Chapter 1: INTRODUCTION

Clinic Management System is a software developed for managing different clinics. It facilitates access to the Clinic information of a particular patient or staff member. This will also help in evaluating Clinic progress.

Clinic Management System is software developed for daily evaluation of clinic continuous assessment records, and performance in accordance with the principle of the clinic. It is facilitated to access the performance and information of the Clinic of a particular patient with their previous health histories. The information is sorted by the admin or receptionist. This system will also enable the analysis of the health histories of patients.

The purpose of the Clinic management system is to computerize the traditional way of maintaining a Clinic. Another purpose for developing this software is to generate the report automatically at the end of the quarter or in between of the session.

Smart Data Management: Apart from the speed and the cost-savings that an automated Clinic tracking system offers, another essential feature of leading-edge e-Clinic software is the capability to manage vast data effectively. Clinic data management includes data compilation, generation of consolidated and category-wise reports, data summaries for given time periods, data storage with data integrity guaranteed, and finally data recovery, including recovery of back-dated reports.

1.1 Study of the problem

Apparently in today's world things cannot be done organized, accurately and efficiently using a file based system so making things globally in a digitized way is a tremendous advantage in this type of Clinic Management System program. Just think what if your clinic does not have an online presence, the chances are you could be losing valuable and efficient data and by doing so you will eventually lose business and customers to your competitors who do have an effective and active online presence. Entries in hard-bound patient registers can easily be tampered with and reports on paper spreadsheets are equally vulnerable to manipulation or destruction. Paperless operations are practically tamper-proof, if the clinic management system provides for role-based access.

1.2 Scope

Daily functions like patient registration, managing admission and overall management of various services can be easily performed with higher accuracy after the installation of clinic software. The modules of clinic management software are user-friendly and easy to access.

1.3 Objective

The purpose of the clinic management system is to computerize the traditional way of managing a clinic. Another purpose for developing this software is to generate the report automatically at the end of the quarter or in between of the session.

- ➔ The User requirements for the new system are to make the system fast, flexible, less prone to errors and reduce expenses and save time.
- ➔ A system that can automate the checking of data which are pre-stored.
- ➔ A facility that can generate result charts as per required.
- ➔ The New system should be more secure in managing patient records and reliable enough to be used in any condition.
- ➔ Finally, it should prove cost effective as compared to the current system.

1.4Infrastructure

Clinic management system is a necessary tool for managing a clinic in any environment where management is critical. However, most of the existing approaches are time consuming, intrusive and require manual work from the users. This research is aimed at developing a less intrusive, cost effective and more efficient automated clinic management system using Visual Studio Code with python with django framework.

Chapter 2: SOFTWARE REQUIREMENTS ANALYSIS

2.1 SRS specifications

There are various software development approaches defined and designed which are used/employed during the development process of software, these approaches are also referred as "Software Development Process Models" (e.g. Waterfall model, Incremental model, etc.). Each process model follows a particular life cycle in order to ensure success in the process of software development. Software life cycle models describe phases of the software cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced according to the design which is called the development phase. After coding and development the testing verifies the deliverable of the implementation phase against requirements.

2.2 Phases, models of the project

These are the following six phases in every Software development life cycle model:

- ➔ Design
- ➔ Implementation or coding
- ➔ Testing
- ➔ Deployment
- ➔ Maintenance

Requirement gathering and analysis: business requirements are gathered in this phase. This phase is the main focus of the project managers and stakeholders. Meetings with managers, stakeholders and users are held in order to determine the requirements like; who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be developed is also studied. Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

Design: in this phase the system and software design is prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

Implementation/ Coding: on receiving system design documents, the work is divided in modules/ units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

Testing: after the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase unit testing, integration testing, system testing, acceptance testing are done.

Deployment: after successful testing the product is delivered/ deployed to the customer for their use.
Maintenance: once when the customers start using the developed software then the actual problems come up and need to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

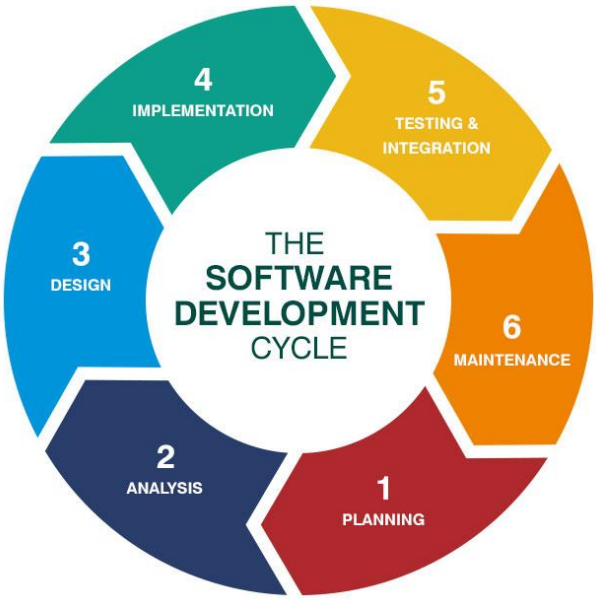


Fig. 2.1: Software Development Life Cycle (SDLC)

2.3 SRS (Software Requirements Specifications)

2.3.1 Introduction

Our project Clinic Management System includes registration of patients, storing their disease details into the system. The software has the facility to give a unique id for every patient and stores the details of every patient. The Clinic Management System can be used by entering respective username and password. It is accessible either by an administrator or receptionist. Only the respective person can add data in the database. The data can be retrieved easily. The interface is very user-friendly. The data is well protected and data processing is very fast, accurate and relevant.

Purpose

A Clinic Management System is a software designed to manage all the areas of a clinic such as medical, administrative and the corresponding processing of services.

Project Scope

Daily functions like patient registration, managing admission and overall management of various services can be easily performed with higher accuracy after the installation of clinic software. The modules of clinic management software are user-friendly and easy to access.

Intended audience

The intended audience of this document would be the client and specific employees like Manager and Receptionist, consultants and System Operators of the specific clinic, and project team, supervisor with the objective to refer and analyze the information. The SRS document can be used in any case regarding the requirements of the project and the solutions that have been taken. The document would finally provide a clear idea about the system that is being built.

Scope

Our software Clinic Management System provides the user interface requirements such as;

- ➔ Booking an appointment for patients
- ➔ After having the treatment saving the prescriptions
- ➔ Having the health histories
- ➔ Giving the feedbacks
- ➔ Keeping the records of medicine suppliers Medicine records and the patient visits

References

- ➔ W3School.com
- ➔ Java script from beginner to professional, author-Laurence, Maaike, Rob Percival.
- ➔ Bootstrap

2.3.2 Overall Description

A clinic management system is software designed to manage all the areas of a clinic such as medical, administrative and the corresponding processing of services. CMS is an abbreviation of Clinic Management System. The Clinic Management System (CMS) is integrated software that handles different directions of clinic workflows. It manages the smooth healthcare performance along with administrative and medical control. That is a cornerstone for the successful operation of the healthcare facility.

User interface

The User Interface for the user software shall be compatible with any browser such as Google Chrome, Mozilla Firefox, and Microsoft Edge by which user can access to the system. The user interface is implemented using tools and software packages associated with HTML, CSS, PYTHON and DJANGO.

System interface

Since the application must run over Internet, all hardware's and system software shall be connected to Internet to get the S/W interface on the system

Constraints

- ➔ Standard development tools, web based tools.
- ➔ There is a memory requirement for storing the data.
- ➔ The computer must be equipped with browsers such as Google Chrome.
- ➔ Response time should not be more than 30 minutes.

Assumptions and Dependencies

- ➔ It is assumed that you can use this website or app offline but you can't. It is dependable on the proper internet & proper OS.
- ➔ The server for the project must be running to access the software.
- ➔ The code should be free with compilation errors/syntax errors.
- ➔ The product must have an interface which is simple enough to understand.

User characteristics

The system will be used in the clinic. The administrators, front-desk staff will be the main users, given the condition that not all the users are computer - literate, some users may have to be trained on using the system.

Operating environment: this system can be used in windows, macos, android and any smart device.

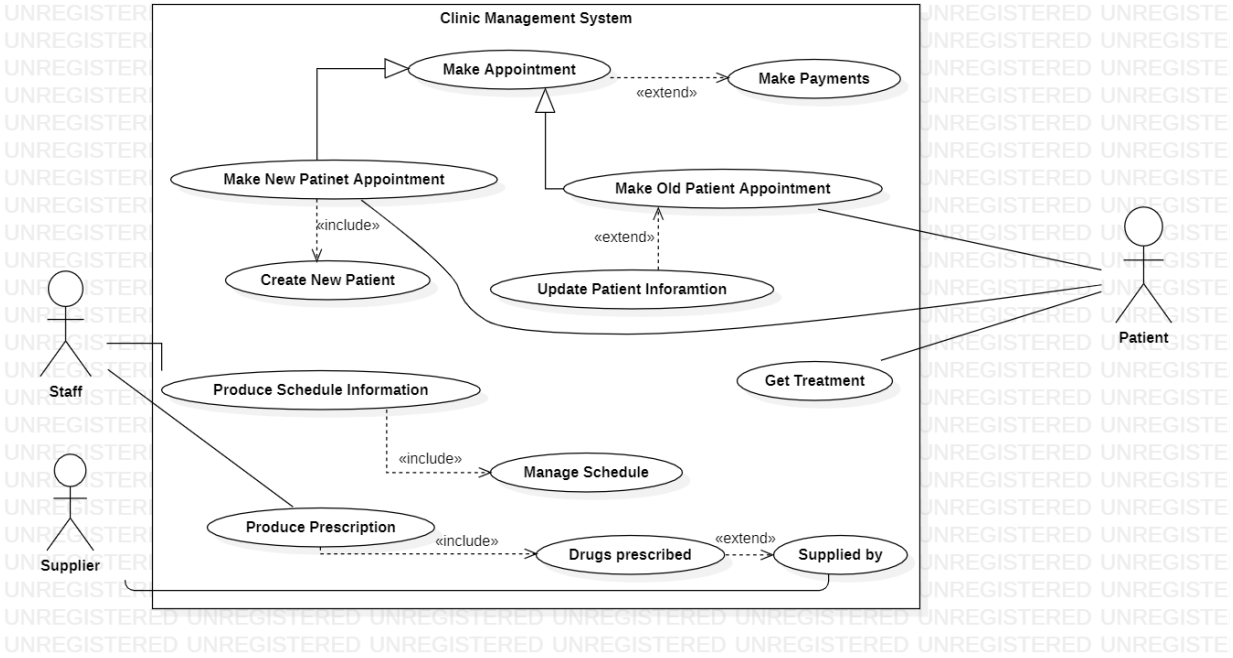
Design & Constraints: python, django framework, HTML, CSS, JS

2.3.3 System Features & Requirements

Functional requirements

- ➔ Firstly the admin or the user we get to the login page and only the admin has the right to update or edit the data.
- ➔ The doctors can view the appointments of their respective patients; they can also go through old prescriptions and old records.
- ➔ The patients have the access to see their old health records, appointment booking ‑facility, saving the prescriptions, medicine records and feedback facility.

Use case Diagram



Usability

- ➔ **Graphical User Interface:**
  - ◆ The system shall provide a uniform look and feel between all the web pages.
  - ◆ The system shall provide a digital image and use of toolbars.
- ➔ **Accessibility:**
  - ◆ Every user can access.
- ➔ **Reliability & availability:**
  - ◆ Every user can access.
- ➔ **Back end Internal Computers:**
  - ◆ The system shall provide storage of all databases on redundant computers with automatic switchover
  - ◆ Django is used to store data.
- ➔ **Internet service provider:**
  - ◆ The system shall provide a contractual agreement with an internet service provider, who can provide 99.99% availability through their network facilities onto the internet.
  - ◆ The system shall provide a contractual agreement with an internet service provider for T3 access with 99.99% availability.

Performance

- ➔ Our software provides high level security.
- ➔ End to end encryption is provided.
- ➔ We give all services fast on any browser.

2.3.4 S/W Quality Attributes

- ➔ **Reliability:** good validations of user inputs will be done to avoid incorrect storage of records.
- ➔ **Maintainability:** during the maintenance stage, SRS documents can be referred for any validations.
- ➔ **Portability:** this system can be installed in any personal computer considering the PC has internet connection and any web browser.
- ➔ **Flexibility:** the system keeps on updating the data according to the transactions that take place.
- ➔ **Timeless:** the system carries out all the operations with consumption of very less time.
- ➔ **Security:** security of the system is maintained by giving access to only authenticated users with their unique id and password.

Security requirements

This system is provided with authentication without which no unauthorized user can pass. So only the known users are allowed to use the application. If the legitimate users share the authentication information then the system is open to outsiders.

Performance

Cos(cost estimation models) manages facilities required by the casual users quickly and easily. It offers to take appointments faster through online. It takes appointment details from the patients and sends the appointment date and timings to the particular patient.

Chapter 3: Data Modeling

3.1 Work Products

All the Work Products specified below must be designed & maintained for further progress of the Software development,

Fig. 3.1: Unified Modeling Language

➔ **Model**

- A model is a simplification of reality.
- A model provides the blueprints of a system.
- A model may be structural, emphasizing the organization of the system, or it may be behavioral, emphasizing the dynamics of the system.
- We build models so that we can better understand the system we are developing.
- We build models of complex systems because we cannot comprehend such a system in its entirety**.**
- **Through modeling, we achieve four aims.**
  - Models help us to visualize a system as it is or as we want it to be.
  - Models permit us to specify the structure or behavior of a system.
  - Models give us a template that guides us in constructing a system.
  - Models document the decisions we have made

➔ **Principles of Modeling**

- The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped
- Every model may be expressed at different levels of precision
- The best models are connected to reality
- No single model is sufficient. Every nontrivial system is best approached through a small set of nearly independent models

UML is a graphical notation used to visualize, specify, construct and document the artifact of software intensive. UML is appropriate for modeling systems ranging from Enterprise Information Systems to Distributed Web-based Application and even to Hard Real-time Embedded systems. UML effectively starts with forming a conceptual modeling of the language. There are two types of UML diagrams, they are;

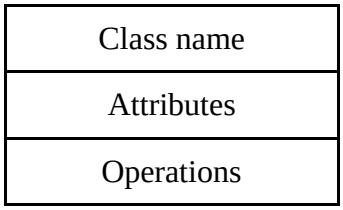
1. Static Diagrams
    1.1. Use case diagrams
    1.2. Class diagrams
    1.3. Object diagrams
    1.4. Component diagrams
    1.5. Deployment diagrams
2. Dynamic diagrams
    2.1. Interaction diagrams
        2.1.1. Sequence diagrams
        2.1.2. Collaboration diagrams
    2.2. State machine diagrams
    2.3. Activity diagram

Applications of UML: UML is intended primarily for software intensive systems. It has been used effectively for such domains as,

- ➔ Enterprise Information Systems
- ➔ Banking and Financial Services
- ➔ Telecommunications
- ➔ Transportation
- ➔ Defense and Aerospace
- ➔ Retail
- ➔ Medical Electronics
- ➔ Scientific
- ➔ Distributed Web-based Services

Basic building blocks of UML: The building blocks of UML can be categorized as,

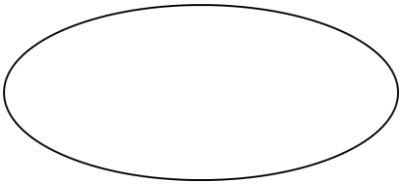
- ➔ **Things:** things are the most important building blocks of UML, things can be
    - ◆ Structural things: They define the static part of the model. They represent physical and conceptual elements. Following are the structural things
        - Class: it describes a set of objects that share the same attributes, operations, relationships and semantics.



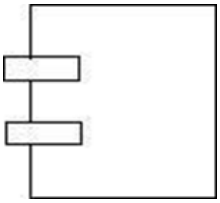
        - Object: it is a collection of operations that specifies a service of a class or component.

- Collaborator: it defines interaction between elements.

