1. SYSTEM TESTING :

❖ INTRODUCTION :

- Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free.
- It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest.
- The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Following are the characteristic that process the testing templates:

- The developer should conduct the successful technical reviews to perform the testing successful.
- Testing starts with the component level and work from outside toward the integration of the whole computer-based system.
- Different testing techniques are suitable at different point in time.
- Testing is organized by the developer of the software and by an independent test group.
- Debugging and testing are different activities, then also the debugging should be accommodated in any strategy of testing.

❖ IMPORTANCE OF TESTING :

- Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product.
- Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

- Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss.

❖ BENEFITS OF TESTING :

- **Cost-Effective:** It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.

- **Security:** It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

- **Product quality:** It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.

- **Customer Satisfaction:** The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience

❖ TESTING STRATEGIES :

- **UNIT TESTING :**

  - Tests a single class or a set of closely coupled classes.
  - These unit tests can either be run using the actual objects that the unit interacts with or by employing the use of test doubles or mocks.
  - In Unit testing, the smallest piece of testable software is tested in the application to determine whether it behaves as expected or not.
  - Tests are typically run at the class level or around a small group of related classes. In unit testing, an important distinction is seen based on whether or not the unit under test is isolated from its collaborators.
  - Unit tests are usually written by the programmers using their regular tools. The only difference being the use of the same sort of unit testing framework.

    There are further two types of testing in Unit Testing :

    a. Sociable Unit Testing :

      - It focuses on testing the behaviour of modules by observing changes in their state. This treats the unit under test as a black box tested entirely through its interface.

    b. Solitary Unit Testing :

      - It looks at the interactions and collaborations between an object and its dependencies, which are replaced by test doubles.

- **INTEGRATION TESTING :**

  - Integration tests are used to test communication between services. These tests are designed to test basic success and error paths over a network boundary.
  - Different components interact with each other for their functional dependency, while communicating with each other integration test verifies

the communication paths between the components and detect the interface defects.

- Here, all test modules are integrated together and tested as a subsystem. It checks that the communication paths between the subsystem work correctly while interacting with its peers. In micro service architecture, they are typically used to verify interactions between layers of integration code and the external components to which they are integrating.

- When the automated tests are written for the modules which are interacting with an external component, the basic goal is to verify the modules are interacting sufficiently with the external component.

- It is very difficult to trigger abnormal behaviour such as a timeout or slow responses from the external component. Special tests are written to ensure that test respond as expected in the unexpected circumstances

- **Persistence integration tests** provide assurances that the schema assumed by the code matches that is available in the data store.

- With unit testing and integration testing, we can have confidence in the correctness of the logic contained in the individual modules that make up the micro service, but we cannot be sure that the micro services work together as a whole to satisfy business requirements.

- **COMPONENT TESTING :**

- Tests the full function of a single micro service. During this type of testing, any calls to external services are mocked in some way.

- A component is any well-encapsulated, coherent and independently replaceable part of a larger system. In a micro service architecture, the components are the services themselves.

- A component's complex behaviour is avoided by isolating it from its peers, also isolation help in controlling the test environment for the component.

- **CONTRACT TESING :**

  - Test the agreed contract for APIs and other resources that are provided by the micro service.

  - At the boundary of the external service, an Integrated contract test is done to verify the contract that is expected by the consuming service. This test verifies that the component meets a contract.

  - A test suite is written to verify only those aspects of the producing service that is in use. The behaviour of the service is not deeply tested, response latency and throughput should be within acceptable limits when input and output of the service call contain required attributes. This test is written by each test consuming team and then packaged. The main aim of this test is to know the impact of the changes made by the maintainers on the consumers.

- **END TO END TESTING :**

  - End-to-End Testing, Tests a complete flow through the application or micro service. Usually used to test a golden path or to verify that the application meets external requirements.

  - End-to-End testing tests the whole system from end to end. It verifies that the entire system meets the external requirements and eventually achieve its goal. Without bothering about the internal architecture of the application business goal should be achieved by the End-to-End testing.

  - The system is fully deployed and is treated as a black box and the test is exercised. With Public interference through GUIs and API, the system is manipulated. End to End Tests is more business facing.

