MARRI EDUCATIONAL SOCIETY'S GROUP OF INSTITUTIONS

MARRI LAXMAN REDDY

INSTITUTE OF TECHNOLOGY & MANAGEMENT

CASETOOLS AND SOFTWARE TESTING

Laboratory Manual

IV YEAR B.TECH (I - SEM)

(CSE)

Department of Computer Science & Engineering

PREFACE

The term "computer networking" conjures up an image of computer nodes linked by telecommunications in such a way that all those Computer resources would be accessible to anyone connected to the net and that a single problem could be handled by two or more computers in the net without human intervention.

Actually the computer networks commonly found today are of two general but simple types: (1) star computer networks, whereby a single, central computer has communication lines radiating to terminal devices; and (2) communication networks, whereby a user at a terminal can connect with *anyone* from a variety of computer-based services remotely located, by dialing the appropriate telephone number.

Technically, all the components of computer networking could be present in a single laboratory or spread over a campus, a region, a country, or the world.

There are several motivations for computer networking, including:

1. Sharing of data and of information resources.

2. Sharing of specialized and expensive computing resources, dedicated to a particular mode of problem solving by users.

3. Provision for loosely-coupled computing support for the experimentalist who has a tightly-coupled controlled system and occasionally needs additional computer resources.

4. Standardization of protocols and algorithms. The variations that now exist, which render software from different sources incompatible,
are often idiosyncratic rather than substantive.

5. Sharing of algorithms *via* a few sites. This means that the best thinking of researchers working in a specific area can be accumulated, used and continually revised within a common mode of representation.

6. A distributed network of minicomputers at one site may be cheaper, easier to program, and more reliable than one large computer.

HOD (CSE) **PRINCIPAL**

LAB CODE

1. Students should report to the concerned lab as per the time table.

2. Students who turn up late to the labs will in no case be permitted to do the program schedule for the day.

3. After completion of the program, certification of the concerned staff in-charge in the observation book is necessary.

4. Student should bring a notebook of 100 pages and should enter the readings /observations into the notebook while performing the experiment.

5. The record of observations along with the detailed experimental procedure of the experiment in the immediate last session should be submitted and certified staff member in-charge.

6. Not more than 3-students in a group are permitted to perform the experiment on the set.

7. The group-wise division made in the beginning should be adhered to and no mix up of students among different groups will be permitted.

8. The components required pertaining to the experiment should be collected from stores in-charge after duly filling in the requisition form.

9. When the experiment is completed, should disconnect the setup made by them, and should return all the components/instruments taken for the purpose.

10. Any damage of the equipment or burn-out components will be viewed seriously either by putting penalty or by dismissing the total group of students from the lab for the semester/year.

11. Students should be present in the labs for total scheduled duration.

12. Students are required to prepare thoroughly to perform the experiment before coming to laboratory.

13. Procedure sheets/data sheets provided to the student's groups should be maintained neatly and to be returned after the experiment.

INDEX

SOFTWARE TESTING LAB

INDEX

UNIFIED MODELING LANGUAGE

Introduction

The unified modeling language(UML)is a standard language for writing software blue prints.

The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting

The artifacts of a software system:

UML is a language that provides vocabulary and the rules for combing words in that vocabulary for the purpose of communication

A modeling language is a language whose vocabulary and rules focus on the concept and physical representation of a system. Vocabulary and rules of a language tell us how to create and real well formed models, but they don't tell you what model you should create and when should create them.

Visualizing

The UML is more than just a bunch of graphical symbols. In UML each symbol has well defined semantics. In this manner one developer can write a model in the UML and another developer or even another tools can interpret the model unambiguously.

SPECIFYING

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

Constructing

UML is not a visual programming language but its models can be directly connected to a variety of programming languages. This means that it is possible to map from a model in the UML to a programming language such as java, c++ or VisualBasic or even to tables in a relational database or the persistent store of an object-oriented database. This mapping permits forward engineering. The generation of code from a UML model into a programming language. The reverse engineering is also possible you can reconstruct a model from an implementation back into the UML.

Documenting

UML is a language for Documenting. A software organization produces all sorts of artifacts in addition to raw executable code. These artifacts include Requirements, Architecture, Design, Source code, Project plans ,Test, Prototype, Release. Such artifacts are not only the deliverables of a project, they are also critical in controlling, measuring and communicating about a system during its development and after its deployment.

Conceptual model of the UML:

To understand the UML, we need to form a conceptual model of the language and this requires learning three major elements.

The UML Basic Building Blocks.

The Rules that direct how those building blocks may be put together. Some common mechanisms that apply throughout the UML. As UML describes the real time systems it is very important to make a conceptual model and then proceed gradually. Conceptual model of UML can be mastered by learning the following three major elements:

UML building blocks

Rules to connect the building blocks

Common mechanisms of UML

UML building blocks. The building blocks of UML can be

defined as:

- Things
- Relationships
- Diagrams

Things:

Things are the most important building blocks of UML. Things can be:

- Structural
- Behavioral
- Grouping
- Annotational

Structural things:

The Structural things define the static part of the model. They represent physical a

conceptual elements. Following are the brief descriptions of the structural things.