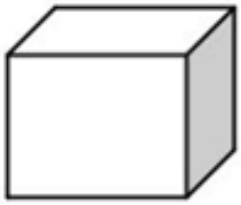
- Use case: they are used to identify different use case components of a particular software project. It is used to model the operation.



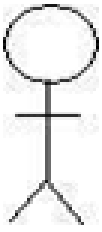
- Component: it is a physical and replaceable part that confirms and provides realization of a set of interfaces.



- Node: a physical resource that exists in runtime and represents a computational resource.



- Actor: the outside entity that communicates with a system. Typically a person playing a role on an external device.

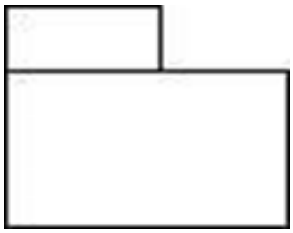


◆ **Behavioral Things:** they consist of dynamic parts of the UML model. The following are behavioral things;

- Interaction: It is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.

- State machine: It is useful when the state of an object in its life cycle. It defines the sequence of states an object goes through in response to events.

- **Grouping Things:** they can be defined as a mechanism to group elements of the UML model together. There is only one grouping thing available which is called a Package. It is used for gathering structural and behavioral things.



- **Annotational things:** they can be defined as a mechanism to capture remarks, description and comments of UML model elements. There is only one annotational thing available i.e., Note. Note is used to render comments, constraints and so on of a UML element.

➔ **Relationships:** the relationship is another most important building block of UML. They show how elements are associated with each other and their association describes the functionality of the application. There are 5 types of relationships, they are;

- **Dependency:** it is a relationship between two things in which change in one element also affects another.
- **Generalization:** it can be defined as a relationship which connects a specialized element with a generalized element. It basically describes the inheritance relationship in the object. It is a hierarchy.
- **Realization:** it can be defined as a relationship in which two elements are connected. One element describes some responsibility which is not implemented and the other one implements them. This relationship exists in the case of interfaces.
- **Association:** it is a set of links that connects elements of an UML model.

3.2 UML Diagrams (StarUML)

1. Use Case Diagram:

Problem Definition: to draw a use case diagram for Clinic Management System.

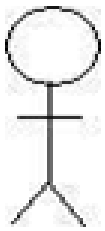
Problem Description: a use case diagram describes a set of sequences in which each sequence indicates the relation with outside things. A use case involves the interaction of actors and systems. There exist three types of relationships;

- ➔ Association
- ➔ Dependency
- ➔ Generalization

Use case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

Actor: an actor represents system users. They help delimit the system requirements and give a clearer picture of what the system should do. An actor is someone or something that;

- ➔ Interacts with or uses the system.
- ➔ Provides input to and receives information from the system.
- ➔ Is external to the system and has no control over the use cases.



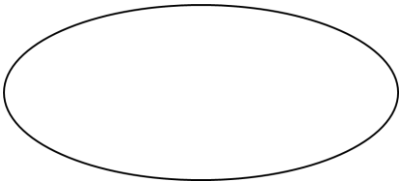
Staff member (actor)

Actors are discovered by examining:

- ➔ Who directly uses the system?
- ➔ Who is responsible for maintaining the system?
- ➔ External hardware used by the system.
- ➔ Other systems that need to interact with the system.

Use case: a use case can be described as a specific way of using the system from users' (actors) perspective. Use case can be characterized as;

- ➔ A pattern of behavior the system exhibits.
- ➔ A sequence of related transactions performed by an actor and the system.
- ➔ Delivering something of value to the actor.



Make appointment

Use cases provide a means to,

- ➔ Capture system requirements.
- ➔ Communicate with end users and domain experts.
- ➔ Test the system.

Every graphical representation has a textual description. The description of each use case is written in a use case specification. Use Case specification has;

- ➔ **Precondition** – which states how and when the use case starts.
- ➔ **Main Flow** – which lists the set of actions performed by the use case.
- ➔ **Alternate Flow** – which lists the exceptions that are possible during executing the use case.
- ➔ **Post condition** – which shows the result after the use case completes successfully.

Use Case Specification For Clinic Use Case

- ➔ Precondition: the user must have a username and password to login to the software.
- ➔ Main flow:
  - ◆ Login to the software with their specific username and password.
  - ◆ Add new patients in the database, or schedule appointments with the doctors, or check the previous health histories.
  - ◆ After treatment, the doctor will prescribe the drugs which are added in the prescription module of the software and the bill gets generated.
- ➔ Alternate flow:
  - ◆ Login to the software with their specific username and password.
  - ◆ Update the information of existing patients.
  - ◆ Checking the supplier details of various drugs and managing the staff details as well as the patient details.
- ➔ Post condition:
  - ◆ The patient's details are added/updated and the appointment details are shared with the staff members.
- ➔ Pseudo code:
  - ◆ Right click on the model
  - ◆ Select add diagram from the drop down menu, click on Use Case Diagram

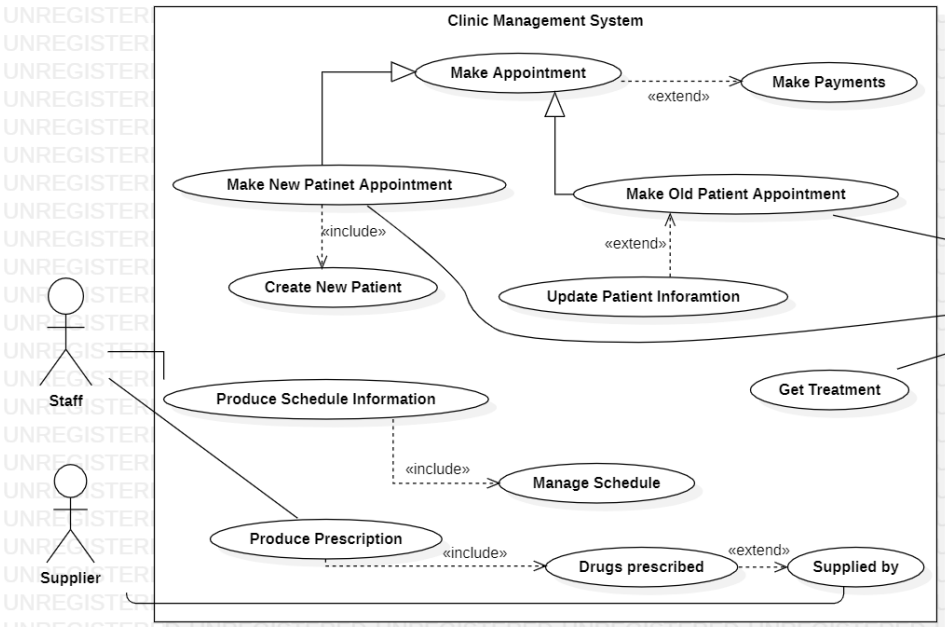


Fig. 3.2.1: Use Case diagram for Clinic Management System

2. Activity Diagram:

Problem Definition: to draw an activity diagram for the Clinic Management System.

Problem Description: an activity diagram is a special case of state diagram. An activity diagram is like a flow Machine showing the flow of control from one activity to another. An activity diagram is used to model dynamic aspects of the system. Activity diagram contains;

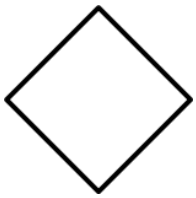
- ➔ Activity states and action states
- ➔ Transition
- ➔ Object

Action state: these are atomic, executable computation which represents the execution of an action.

Activity State: they can be decomposed. That is, their activity is represented by other activity

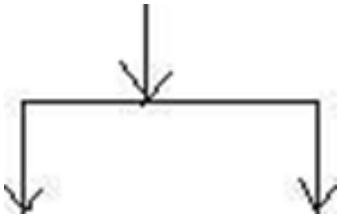
Diagrams;

- ➔ Branching: in branching, we have one incoming transition and two or more outgoing transitions.

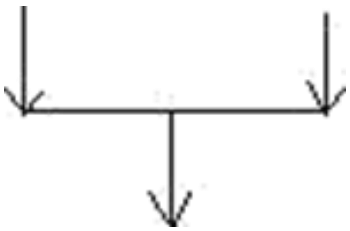


Decision

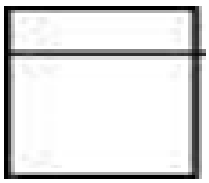
- ➔ Forking: it is a process of splitting a single flow of control into multiple flow of controls. Generally a fork has a single incoming flow of control but multi outgoing flow of control.



- ➔ Joining: it is exactly opposite of forking. It generally has multiple incoming flows of control but single outgoing flow of control.



➔ Swim-lanes: They represent the columns in the activity diagram to group the related activities. These are represented in the form of partitioned region.Swim-lanes are helpful when modeling a business workflow because they can represent organizational units or roles within a business model. Swim-lanes are very similar to an object because they provide a way to tell who is performing a certain role.



➔ Pseudo code:

- Right click on the model
- Select add diagram from the dropdown menu, click on add Activity Diagram

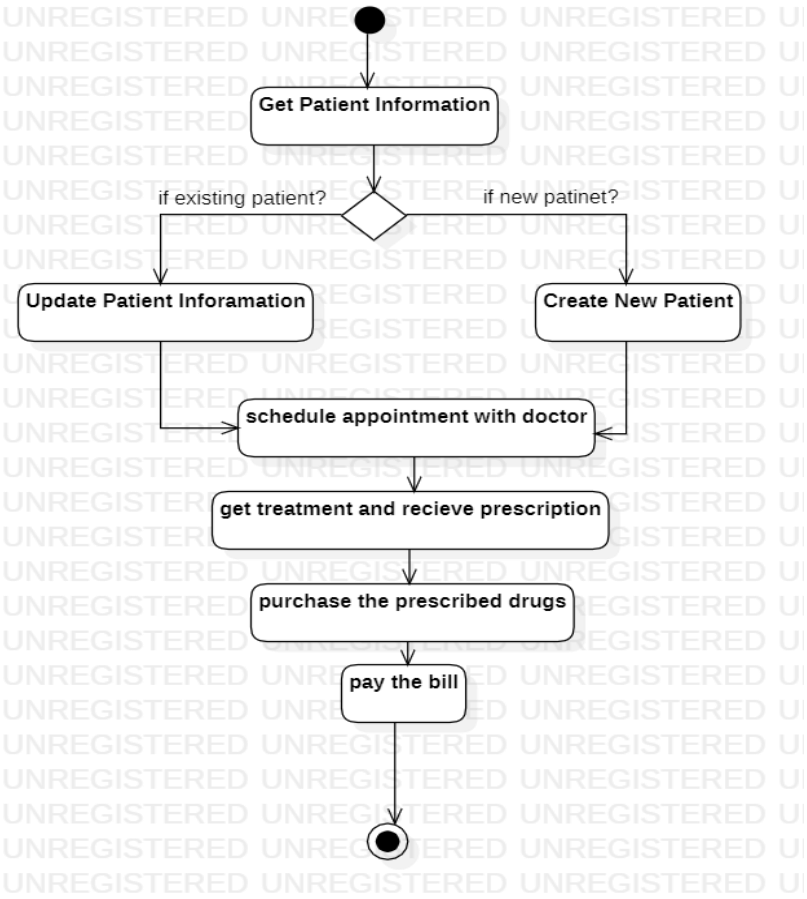


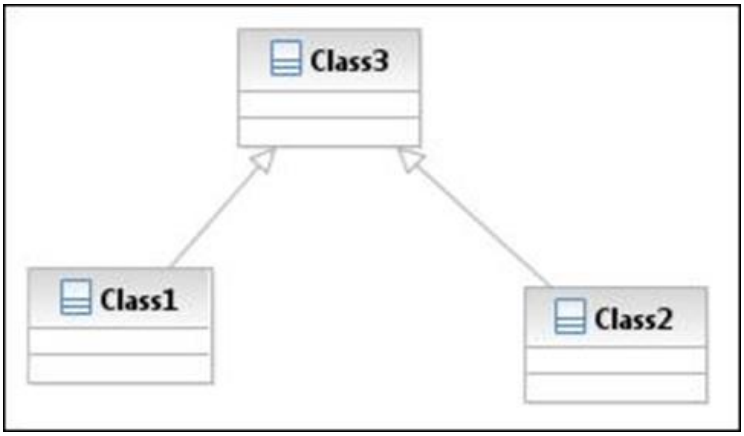
Fig. 3.2.2: Activity Diagram for Clinic Management System

3. Class Diagram:

Problem Definition: to draw a class diagram for Clinic Management Clinic.
Problem Description: class diagram contains icons representing classes, interfaces and their relationships.
Class: a class is a set of objects that share a common structure and common behavior (the same attributes, operations and semantics). A class is an abstraction of real-world items. When these items exist in the real-world, they are instances of the class and are referred to as objects. A class icon is a 3-part box.

Class name
Attributes
Operations

Generalization relationship for classes: it shows that sub classes share the structure or behavior defined in one or more super classes. Use a generalized relationship to show "is_a" relationship.



Here, Class3 is Super Class
And Class1, Class2 are the Subclasses

Dependency Relationship: the dependency is a relationship between two model elements in which change in one element will affect the other model element. Typically in class diagrams, a dependency relationship indicates that the operations of the client invoke operation of the supplier. **Cardinality Adornment:-** Cardinality specifies how many instances of one class may be associated with a single instance of another class. When you apply a cardinality adornment to a class, you are indicating the number of instances allowed for that class. A relationship, you are indicating the number of links allowed between one instance of a class and the instances of another class.

Valid Values	Value Description
0..0	Zero
0..1	Zero or one
0..n	Zero or more
1..1	One
1..n	One or more
n	Unlimited number

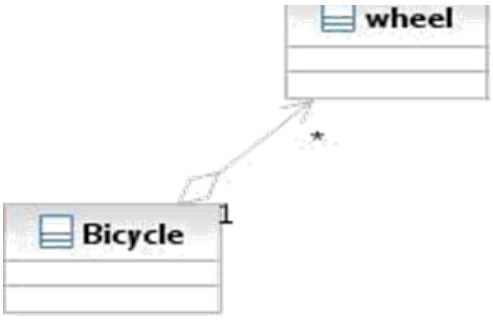
Table 3.1.1

Interface: an interface specifies the externally visible operations of a class and/or component, and has no implementation of its own. An interface specifies only a limited part of behavior of class or a component.

Association relationship: an association provides a pathway of communications. The communication can be between use cases, actors, classes or interfaces. If two classes are usually considered independently, the relationship is an association.

Unidirectional(left), Bidirectional(right)

Aggregate relationship: use the aggregate relationship to show a whole or part relationship between two classes.



The diamond end represents the class which is whole.