  - This test verifies that the firewall, proxies, and load balancers are correctly configured.

  - In micro service architecture, for one behaviour, there are many micro services which interact to respond to that behaviour, an end-to-end testing provides value by adding coverage of gaps between the system

2. TEST CASE :

- **1st Test Case :**

TEST CASE ID	1-A
Name	Change Password Validation
Test Scenario	It Will Use To Change Password Of Any User At Login Stage.
Test Step	1.Open Web-site 2.Open Login Page 3.Click On Change Password 4.Enter Detail 5.Click On Update
Test Data	Username : admin Old Password : admin123 New Password : admin1 Re-Enter Password : admin1
Expected Result	Message : Password Changed Successfully !!
Actual Result	As Expected
Pass / Fail	Pass

- **2nd Test Case :**

TEST CASE ID	1-B
Name	Change Password Validation
Test Scenario	This Test Show If User Enter Wrong Username For Change Password.
Test Step	1.Open Web-site 2.Open Login Page 3.Click On Change Password 4.Enter Detail 5.Click On Update
Test Data	Username : admin1 Old Password : admin1 New Password : admin123 Re-Enter Password : admin123
Expected Result	Message : Username Not Match !!
Actual Result	As Expected
Pass / Fail	Pass

- **3rd Test Case :**

TEST CASE ID	1-C
Name	Change Password Validation
Test Scenario	This Test Show If User Enter Wrong Password For Change Password.
Test Step	1.Open Web-site 2.Open Login Page 3.Click On Change Password 4.Enter Detail 5.Click On Update
Test Data	Username : admin Old Password : admin123 New Password : admin12 Re-Enter Password : admin12
Expected Result	Message : Old Password Is Invalid !!
Actual Result	As Expected
Pass / Fail	Pass

- **4th Test Case :**

TEST CASE ID	1-D
Name	Change Password Validation
Test Scenario	This Test Show If User Enter Password And New Password Same For Change Password.
Test Step	1.Open Web-site 2.Open Login Page 3.Click On Change Password 4.Enter Detail 5.Click On Update
Test Data	Username : admin Old Password : admin1 New Password : admin1 Re-Enter Password : admin1
Expected Result	Message : New Password Must Be Different From Previous Password !!
Actual Result	As Expected
Pass / Fail	Pass

- **5th Test Case :**

TEST CASE ID	1-E
Name	Change Password Validation
Test Scenario	This Test Show If User Enter New Password And Re-Enter Password Different For Change Password.
Test Step	1.Open Web-site 2.Open Login Page 3.Click On Change Password 4.Enter Detail 5.Click On Update
Test Data	Username : admin Old Password : admin1 New Password : admin123 Re-Enter Password : admin123
Expected Result	Message : New Password And Re-Enter Password Must Be Same !!
Actual Result	As Expected
Pass / Fail	Pass

- **6th Test Case :**

TEST CASE ID	1-F
Name	Change Password Validation
Test Scenario	This Test Show If User Not Enter Any Of The Filed In Change Password Form.
Test Step	1.Open Web-site 2.Open Login Page 3.Click On Change Password 4.Enter Detail 5.Click On Update
Test Data	Username : … Old Password : … New Password : … Re-Enter Password : …
Expected Result	Message : Enter All Filed First !!
Actual Result	As Expected
Pass / Fail	Pass

- **7th Test Case :**

TEST CASE ID	2-A
Name	Change Username Validation
Test Scenario	This Test For Username Change Successfully.
Test Step	1.Open Web-site 2.Login 3.After Login Goto Manage Profile 4.Click On Change Username 5.Click On Login
Test Data	Username : admin1
Expected Result	Message : Username Change Successfully !!
Actual Result	As Expected
Pass / Fail	Pass

- **8th Test Case :**

TEST CASE ID	2-B
Name	Change Username Validation
Test Scenario	This Test For Username Is Already Exist.
Test Step	1.Open Web-site 2.Login 3.After Login Goto Manage Profile 4.Click On Change Username 5.Click On Login
Test Data	Username : manager
Expected Result	Message : Username Is Already Exist !!
Actual Result	As Expected
Pass / Fail	Pass