Class:

Class represents set of objects having similar responsibilities.

*Interface***:**

Interface defines a set of operations which specify the responsibility of a class

*Use case***:**

Use case represents a set of actions performed by a system for a specific goal.

Component:

Component describes physical part of a system.

*Node***:** A node can be defined as a physical element that exists at run time.

Behavioral things:

A behavioral thing consists of the dynamic parts of UML models. Following are the behavioral things:

Interaction:

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

State machine:

State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change.

Grouping things

Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available.

Package: Package is the only one grouping thing available for gathering structural and behavioral things.

Annotational things: Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. Note is the only one Annotational thing available.

Note: A note is used to render comments, constraints etc of an UML element.

Relationships In UML:

Relationship is another most important building block of UML. It shows how elements are associated with each other and this association describes the functionality of an application. There are four kinds of relationships available.

Dependency:

Dependency is a relationship between two things in which change in one element also affects the other one.

Association:

Association is basically a set of links that connects elements of an UML model. It also describes how many objects are taking part in that relationship.

Generalization:

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes inheritance relationship in the world of objects.

Realization:

Realization 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 case of interfaces.

UML Diagrams:

UML diagrams are the ultimate output of the entire discussion. All the elements,

relationships are used to make a complete UML diagram and the diagram represents a

system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it a complete one.

UML includes the following nine diagrams and the details are described in the following

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram

- Statechart diagram
- Deployment diagram
- Component diagram

ARCHITECTURE OF UML

Any real world system is used by different users. The users can be developers, testers, business people, analysts and many more. So before designing a system the architecture is made with different perspectives in mind. The most important part is to visualize the system from different viewer's perspective. The better we understand the better we make the system. UML plays an important role in defining different perspectives of a system. These perspectives are:

- Design
- Implementation
- Process
- Deployment

And the centre is the **Use Case view** which connects all these four. A Use case represents the functionality of the system. So the other perspectives are connected with use case.

Design of a system consists of classes, interfaces and collaboration. UML provides class diagram, object diagram to support this. Implementation defines the components assembled together to make a complete physical system. UML component diagram is used to support implementation perspective.

Process defines the flow of the system. So the same elements as used in Design are also used to support this perspective.

Deployment represents the physical nodes of the system that forms the hardware. UML deployment diagram is used to support this perspective.

Automatic Teller Machine

Description of ATM System

The software to be designed will control a simulated automated teller machine (ATM) having a magnetic stripe reader for reading an ATM card, a customer console (keyboard and display) for interaction with the customer, a slot for depositing envelopes, a dispenser for cash, a printer for printing customer receipts, and a key-operated switch to allow an operator to start or stop the machine. The ATM will communicate with the bank's computer over an appropriate communication link. (The software on the latter is not part of the requirements for this problem.)

The ATM will service one customer at a time. A customer will be required to insert an ATM card and enter a personal identification number (PIN) – both of which will be sent to the bank for validation as part of each transaction. The customer will then be able to perform one or more transactions. The card will be retained in the machine until the customer indicates that he/she desires no further transactions, at which point it will be returned – except as noted below.

The ATM must be able to provide the following services to the customer:

1. A customer must be able to make a cash withdrawal from any suitable account linked to the card. Approval must be obtained from the bank before cash is dispensed.
2. A customer must be able to make a deposit to any account linked to the card, consisting of cash and/or checks in an envelope. The customer will enter the amount of the deposit into the ATM, subject to manual verification when the envelope is removed from the machine by an operator. Approval must be obtained from the bank before physically accepting the envelope.
3. A customer must be able to make a transfer of money between any two accounts linked to the card.
4. A customer must be able to make a balance inquiry of any account linked to the card.
5. A customer must be able to abort a transaction in progress by pressing the Cancel key instead of responding to a request from the machine.

The ATM will communicate each transaction to the bank and obtain verification that it was allowed by the bank. Ordinarily, a transaction will be considered complete by the bank once it has been approved. In the case of a deposit, a second message will be sent to the bank indicating that the customer has deposited the envelope. (If the customer fails to deposit the envelope within the timeout period, or presses cancel instead, no second message will be sent to the bank and the deposit will not be credited to the customer.)

If the bank determines that the customer's PIN is invalid, the customer will be required to re-enter the PIN before a transaction can proceed. If the customer is unable to successfully enter the PIN after three tries, the card will be permanently retained by the machine, and the customer will have to contact the bank to get it back.

If a transaction fails for any reason other than an invalid PIN, the ATM will display an explanation of the problem, and will then ask the customer whether he/she wants to do another transaction.

The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account ("to" account for transfers).

The ATM will have a key-operated switch that will allow an operator to start and stop the servicing of customers. After turning the switch to the "on" position, the operator will be required to verify and enter the total cash on hand. The machine can only be turned off when it is not servicing a customer. When the switch is moved to the "off" position, the machine will shut down, so that the operator may remove deposit envelopes and reload the machine with cash, blank receipts, etc.