Pseudo Code:

- ➔ Right click on the model
- ➔ Select add diagram from the dropdown menu, click on add Class Diagram

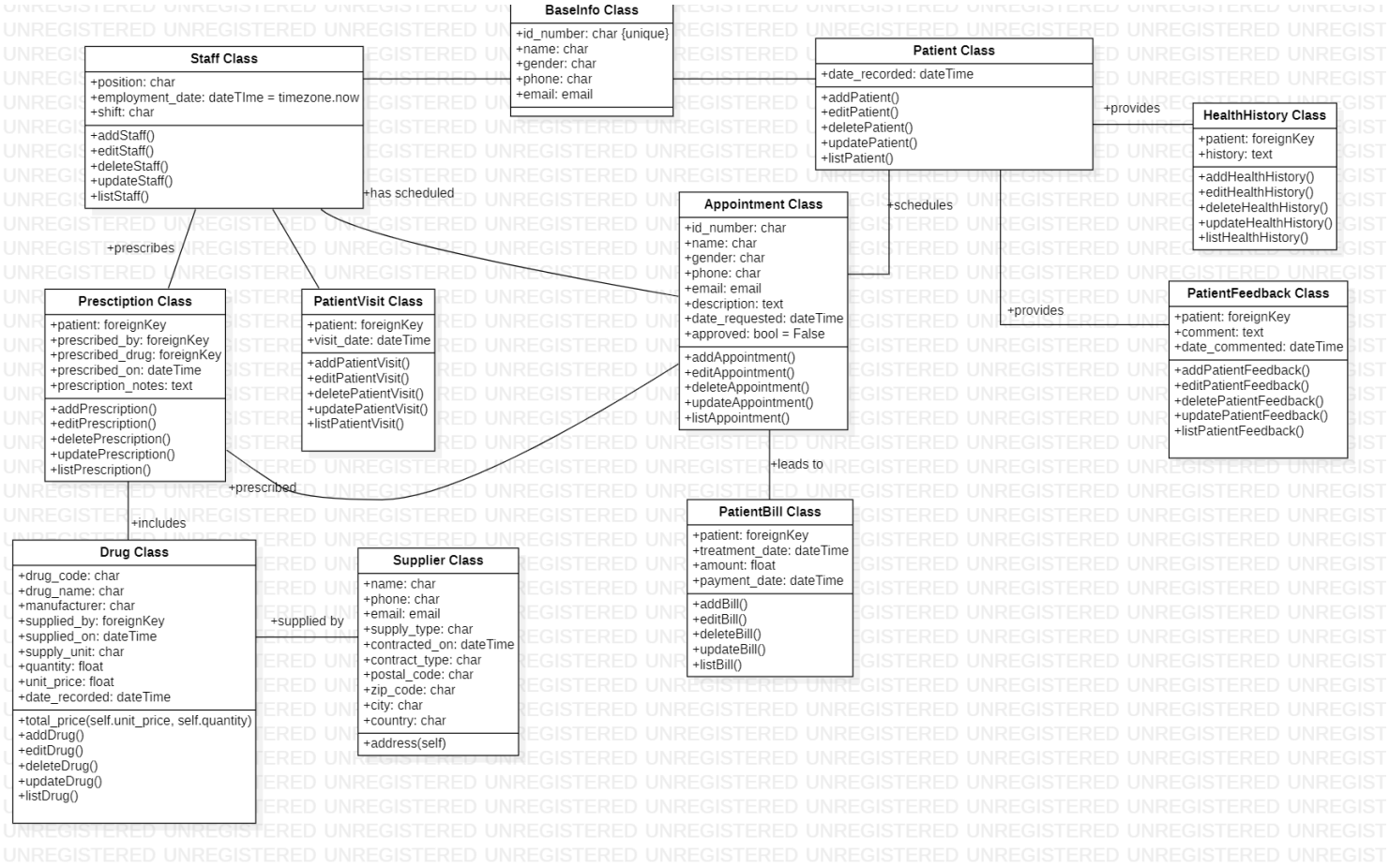


Fig. 3.2.3: Class diagram for Clinic Management System

4. Interaction Diagram:

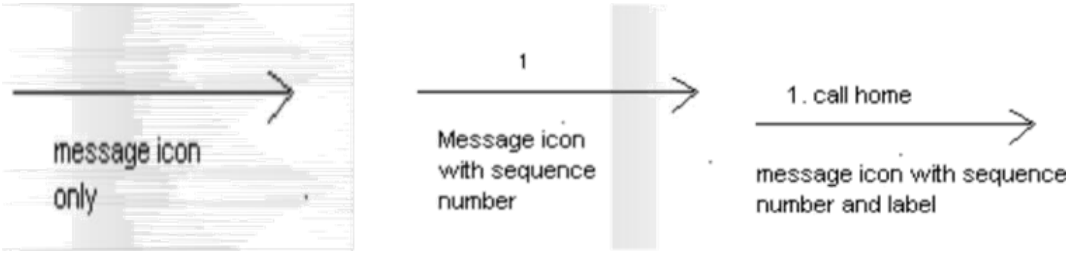
Problem Definition: to draw interaction diagrams for Clinic Management System.

Problem description: an interaction is an important sequence of interactions between objects. There are two types of interaction diagrams;

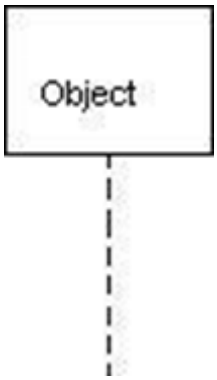
- **Sequence Diagrams:** a sequence diagram is a graphical view of a scenario that shows object interaction in a time based sequence, what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. A sequence diagram has two dimensions: vertical placement represents time and horizontal placement represents different objects.

  Link: objects interact through their links to other objects. A link is an instance of an association, analogous to an object being an instance of a class. A link should exist between two objects, including class utilities, only if there is a relationship between their corresponding classes.

Message icons: a message icon represents the communication between objects indicating that an action will follow. The message icon is a horizontal, solid arrow connecting two lifelines together.A message icon in a sequence diagram represents exactly one message.



Lifeline: each object appearing on the sequence diagram contains a dashed vertical line, called lifeline, which represents the location of an object at a particular point in time. The lifeline also as a Serves place for messages to start and stop and a place for the focus of control to reside.

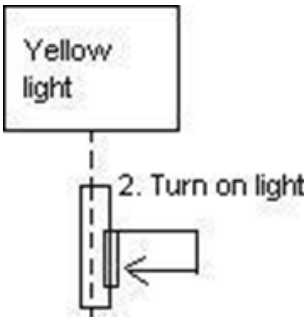


Lifeline

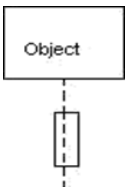
Message or event: a message is a communication carried between two objects that trigger an event. A message is represented in collaboration and sequence diagrams by a message icon which usually indicates its synchronization. Synchronization types that are supported.

- ➔ Synchronous Call
- ➔ Asynchronous Call
- ➔ Asynchronous Signal
- ➔ Create
- ➔ Delete
- ➔ Reply

Message to self: It is a tool that sends a message from one object back to the same object. The sender of the message is the same as the receiver.



Tools:

1.	Object	
2.	Object message	
3.	Message to self	
4.	Return message	
5.	Destruction marker	

Pseudo Code:

- ➔ Right click on the model
- ➔ Select add diagram from the dropdown menu, click on add Diagram

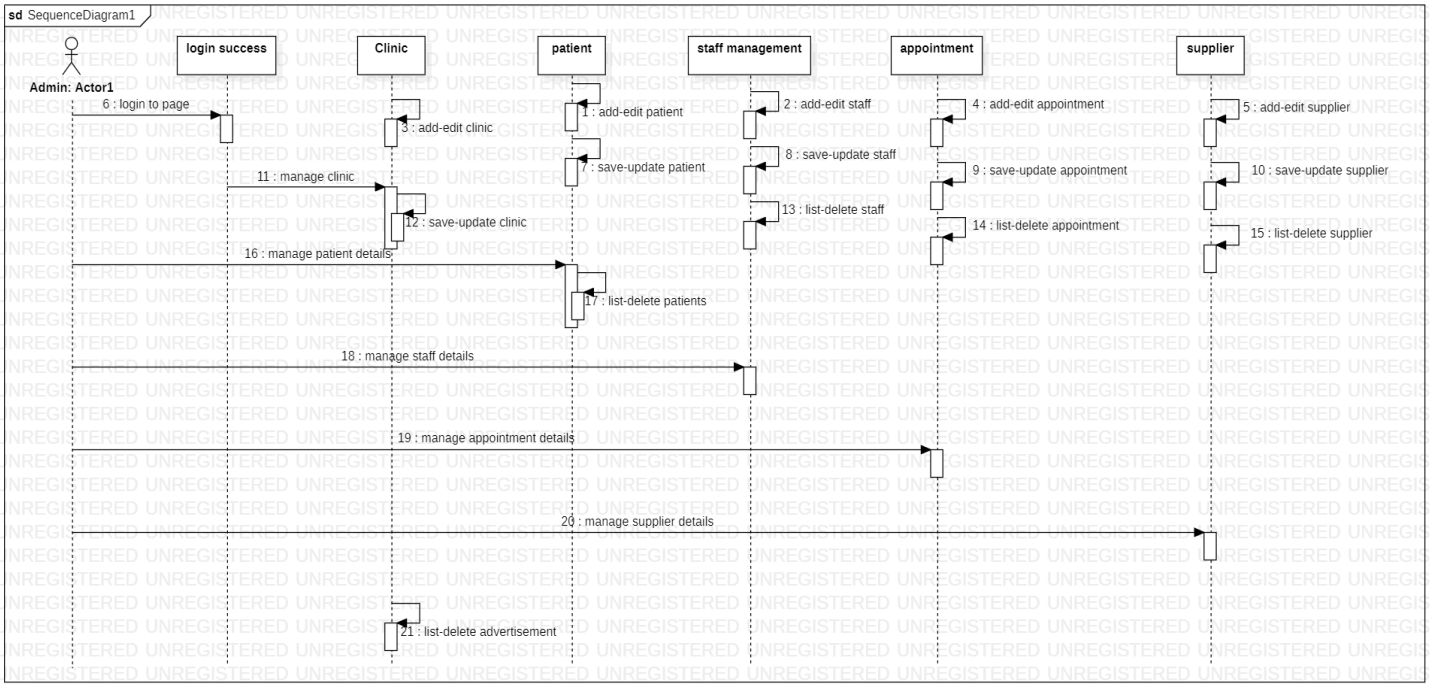


Fig. 3.2.4: Sequence diagram for Clinic Management System

➔ **Collaboration diagrams:** a collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. Collaboration diagrams show objects, their links, and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model. Collaboration diagrams contain icons representing objects. Sequence and collaboration diagrams are semantically equivalent as both show the interaction among objects. From one diagram we can generate another diagram. To generate a collaboration diagram from sequence diagram, right click on sequence diagram, select - Add Diagram - communication diagram . Similarly, a sequence diagram can be generated from a collaboration diagram.

Pseudo Code:

➔ Right click on the model
➔ Select add diagram from the dropdown menu, click on add Diagram

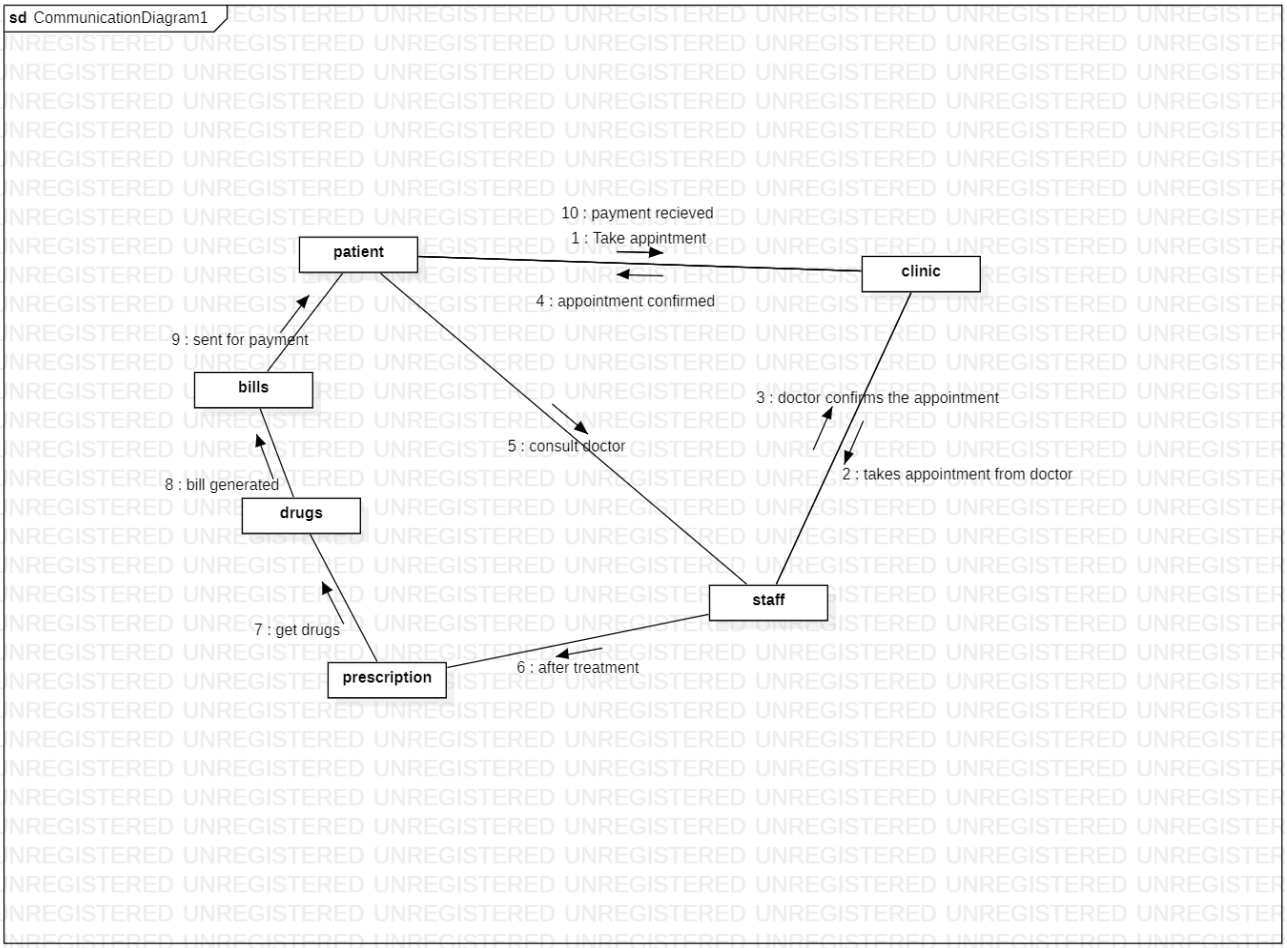


Fig. 3.2.5: Communication Diagram for Clinic Management System

5. State Machine Diagram:

Problem Definition: to draw the state machine diagram for the clinic management system.

Problem Description: state Machine diagrams model the dynamic behavior of individual classes or any other kind of object. They show the sequence of states that an object goes through the events that cause a transition from one state to another and the actions that result from a state change. A state Machine diagram is typically used to model the discrete stages of an object's lifetime. A state Machine diagram typically contains one start state and multiple end states.

State: a state represents a condition or situation during the life of an object during which it satisfies some condition or waits for an event. Each state represents a cumulative history of its behavior. States can be shared between state machines. Transitions cannot be shared.

Naming: the name of the state must be unique to its enclosing class, within the state.

Actions: actions on states can occur one of four times on entry, on exit, on event.

Start state: a start state (also called an "initial state") explicitly shows the beginning of the execution of the state machine on the state Machine diagram or beginning of the workflow on an activity diagram. Normally, one outgoing transition can be placed from the start state. However, multiple transitions may be placed in the start state, if at least one of them is labeled with a condition. No incoming transitions are allowed. The start state icon is a small, filled circle that may contain the name (Begin process).

Begin process

End state: an end state represents a final or terminal state on an activity or state Machine diagram. transitions can only occur into an end state. The end state icon is a filled circle inside a slightly larger unfilled circle that may contain the name (End process).

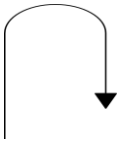
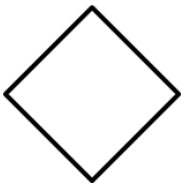
End process

State transition: a state transition indicates that an action in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied. A state transition is a relationship between two states, two activities or between an activity or a state. The icon for a state transition is a line with an arrow head pointing toward the destination state or activity.

We can show one or more state transitions from a state as long as each transition is unique. Transitions originating from a state cannot have the same event, unless there are conditions on the event.

Naming: We should label each state transition with the name of at least one event that causes the state transition. We do not have to use unique labels for the state transitions because the same event can cause a transition to many different states or activities.

