one team member waits on the outcome of three other tasks before starting a crucial piece of the assignment, he or she can perform other project tasks. Visualizing resource usage during projects allows managers to make better use of people, places, and things.

Accountability:

When project teams face major organizational change, documenting effort and outcomes becomes crucial to career success. Using Gantt charts during critical projects allows both project managers and participants to track team progress, highlighting both big wins and major failures during professional review periods; team members who frequently exceed expectations can leverage this documentation into larger raises or bonuses.

Gantt chart Importance:

The project's summary and terminal elements, which combine to form the project's internal structure, are shown on the Gantt chart. Many charts will also depict the precedence rankings and dependencies of various tasks within the project. The charts can illustrate the start and finish project terminal elements in project management. It can also show summary elements and terminal dependencies. The smallest task tracked as part of the project effort is known as a terminal element. Gantt chart represents the tasks in most modern project scheduling packages. However other management applications use simpler communication tools such as message boards, to-do lists and simple scheduling etc., therefore, they do not use Gantt charts as heavily.

The way to create this chart begins by determining and listing the necessary activities. Next, sketch out how you expect the chart to look. List which items depend on others and what activities take place when. For each activity, list how many man-hours it will require, and who is responsible. Lastly, determine the throughput time.

This technique's primary advantage is its good graphical overview that is easy to understand for nearly all project participants and stakeholders.
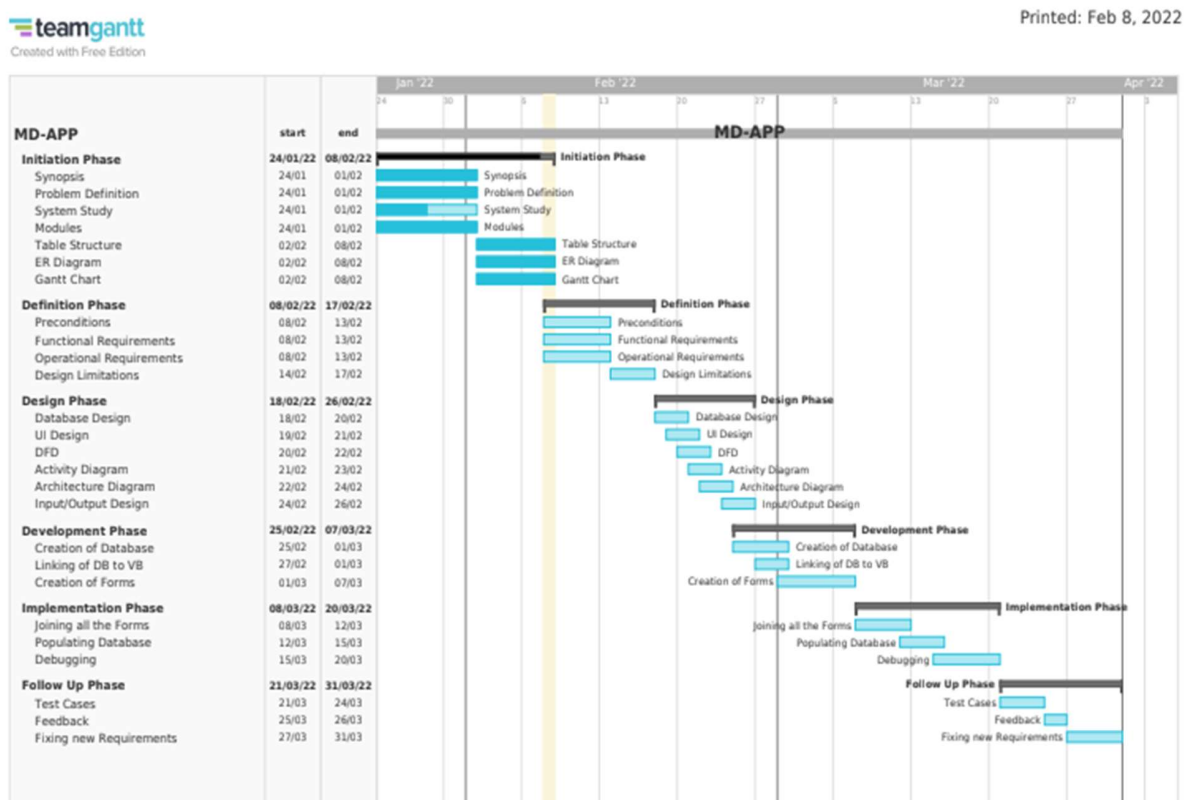
3.4 Input/Output Design

Main menu:

```
Public Class frm_MainMenu
#Region " Top Panel "
#Region " Move Form "
  Public MoveForm As Boolean
  Public MoveForm_MousePosition As Point
  Public Sub MoveForm_MouseDown(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseDown
    If e.Button = MouseButtons.Left Then
      MoveForm = True
      MoveForm_MousePosition = e.Location
    End If
  End Sub
  Public Sub MoveForm_MouseMove(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseMove
    If MoveForm Then
      Me.Location += (e.Location - MoveForm_MousePosition)
    End If
  End Sub
  Public Sub MoveForm_MouseUp(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseUp
    If e.Button = MouseButtons.Left Then
      MoveForm = False
    End If
  End Sub
#End Region
  Private Sub CloseApp(sender As Object, e As EventArgs) _
    Handles btn_Close.Click, ctxItem_Exit.Click, img_TopBar_Logo.DoubleClick
    Me.Close()
  End Sub
```

```vbnet
    Private Sub InvokeCtx(sender As Object, e As EventArgs) _
      Handles img_TopBar_Logo.Click
      ctx_Main.Show(MousePosition.X + 2, MousePosition.Y + 2)
    End Sub
#End Region
#Region "Embed Auth Forms"
   Dim AuthForm As Form
   Dim AuthState As Integer = 0 ' 0 means login, 1 means registration
   Private Sub SetAuthForm(frm As Form, lblText As String, btnText As String)
      With frm
        .TopLevel = False
        pnl_Auth.Controls.Clear()
        pnl_Auth.Controls.Add(frm)
        .Dock = DockStyle.Fill
        .BringToFront()
        .Show()
        AuthForm = frm
      End With
      lbl_ToggleAuthText.Text = lblText
      btn_ToggleAuth.Text = btnText
    End Sub
    Private Sub AuthPageLoad(sender As Object, e As EventArgs) _
      Handles Me.Load
      Me.CenterToScreen()
      SetAuthForm(frm_LoginAuth, "New User? Register instead", "Register")
      AuthState = 0
    End Sub
    Private Sub ToggleAuth(sender As Object, e As EventArgs) _
      Handles btn_ToggleAuth.Click
      If AuthForm IsNot Nothing Then
        AuthForm.Close()
      End If
```

```
   If AuthState = 0 Then
     SetAuthForm(frm_RegisterAuth, "Already Registered? Login Instead", "Login")
   Else
     SetAuthForm(frm_LoginAuth, "New User? Register instead", "Register")
   End If
   AuthState = AuthState Xor 1
  End Sub
#End Region
End Class
```