EXPERIMENT: 1

Name of the experiment: Class diagram for ATM System

1. AIM: To design and implement class diagram for ATM system

2. THEORY:

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application. The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.

Purpose:

The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction. The UML diagrams like activity diagram, sequence diagram can only give the sequence

flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community. So the purpose of the class diagram can be summarized as:

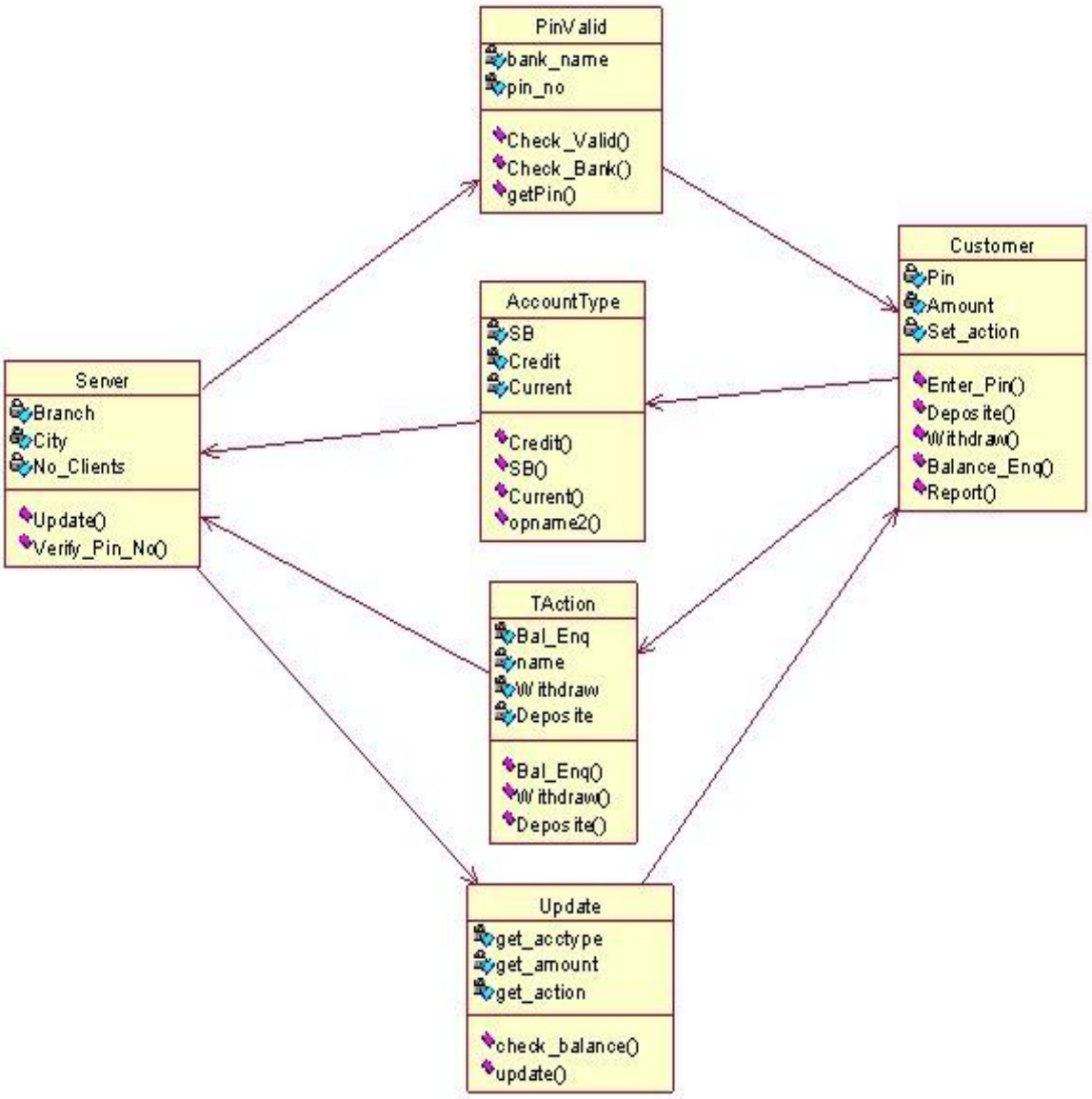
- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

Contents:

Class diagrams commonly contain the following things

- Classes
- Interfaces
- Collaborations
- Dependency, generalization and association relationships

ALGORITHM/ FLOWCHART/PSEUDO CODE:



RESULT:

Input: _____

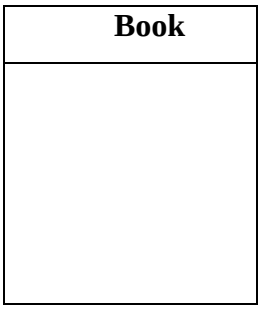
Output:_____

Inferences:

1. **understand the concept of classes**
2. **identify classes and attributes and operations for a class**
3. **model the class diagram for the system**

sample input:

Class: Book



Applications:

Online transaction

Online banking

Questions

1. What do you mean by class diagram?
2. What are properties of class diagram?
3. What are the common uses of class diagram?
4. What are the visibilities of class diagram?
5. What do you mean by classifiers?
6. What is the use of multiplicity in class diagram?
7. What is the scope of classifiers ?