Tools:

1.	State	
2.	Activity	
3.	Start state	
4.	End state	
5.	State transition	
6.	Transition to self	
7.	Horizontal synchronization	
8.	Vertical synchronization	
9.	Decision	

Pseudo code:

- ➔ Right click on the model
- ➔ Select add diagram from the dropdown menu, click on add State Chart Diagram

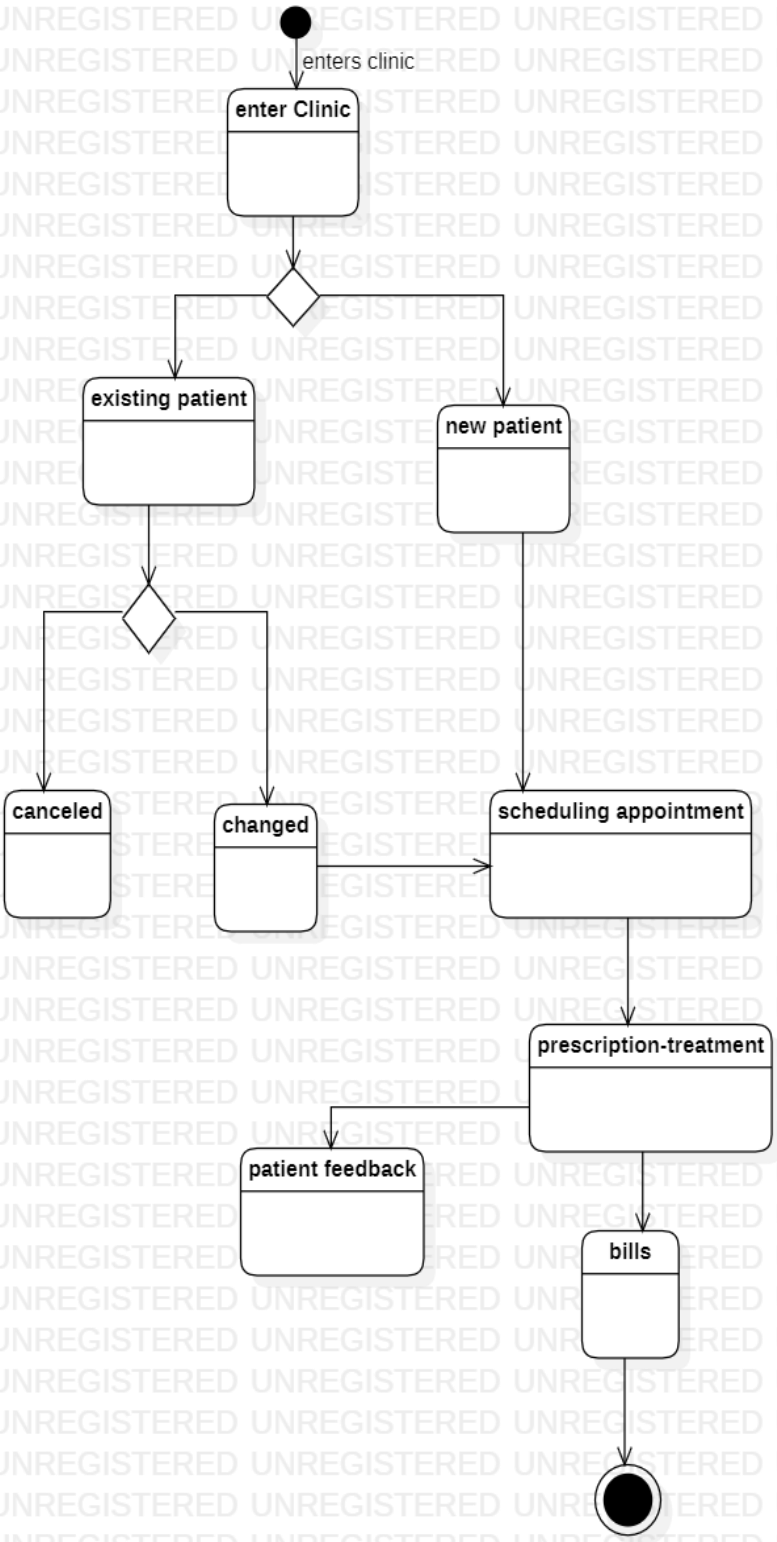


Fig. 3.2.6: State Chart diagram for Clinic Management System

6. Deployment Diagram:

Problem Definition: to draw a deployment diagram for the Clinic Management System.
Problem Description: a deployment diagram shows processors, devices and connections. Each model contains a single deployment diagram which shows the connections between nodes.
Node: a node is a hardware component capable of executing programs. Each node must have a name, there are no constraints on the node name because nodes denote hardware rather than software entities.
Pseudo Code:

- ➔ Right click on the model
- ➔ Select add diagram from the dropdown menu, click on add Deployment Diagram

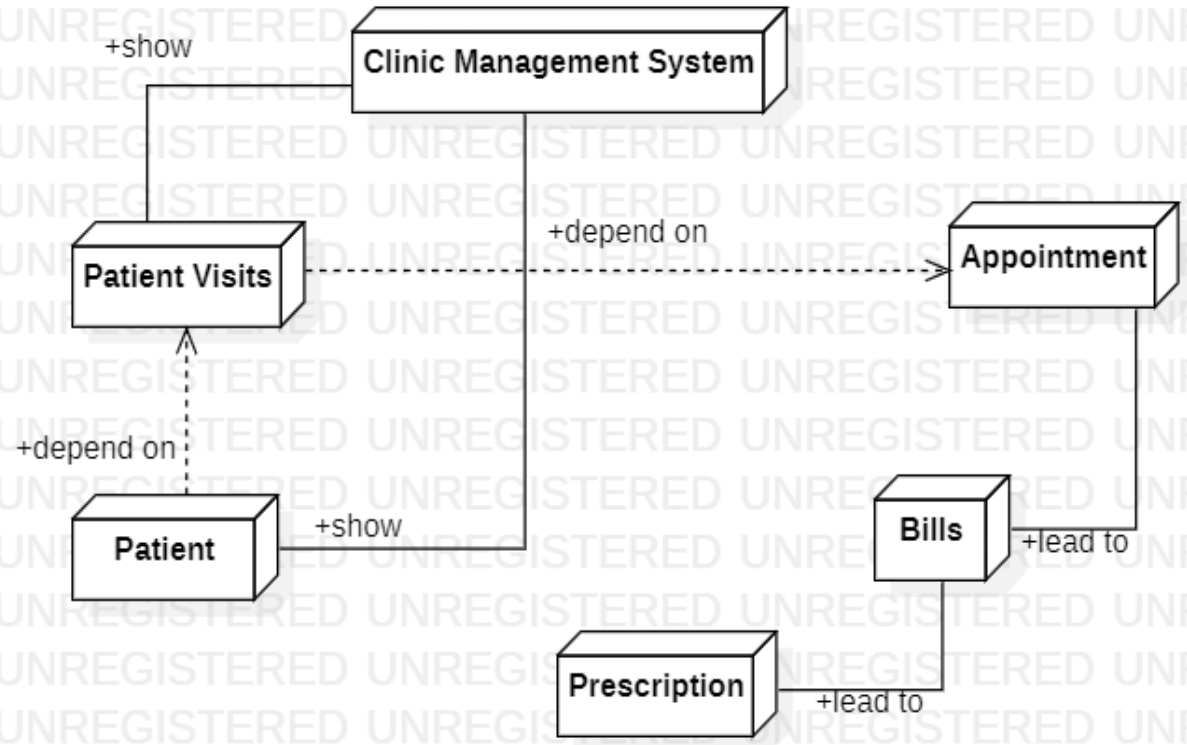
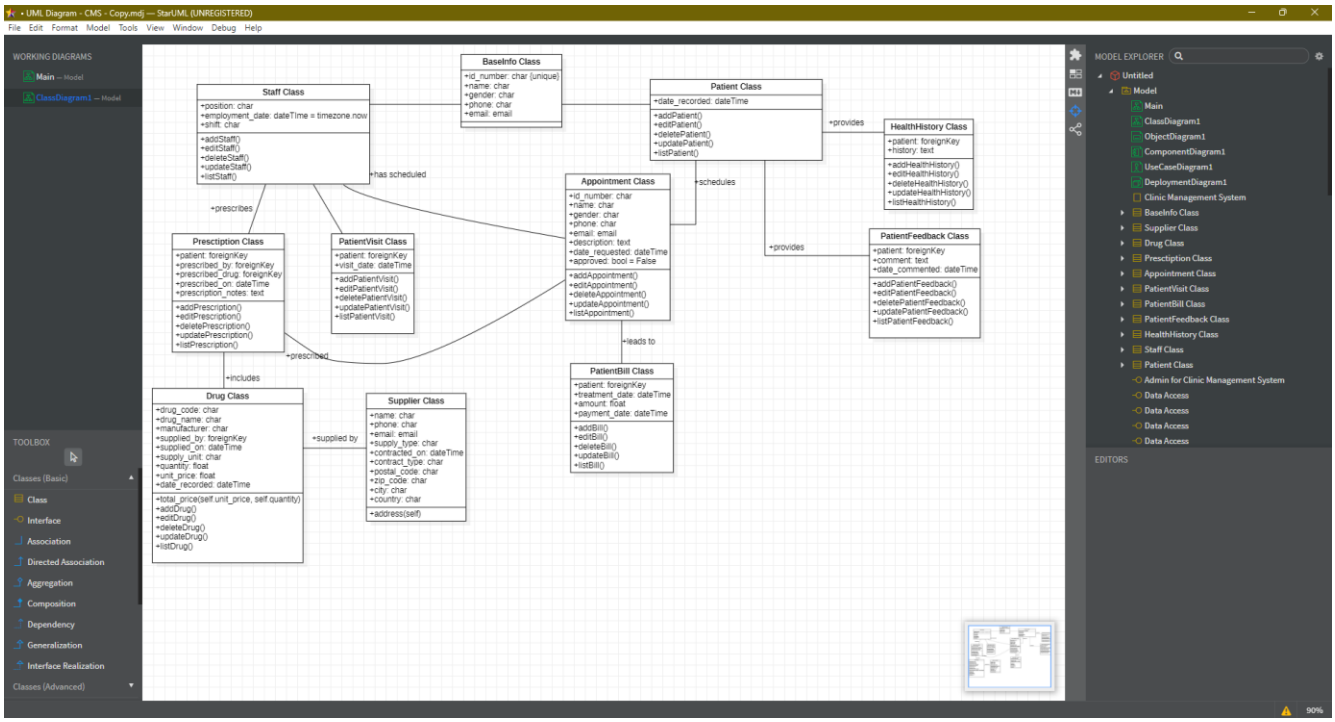


Fig. 3.2.7: Deployment diagram for Clinic Management System

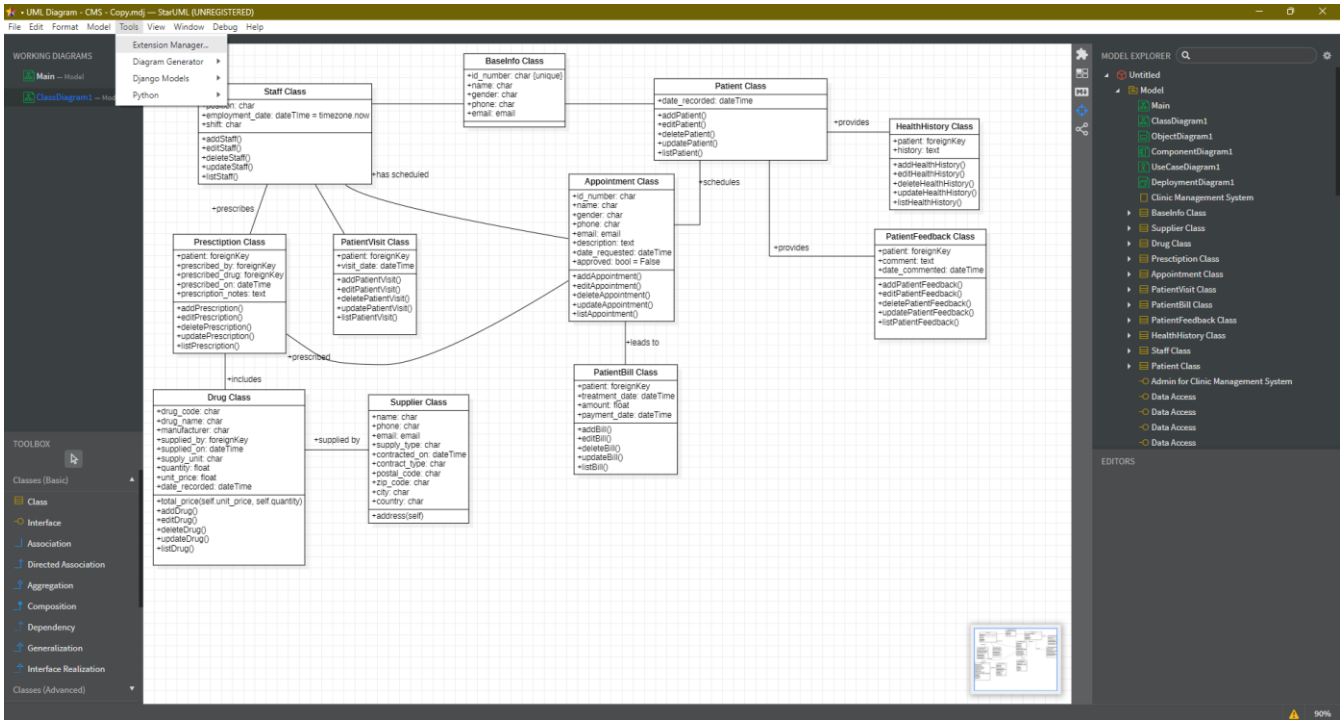
3.3 Forward Engineering

StarUML Forward Engineering Operation Steps:

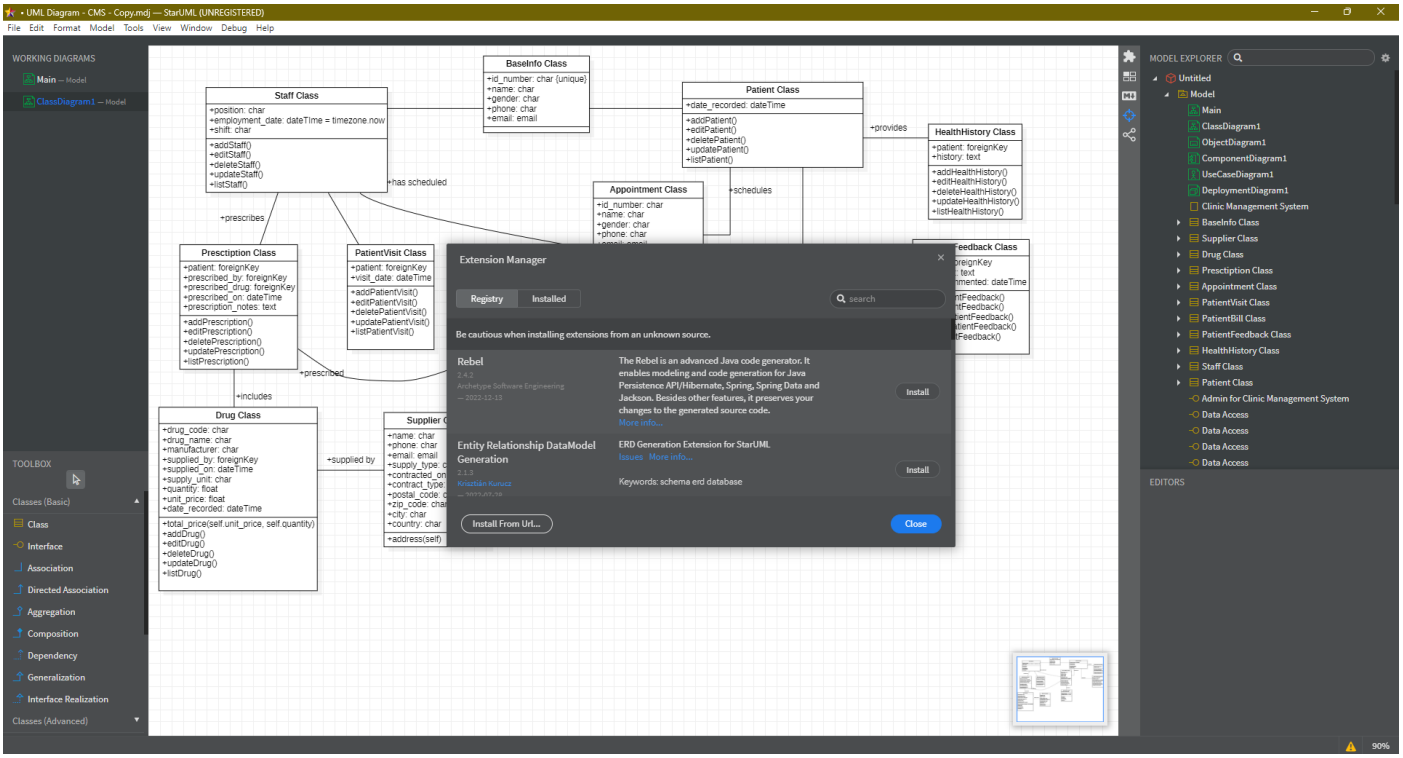
1) Create a new logical view through StarUML.
2) Build a class relationship diagram, add package, class relationship, structure, comments etc.