Login authentication:

```
Imports System.Data.SQLite
Imports System.Runtime.InteropServices
Imports System.Text
Imports System.Security.Cryptography
Public Class frm_LoginAuth
  Dim DBPath As String
  ReadOnly TableName As String = "users"
  Public user_id, passwordhash, username, dept As String
  Public is_admin As Integer
  Private Sub LoadDB(ByVal q As String, ByVal tbl As DataTable, ByVal cn As SQLiteConnection)
    Dim SQLiteDA As New SQLiteDataAdapter(q, cn)
    SQLiteDA.Fill(tbl)
    SQLiteDA.Dispose()
  End Sub
  Private Sub ExecuteNonQuery(ByVal query As String, ByVal cn As SQLiteConnection)
    Dim SQLiteCM As New SQLiteCommand(query, cn)
    SQLiteCM.ExecuteNonQuery()
    SQLiteCM.Dispose()
  End Sub
  Private Function SHA512(ByVal input) As String
    Dim hash As Byte() = SHA512Managed.Create().ComputeHash(Encoding.UTF8.GetBytes(input))
```

```vbnet
    Dim stringBuilder As New StringBuilder()
    For i As Integer = 0 To hash.Length - 1
      stringBuilder.Append(hash(i).ToString("X2"))
    Next
    Return stringBuilder.ToString
  End Function
  Private Sub btn_Login_Click(sender As Object, e As EventArgs) Handles btn_Login.Click
    If tb_LoginAuth_Username.Text = Nothing _
      Or tb_LoginAuth_Username.Text = "" _
      Or tb_LoginAuth_Password.Text = Nothing _
      Or tb_LoginAuth_Password.Text = "" Then
     MsgBox("Invalid Username/Password")
     Exit Sub
    End If
    user_id = tb_LoginAuth_Username.Text.Trim()
       passwordhash = SHA512(tb_LoginAuth_Username.Text & tb_LoginAuth_Password.Text)
    DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
    Dim SQLiteCon As New SQLiteConnection(DBPath)
    Try
      SQLiteCon.Open()
    Catch ex As Exception
      SQLiteCon.Dispose()
      MsgBox("Error Opening Database: " & ex.Message)
      Exit Sub
    End Try
    Dim TableDB As New DataTable
    Try
     LoadDB("select * from " & TableName & " where user_id='" & user_id & "'", TableDB, SQLiteCon)
     If TableDB.Rows.Count = 1 AndAlso TableDB.Rows(0)(1) = passwordhash Then
       Dim row As DataRow = TableDB.Rows(0)
       username = row(2)
       dept = row(3)
```

```vbnet
        is_admin = row(4)
        If is_admin = 0 Then
          frm_UserHome.Show()
          frm_MainMenu.Hide()
        Else
          frm_AdminHome.Show()
          frm_MainMenu.Hide()
        End If
      Else
        MsgBox("Username/Password invalid")
        Exit Sub
      End If
    Catch ex As Exception
        MsgBox("Error loading database: " & ex.Message)
        Exit Sub
    Finally
      TableDB.Dispose()
      SQLiteCon.Dispose()
      tb_LoginAuth_Username.Text = Nothing
      tb_LoginAuth_Password.Text = Nothing
    End Try
  End Sub
  Private Sub tb_LoginAuth_KeyDown(sender As Object, e As KeyEventArgs) _
    Handles tb_LoginAuth_Password.KeyDown, tb_LoginAuth_Username.KeyDown
    If e.KeyCode = Keys.Enter Then
      btn_Login_Click(sender, e)
    End If
  End Sub
End Class
```

Register authentication:

```
Imports System.Data.SQLite

Imports System.Runtime.InteropServices

Imports System.Text

Imports System.Security.Cryptography

Public Class frm_RegisterAuth

  Dim DBPath As String

  ReadOnly TableName As String = "users"

  Dim user_id, passwordhash, username, dept As String

  Dim is_admin As Integer

  Private Sub LoadDB(ByVal q As String, ByVal tbl As DataTable, ByVal cn As SQLiteConnection)

    Dim SQLiteDA As New SQLiteDataAdapter(q, cn)

    SQLiteDA.Fill(tbl)

    SQLiteDA.Dispose()

  End Sub

  Private Sub ExecuteNonQuery(ByVal query As String, ByVal cn As SQLiteConnection)

    Dim SQLiteCM As New SQLiteCommand(query, cn)

    SQLiteCM.ExecuteNonQuery()

    SQLiteCM.Dispose()

  End Sub

  Private Function SHA512(ByVal input) As String

    Dim hash As Byte() = SHA512Managed.Create().ComputeHash(Encoding.UTF8.GetBytes(input))

    Dim stringBuilder As New StringBuilder()

    For i As Integer = 0 To hash.Length - 1

      stringBuilder.Append(hash(i).ToString("X2"))

    Next

    Return stringBuilder.ToString

  End Function

  Private Sub btn_Register_Click(sender As Object, e As EventArgs) Handles btn_Register.Click

    If tb_RegisterAuth_Username.Text = Nothing _

      Or tb_RegisterAuth_Username.Text = "" _

      Or tb_RegisterAuth_Password.Text = Nothing _
```

```
    Or tb_RegisterAuth_Password.Text = "" _
    Or tb_RegisterAuth_Name.Text = Nothing _
    Or tb_RegisterAuth_Name.Text = "" Then
  MsgBox("Fill all the required fields")
  Exit Sub
End If
user_id = tb_RegisterAuth_Username.Text.Trim()
passwordhash = SHA512(tb_RegisterAuth_Username.Text & tb_RegisterAuth_Password.Text)
username = tb_RegisterAuth_Name.Text
dept = tb_RegisterAuth_Dept.Text
is_admin = If(chkbx_RegisterAuth_is_admin.Checked, 1, 0)
If user_id.Length > 16 Then
  MsgBox("Username can be 16 characters at max")
  Exit Sub
End If
If username.Length > 50 Then
  MsgBox("Name can be 50 characters at max")
  Exit Sub
End If
If dept.Length > 15 Then
  MsgBox("Department can be 15 characters at max")
  Exit Sub
End If
DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
Dim SQLiteCon As New SQLiteConnection(DBPath)
Try
  SQLiteCon.Open()
Catch ex As Exception
  SQLiteCon.Dispose()
  MsgBox("Error connecting to database:" & ex.Message)
  Exit Sub
End Try
```

```vbnet
    Try
      ExecuteNonQuery("insert into users values('" &
              user_id &
              "','" & passwordhash &
              "','" & username &
              "','" & dept &
              "','" & is_admin &
              "');", SQLiteCon)
      MsgBox(If(is_admin = 1, "Admin", "User") & " Registered Successfully")
    Catch ex As Exception
      If ex.Message.Contains("UNIQUE") Then
        MsgBox("Username is already registered")
      Else
        MsgBox("Error Registering User: " & ex.Message)
      End If
      Exit Sub
    Finally
      tb_RegisterAuth_Dept.Text = Nothing
      tb_RegisterAuth_Name.Text = Nothing
      tb_RegisterAuth_Password.Text = Nothing
      tb_RegisterAuth_Username.Text = Nothing
      SQLiteCon.Dispose()
    End Try
  End Sub
  Private Sub tb_RegisterAuth_KeyDown(sender As Object, e As KeyEventArgs) _
   Handles        tb_RegisterAuth_Dept.KeyDown,        tb_RegisterAuth_Name.KeyDown,
tb_RegisterAuth_Password.KeyDown,
   tb_RegisterAuth_Username.KeyDown
    If e.KeyCode = Keys.Enter Then
      btn_Register_Click(sender, e)
    End If
  End Sub
```

End Class

User home:

```vb
Public Class frm_UserHome
#Region " Top Panel "
#Region " Move Form "
  Public MoveForm As Boolean
  Public MoveForm_MousePosition As Point
  Public Sub MoveForm_MouseDown(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseDown
    If e.Button = MouseButtons.Left Then
      MoveForm = True
      MoveForm_MousePosition = e.Location
    End If
  End Sub
  Public Sub MoveForm_MouseMove(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseMove
    If MoveForm Then
      Me.Location += (e.Location - MoveForm_MousePosition)
    End If
  End Sub
  Public Sub MoveForm_MouseUp(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseUp
    If e.Button = MouseButtons.Left Then
      MoveForm = False
    End If
  End Sub
#End Region
  Private Sub CloseApp(sender As Object, e As EventArgs) _
    Handles btn_Close.Click, ctxItem_Exit.Click, img_TopBar_Logo.DoubleClick
    frm_MainMenu.Show()
```

```vbnet
    Me.Close()
  End Sub
  Private Sub InvokeCtx(sender As Object, e As EventArgs) _
    Handles img_TopBar_Logo.Click
    ctx_Main.Show(MousePosition.X + 2, MousePosition.Y + 2)
  End Sub
#End Region
  Private Sub btn_NewPres_Click(sender As Object, e As EventArgs) Handles btn_NewPres.Click
    frm_PrescriptionEditor.Show()
    Me.Hide()
  End Sub
  Private Sub btn_PatHist_Click(sender As Object, e As EventArgs) Handles btn_PatHist.Click
    frm_PatHist.Show()
    Me.Hide()
  End Sub
End Class
```