- **9th Test Case :**

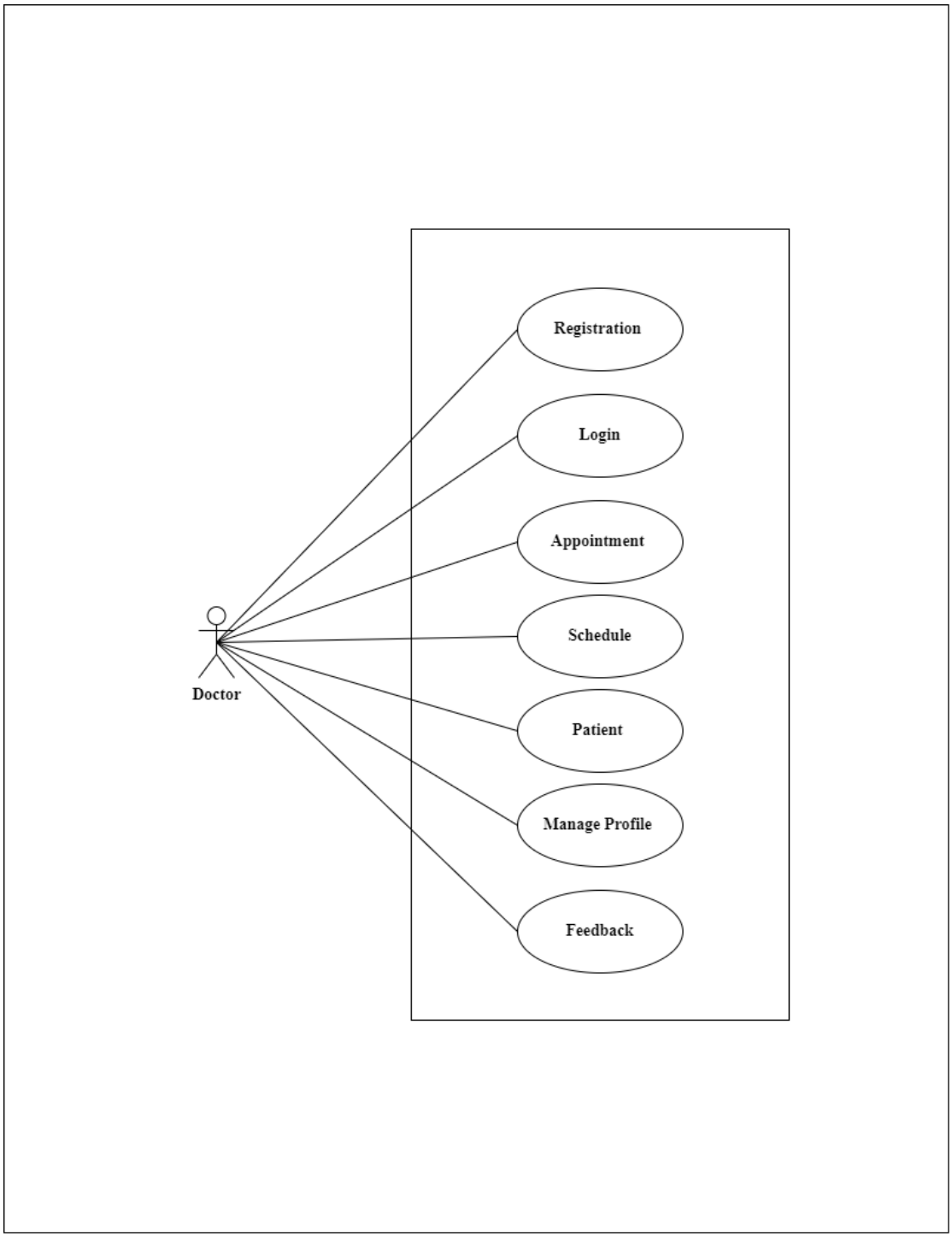
TEST CASE ID	2-C
Name	Change Username Validation
Test Scenario	This Test For Username Is Same As Previous Username Already Exist.
Test Step	1.Open Web-site 2.Login 3.After Login Goto Manage Profile 4.Click On Change Username 5.Click On Login
Test Data	Username : admin1
Expected Result	Message : Enter Different username !!
Actual Result	As Expected
Pass / Fail	Pass