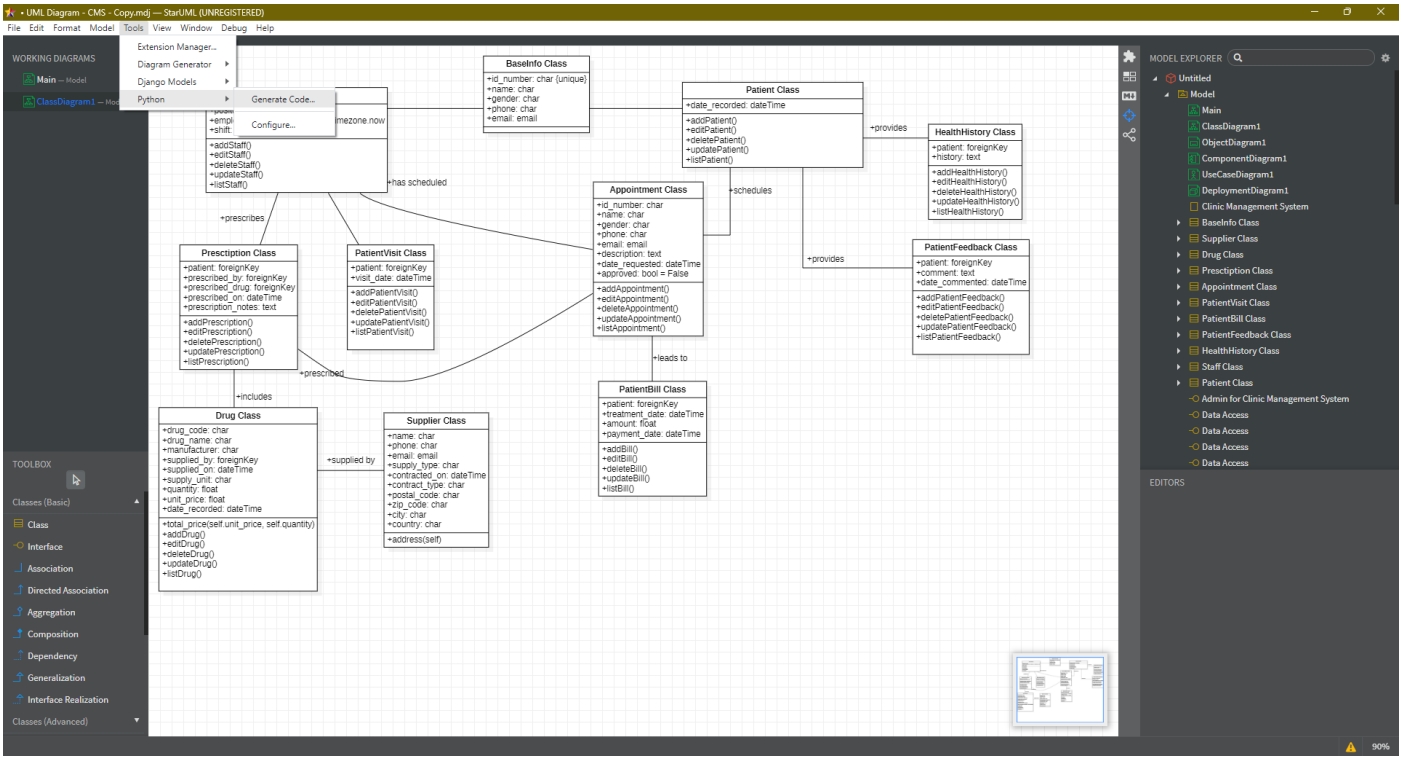
3) Click the tools drop down menu in the upper menu bar, then extension manager.



4) Find python and django, click install and reboot the application.

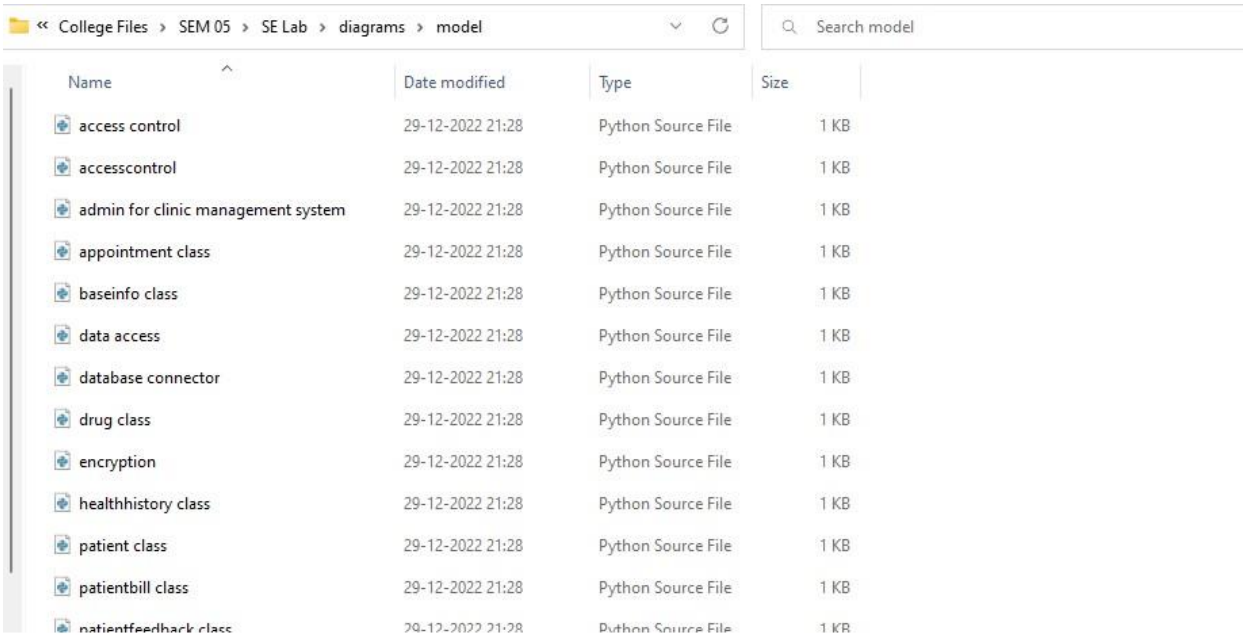


5) Click tools, python, generate code



6) Select the model you want to perform forward engineering, specify the directory and then click on OK.

7) The files will be generated in the directory specified.



3.4 Reverse engineering

StarUML Reverse Engineering Operation Steps:

1) Create a new logical view through StarUML
2) Click on the tools drop down menu, python, reverse code in the upper menu bar and specify the location of the code files.
3) The class in the code appears in the tree structure on the left
4) Drag and drop the generated UML primitives to the workspace, as shown in the figure below,
5) Save the UML diagrams in .mdj files.

3.5 Data Dictionary

A data dictionary is a collection of descriptions of the data objects or items in a data model for the benefit of programmers and others who need to refer to them. A first step in analyzing a system of objects with which users interact is to identify each object and its relationship to other objects. This process is called data modeling and results in a picture of object relationships. After each data object or item is given a descriptive name, its relationship is described (or it becomes part of some structure that implicitly describes relationship), the type of data (such as text or image or binary value) is described, possible predefined values are listed, and a brief textual description is provided. This collection can be organized for reference into a book called a data dictionary.

When developing programs that use the data model, a data dictionary can be consulted to understand where a data item fits in the structure, what values it may contain, and basically what the data item means in real-world terms. For example, a clinic could model the data objects involved in their clinic. They could then provide a data dictionary for their programmer. The data dictionary would describe each of the data items in its data model for clinic management (for example, "Health Histories" and "Prescription").

3.5.1 Data Structures

DTD for Patient Registration

Field Name	Type	Constraints
Id Number	integer	primary_key
Name	varchar	max_length=200, unique
Gender	varchar	max_length=20
Phone	varchar	max_length=200
Email	varchar	max_length=200
date_recorded	datetime	-

DTD for Staff Registration

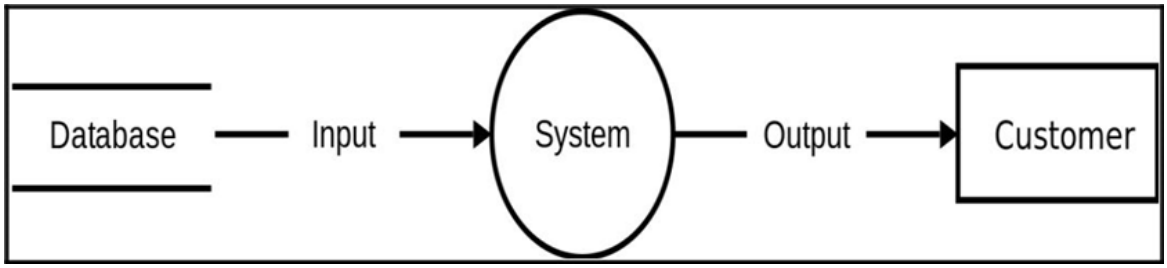
Field Name	Type	Constraints
Id	integer	primary_key
staff_id_number	varchar	max_length=200, unique
name	varchar	max_length=200
gender	varchar	max_length=20
phone	varchar	max_length=200
email	varchar	max_length=200
position	varchar	max_length=100
employment_date	datetime	-
shift	varchar	max_length=20

DTD for Appointment Registration

Field Name	Type	Constraints
id	integer	Primary key
appointment_id_number	varchar	max_length=200
name	varchar	max_length=200
gender	varchar	max_length=20
phone	varchar	max_length=200
email	varchar	max_length=200
description	text	-
date_requested	datetime	-
approved	bool	-

3.6 Data Flow Diagrams

A data-flow diagram (DFD) is a way of representing the flow of data of a process or a system (usually an information system) The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.



For each data flow, at least one of the endpoints (source and / or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes.

The data-flow diagram is part of the structured-analysis modeling tools. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan. DFD consists of processes, flows, warehouses, and terminators. There are several ways to view these DFD components.

Process: The process (function, transformation) is part of a system that transforms inputs to outputs. The symbol of a process is a circle, an oval, a rectangle or a rectangle with rounded corners (according to the type of notation). The process is named in one word, a short sentence, or a phrase that is clearly to express its essence.

Data Flow: Data flow (flow, dataflow) shows the transfer of information (sometimes also material) from one part of the system to another. The symbol of the flow is the arrow. The flow should have a name that determines what information (or what material) is being moved. Exceptions are flows where it is clear what information is transferred through the entities that are linked to these flows. Material shifts are modeled in systems that are not merely informative. Flow should only transmit one type of information (material). The arrow shows the flow direction (it can also be bi-directional if the information to/from the entity is logically dependent - e.g. question and answer). Flows link processes, warehouses and terminators.

Warehouse: The warehouse (datastore, data store, file, database) is used to store data for later use. The symbol of the store is two horizontal lines, the other way of view is shown in the DFD Notation. The name of the warehouse is a plural noun (e.g. orders) - it derives from the input and output streams of the warehouse. The warehouse does not have to be just a data file, for example, a folder with documents, a filing cabinet, and optical discs. Therefore, viewing the warehouse in DFD is independent of implementation. The flow from the warehouse usually represents the reading of the data stored in the warehouse, and the flow to the warehouse usually expresses data entry or updating (sometimes also deleting data). Warehouse is represented by two parallel lines between which the memory name is located (it can be modeled as a UML buffer node).

Terminator: The Terminator is an external entity that communicates with the system and stands outside of the system. It can be, for example, various organizations (eg a bank), groups of people (e.g. customers), authorities (e.g. a tax office) or a department (e.g. a human-resources department) of the same organization, which does not belong to the model system. The terminator may be another system with which the modeled system communicates.