Prescription editor:

```vbnet
Imports System.Data.SQLite
Imports System.Runtime.InteropServices
Public Class frm_PrescriptionEditor
#Region " Top Panel "
#Region " Move Form "
  Public MoveForm As Boolean
  Public MoveForm_MousePosition As Point
  Public Sub MoveForm_MouseDown(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseDown
    If e.Button = MouseButtons.Left Then
      MoveForm = True
      MoveForm_MousePosition = e.Location
    End If
```

```vbnet
    End Sub
    Public Sub MoveForm_MouseMove(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseMove
        If MoveForm Then
            Me.Location += (e.Location - MoveForm_MousePosition)
        End If
    End Sub
    Public Sub MoveForm_MouseUp(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseUp
        If e.Button = MouseButtons.Left Then
            MoveForm = False
        End If
    End Sub
#End Region
    Private Sub CloseApp(sender As Object, e As EventArgs) _
        Handles btn_Close.Click, ctxItem_Exit.Click, img_TopBar_Logo.DoubleClick
        frm_UserHome.Show()
        Me.Close()
    End Sub
    Private Sub InvokeCtx(sender As Object, e As EventArgs) _
        Handles img_TopBar_Logo.Click
        ctx_Main.Show(MousePosition.X + 2, MousePosition.Y + 2)
    End Sub
#End Region
    Dim DBPath As String
    Dim TableName As String = "visits"
    Private Sub LoadDB(ByVal q As String, ByVal tbl As DataTable, ByVal cn As SQLiteConnection)
        Dim SQLiteDA As New SQLiteDataAdapter(q, cn)
        SQLiteDA.Fill(tbl)
        SQLiteDA.Dispose()
    End Sub
```

```vbnet
Private Sub ExecuteNonQuery(ByVal query As String, ByVal cn As SQLiteConnection)
  Dim SQLiteCM As New SQLiteCommand(query, cn)
  SQLiteCM.ExecuteNonQuery()
  SQLiteCM.Dispose()
End Sub
Public dtb_med As New DataTable
Public dtb_proc As New DataTable
Public dtb_consol As New DataTable
Private Sub ClearAdvice(sender As Object, e As EventArgs) Handles btn_ClearAdvice.Click
  rtb_Advice.Clear()
End Sub
Private Sub AddMedicine(sender As Object, e As EventArgs) Handles btn_AddMed.Click
  frm_MedicineSearch.Show()
End Sub
Private Sub btn_AddProc_Click(sender As Object, e As EventArgs) Handles btn_AddProc.Click
  frm_ProcedureSearch.Show()
End Sub
Private Sub frm_PrescriptionEditor_Load(sender As Object, e As EventArgs) Handles Me.Load
  ' Input boxes
  dtp_date.Value = DateTime.Now
  ' Data Tables
  dtb_med = New DataTable()
  dtb_proc = New DataTable()
  dtb_consol = New DataTable()
  dtb_med.Columns.Clear()
  dtb_med.Columns.Add("id", GetType(Integer))
  dtb_med.Columns.Add("name", GetType(String))
  dtb_med.Columns.Add("dosage", GetType(Integer))
  dtb_med.Columns.Add("unit", GetType(String))
  dtb_proc.Columns.Clear()
  dtb_proc.Columns.Add("id", GetType(Integer))
  dtb_proc.Columns.Add("name", GetType(String))
```

```vbnet
    dtb_consol.Columns.Clear()
    dtb_consol.Columns.Add("Medicines/Procedures", GetType(String))
    dtb_consol.Columns.Add("Additional Note", GetType(String))
    dgv_PresTable.DataSource = dtb_consol
    dgv_PresTable.Columns("Medicines/Procedures").ReadOnly = True
  End Sub
  Private Function getFormattedDate(dt As Date) As String
    Dim year As String = dt.Year.ToString()
    Dim month As String = dt.Month.ToString().PadLeft(2, "0")
    Dim day As String = dt.Day.ToString().PadLeft(2, "0")
    Return year + "-" + month + "-" + day
  End Function
  Private Sub btn_Save_Click(sender As Object, e As EventArgs) Handles btn_Save.Click
    If tb_Name.Text.Length > 50 Then
      MsgBox("Name max length is 50")
      Exit Sub
    End If
    DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
    Dim SQLiteCon As New SQLiteConnection(DBPath)
    Try
      SQLiteCon.Open()
    Catch ex As Exception
      SQLiteCon.Dispose()
      MsgBox("Error connecting to database:" & ex.Message)
      Exit Sub
    End Try
    Try
      ExecuteNonQuery("insert into " & TableName &
             "(user_id, patient_name , patient_age , patient_gender , date )" &
             " values('" &
             frm_LoginAuth.user_id &
```

```vbnet
                "','" & tb_Name.Text &

                "','" & num_age.Value &

                "','" & cb_gender.SelectedItem.ToString &

                "','" & getFormattedDate(dtp_date.Value) &

                "');", SQLiteCon)
    Dim dtb As New DataTable

    LoadDB("select last_insert_rowid()", dtb, SQLiteCon)

    Dim visit_id = dtb.Rows(0)(0)

    ' insert each medicine and procedure into med/proc table

    TableName = "prescriptions"

    ' medicines

    For Each row In dtb_med.Rows

      ExecuteNonQuery("insert into " & TableName &

            "(visit_id, medicine_id)" &

            " values('" &

            visit_id &

            "','" & row(0) &

            "');", SQLiteCon)

    Next

    ' procedures

    For Each row In dtb_proc.Rows

      ExecuteNonQuery("insert into " & TableName &

            "(visit_id, procedure_id)" &

            " values('" &

            visit_id &

            "','" & row(0) &

            "');", SQLiteCon)

    Next

    MsgBox("Prescription Saved")

  Catch ex As Exception

    MsgBox("Error Saving Data to Database: " & ex.Message)

    Exit Sub
```

```vbnet
      Finally
        SQLiteCon.Dispose()
        Me.Close()
        frm_UserHome.Show()
      End Try
    End Sub
    Private Sub Next_Enabled_Check(sender As Object, e As EventArgs) _
      Handles      tb_Name.TextChanged,      dtp_date.ValueChanged,      num_age.ValueChanged,
cb_gender.SelectedIndexChanged
        If dtp_date.Value = Nothing Or
          tb_Name.Text = Nothing Or tb_Name.Text = "" Or
          num_age.Value = Nothing Or
          cb_gender.SelectedIndex = -1 Then
          btn_Save.Enabled = False
          btn_Print.Enabled = False
        Else
          btn_Save.Enabled = True
          btn_Print.Enabled = True
        End If
      End Sub
    Private Sub btn_Print_Click(sender As Object, e As EventArgs) Handles btn_Print.Click
      Dim strPrint As String = ""
      strPrint += "------------------------------" & vbCrLf
      strPrint += "          MD-APP            " & vbCrLf
      strPrint += "------------------------------" & vbCrLf
      strPrint += "      Patient Details        " & vbCrLf
      strPrint += "------------------------------" & vbCrLf
      strPrint += "Name:" & vbTab & tb_Name.Text & vbCrLf
      strPrint += "Age:" & vbTab & num_age.Text & vbCrLf
      strPrint += "Gender:" & vbTab & cb_gender.SelectedItem & vbCrLf
      strPrint += "------------------------------" & vbCrLf
      strPrint += "      Visit Details         " & vbCrLf
```

```vb
    strPrint += "-----------------------------" & vbCrLf
    strPrint += "Doctor:" & vbTab & frm_LoginAuth.username & vbCrLf
    strPrint += "Date:" & vbTab & dtp_date.Value.ToLongDateString & vbCrLf
    strPrint += "-----------------------------" & vbCrLf
    strPrint += "        Prescription        " & vbCrLf
    strPrint += "-----------------------------" & vbCrLf
    For Each row As DataRow In dtb_consol.Rows()
      strPrint += "• " & row(0).ToString & vbTab & "(" & row(1).ToString & ")" & vbCrLf
    Next
    strPrint += "-----------------------------" & vbCrLf
    strPrint += "      Doctor's Advice        " & vbCrLf
    strPrint += "-----------------------------" & vbCrLf
    strPrint += rtb_Advice.Text & vbCrLf
    strPrint += "                            " & vbCrLf
    strPrint += "--------------*---------------" & vbCrLf
    Printer.Print(strPrint)
  End Sub
End Class
```