8. What are the relationships used in class diagrams?
9. What are the contents of class diagram?
10. What are the standard elements in class diagram?

EXPERIMENT : 2

NAME OF EXPERIMENT: Use case diagram for ATM System.

AIM: To design and implement Use case diagram for ATM System.

THEORY: A Use case Diagram is a diagram that shows a set of Use cases and actors and their relationships. These diagrams are used to model the static use case view of a system.

To model a system the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behavior means the behaviour of the system when it is running /operating. So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem ofan application. A single use case diagram captures a particular functionality of a system. So to model the entire system numbers of use case diagrams are used.

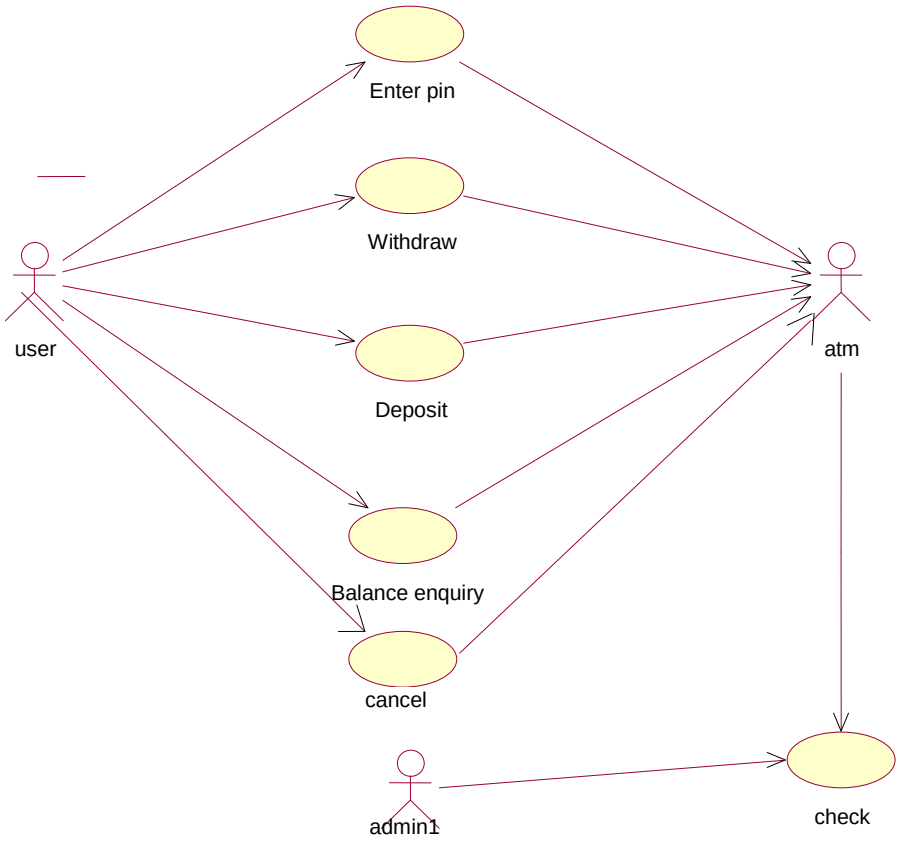
Purpose:

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose. Because other four diagrams (activity, sequence, collaboration and Statechart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

So in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.

ALGORITHM/ FLOWCHART/PSEUDO CODE:



RESULT:

Input: _____

Output:_____

Inferences:

1.Identification of use cases.

2. Identification of actors.

Sample input:

Actor:Student

Use case: 

Withdrawal Use Case

A withdrawal transaction asks the customer to choose a type of account to withdraw from (e.g. checking) from a menu of possible accounts, and to choose a dollar amount from a menu of possible amounts. The system verifies that it has sufficient money on hand to satisfy the request before sending the transaction to the bank. (If not, the customer is informed and asked to enter a different amount.) If the transaction is approved by the bank, the appropriate amount of cash is dispensed by the machine before it issues a receipt. A withdrawal transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the dollar amount.

Deposit Use Case

A deposit transaction asks the customer to choose a type of account to deposit to (e.g. checking) from a menu of possible accounts, and to type in a dollar amount on the keyboard. The transaction is initially sent to the bank to verify that the ATM can accept a deposit from

this customer to this account. If the transaction is approved, the machine accepts an envelope from the customer containing cash and/or checks before it issues a receipt. Once the envelope has been received, a second message is sent to the bank, to confirm that the bank can credit the customer's account – contingent on manual verification of the deposit envelope contents by an operator later.

A deposit transaction can be cancelled by the customer pressing the Cancel key any time prior to inserting the envelope containing the deposit. The transaction is automatically cancelled if the customer fails to insert the envelope containing the deposit within a reasonable period of time after being asked to do so.