- **10th Test Case :**

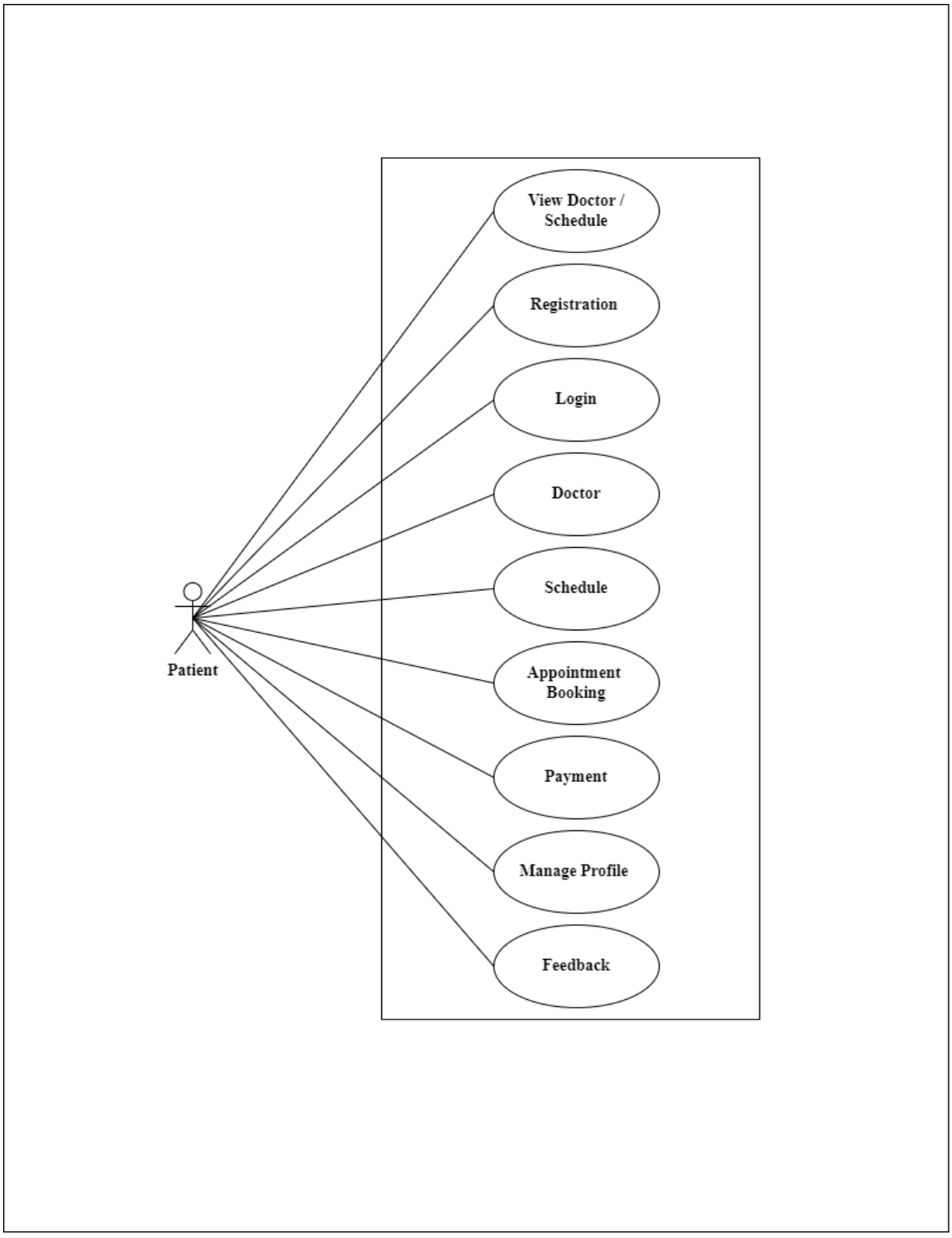
TEST CASE ID	2-D
Name	Change Username Validation
Test Scenario	This Test Show If User Not Enter Any Data In Change Username Form.
Test Step	1.Open Web-site 2.Login 3.After Login Goto Manage Profile 4.Click On Change Username 5.Click On Login
Test Data	Username : …
Expected Result	Message : Enter New Username !!
Actual Result	As Expected
Pass / Fail	Pass

3. USE-CASE DIAGRAM :

❖ DOCTOR'S USE CASE :



❖ PATIENT'S USE CASE :



4. ACTIVITY DIAGRAM :

❖ WHAT IS ACTIVITY DIAGRAM ?