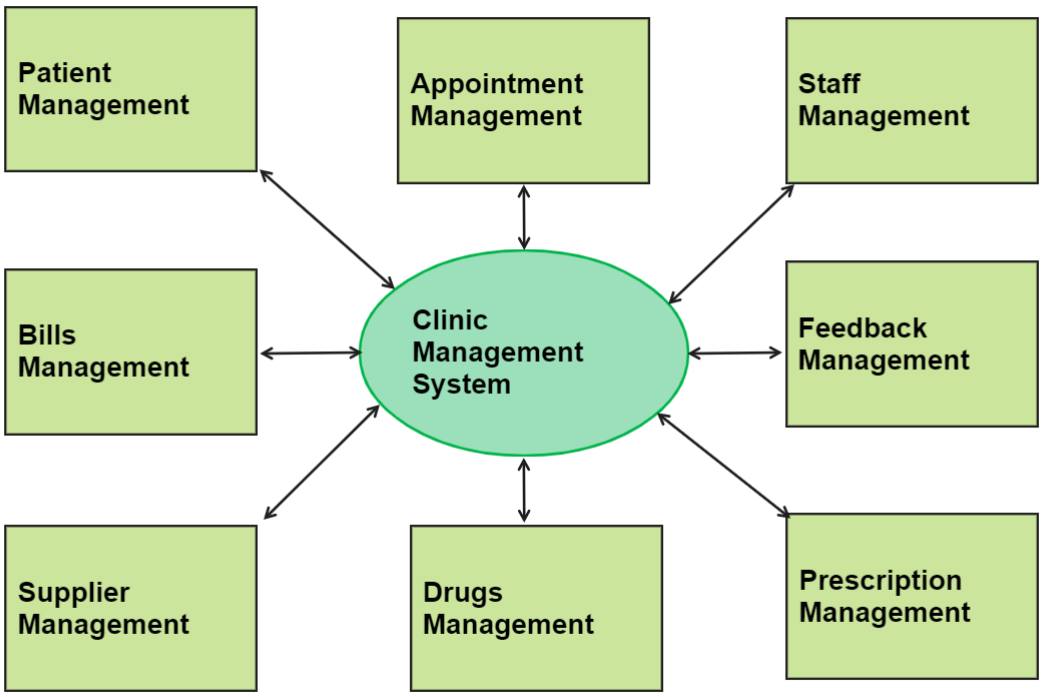


Fig. 3.6.1: DFD Level 01

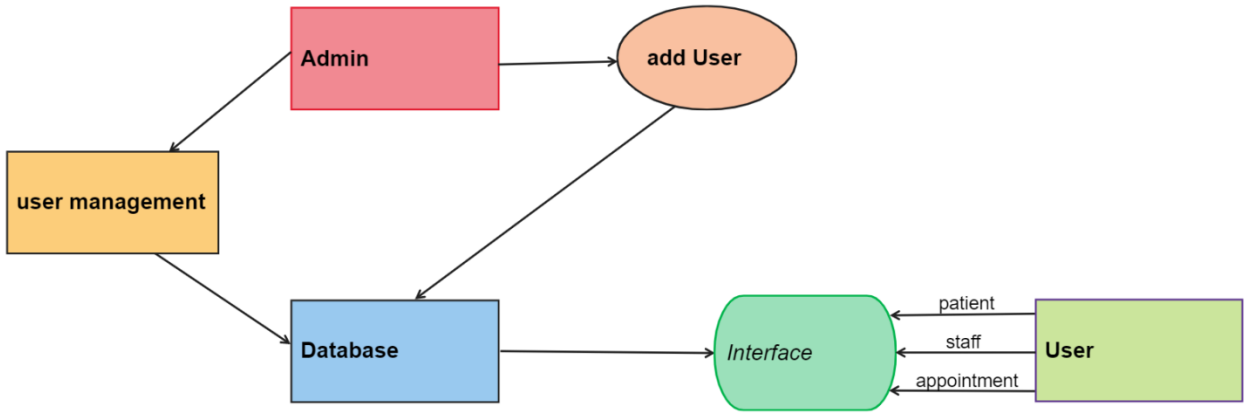


Fig. 3.6.2: DFD Level 01

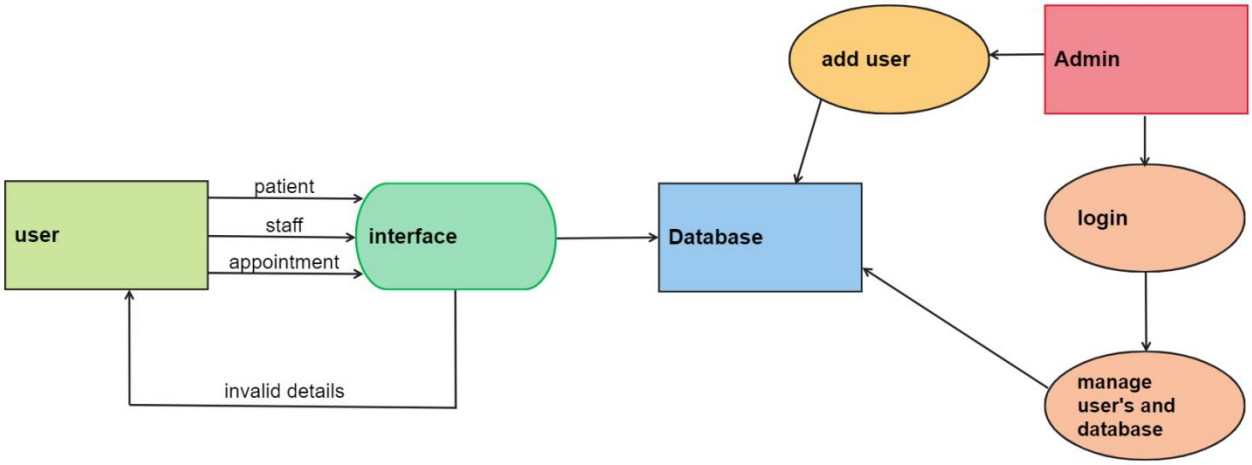


Fig. 3.6.3: DFD Level 02

Chapter 4: Software Development & Debugging

4.1 Sample Code

```
{% block content %}
    <div class="container-fluid" style="background-color: white;">
        <div class="row">
            <div class="col-xl-4 col-md-6 mt-5">
                <div class="card text-white mb-4" style="background-color: rgb(24, 236,
180);">
                    <div class="card-body">
                        <h4>Patients</h4>
                        </div>
                    <div class="card-footer d-flex align-items-center justify-content-
between">
                        <a class="small text-white btn btn-primary stretched-link"
href="{% url 'patients' %}">See Patients Details</a>
                        <div class="small text-white"><i class="fas fa-angle-
right"></i></div>
                    </div>
                </div>
            </div>
            <div class="col-xl-4 col-md-6 mt-5">
                <div class="card text-white mb-4" style="background-color: rgb(24, 236,
180);">
                    <div class="card-body">
                        <h4>Health Histories</h4>
                        </div>
                    <div class="card-footer d-flex align-items-center justify-content-
between">
                        <a class="small text-white btn btn-primary stretched-link"
href="{% url 'health_histories' %}">See Health Histories</a>
                        <div class="small text-white"><i class="fas fa-angle-
right"></i></div>
                    </div>
                </div>
            </div>
            <div class="col-xl-4 col-md-6 mt-5">
                <div class="card text-white mb-4" style="background-color: rgb(24, 236,
180);">
                    <div class="card-body">
                        <h4>Patients Visits</h4>
                        </div>
                    <div class="card-footer d-flex align-items-center justify-content-
between">
                        <a class="small text-white btn btn-primary stretched-link"
href="{% url 'visits' %}">See Patients Visits</a>
                        <div class="small text-white"><i class="fas fa-angle-
right"></i></div>
                    </div>
                </div>
```

```
            </div>
        </div>

      <div class="row">
            <div class="col-xl-4 col-md-6 mt-5">
                <div class="card text-white mb-4" style="background-color: rgb(24, 236,
180);">
                    <div class="card-body">
                        <h4>Prescriptions</h4>
                        </div>
                    <div class="card-footer d-flex align-items-center justify-content-
between">
                        <a class="small text-white btn btn-primary stretched-link"
href="{% url 'prescriptions' %}">See Prescriptions Details</a>
                        <div class="small text-white"><i class="fas fa-angle-
right"></i></div>
                    </div>
                </div>
            </div>
            <div class="col-xl-4 col-md-6 mt-5">
                <div class="card text-white mb-4" style="background-color: rgb(24, 236,
180);">
                    <div class="card-body">
                        <h4>Clinic Staff</h4>
                        </div>
                    <div class="card-footer d-flex align-items-center justify-content-
between">
                        <a class="small text-white btn btn-primary stretched-link"
href="{% url 'staff' %}">See Staff Details</a>
                        <div class="small text-white"><i class="fas fa-angle-
right"></i></div>
                    </div>
                </div>
            </div>
            <div class="col-xl-4 col-md-6 mt-5">
                <div class="card text-white mb-4" style="background-color: rgb(24, 236,
180);">
                    <div class="card-body">
                        <h4>Suppliers</h4>
                        </div>
                    <div class="card-footer d-flex align-items-center justify-content-
between">
                        <a class="small text-white btn btn-primary stretched-link"
href="{% url 'suppliers' %}">See Suppliers Details</a>
                        <div class="small text-white"><i class="fas fa-angle-
right"></i></div>
                    </div>
                </div>
            </div>
        </div>
```

```
        <div class="row">
            <div class="col-xl-4 col-md-6 mt-5">
                <div class="card text-white mb-4" style="background-color: rgb(24, 236,
180);">
                    <div class="card-body">
                        <h4>Feedbacks</h4>
                        </div>
                    <div class="card-footer d-flex align-items-center justify-content-
between">
                        <a class="small text-white btn btn-primary stretched-link"
href="{% url 'feedbacks' %}">See Feedbacks</a>
                        <div class="small text-white"><i class="fas fa-angle-
right"></i></div>
                    </div>
                </div>
            </div>
            <div class="col-xl-4 col-md-6 mt-5">
                <div class="card text-white mb-4" style="background-color: rgb(24, 236,
180);">
                    <div class="card-body">
                        <h4>Drug Records</h4>
                        </div>
                    <div class="card-footer d-flex align-items-center justify-content-
between">
                        <a class="small text-white btn btn-primary stretched-link"
href="{% url 'drugs' %}">See Drug Records</a>
                        <div class="small text-white"><i class="fas fa-angle-
right"></i></div>
                    </div>
                </div>
            </div>
            <div class="col-xl-4 col-md-6 mt-5">
                <div class="card text-white mb-4" style="background-color: rgb(24, 236,
180);">
                    <div class="card-body">
                        <h4>Appointments</h4>
                        </div>
                    <div class="card-footer d-flex align-items-center justify-content-
between">
                        <a class="small text-white btn btn-primary stretched-link"
href="{% url 'appointments' %}">See Appointments Details</a>
                        <div class="small text-white"><i class="fas fa-angle-
right"></i></div>
                    </div>
                </div>
            </div>
        </div>

    <footer class="footer mt-auto py-3" style="background-color: rgb(18, 237, 211);">
  <div class="container">
      <code>
```

```
          <h5 class="text-center text-white">Clinic Management System 2022</h5>
      </code>

  </div>
</footer>
    </div>

{% endblock content %}
```

4.2 Application Running

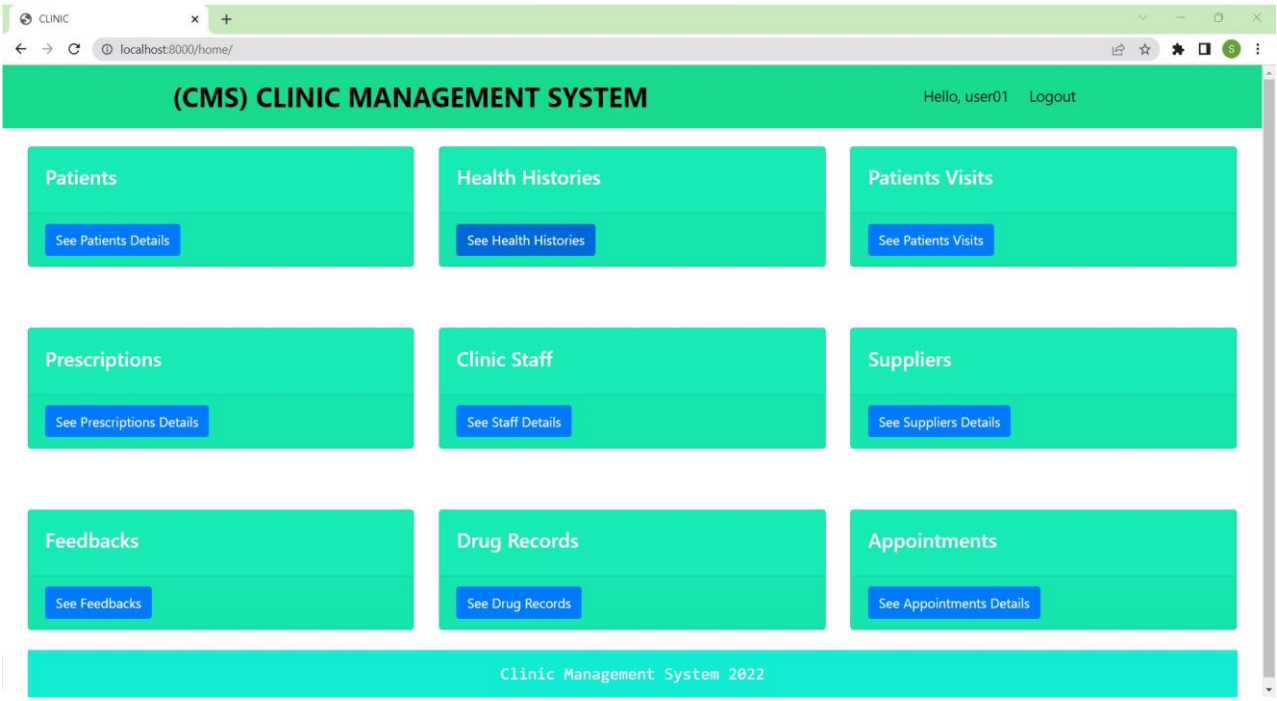
The below screen describes the development success of an email client system application from the android console which is being run in the emulator**.** Android Studio provides you with various ways to test web pages as you are developing them. You have the following choices:

- **Start Debugging and run** (ctrl+alt+F5). This option allows you to perform debugging and run the application in the emulator.

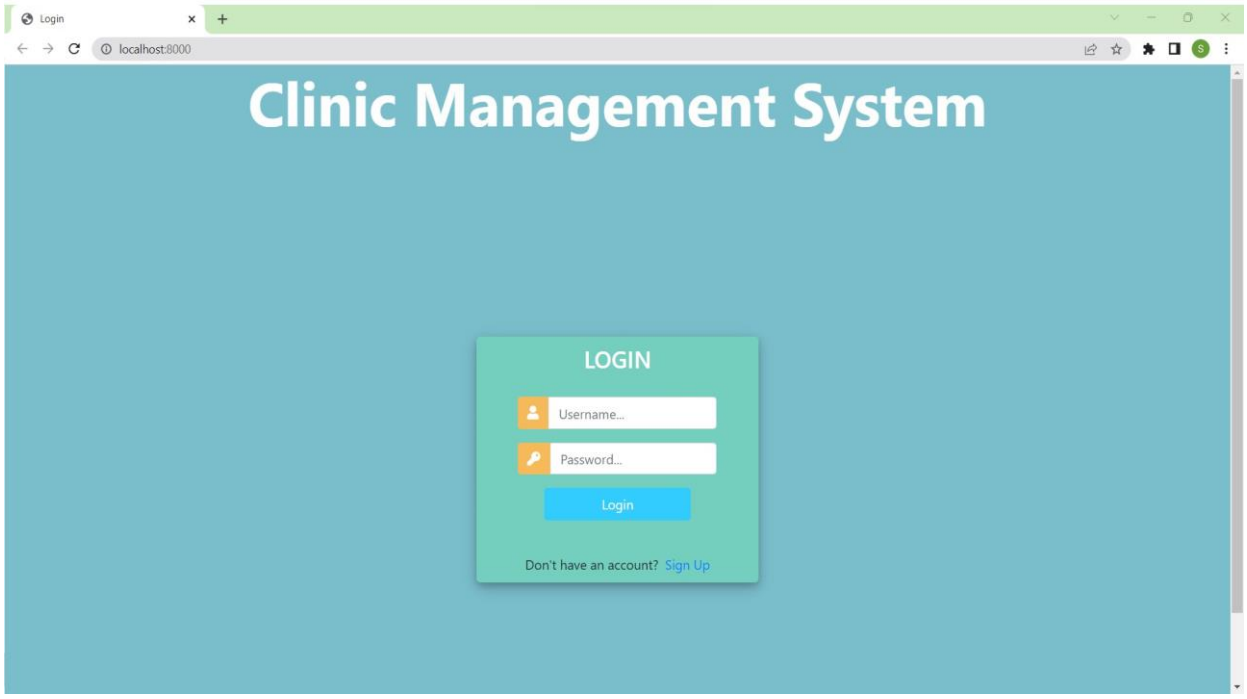
4.3 Sample Screenshots

Screen shots are the screens that clearly explain the project through Graphical User Interface (GUI). The following are the screens that explain the project.

Home page: the first page displayed after execution of the project is the home page screen. In this project the users are added by the admin of the clinic with the help of the admin interface, the admin adds the user by adding the username and its password.



Login screen: login is the process by which an individual gains access to a computer system by identifying and authenticating themselves. The user credentials are typically some form of "username" and a matching "password" and these credentials themselves are sometimes referred to as login, (or a log on or sign-in). When access is no longer needed, the user can log out or sign-off) from the application.

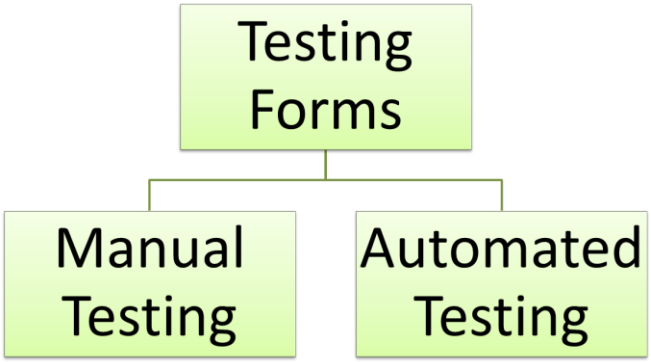


Chapter 5: Software Testing

5.1 Testing description

The purpose of testing is to assess product quality. It helps to strengthen and stabilize the architecture early in the development cycle. We can verify through testing, the various interactions, integration of components and the requirements which were implemented. It provides timely feedback to resolve the quality issues, in a timely and cost effective manner. The test workflow involves the following:

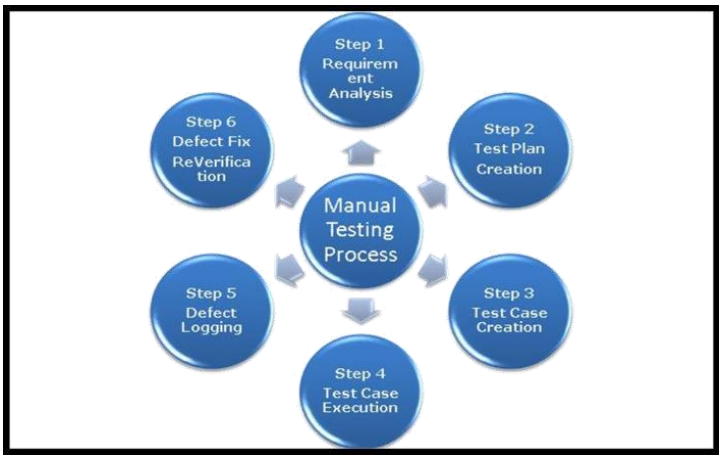
- ➔ Verifying the interactions of components.
- ➔ Verifying the proper integration of components.
- ➔ Verifying that all requirements have been implemented correctly.
- ➔ Identifying and ensuring that all discovered defects are addressed before the software is deployed.