Transfer UseCase

A transfer transaction asks the customer to choose a type of account to transfer from (e.g. checking) from a menu of possible accounts, to choose a different account to transfer to, and to type in a dollar amount on the keyboard. No further action is required once the transaction is approved by the bank before printing the receipt.

A transfer transaction can be cancelled by the customer pressing the Cancel key any time prior to entering a dollar amount.

Inquiry Use Case

An inquiry transaction asks the customer to choose a type of account to inquire about from a menu of possible accounts. No further action is required once the transaction is approved by the bank before printing the receipt. An inquiry transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the account to inquire about.

ValidateUser usecase:

This usecase is for validate the user i.e check the pin number, when the bank reports that the customer's transaction is disapproved due to an invalid PIN. The customer is required to re-enter the PIN and the original request is sent to the bank again. If the bank now approves the transaction, or disapproves it for some other reason, the original use case is continued;

otherwise the process of re-entering the PIN is repeated. Once the PIN is successfully re-entered

If the customer fails three times to enter the correct PIN, the card is permanently retained, a screen is displayed informing the customer of this and suggesting he/she contact the bank, and the entire customer session is aborted.

PrintBill usecase

This usecase is for printing corresponding bill after transactions(withdraw or deposit ,or balance enquiry, transfer) are completed.

Update Account

This usecase is for updating corresponding user accounts after transactions (withdraw or deposit or transfer) are completed.

Questions:

1. What do you mean by Use case diagram
2. What are the common modeling techniques of use case diagram
3. What do you mean by Use case .
4. What are common properties of Use case diagram
5. What are the common uses of Use case diagram
6. What are contens of Use case diagram
7. What do you mean by extend relationship.
8. What do you mean by include relationship.
9. What is the f low of events in usecases
10. What do you mean by actor and how can you represent an actor

3.INTERACTION DIAGRAMS

We have two types of interaction diagrams in UML. One is sequence diagram and the other is a collaboration diagram. The sequence diagram captures the time sequence of message flow from one object to another and the collaboration diagram describes the organization of objects in a system taking part in the message flow.

So the following things are to identified clearly before drawing the interaction diagram:

1. Objects taking part in the interaction.
2. Message flows among the objects.
3. The sequence in which the messages are flowing.
4. Object organization.

Purpose:

1. To capture dynamic behaviour of a system.
2. To describe the message flow in the system.
3. To describe structural organization of the objects.
4. To describe interaction among objects.

Contents of a Sequence Diagram

Objects

Focus of control

Messages

Life line

Contents.