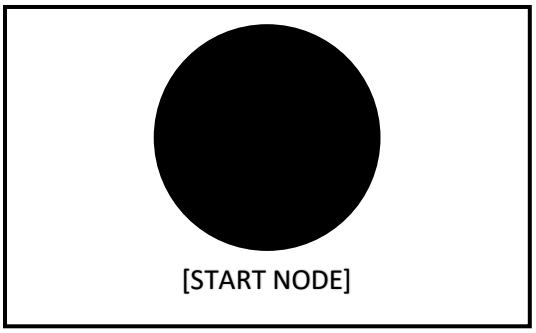
- Activity diagram is another important behavioural diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modelling the flow from one activity to another activity.

❖ WHEN TO USE ACTIVITY DIAGRAM :

- Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction.
- Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination.
- It is also suitable for modelling how a collection of use cases coordinate to represent business workflows.
- Either model generic interactions (showing all possible paths through the interaction) or specific instances of a interaction

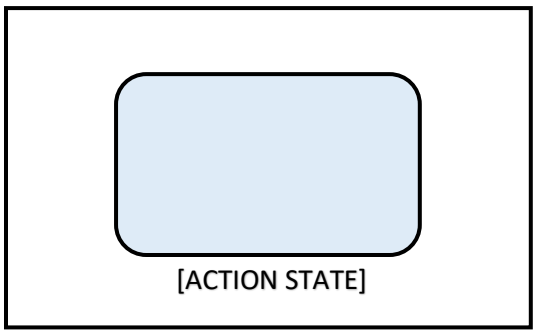
❖ SYMBOL AND NOTATION OF ACTIVITY DIAGRAM :

- **START NODE :**



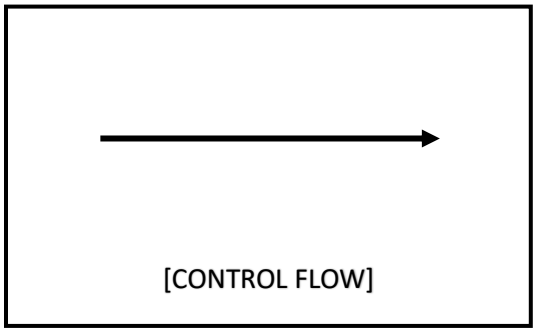
- A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram. For activity diagram using swimlanes, make sure the start point is placed in the top left corner of the first column.

- **ACTION STATE :**



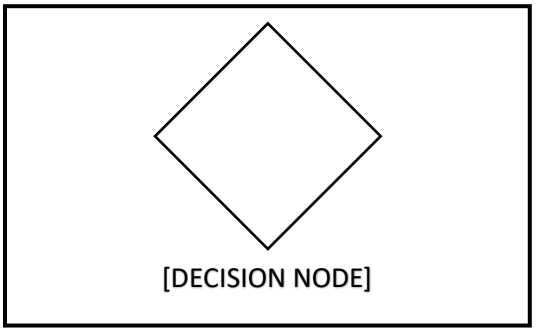
- An action state represents the non-interruptible action of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.

- **CONTROL F LOW :**



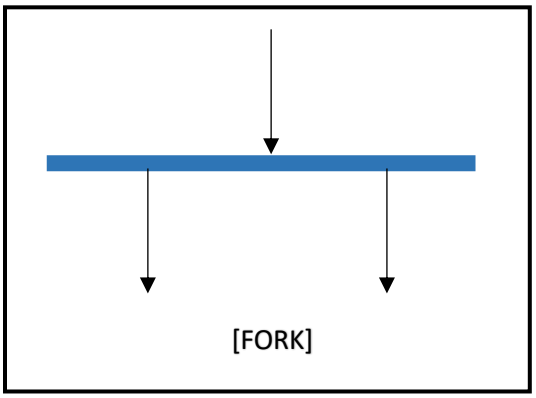
- Control flows, also called edges and paths, illustrate the transitions from one action state to another. They are usually drawn with an arrowed line.