Medicine search:

```vb
Imports System.Data.SQLite
Imports System.Runtime.InteropServices
Public Class frm_MedicineSearch
#Region " Top Panel "
#Region " Move Form "
  Public MoveForm As Boolean
  Public MoveForm_MousePosition As Point
  Public Sub MoveForm_MouseDown(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseDown
    If e.Button = MouseButtons.Left Then
      MoveForm = True
      MoveForm_MousePosition = e.Location
```

```vbnet
    End If
  End Sub
  Public   Sub   MoveForm_MouseMove(sender   As   Object,   e   As   MouseEventArgs)   Handles
pnl_TopBar.MouseMove
    If MoveForm Then
      Me.Location += (e.Location - MoveForm_MousePosition)
    End If
  End Sub
  Public   Sub   MoveForm_MouseUp(sender   As   Object,   e   As   MouseEventArgs)   Handles
pnl_TopBar.MouseUp
    If e.Button = MouseButtons.Left Then
      MoveForm = False
    End If
  End Sub
#End Region
  Private Sub CloseApp(sender As Object, e As EventArgs) _
    Handles btn_Close.Click, ctxItem_Exit.Click, img_TopBar_Logo.DoubleClick
    Me.Close()
  End Sub
  Private Sub InvokeCtx(sender As Object, e As EventArgs) _
    Handles img_TopBar_Logo.Click
    ctx_Main.Show(MousePosition.X + 2, MousePosition.Y + 2)
  End Sub
#End Region
  Dim DBPath As String
  ReadOnly TableName As String = "medicines"
  Dim TableDB As New DataTable
  Private Sub LoadDB(ByVal q As String, ByVal tbl As DataTable, ByVal cn As SQLiteConnection)
    Dim SQLiteDA As New SQLiteDataAdapter(q, cn)
    SQLiteDA.Fill(tbl)
    SQLiteDA.Dispose()
  End Sub
```

```vb
 Private Sub LoadTable(sender As Object, e As EventArgs) _
   Handles tb_SearchInput.TextChanged, MyBase.Load
   DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
   Dim SQLiteCon As New SQLiteConnection(DBPath)
   Try
     SQLiteCon.Open()
   Catch ex As Exception
     SQLiteCon.Dispose()
     MsgBox("Error Opening Database: " & ex.Message)
     Exit Sub
   End Try
   Try
     TableDB.Clear()
     LoadDB("select * from " & TableName & " where name like '%" &
          tb_SearchInput.Text.Trim().ToLower() &
          "%'", TableDB, SQLiteCon)
     dgv_Medicines.DataSource = TableDB
     If dgv_Medicines.Columns.Count <> 0 Then
       dgv_Medicines.Columns(0).Visible = False
     End If
   Catch ex As Exception
     MsgBox("Error loading database: " & ex.Message)
     Exit Sub
   Finally
     TableDB.Dispose()
     SQLiteCon.Dispose()
   End Try
 End Sub
 Private Sub AddMed(sender As Object, e As EventArgs) _
   Handles btn_OK.Click, dgv_Medicines.CellDoubleClick
   If dgv_Medicines.SelectedRows.Count <> 1 Then
```

```
      MsgBox("Select a Medicine first!")
      Exit Sub
    End If
    Dim row As DataRow = dgv_Medicines.SelectedRows(0).DataBoundItem.Row
    Try
     frm_PrescriptionEditor.dtb_med.Rows.Add(row.ItemArray())
     frm_PrescriptionEditor.dtb_consol.Rows.Add(
       row(1).ToString & " (" &
       row(2).ToString & " " &
       row(3).ToString & ")")
    Catch ex As Exception
      MsgBox(ex.Message)
    End Try
    Me.Hide()
  End Sub
End Class
```

Procedure search:

```
Imports System.Data.SQLite
Imports System.Runtime.InteropServices
Public Class frm_ProcedureSearch
#Region " Top Panel "
#Region " Move Form "
  Public MoveForm As Boolean
  Public MoveForm_MousePosition As Point
  Public Sub MoveForm_MouseDown(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseDown
    If e.Button = MouseButtons.Left Then
      MoveForm = True
      MoveForm_MousePosition = e.Location
    End If
```

```vbnet
    End Sub
    Public Sub MoveForm_MouseMove(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseMove
      If MoveForm Then
        Me.Location += (e.Location - MoveForm_MousePosition)
      End If
    End Sub
    Public Sub MoveForm_MouseUp(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseUp
      If e.Button = MouseButtons.Left Then
        MoveForm = False
      End If
    End Sub
#End Region


    Private Sub CloseApp(sender As Object, e As EventArgs) _
      Handles btn_Close.Click, ctxItem_Exit.Click, img_TopBar_Logo.DoubleClick
      Me.Close()
    End Sub
    Private Sub InvokeCtx(sender As Object, e As EventArgs) _
      Handles img_TopBar_Logo.Click
      ctx_Main.Show(MousePosition.X + 2, MousePosition.Y + 2)
    End Sub
#End Region
    Dim DBPath As String
    ReadOnly TableName As String = "procedures"
    Dim TableDB As New DataTable
    Private Sub LoadDB(ByVal q As String, ByVal tbl As DataTable, ByVal cn As SQLiteConnection)
      Dim SQLiteDA As New SQLiteDataAdapter(q, cn)
      SQLiteDA.Fill(tbl)
      SQLiteDA.Dispose()
    End Sub
```

```vb
Private Sub AddProc(sender As Object, e As EventArgs) _
 Handles btn_OK.Click, dgv_Procedures.CellDoubleClick
 If dgv_Procedures.SelectedRows.Count <> 1 Then
  MsgBox("Select a Procedure first!")
  Exit Sub
 End If
 Dim row As DataRow = dgv_Procedures.SelectedRows(0).DataBoundItem.Row
 frm_PrescriptionEditor.dtb_proc.Rows.Add(row(0).ToString, row(1).ToString)
 frm_PrescriptionEditor.dtb_consol.Rows.Add(row(1).ToString)
 Me.Hide()
End Sub
Private Sub LoadTable(sender As Object, e As EventArgs) _
 Handles tb_SearchInput.TextChanged, MyBase.Load
 DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
 Dim SQLiteCon As New SQLiteConnection(DBPath)
 Try
   SQLiteCon.Open()
 Catch ex As Exception
   SQLiteCon.Dispose()
   MsgBox("Error Opening Database: " & ex.Message)
   Exit Sub
 End Try
 Try
   TableDB.Clear()
   LoadDB("select * from " & TableName & " where name like '%" &
       tb_SearchInput.Text.Trim().ToLower() &
       "%'", TableDB, SQLiteCon)
   dgv_Procedures.DataSource = TableDB
   If dgv_Procedures.Columns.Count <> 0 Then
     dgv_Procedures.Columns(0).Visible = False
   End If
```

```
      Catch ex As Exception
        MsgBox("Error loading database: " & ex.Message)
        Exit Sub
      Finally
        TableDB.Dispose()
        SQLiteCon.Dispose()
      End Try
    End Sub
End Class
```

Admin home:

```
Public Class frm_AdminHome
#Region " Top Panel "
#Region " Move Form "
  Public MoveForm As Boolean
  Public MoveForm_MousePosition As Point
  Public Sub MoveForm_MouseDown(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseDown
    If e.Button = MouseButtons.Left Then
      MoveForm = True
      MoveForm_MousePosition = e.Location
    End If
  End Sub
  Public Sub MoveForm_MouseMove(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseMove
    If MoveForm Then
      Me.Location += (e.Location - MoveForm_MousePosition)
    End If
  End Sub
  Public Sub MoveForm_MouseUp(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseUp
    If e.Button = MouseButtons.Left Then
```

```vbnet
      MoveForm = False
    End If
  End Sub
#End Region
  Private Sub CloseApp(sender As Object, e As EventArgs) _
    Handles btn_Close.Click, ctxItem_Exit.Click, img_TopBar_Logo.DoubleClick
    frm_MainMenu.Show()
    Me.Close()
  End Sub
  Private Sub InvokeCtx(sender As Object, e As EventArgs) _
    Handles img_TopBar_Logo.Click
    ctx_Main.Show(MousePosition.X + 2, MousePosition.Y + 2)
  End Sub
#End Region
  Private Sub btn_EditMed_Click(sender As Object, e As EventArgs) Handles btn_EditMed.Click
    frm_EditMed.Show()
    Me.Hide()
  End Sub
  Private Sub btn_EditProc_Click(sender As Object, e As EventArgs) Handles btn_EditProc.Click
    frm_EditProc.Show()
    Me.Hide()
  End Sub
    Private Sub btn_PatHist_Click(sender As Object, e As EventArgs) Handles btn_PatHist.Click
    frm_PatHist.Show()
    Me.Hide()
  End Sub
End Class
```