Manual Testing is a process of finding out the defects or bugs in a software program. In this method, the tester plays an important role as the end-user and verifies that all the features of the application are working correctly. **The tester manually executes test cases without using any automation tools**. The tester prepares a test plan document which describes the detailed and systematic approach to testing of software applications. Test cases are planned to cover almost 100% of the software application. As manual testing involves complete test cases it is a time-consuming test.

The differences between actual and desired results are treated as defects. The defects are then fixed by the developer of the software application. The tester retests the defects to ensure that defects are fixed. The goal of Manual testing is to ensure that application is defect & error-free and is working fine to provide good quality work to customers.

5.2 Procedure Of Manual Testing



- ➔ Requirement Analysis
- ➔ Test Plan Creation
- ➔ Test case Creation
- ➔ Test case Execution
- ➔ Defect Logging
- ➔ Defect Fix & Re-Verification

Check Login Functionality there are many possible test cases:

- ➔ Test Case 1: Check results on entering valid User Id & Password
- ➔ Test Case 2: Check results on entering Invalid User ID & Password
- ➔ Test Case 3: Check response when a User ID is Empty & Login Button is pressed, and many more

The following links are to learn how to write test cases in manual testing with example :

- How to Write Test Cases in Manual Testing
- The format of Standard Test Cases
- Best Practice for writing good Test Case Example.
- Test Case Management Tools
- Resources

Writing Test Cases in Manual Testing

Test Case For Login Page

Description	Expected Output	Actual Output	Tester Name	Date
To validate/verify username and password	To be directed to the homepage	An error message invalid user and password display	Shariquddin	03-01-2023
To validate/verify username and password	To be directed to the homepage	Display home page	Shariquddin	03-01-2023

Test Case For Patient Registration Page

Description	Expected Output	Actual Output	Tester Name	Date
Create Patient details	Insert successfully	An error message student name should not be equal to null	Ahmed	03-01-2023
Create patient details	Insert successfully	Insert successfully	Ahmed	03-01-2023

Test Case For Home Page

Description	Expected Output	Actual Output	Tester Name	Date
To check the interface is link between login page and home page	To be directed to home page	Remain unchanged	Daniyaal	03-01-2023
To check the interface is link between login page and home page	To be directed to home page	To be directed to view/add home page	Daniyaal	03-01-2023

5.3 Testing using JMeter

Jmeter and Selenium IDE is used here to test a sample Web Application

JMeter Installation

Apache JMeter installation on Windows 10.

1. Install Java 8
   1.1. Download latest Java 8 from Oracle website. You will need to have an Oracle account for downloading Java.
   1.2. Run the Java installer package.
   1.3. Verify installed Java version by running this command on Command Prompt.
2. Download Apache JMeter
   2.1. You can download the latest version of Apache Jmeter from http://jmeter.apache.org/download_jmeter.cgi
   2.2. Download the files
   2.3. In the Apache-JMeter, go to the bin folder, then open the jmeter.bat file.
   2.4. Once you open the file, the JMeter GUI will open.

TESTING A SAMPLE WEB PAGE OF MAILING APPLICATION USING JMETER:

JMeter is an open-source, Java-based functional and load testing tool which was initially developed for web but has since extended to support all major protocols, like FTP, SMTP, JMS, and JDBC.

The basics of JMeter and you can configure and run load tests using JMeter.

Objective

When running a prolonged load test, you might want to receive the test reports automatically via email upon test completion, or you might be running scheduled tests and want to get test reports by email. Whatever the case, our goal is to configure JMeter to send the generated test reports automatically by email.

There are two ways you can achieve this using JMeter. The first method doesn't require any external plugins but depends on JMeter's configuration to overcome the inherent problem (described below) while sending test reports via SMTP sampler.

The second method requires you to install JMeter Standard Plugins (via Plugins Manager or via external jars) but provides more flexibility in the reporting section.

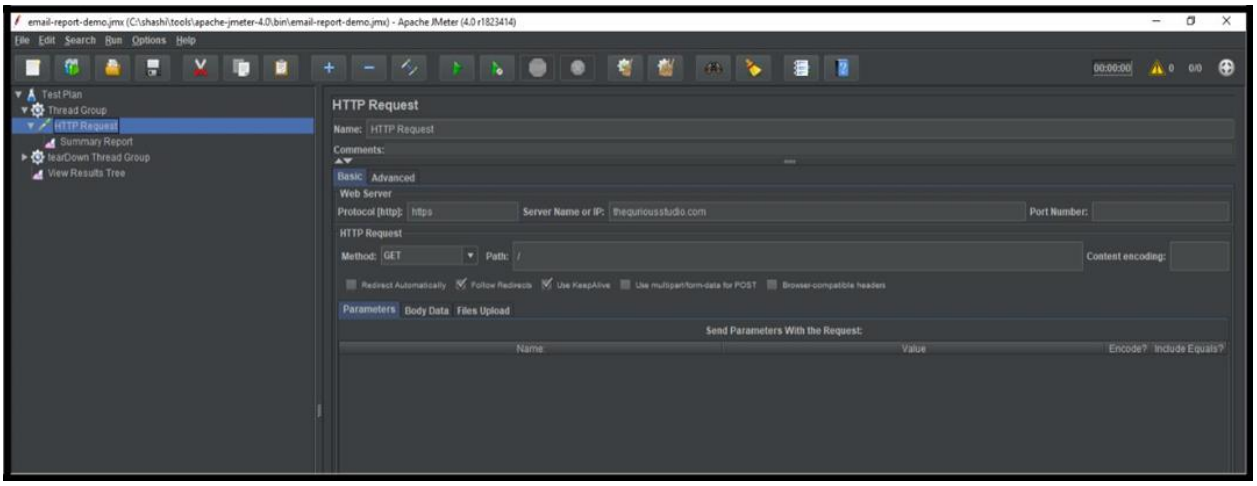
Prerequisites

The test has been performed with Apache JMeter v4.0, which requires JDK 8. The test script consists of a basic HTTP sampler with SMTP Sampler and "Summary Report" listener for the first method and "jp@gc Flexible File Writer" for the second method.

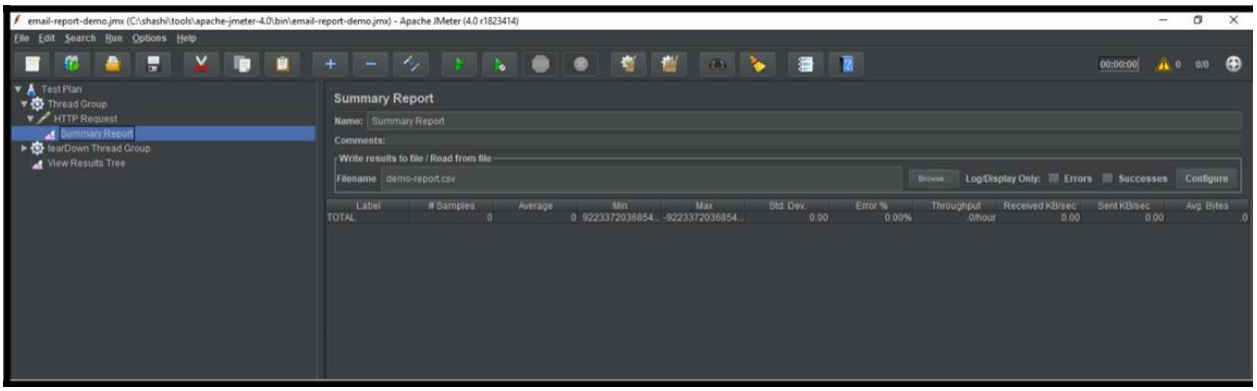
Using SMTP Sampler and Default "Summary Report"

Listener Add a Thread Group to the Test Plan.

Add an HTTP Sampler and set up the required parameters, like Server Name, Method, Path, etc.



Add the "Summary Report" listener to the Thread Group. Configure the File Name parameter to define the file where reports will be saved (the default file path in JMeter's bin directory). The File Name parameter is important as it will be referenced by the SMTP sampler.



Add a tearDown Thread group to the Test Plan. The tearDown thread group will run only after the test has been executed, so it will run after the first Thread Group is completed.

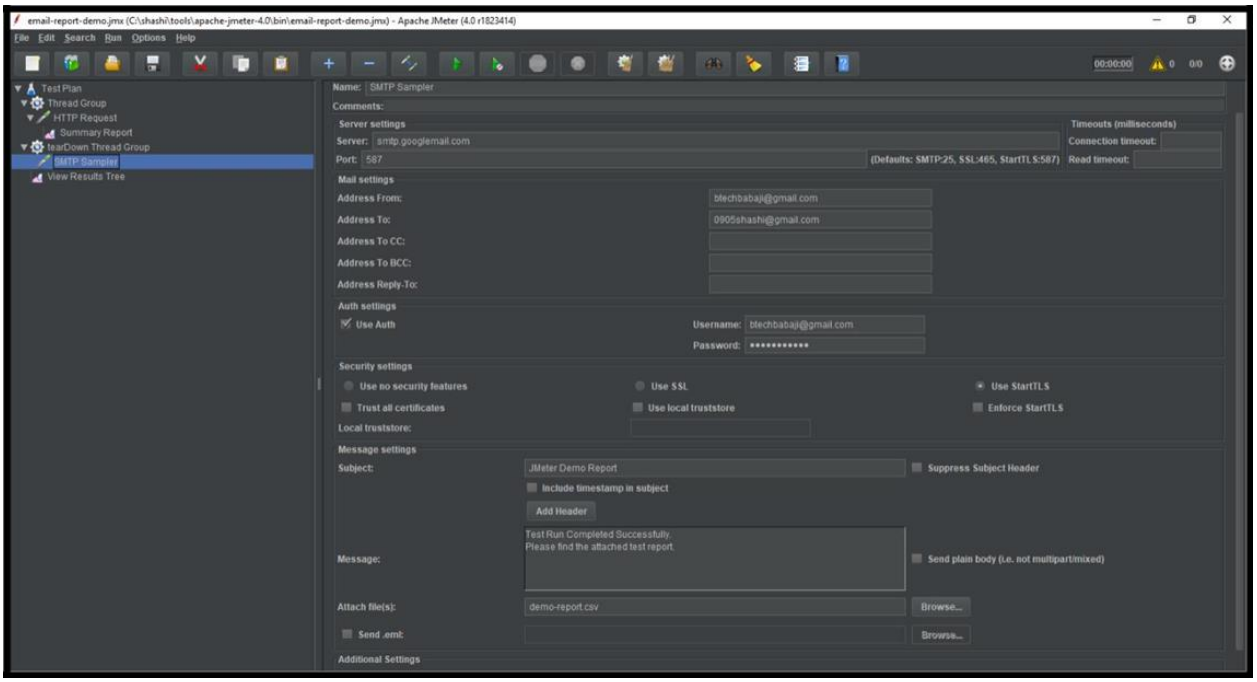
Add the SMTP Sampler to the Tear-Down Thread group and configure the mail server parameters. A demo configuration using Gmail. The SMTP server would use the following parameter values:

➔ **Server:** smtp.googlemail.com

➔ **Port:** 587

- ◆ **Address form:**
  - **Address from:** sender@gmail.com
  - **Address to:** recipient@gmail.com
  - **Auth settings:**
    - **Username:** sender@gmail.com
    - **Password:** sender's gmail password.
  - **Security settings:** use StartTLS.
  - **Message settings:**
    - **Subject:** email subject.
    - **Message:** email body is here.
    - **Attach file:** enter the filename used in the summary report listener in step 3.

When using Gmail as an SMTP server, you might need to configure Gmail's settings to allow "Less-Secure" apps to sign in; otherwise, Google may block JMeter from sending any emails.



Now your test plan is almost complete, but if you run the test now, you will get an empty file in the mail without any data. This is due to the fact that the report file is written only after the test script has finished execution (which includes our tearDown Thread Group). When the SMTP Sampler is executed, it gets only an empty file without any data. The reason JMeter does this is to provide better performance — instead of executing write operations on a per-line basis, it is done all at once after test execution.

However, this property of JMeter can easily be controlled by modifying the "user.properties" (or "jmeter.properties") configuration file located in JMeter's bin directory.

Find and modify (or add, if not present) the following section in the file, setting the autoflush parameter to true:

- ➔ AutoFlush on each line written in XML or CSV output
- ➔ Setting this to true will result in less test results data loss in case of Crash
- ➔ but with impact on performances, particularly for intensive tests (low or no pauses)
- ➔ Since JMeter 2.10, this is false by default
- ➔ jmeter.save.saveservice.autoflush=true

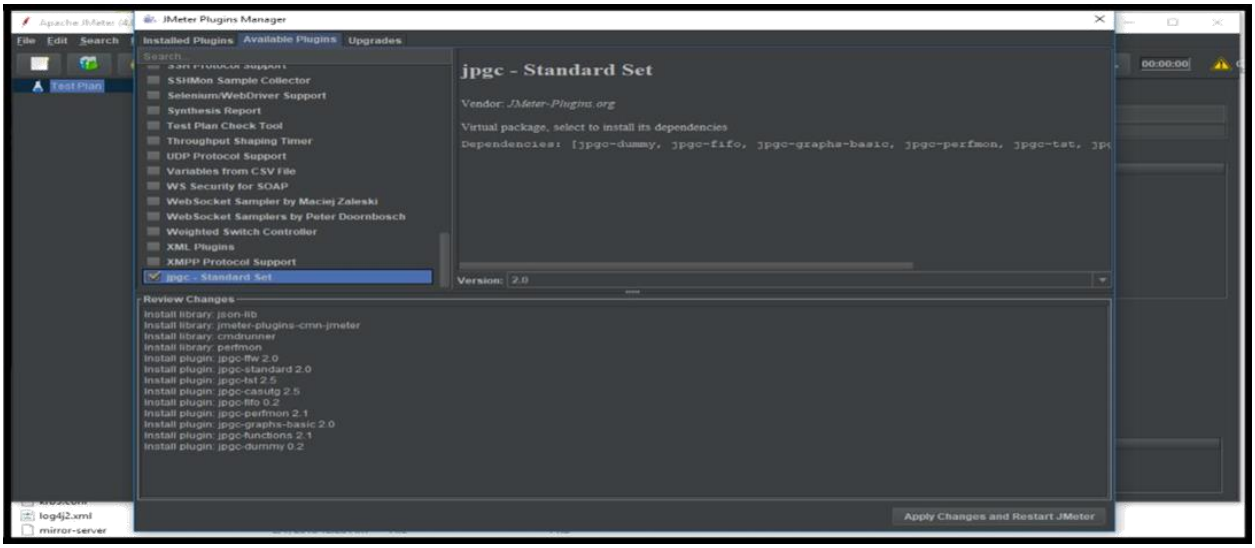
Now, restart the JMeter and run the test again. You should be able to get the test report on mail with complete data.

Using SMTP Sampler With jp@gc Flexible File Writer

The jp@gc standard plugin set for JMeter can be used to extend its capabilities, including features like writing test results in flexible formats, enabling server monitoring, and recording performance metrics.

The standard plugin set can be installed via Plugins Manager or by directly downloading the jar and putting it into the /lib/ext directory. However, Plugins Manager provides easy-to-use plugin management capabilities via UI. To install Plugins Manager, download its jar.