Contents of a Collaboration Diagram

Objects

Links

Messages

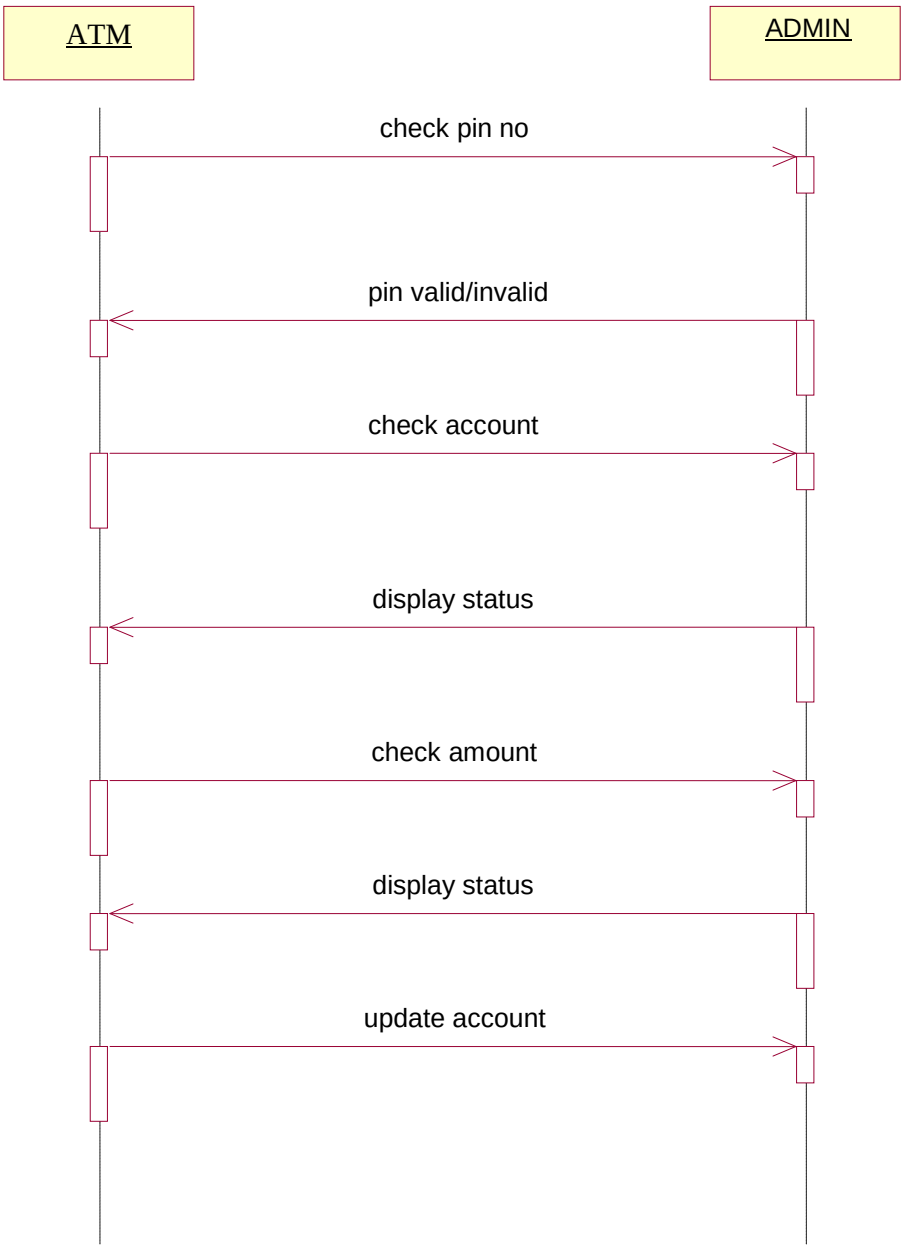
EXPERIMENT : 3

Name of the experiment: Sequence diagram for ATM System

AIM: To design and implement Sequence diagram for ATM System.

THEORY: A Sequence diagram is an interaction diagram that emphasizes the time ordering of messages.This diagram is used to show the dynamic view of a system.

ALGORITHM/ FLOWCHART/PSEUDO CODE



QUESTIONS:

11. **1.** What do you mean by sequence diagram?
12. What are the properties of Interaction diagram.?
13. What do you mean by life line?
14. What are the common modeling techniques of Interaction diagram?
15. What are the common uses of Interaction diagram?
16. What are contents of Interaction diagram?
17. What do you mean by messages?
18. What do you mean by focus of control?
19. What do you mean by stereotype?
20. What do you mean by Interaction diagram?

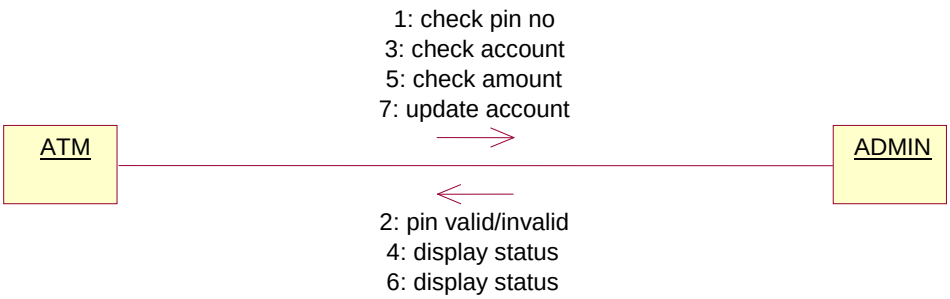
EXPERIMENT : 4

NAME OF EXPERIMENT: collaboration for ATM System.

AIM: To design and implement Use case diagram for ATM System.

THEORY: A Usecase diagram is a diagram that shows a set of use cases and actors and their relationships. These diagrams are used to model the static usecase view of a system.

ALGORITHM/ FLOWCHART/PSEUDO CODE



5. STATE CHART Diagram

Statechart diagram is used to model dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events. So Statechart diagrams are useful to model reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. So the most important purpose of Statechart diagram is to model life time of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. But the main purpose is to model reactive system.

Following are the main purposes of using Statechart diagrams:

1. To model dynamic aspect of a system.
2. To model life time of a reactive system.
3. To describe different states of an object during its life time.
4. Define a state machine to model states of an object.

Contents

Simply state and composite states

Transitions, including events and actions

Common use

They are use to model the dynamic aspects of a system.

Event ordered behavior of any kind of objects, to model reactive objects.

1. Identify the states of the objects .
2. Model the state diagram for the system.

:

EXPERIMENT : 5

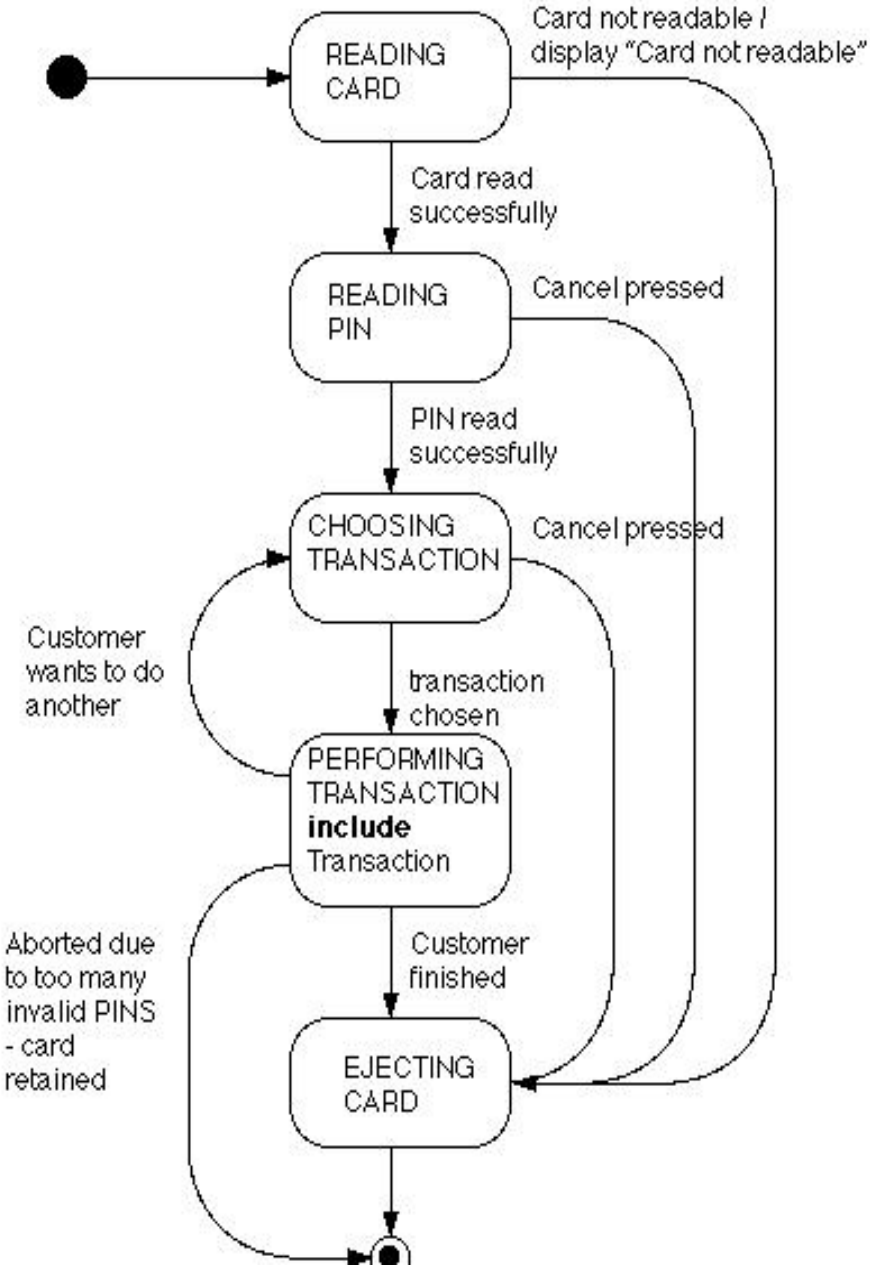
NAME OF EXPERIMENT: State chart diagram for ATM System.

AIM: To design and implement State chart diagram for ATM System.

THEORY: A statechart diagram shows a state machine,emphazing the flow of control from state to state .A state machine is a behavior that specifies the sequences of states an object goes through during its life time in response to events together with responses to those events. We can use these diagrams to model the dynamic view of a system.

ALGORITHM/ FLOWCHART/PSEUDO CODE

State-Chart for One Session



RESULT:

Input: _____

Output:_____

Inferences:

1. Identify the states of the objects .
2. Model the state diagram for the system.

QUESTIONS:

1. **1.** What do you mean by state chart diagram?
2. What do you mean by statemachine?
3. What do you mean by state?
4. What are common properties of state chart diagram?
5. What are the common uses of state chart diagram?
6. What are contens of state chart diagram?
7. What are twelve stereotypes that may be applied to dependency relationships?
8. What do you mean by dependency,genrelization,association?
9. What are the steps to model the group the elements?
10. What do you mean by composite states. ?

EXPERIMENT : 6

NAME OF EXPERIMENT: Activity diagram for ATM System.

AIM: To design and implement Activity diagram for ATM System**.**

THEORY: An activity diagram shows the flow from activity to activity .An activity is an ongoing non atomic execution within a state machine .Activities ultimately result in some action ,which is made up of executable atomic computations. we can use these diagrams to model the dynamic aspects of a system.

Activity diagram is basically a flow chart to represent the flow form one activity to another . The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow by using elements like fork, join etc.

Contents

Initial/Final State , Activity , Fork & Join , Branch , Swimlanes

Fork

A fork represents the splitting of a single flow of control into two or more concurrentFlow of control. A fork may have one incoming transition and two or more outgoing transitions, each of which represents an independent flow of control. Below fork the activities associated with each of these path continues in parallel.

Join

A join represents the synchronization of two or more concurrent flows of control. A join may have two or more incoming transition and one outgoing transition. Above the join the activities associated with each of these paths continues in parallel.