Edit medicine:

```vbnet
Imports System.Data.SQLite
Imports System.Runtime.InteropServices
Public Class frm_EditMed
```

```vbnet
#Region " Top Panel "
#Region " Move Form "
  Public MoveForm As Boolean
   Public MoveForm_MousePosition As Point
   Public Sub MoveForm_MouseDown(sender As Object, e As MouseEventArgs) Handles pnl_TopBar.MouseDown
     If e.Button = MouseButtons.Left Then
       MoveForm = True
       MoveForm_MousePosition = e.Location
     End If
   End Sub
   Public Sub MoveForm_MouseMove(sender As Object, e As MouseEventArgs) Handles pnl_TopBar.MouseMove
     If MoveForm Then
       Me.Location += (e.Location - MoveForm_MousePosition)
     End If
   End Sub
   Public Sub MoveForm_MouseUp(sender As Object, e As MouseEventArgs) Handles pnl_TopBar.MouseUp
     If e.Button = MouseButtons.Left Then
       MoveForm = False
     End If
   End Sub
#End Region
  Private Sub CloseApp(sender As Object, e As EventArgs) _
  Handles btn_Close.Click, ctxItem_Exit.Click, img_TopBar_Logo.DoubleClick
    frm_AdminHome.Show()
    Me.Close()
  End Sub
  Private Sub InvokeCtx(sender As Object, e As EventArgs) _
   Handles img_TopBar_Logo.Click
    ctx_Main.Show(MousePosition.X + 2, MousePosition.Y + 2)
```

```vbnet
    End Sub
#End Region
  Dim DBPath As String
  ReadOnly TableName As String = "medicines"
  Dim TableDB As New DataTable
  Dim med_id As Integer = -1
  Private Sub LoadDB(ByVal q As String, ByVal tbl As DataTable, ByVal cn As SQLiteConnection)
    Dim SQLiteDA As New SQLiteDataAdapter(q, cn)
    SQLiteDA.Fill(tbl)
    SQLiteDA.Dispose()
  End Sub
  Private Sub ExecuteNonQuery(ByVal query As String, ByVal cn As SQLiteConnection)
    Dim SQLiteCM As New SQLiteCommand(query, cn)
    SQLiteCM.ExecuteNonQuery()
    SQLiteCM.Dispose()
  End Sub
  Private Sub LoadTable(sender As Object, e As EventArgs) _
    Handles tb_SearchInput.TextChanged, MyBase.Load
    DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
    Dim SQLiteCon As New SQLiteConnection(DBPath)
    Try
      SQLiteCon.Open()
    Catch ex As Exception
      SQLiteCon.Dispose()
      MsgBox("Error Opening Database: " & ex.Message)
      Exit Sub
    End Try
    Try
      TableDB.Clear()
      LoadDB("select * from " & TableName & " where name like '%" &
          tb_SearchInput.Text.Trim().ToLower() &
          "%'", TableDB, SQLiteCon)
```

```vb
    dgv_Medicines.DataSource = TableDB
    If dgv_Medicines.Columns.Count <> 0 Then
      dgv_Medicines.Columns(0).Visible = False
    End If
    dgv_Medicines.ClearSelection()
  Catch ex As Exception
    MsgBox("Error loading database: " & ex.Message)
    Exit Sub
  Finally
    TableDB.Dispose()
    SQLiteCon.Dispose()
  End Try
End Sub
Private Sub SelectMed(sender As Object, e As EventArgs) _
  Handles dgv_Medicines.CellClick
  If dgv_Medicines.SelectedRows.Count <> 1 Then
    MsgBox("Select a Medicine first!")
    Exit Sub
  End If
  Dim row As DataRow = dgv_Medicines.SelectedRows(0).DataBoundItem.Row
  Try
    med_id = row("medicine_id")
    tb_medname.Text = row("name")
    tb_meddose.Text = row("dosage")
    tb_medunit.Text = row("unit")
  Catch ex As Exception
    MsgBox(ex.Message)
  End Try
End Sub
Private Sub UpdateMed(sender As Object, e As EventArgs) Handles btn_update.Click
  If med_id = -1 Then
    MsgBox("Select a row to update first")
```

```vbnet
      Exit Sub
    End If
    DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
    Dim SQLiteCon As New SQLiteConnection(DBPath)
    Try
      SQLiteCon.Open()
    Catch ex As Exception
      SQLiteCon.Dispose()
      MsgBox("Error Opening Database: " & ex.Message)
      Exit Sub
    End Try
    Try
      ExecuteNonQuery("update " & TableName &
              " set name = '" & tb_medname.Text.Trim() & "'" &
              ", dosage = '" & tb_meddose.Text.Trim() & "'" &
              ", unit = '" & tb_medunit.Text.Trim() & "'" &
              " where medicine_id = '" & med_id & "'",
              SQLiteCon)
      LoadTable(sender, e)
    Catch ex As Exception
      MsgBox("Error updating database: " & ex.Message)
      Exit Sub
    Finally
      TableDB.Dispose()
      SQLiteCon.Dispose()
      tb_meddose.Text = ""
      tb_medname.Text = ""
      tb_medunit.Text = ""
      tb_SearchInput.Text = ""
      med_id = -1
      dgv_Medicines.ClearSelection()
    End Try
```

```vbnet
  End Sub
  Private Sub InsertMed(sender As Object, e As EventArgs) Handles btn_new.Click
    DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
    Dim SQLiteCon As New SQLiteConnection(DBPath)
    Try
      SQLiteCon.Open()
    Catch ex As Exception
      SQLiteCon.Dispose()
      MsgBox("Error Opening Database: " & ex.Message)
      Exit Sub
    End Try
    Try
      ExecuteNonQuery("insert into " & TableName & "(name, dosage, unit) values('" &
              tb_medname.Text.Trim() & "','" &
              tb_meddose.Text.Trim() & "','" &
              tb_medunit.Text.Trim() &
              "')", SQLiteCon)
      LoadTable(sender, e)
    Catch ex As Exception
      MsgBox("Error inserting database: " & ex.Message)
      Exit Sub
    Finally
      TableDB.Dispose()
      SQLiteCon.Dispose()
      tb_meddose.Text = ""
      tb_medname.Text = ""
      tb_medunit.Text = ""
      tb_SearchInput.Text = ""
      med_id = -1
      dgv_Medicines.ClearSelection()
    End Try
  End Sub
```

```vbnet
 Private Sub DeleteMed(sender As Object, e As EventArgs) Handles btn_delete.Click
   If med_id = -1 Then
     MsgBox("Select a row to delete first")
     Exit Sub
   End If
   DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
   Dim SQLiteCon As New SQLiteConnection(DBPath)
   Try
     SQLiteCon.Open()
   Catch ex As Exception
     SQLiteCon.Dispose()
     MsgBox("Error Opening Database: " & ex.Message)
     Exit Sub
   End Try
   Try
     ExecuteNonQuery("delete from " & TableName & " where medicine_id='" & med_id & "'",
SQLiteCon)
     LoadTable(sender, e)
   Catch ex As Exception
     MsgBox("Error deleting database: " & ex.Message)
     Exit Sub
   Finally
     TableDB.Dispose()
     SQLiteCon.Dispose()
     tb_meddose.Text = ""
     tb_medname.Text = ""
     tb_medunit.Text = ""
     tb_SearchInput.Text = ""
     med_id = -1
     dgv_Medicines.ClearSelection()
   End Try
```