- **DECISION NODE :**



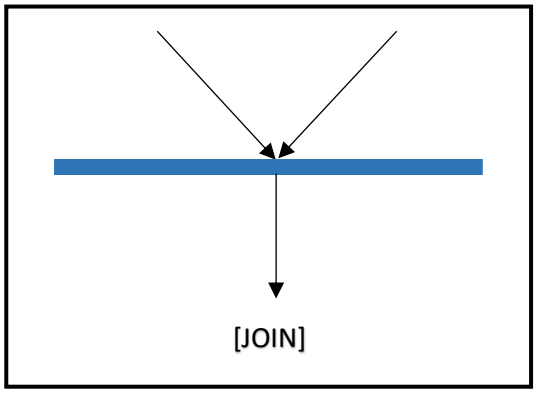
- A diamond represents a decision with alternate paths. When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."

- **FORK :**



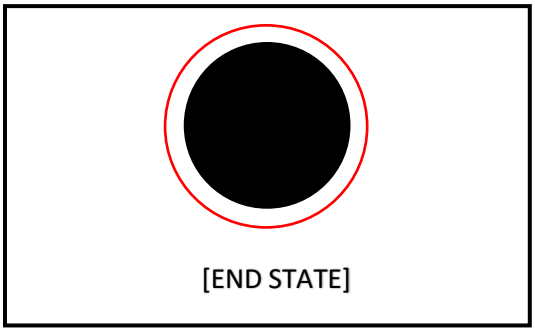
- Splits a single activity flow into two concurrent activities. Symbolized with multiple arrowed lines from a join.

- **JOIN :**



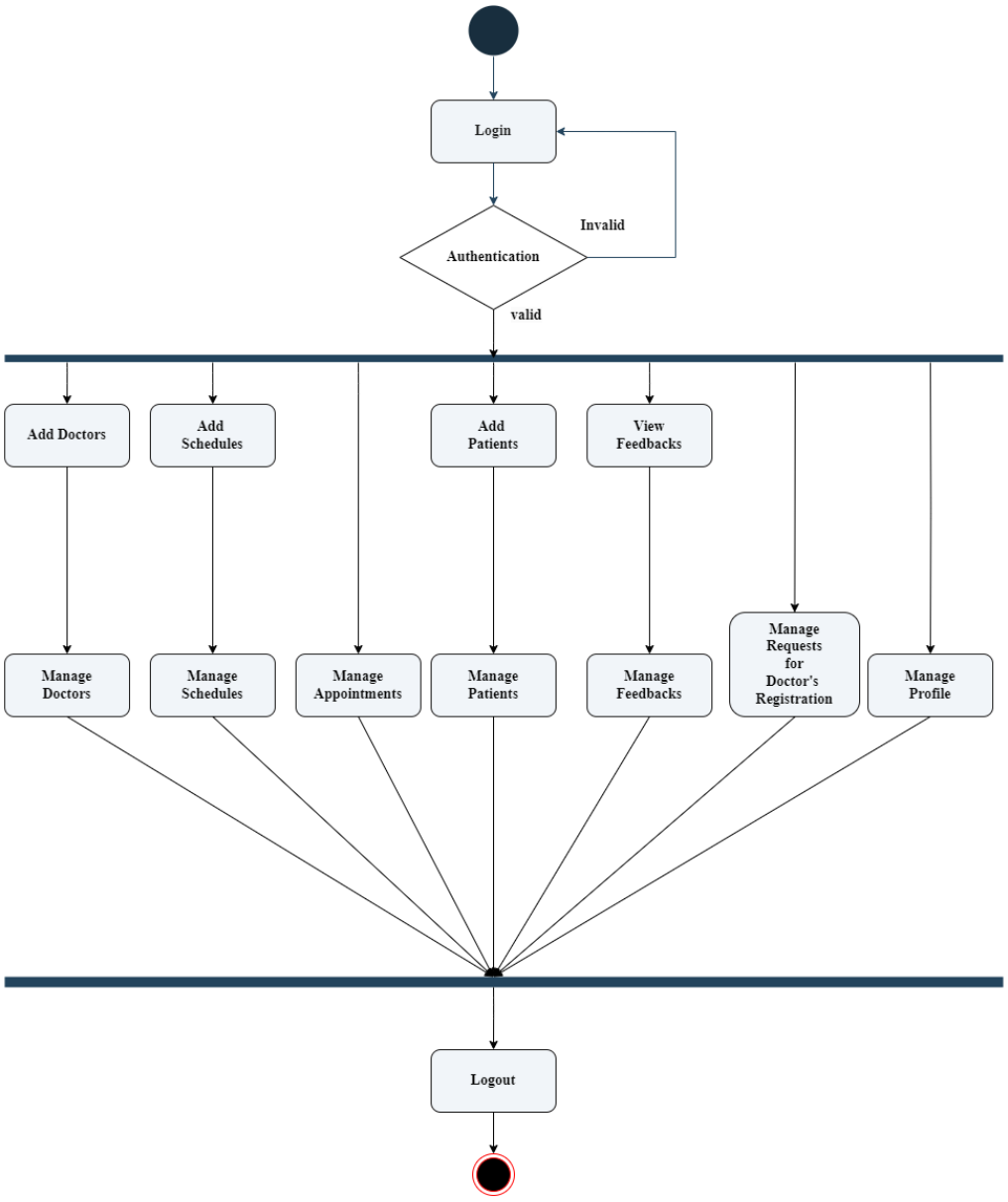
➢ Combines two concurrent activities and re-introduces them to a flow where only one activity occurs at a time. Represented with a thick vertical or horizontal line.