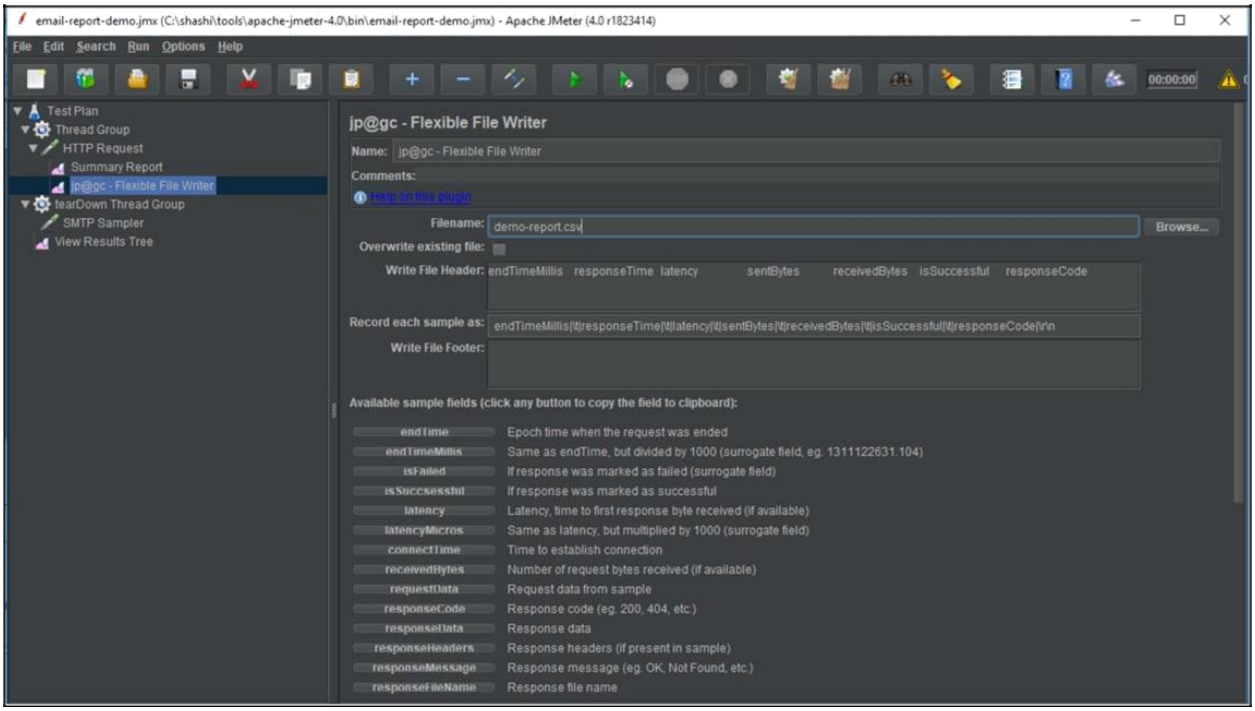
Once installed, you can access the Plugins Manager via File Menu in JMeter (under the "options" tab). To install the jp@gc Standard Plugin Set (which includes Flexible File Writer), go to the "Available Plugins" tab and mark the checkbox for "jp@gc Standard Set." Click the button in the bottom-right corner to "Apply Changes and Restart JMeter."



After restarting JMeter, you can find Flexible File Writer in the list of Listeners. Now, replace the "Summary Report" listener in our test script with the "Flexible File Writer."

When you use Flexible File Writer, you don't have to modify the configuration file to enable Autoflush (done in Step 6) as it keeps writing the report file during test execution.

Notice the flexibility with the Flexible File Writer Listener. You can select the parameters to be included in the report by just selecting them from the list. You can also specify the format in which the data is written in the file.



You can read more about the usage and features of Flexible File Writer.

The choice between the two methods above depends solely on your test requirements. If you only need a basic summary report, you can go with the inbuilt Summary Report Listener. In case you need more flexibility or control over the reports and want to explore more new features, then try out the jp@gc set of plugins.

5.4 Testing using Selenium IDE

Selenium IDE (Integrated Development Environment) is primarily a record/run tool that a test case developer uses to develop **Selenium** Test cases. **Selenium IDE** is an easy to use tool from the **Selenium** Test Suite and **can** even be used by someone new to developing automated test cases for their web applications

It is a Firefox add-o that creates tests very quickly through its record-and-playback functionality.

This feature is similar to that of QTP. It is effortless to install and easy to learn.

Installation of Selenium IDE

What you need;

- ➔ Mozilla Firefox
- ➔ Active Internet Connection

If you do not have Mozilla Firefox yet, you can download it from http://www.mozilla.org/en-US/firefox/new.

1. Launch firefox and navigate to https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/, click on add to firefox.
2. Complete the installation when prompted whether to add or cancel the add-on.
3. Open selenium by clicking on its icon.

Firefox DevTools

1. Right click anywhere on the page and select Inspect Element. You can also use the shortcut Ctrl + Shift + I.
2. You can right click on an element and choose CSS or XPath. This is useful in object identification.

Plugins

Selenium IDE can support additional Firefox add-ons or plugins created by other users. You can visit here for a list of Selenium add-ons available to date. Install them just as you do with other Firefox add-ons.

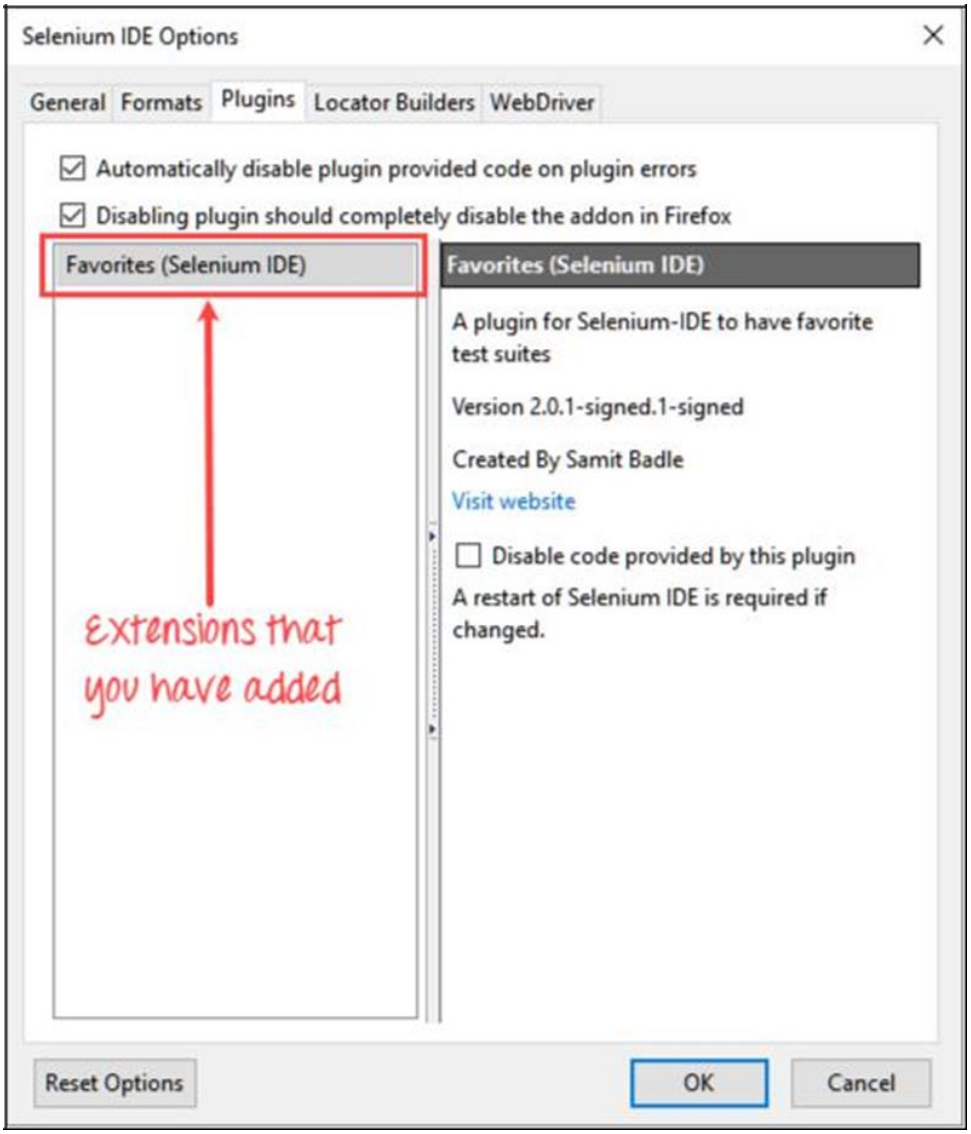
By default, Selenium IDE comes bundled with 4 plugins:

1. Selenium IDE: C# Formatters
2. Selenium IDE: Java_Formatters
3. Selenium IDE: Python_Formatters

4. Selenium IDE: Ruby Formatters

These four plugins are required by Selenium IDE to convert Selenese into different formats.

The Plugins tab shows a list of all your installed add-ons, together with the version number and name of the creator of each.

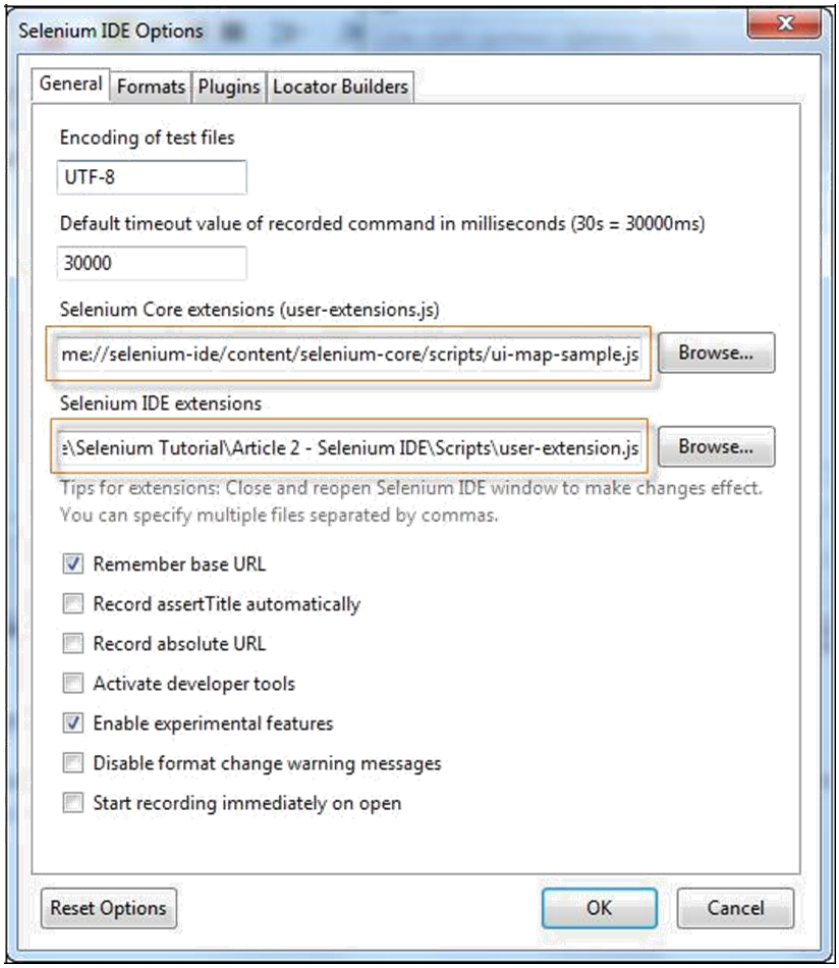


User Extensions

Selenium IDE can support user extensions to provide advanced capabilities. User extensions are in the form of JavaScript files. You install them by specifying their absolute path in either of these two fields in the Options dialog box.

Selenium Core extensions (user-extensions.js)

Selenium IDE extensions



Creating A Test Case In Selenium

1. Start Selenium IDE from Firefox's ToolBar.
2. Add "Base URL" to specify the web page on which the test is going to be performed. This URL can be changed to perform the same test case on a different URL.
3. Once you've added the Base URL you can press the "Record" button (red round button top right) and start performing a manual test to the site (filling inputs, clicking links, etc.).
4. As you perform your manual test, Selenium will populate its commands table with all your interactions on the page.
5. After you've finished recording your test, press the "Record" button again to stop the recording process.
6. Once your test is done recording, you can use the "Play" buttons to play either and entire Test Suite or current Test Case.
7. Right click on the test case's title, at the moment "Untitled *" and click on "Properties".
8. Here you can change the Test Case's Title to something more descriptive as you will probably have many test cases in a single Test Suite.
9. Once your test case has a title, you can proceed to save it. Click Firefox's menu File> Save Test Case As … and add a name to the file. Test Cases could be saved as HTML files.
10. If you save a Test Case as an HTML file you can later double click on it and see the table on any browser.

Editing an existing Test Case

1. Start Selenium IDE from Firefox's ToolBar.
2. Click Firefox's menu File> Open … and browse to the Test Case file that you want to work with.

Once loaded there are several things you can do to a Test Case:

- ➔ You can "Edit" a command
- ➔ You can "Add" new commands
- ➔ You can "Delete" commands

Editing a Command

1. Click on a command from the Commands Table to select it.
2. Right below the commands table you'll see the selected command's information
3. Here you can change the command executed selecting from the drop down menu. You CAN Find an Entire list of Selenium's commands in the Selenium Reference ("broken link").
4. Once a command is selected you can see a short reference at the bottom of the app, where you'll find the command's arguments and description.
5. You can also edit the target of the command by changing the target's id.
6. If you do not know the target's id you can click on "Select" and then click on the website's element you want to apply the command to, this will get you its id.
7. And finally you can change the value used with the selected command by changing the "Value" field.
8. After you've finished editing commands you can proceed to save your test case as previously explained.

Adding a new command

1. Right click anywhere on the command's table and select the "Insert New Command" option.
2. An empty new command will be added to the table.
3. Edit it as previously explained.
4. After editing, drag and drop the command to its required position.
5. After you've finished adding commands you can proceed to save your test case as previously explained.

Deleting command

1. Go to your commands table and right click on the command you want to eliminate.
2. Select "Delete" from the menu.
3. After you've finished deleting commands you can proceed to save your test case as previously explained.

REUSING SELENIUM TEST CASES

The Base URL field at the top of the Selenium-IDE window is very useful for allowing test cases to be run across different domains. Suppose that a site named http://news.portal.com had an in-house beta site named http://beta.news.portal.com. Any test cases for these sites that begin with an open statement should specify a relative URL as the argument to open rather than an absolute URL (one starting with a protocol such as http: or https:). Selenium-IDE will then create an absolute URL by appending the open command's argument onto the end of the value of Base URL…

At the End of Practical's

This Email Client Software is developed following SDLC phases with Forward & Reverse Engineering, it can be implemented to fulfill all the client requirements. The system interface is very user friendly, and the overall system has been successfully tested. It has a broad future scope as new features can be incorporated in the present proposed system. The system can be used for online sharing of data without the involvement of authority by which users can read messages from any part of the world.

This system is designed using Unified Modelling Language concepts. The interfaces designed for the system are very user friendly and attractive. It has successfully implemented the receive messages as per the client requirement.

The system has successfully passed both testing at the development site and is under the testing phase in the presence of the client. The system is waiting for the client.

Chapter 6: CONCLUSION

The Clinic Management System is developed using Visual Basic .NET fully meets the objectives of the system which it has been developed. The system has reached a steady state where all bugs have been eliminated. The system is operated at a high level of efficiency and all the teachers and users associated with the system understand its advantage. The system solves the problem. It was intended to solve the requirement specification.

6.1 Scope for Future Development

The project has a very vast scope in future. The project can be implemented on intranet in future. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion. With the proposed software of database Space Manager ready and fully functional the client is now able to manage and hence run the entire work in a much better, accurate and error free manner.

References

- ➔ W3School.com
- ➔ Java script from beginner to professional, author-Laurence, Maaike, Rob Percival.
- ➔ Bootstrap