End Sub

End Class

Edit Procedure:

```vb
Imports System.Data.SQLite

Imports System.Runtime.InteropServices

Public Class frm_EditProc

#Region " Top Panel "

#Region " Move Form "

  Public MoveForm As Boolean

    Public MoveForm_MousePosition As Point

    Public Sub MoveForm_MouseDown(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseDown

      If e.Button = MouseButtons.Left Then

        MoveForm = True

        MoveForm_MousePosition = e.Location

      End If

    End Sub

    Public Sub MoveForm_MouseMove(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseMove

      If MoveForm Then

        Me.Location += (e.Location - MoveForm_MousePosition)

      End If

    End Sub

    Public Sub MoveForm_MouseUp(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseUp

      If e.Button = MouseButtons.Left Then

        MoveForm = False

      End If

    End Sub

#End Region
```

```vbnet
    Private Sub CloseApp(sender As Object, e As EventArgs) _
    Handles btn_Close.Click, ctxItem_Exit.Click, img_TopBar_Logo.DoubleClick
      frm_AdminHome.Show()
      Me.Close()
    End Sub
    Private Sub InvokeCtx(sender As Object, e As EventArgs) _
      Handles img_TopBar_Logo.Click
        ctx_Main.Show(MousePosition.X + 2, MousePosition.Y + 2)
      End Sub
#End Region
  Dim DBPath As String
  ReadOnly TableName As String = "procedures"
  Dim TableDB As New DataTable
  Dim proc_id As Integer = -1
  Private Sub LoadDB(ByVal q As String, ByVal tbl As DataTable, ByVal cn As SQLiteConnection)
      Dim SQLiteDA As New SQLiteDataAdapter(q, cn)
      SQLiteDA.Fill(tbl)
      SQLiteDA.Dispose()
    End Sub
    Private Sub ExecuteNonQuery(ByVal query As String, ByVal cn As SQLiteConnection)
      Dim SQLiteCM As New SQLiteCommand(query, cn)
      SQLiteCM.ExecuteNonQuery()
      SQLiteCM.Dispose()
    End Sub
    Private Sub LoadTable(sender As Object, e As EventArgs) _
    Handles tb_SearchInput.TextChanged, MyBase.Load
      DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
      Dim SQLiteCon As New SQLiteConnection(DBPath)
      Try
        SQLiteCon.Open()
      Catch ex As Exception
        SQLiteCon.Dispose()
```

```vbnet
        MsgBox("Error Opening Database: " & ex.Message)
        Exit Sub
      End Try
      Try
        TableDB.Clear()
        LoadDB("select * from " & TableName & " where name like '%" &
            tb_SearchInput.Text.Trim().ToLower() &
            "%'", TableDB, SQLiteCon)
      dgv_Procedures.DataSource = TableDB
      If dgv_Procedures.Columns.Count <> 0 Then
        dgv_Procedures.Columns(0).Visible = False
      End If
      dgv_Procedures.ClearSelection()
    Catch ex As Exception
      MsgBox("Error loading database: " & ex.Message)
      Exit Sub
    Finally
      TableDB.Dispose()
      SQLiteCon.Dispose()
    End Try
  End Sub
  Private Sub SelectProc(sender As Object, e As EventArgs) _
    Handles dgv_Procedures.CellClick
    If dgv_Procedures.SelectedRows.Count <> 1 Then
      MsgBox("Select a Procedure first!")
      Exit Sub
    End If
    Dim row As DataRow = dgv_Procedures.SelectedRows(0).DataBoundItem.Row
    Try
      proc_id = row("procedure_id")
      tb_procname.Text = row("name")
    Catch ex As Exception
```

```vb
        MsgBox(ex.Message)
      End Try
    End Sub
    Private Sub UpdateProc(sender As Object, e As EventArgs) Handles btn_update.Click
      If proc_id = -1 Then
        MsgBox("Select a row to update first")
        Exit Sub
      End If
      DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
      Dim SQLiteCon As New SQLiteConnection(DBPath)
      Try
        SQLiteCon.Open()
      Catch ex As Exception
        SQLiteCon.Dispose()
        MsgBox("Error Opening Database: " & ex.Message)
        Exit Sub
      End Try
      Try
        ExecuteNonQuery("update " & TableName &
                " set name = '" & tb_procname.Text.Trim() & "'" &
                " where procedure_id = '" & proc_id & "'",
                SQLiteCon)
        LoadTable(sender, e)
      Catch ex As Exception
        MsgBox("Error updating database: " & ex.Message)
        Exit Sub
      Finally
        TableDB.Dispose()
        SQLiteCon.Dispose()
        tb_procname.Text = ""
        tb_SearchInput.Text = ""
        proc_id = -1
```

```vb
        dgv_Procedures.ClearSelection()
      End Try
    End Sub
    Private Sub InsertProc(sender As Object, e As EventArgs) Handles btn_new.Click
      DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
      Dim SQLiteCon As New SQLiteConnection(DBPath)
      Try
        SQLiteCon.Open()
      Catch ex As Exception
        SQLiteCon.Dispose()
        MsgBox("Error Opening Database: " & ex.Message)
        Exit Sub
      End Try
      Try
        ExecuteNonQuery("insert into " & TableName & "(name) values('" &
                tb_procname.Text.Trim() &
                "')", SQLiteCon)
        LoadTable(sender, e)
      Catch ex As Exception
        MsgBox("Error inserting database: " & ex.Message)
        Exit Sub
      Finally
        TableDB.Dispose()
        SQLiteCon.Dispose()
        tb_procname.Text = ""
        tb_SearchInput.Text = ""
        proc_id = -1
        dgv_Procedures.ClearSelection()
      End Try
    End Sub
    Private Sub DeleteProc(sender As Object, e As EventArgs) Handles btn_delete.Click
      If proc_id = -1 Then
```

```vb
        MsgBox("Select a row to delete first")
        Exit Sub
    End If
    DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
    Dim SQLiteCon As New SQLiteConnection(DBPath)
    Try
        SQLiteCon.Open()
    Catch ex As Exception
        SQLiteCon.Dispose()
        MsgBox("Error Opening Database: " & ex.Message)
        Exit Sub
    End Try
    Try
        ExecuteNonQuery("delete from " & TableName & " where procedure_id='" & proc_id & "'",
SQLiteCon)
        LoadTable(sender, e)
    Catch ex As Exception
        MsgBox("Error deleting database: " & ex.Message)
        Exit Sub
    Finally
        TableDB.Dispose()
        SQLiteCon.Dispose()
        tb_procname.Text = ""
        tb_SearchInput.Text = ""
        proc_id = -1
        dgv_Procedures.ClearSelection()
    End Try
  End Sub
End Class
```

Patient History:

```vbnet
Imports System.Data.SQLite
Imports System.Runtime.InteropServices
Public Class frm_PatHist
#Region " Top Panel "
#Region " Move Form "
  Public MoveForm As Boolean
  Public MoveForm_MousePosition As Point
  Public Sub MoveForm_MouseDown(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseDown
    If e.Button = MouseButtons.Left Then
      MoveForm = True
      MoveForm_MousePosition = e.Location
    End If
  End Sub
  Public Sub MoveForm_MouseMove(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseMove
    If MoveForm Then
      Me.Location += (e.Location - MoveForm_MousePosition)
    End If
  End Sub
  Public Sub MoveForm_MouseUp(sender As Object, e As MouseEventArgs) Handles
pnl_TopBar.MouseUp
    If e.Button = MouseButtons.Left Then
      MoveForm = False
    End If
  End Sub
#End Region
  Private Sub CloseApp(sender As Object, e As EventArgs) _
    Handles btn_Close.Click, ctxItem_Exit.Click, img_TopBar_Logo.DoubleClick
    If frm_LoginAuth.is_admin = 0 Then
      frm_UserHome.Show()
```

```vbnet
      Else
        frm_AdminHome.Show()
      End If
      Me.Close()
    End Sub
    Private Sub InvokeCtx(sender As Object, e As EventArgs) _
      Handles img_TopBar_Logo.Click
      ctx_Main.Show(MousePosition.X + 2, MousePosition.Y + 2)
    End Sub
#End Region
  Dim DBPath As String
  ReadOnly TableName As String = "visits"
  Dim TableDB As New DataTable
  Dim visit_id As Integer = -1
  Private Sub LoadDB(ByVal q As String, ByVal tbl As DataTable, ByVal cn As SQLiteConnection)
    Dim SQLiteDA As New SQLiteDataAdapter(q, cn)
    SQLiteDA.Fill(tbl)
    SQLiteDA.Dispose()
  End Sub
  Private Sub ExecuteNonQuery(ByVal query As String, ByVal cn As SQLiteConnection)
    Dim SQLiteCM As New SQLiteCommand(query, cn)
    SQLiteCM.ExecuteNonQuery()
    SQLiteCM.Dispose()
  End Sub
  Private Sub LoadTable(sender As Object, e As EventArgs) _
    Handles tb_SearchInput.TextChanged, MyBase.Load
    DBPath = "Data Source=" & Application.StartupPath & "\data.db;"
    Dim SQLiteCon As New SQLiteConnection(DBPath)
    Try
      SQLiteCon.Open()
    Catch ex As Exception
      SQLiteCon.Dispose()
```

```vbnet
      MsgBox("Error Opening Database: " & ex.Message)
      Exit Sub
    End Try
    Try
      TableDB.Clear()
      LoadDB("select * from " & TableName & " where ( patient_name like '%" &
            tb_SearchInput.Text.Trim().ToLower() &
            "%' or  patient_age like '%" &
            tb_SearchInput.Text.Trim().ToLower() &
            "%' or  patient_gender like '%" &
            tb_SearchInput.Text.Trim().ToLower() &
            "%' or date like '%" &
            tb_SearchInput.Text.Trim().ToLower() &
            If(frm_LoginAuth.is_admin = 0, "%' ) and user_id = '" & frm_LoginAuth.user_id & "'", "%'
)") _
            & "order by date desc" _
          , TableDB, SQLiteCon)
      dgv_Visits.DataSource = TableDB
      If dgv_Visits.Columns.Count <> 0 Then
        dgv_Visits.Columns(0).Visible = False
      End If
      dgv_Visits.ClearSelection()
    Catch ex As Exception
      MsgBox("Error loading database: " & ex.Message)
      Exit Sub
    Finally
      TableDB.Dispose()
      SQLiteCon.Dispose()
    End Try
  End Sub
End Class
```