- **END STATE :**



➢ Marks the end state of an activity and represents the completion of all flows of a process.

❖ ADMIN ACTIVITY DIAGRAM :



5. FUTURE SCOPE :

i. Integration of Payment Options :

To enhance the payment options on your platform, you're looking to incorporate popular methods such as BHIM Pay and UPI (Unified Payments Interface). BHIM Pay is a mobile app developed by the National Payments Corporation of India (NPCI) to facilitate digital transactions through UPI. Integrating these options will offer users more flexibility and convenience in making payments on your platform. This integration involves collaborating with relevant financial institutions, implementing secure payment gateways, and ensuring seamless user experience during transactions. By diversifying payment methods, you're accommodating a broader range of users with varying preferences and access to banking services.

ii. Scaling up the Platform :

Your current platform caters to a limited number of cities, but you aim to expand its reach to cover entire states and counties. This expansion requires a strategic approach encompassing infrastructure development, resource allocation, and market penetration strategies. Scaling up involves optimizing server capacity, enhancing network bandwidth, and employing scalable software architecture to accommodate increased traffic and data volume. Moreover, it entails market research to understand regional preferences and needs, localization of content and services, and forging partnerships with local stakeholders such as healthcare providers, government agencies, and community organizations. By scaling up, you're positioning your platform to become a comprehensive solution accessible to a wider audience, thereby maximizing its impact and potential.

iii. Implementation of Virtual Doctor Consultations :

In response to the growing demand for telemedicine services, you're introducing virtual sessions where patients can consult with doctors remotely. This feature requires the integration of video conferencing technology, secure communication channels, and electronic health record (EHR) systems to facilitate seamless interaction between patients and healthcare providers. Doctors need a user-friendly interface to schedule appointments, review patient histories, and conduct virtual consultations effectively. Patients, on the other hand, require intuitive platforms to book appointments, share medical records securely, and engage in real-time consultations from the comfort of their homes. Ensuring

compliance with healthcare regulations, safeguarding patient privacy, and maintaining high standards of medical practice are paramount in implementing virtual doctor consultations. By embracing telemedicine, you're enhancing accessibility to healthcare services, improving patient outcomes, and fostering a more patient-centered approach to medical care delivery.