Prescription Printing:

```
Public Class Printer
  Private Shared Lines As New Queue(Of String)
  Private Shared _myfont As Font
  Private Shared prn As Printing.PrintDocument
  Shared Sub New()
    _myfont = New Font("Courier New",
          8, FontStyle.Regular, GraphicsUnit.Point)
    prn = New Printing.PrintDocument
    AddHandler prn.PrintPage, AddressOf PrintPageHandler
  End Sub
  Public Shared Sub Print(ByVal text As String)
    Dim linesarray() = text.Split(New String() _
       {Environment.NewLine}, StringSplitOptions.None)
    For Each line As String In linesarray
      Lines.Enqueue(line)
    Next
    prn.Print()
  End Sub
  Private Shared Sub PrintPageHandler(ByVal sender As Object,
     ByVal e As Printing.PrintPageEventArgs)
    Dim sf As New StringFormat()
    Dim vpos As Single = e.PageSettings.HardMarginY + 30
    Do While Lines.Count > 0
      Dim line As String = Lines.Dequeue
      Dim sz As SizeF = e.Graphics.MeasureString(
         line, _myfont, e.PageSettings.Bounds.Size, sf)
      Dim rct As New RectangleF(
         e.PageSettings.HardMarginX + 65, vpos,
         e.PageBounds.Width - e.PageSettings.HardMarginX * 2 - 65,
         e.PageBounds.Height - e.PageSettings.HardMarginY * 2 - 65)
      e.Graphics.DrawString(line, _myfont, Brushes.Black, rct)
```

```
        vpos += sz.Height
        If vpos > e.PageSettings.Bounds.Height -
            e.PageSettings.HardMarginY * 2 Then
          e.HasMorePages = True
          Exit Sub
        End If
      Loop
    End Sub
  End Class
```

4. SYSTEM CONFIGURATION

4.1 Hardware Requirements

DESKTOP/LAPTOP : BOTH

PROCESSOR : INTEL® PENTIUM® 4CPU 3.06GHz

RAM : 2GB SYSTEM TYPE : 32BIT OPERATING SYSTEM OR 64BIT OPERATING SYSTEM

HARD DISK : 30GB

4.2 Software Requirements

OPERATIONAL SYSTEM : WINDOWS XP OR BEYOND

PROGRAMMING LANGUAGE : VB.NET

DATABASE OR DBMS : SQLite3

TOOL(S) : MICROSOFT PROJECT PLANNER 2020

DOCUMENTATION : MICROSOFT WORD 2019

5. DETAILS OF SOFTWARE

5.1 Overview of Front-End

Microsoft Visual Studio 2012 is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silver light. It can produce both native code and managed code.

Visual Studio supports different programming languages and allows the code editor and debugger to support nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, Visual C++ and VB.NET. Support for other languages such as Python, Ruby, Node.js, and M among others is available via language services installed separately. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Java (and J#) was supported in the past.

Microsoft provides a free version of Visual Studio called the Community edition that supports plug-in and is available at no cost for all users. Support for programming languages is added by using a specific VSPackage called a LanguageService. A language service defines various interfaces which the VSPackage implementation can implement to add support for various functionalities. Functionalities that can be added this way include syntax coloring, statement completion; brace matching, parameter information tooltips, member lists and error markers for background compilation. If the interface is implemented, the functionality will be available for the language. Language services are implemented on a per-language basis.

FEATURES:

Boolean Conditions
- Automatic Garbage Collection
- Standard Library
- Assembly Versioning
- Properties and Events
- Delegates and Events Management

- Easy-to-use Generics
- Indexers
- Conditional Compilation
- Simple Multithreading

ADVANTAGES:

The structure of the Basic programming language is very simple, particularly as to the executable code.

1. VB is not only a language but primarily an integrated, interactive development environment ("IDE").

2. The VB-IDE has been highly optimized to support rapid application development ("RAD"). It is particularly easy to develop graphical user interfaces and to connect them to handler functions provided by the application.

3. The graphical user interface of the VB-IDE provides intuitively appealing views for the management of the program structure in the large and the various types of entities (classes, modules, procedures, forms, …)

4. VB provides a comprehensive interactive and context-sensitive online help system.

5. When editing program texts the "IntelliSense" technology informs you in a little popup window about the types of constructs that may be entered at the current cursor location.

6. VB is a component integration language which is attuned to Microsoft's Component Object Model ("COM").

7. COM components can be written in different languages and then integrated using VB.

8. Interfaces of COM components can be easily called remotely via Distributed COM ("DCOM"), which makes it easy to construct distributed applications.

9. COM components can be embedded in / linked to your application's user interface and also in/to stored documents (Object Linking and Embedding "OLE", "Compound Documents").

10. There is a wealth of readily available COM components for many different purposes.

11. Visual Basic is built around the .NET environment used by all Microsoft Visual languages, so there is very little that can't be done in Visual Basic that can be done in other languages (such as C#)

DISADVANTAGES:

1. Visual basic is a proprietary programming language written by Microsoft, so programs written in Visual basic cannot, easily, be transferred to other operating systems.

2. There are some, fairly minor disadvantages compared with C. C has better declaration of arrays – it's possible to initialize an array of structures in C at declaration time; this is impossible in VB5.2

5.2 Overview of Back-End

SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.

SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. SQLite accesses its storage files directly.

Why SQLite?

- SQLite does not require a separate server process or system to operate (serverless).

- SQLite comes with zero-configuration, which means no setup or administration needed.

- A complete SQLite database is stored in a single cross-platform disk file.

- SQLite is very small and light weight, less than 400KiB fully configured or less than 250KiB with optional features omitted.

- SQLite is self-contained, which means no external dependencies.

- SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads.

SQLite Limitations

There are few unsupported features of SQL92 in SQLite which are listed in the following table.

Sr.No.	Feature & Description
1	**RIGHT OUTER JOIN** Only LEFT OUTER JOIN is implemented.

2	**FULL OUTER JOIN** Only LEFT OUTER JOIN is implemented.
3	**ALTER TABLE** The RENAME TABLE and ADD COLUMN variants of the ALTER TABLE command are supported. The DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT are not supported.
4	**Trigger support** FOR EACH ROW triggers are supported but not FOR EACH STATEMENT triggers.
5	**VIEWs** VIEWs in SQLite are read-only. You may not execute a DELETE, INSERT, or UPDATE statement on a view.
6	**GRANT and REVOKE** The only access permissions that can be applied are the normal file access permissions of the underlying operating system.

SQLite Commands

The standard SQLite commands to interact with relational databases are similar to SQL. They are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into groups based on their operational nature −

DDL - Data Definition Language

Sr.No.	Command & Description
1	**CREATE** Creates a new table, a view of a table, or other object in database.
2	**ALTER** Modifies an existing database object, such as a table.
3	**DROP** Deletes an entire table, a view of a table or other object in the database.

DML - Data Manipulation Language

Sr.No.	Command & Description
1	**INSERT** Creates a record
2	**UPDATE** Modifies records
3	**DELETE** Deletes records

DQL - Data Query Language

Sr.No.	Command & Description
1	**SELECT** Retrieves certain records from one or more tables

5.3 ABOUT THE PLATFORM

Windows is a series of Operating Systems developed by Microsoft. Each version of Windows includes a Graphical User Interface, with a desktop that allows users to view files and folders in Windows. For the past two decades, Windows has been the most widely used operating system for personal computers PCs.

Microsoft Windows is designed for both home computing and professional purposes. Past versions of Windows home editions include Windows 3.0 (1990), Windows 3.1 (1992), Windows 95 (1995), Windows 98 (1998), Windows Me (2000), Windows XP (2001), and Windows Vista (2006). The current version, Windows 7, was released in 2009.

The first business-oriented version of Windows, called Windows NT 3.1, was in 1993. This was followed by Windows 3.5, 4.0, and Windows 2000. When Microsoft released Windows XP in 2001, the company simply created different editions of the operating system for personal and business purposes. Windows Vista and Windows 7 have followed the same release strategy. Windows is designed to run on standard x86 hardware, such as Intel and AMD processors.

6. TESTING PHASE

Testing is a vital part of software development, and it is important to start it as early as possible, and to make testing a part of the process of deciding requirements. To get the most useful perspective on your development project, it is worthwhile devoting some thought to the entire lifecycle including how feedback from users will influence the future of the application. The tools and techniques we've discussed in this book should help your team to be more responsive to changes without extra cost, despite the necessarily wide variety of different development processes. Nevertheless, new tools and process improvements should be adopted gradually, assessing the results after each step.

Testing is part of a lifecycle. The software development lifecycle is one in which you hear of a need, you write some code to fulfil it, and then you check to see whether you have pleased the stakeholders—the users, owners, and other people who have an interest in what the software does. Hopefully they like it, but would also like some additions or changes, so you update or augment your code; and so the cycle continues. This cycle might happen every few days, as it does in Fabrikam's ice cream vending project, or every few years, as it does in Contoso's carefully specified and tested healthcare support system. Software development lifecycle

Testing is a proxy for the customer. You could conceivably do your testing by releasing it into the wild and waiting for the complaints and compliments to come back. Some companies have been accused of having such a strategy as their business model even before it became fashionable. But on the whole, the books are better balanced by trying to make sure that the software will satisfy the customer before we hand it over. We therefore design tests based on the stakeholders' needs, and run the tests before the product reaches the users. Preferably well before then, so as not to waste our time working on something that isn't going to do the job.

In this light, two important principles become clear:

• Tests represent requirements. Whether you write user stories on sticky notes on the wall, or use cases in a big thick document, your tests should be derived from and linked to those requirements. And as we've said, devising tests is a good vehicle for discussing the requirements.

• We're not done till the tests pass. The only useful measure of completion is when tests have been performed successfully

Those principles apply no matter how you develop your software.

Software Testing Types:

• Black box testing – Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.

• White box testing – This testing is based on knowledge of the internal logic of an application's code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.

• Unit testing – Testing of individual software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. May require developing test drive modules or test harnesses.

• Functional testing – This type of testing ignores the internal parts and focus on the output is as per requirement or not. Black-box type testing geared to functional requirements of an application.

• System testing – Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system

• Performance testing – Term often used interchangeably with 'stress' and 'load' testing. To check whether system meets performance requirements. Used different performance and load tools to do this.

• Usability testing – User-friendliness check. Application flow is tested, Can new user understand the application easily, Proper help documented whenever user stuck at any point. Basically system navigation is checked in this testing.

• Security testing – Can system be penetrated by any hacking way. Testing how well the system protects against unauthorized internal or external access. Checked if system, database is safe from external attacks.

• Alpha testing – In house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing

7. CONCLUSION AND FUTURE ENHANCEMENT

Conclusion:

In this modern day and age the only thing done in the primitive way is writing prescriptions by practitioners. Most of the time the prescription will be written by them using pen and paper in a hurry, resulting in oftentimes the writing being illegible. The doctors are unable to access their patient's previous records too for future consultations.

MD.APP solves that requirement by computerizing the prescription composing and printing procedure, aided by a gigantic database of all the medicines and medical procedures present. The software backs up the prescription and patient details as well, so the data is never lost. The prescribing doctor can thus, in future, search and access their past prescriptions. This also helps the clinic management to monitor all the doctor's prescriptions. Records are immutable and cannot be changed once prescribed.

Future Enhancement:

MD-APP currently works totally offline, which is beneficial for places with low connectivity but the app thus works in a standalone fashion, and its database is stored locally only.

So multiple practitioners in a hospital running separate instances of the applications maintain different local databases and they are not centralized or backed up centrally. Making the administration work difficult. The future versions can include networking support to centralize and backup the databases.

Other future scopes:

- Including signature of the doctor
- Smart authentication like biometric, etc
- Email/SMS of prescription
- Reminder system for check-up appointments
- Photo of patient

8. BIBLIOGRAPHY

1. www.youtube.com
2. www.github.com
3.   www.tutorialspoint.com
4.   www.geeksforgeeks.org

APPENDICES A-Table Structure

Table structure

Table name: users

FIELD	DATA TYPE	CONSTRAINTS	DESCRIPTION
User_id	Varchar(16)	Primary key	The unique id of each user
Password_hash	Varchar(128)	Not null	The SHA512 hash of username and password used for authentication
Name	Varchar(50)	Not null	Name of the user
Dept	Varchar(128)	Not null	Department of the user
Is_admin	Number(1)	Not null	Whether user is admin or not

Table name: medicines

FIELD	DATA TYPE	CONSTRAINTS	DESCRIPTION
Medicine_id	Integer	Primary key, Auto Increment	Unique key of each medicine
Name	Varchar(100)	Not null	Name of the medicine
dosage	Varchar(6,2)	Nullable	Dosage of the medicine
unit	Varchar(5)	Nullable	Unit of the dosage

Table name: procedures

FIELD	DATA TYPE	CONSTRAINTS	DESCRIPTION
Procedure_id	Integer	Primary key, Auto Increment	Unique key of each procedure
name	Varchar(100)	Not null	Name of the procedure

Table name: visits

FIELD	DATA TYPE	CONSTRAINTS	DESCRIPTION
Visit_id	Integer	Primary key, Auto Increment	Unique id of the visit
User_id	Varchar(16)	Not null, foreign key	Id of the doctor attending the visit
Patient_name	Varchar(50)	Not null	Name of the patient
Patient_age	Int	Not null	Age of the patient
Patient_gender	Varchar(6)	Not null	Gender of the person
date	date	Not null	Date of the visit

Table name: prescriptions

FIELD	DATA TYPE	CONSTRAINTS	DESCRIPTION
Prescription_id	Integer	Primary key, Auto Increment	Unique id of the prescription
Visit_id	Integer	Not null, foreign key	Id of the visit where it was prescribed
Medicine_id	Integer	Not null, foreign key	Id of the medicine prescribed
Procedure_id	Integer	Not null, foreign key	Id of the procedure prescribed

APPENDICES B-Screenshots

MD-APP MODULE DESIGN:

APPENDICES C-Sample Report of test cases

Test cases:

Sr. No.	Test ID	Test Description	Steps to Execute	Test Data	Expected Result	Actual Result	Status
1	T1	Correct username and password	enter the username and password, login	Username: sayan Password: password	You are authenticated	You are authenticated	Pass
2	T2	Wrong username and correct password	enter the username and password, login	Username: Admin123 Password: password	Invalid Credentials.	Invalid Credentials	Pass
3	T3	Correct username and wrong password	enter the username and password, login	Username: sayan Password: 123456	Invalid Credentials.	Invalid Credentials.	Pass
4	T4	Wrong username and wrong password	enter the username and password, login	Username: Admin123 Password: 12345	Invalid Credentials.	Invalid Credentials.	Pass