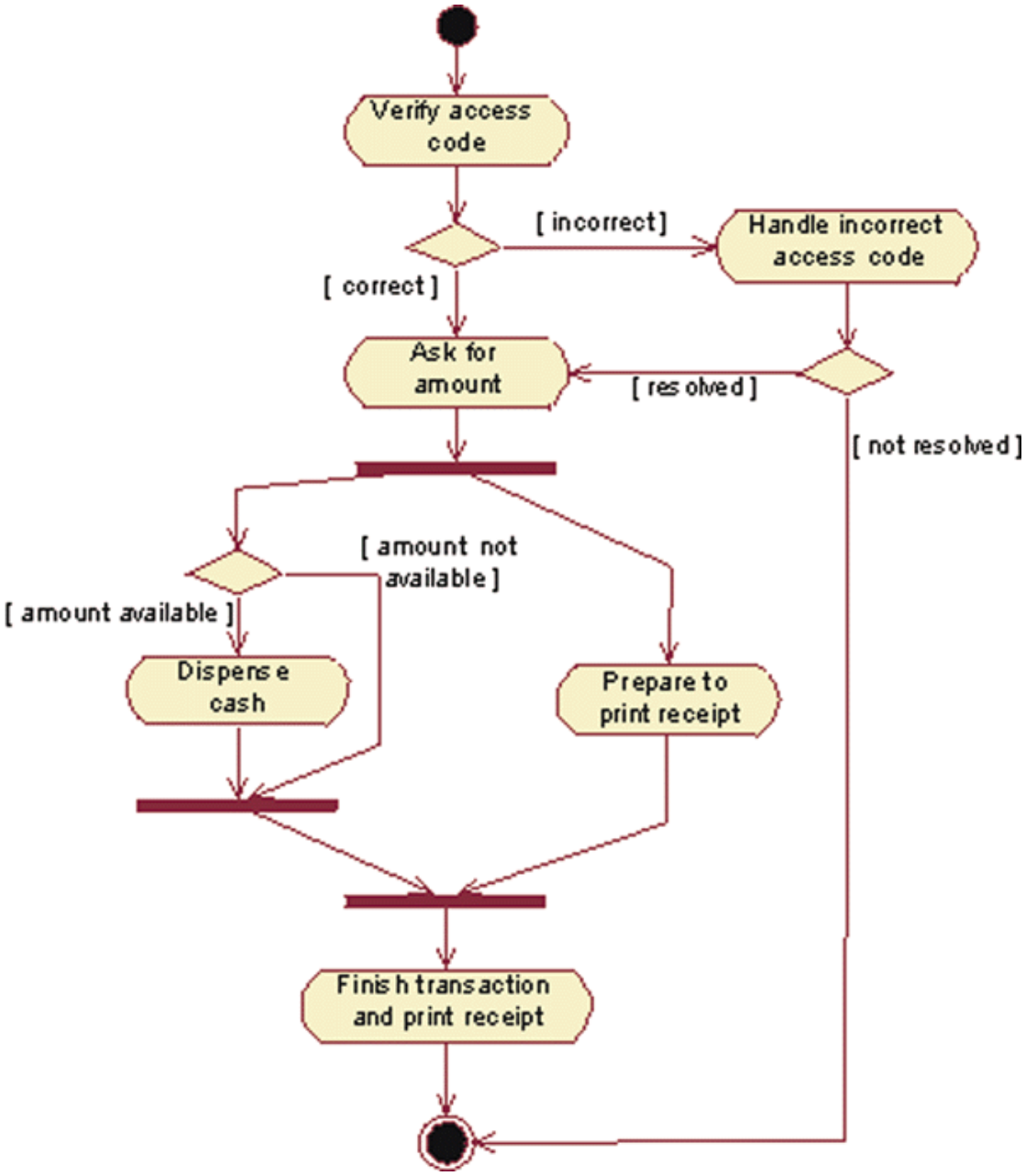
Branching

A branch specifies alternate paths takes based on some Boolean expression Branch is represented by diamond Branch may have one incoming transition and two or more outgoing one on each outgoing transition, you place a Boolean expression shouldn't overlap but they should cover all possibilities.

Swimlane:

Swimlanes are useful when we model workflows of business processes to partition the activity states on an activity diagram into groups. Each group representing the business organization responsible for those activities ,these groups are called Swimlanes .

ALGORITHM/ FLOWCHART/PSEUDO CODE


Verify access code
[incorrect]
Handle incorrect access code
[correct]
Ask for amount
[resolved]
[not resolved]
[amount not available]
[amount available]
Dispense cash
Prepare to print receipt
Finish transaction and print receipt

RESULT:

Input: _____

Output:_____

Inferences:

1. Identify the states of the objects .
2. Understand the transitions and events for various objects.
3. Model the state diagram for the system.

QUESTIONS:

1. What do you mean by state chart diagram?

2.What do you mean by state machine?

3.What do you mean by state?

4.What are common properties of state chart diagram?

11. What are the common uses of state chart diagram?

12. What are contens of state chart diagram?

13. What are twelve stereotypes that may be applied to dependency relationships?

14. What do you mean by dependency, generalization, association?

15. What are the steps to model the group the elements?

16. What do you mean by composite states. ?

EXPERIMENT : 7

NAME OF EXPERIMENT: Component diagram for ATM System.

AIM: To design and implement Component diagram for ATM System**.**

THEORY:

Component diagrams are used to model physical aspects of a system. Now the question is what are these physical aspects? Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node. So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Purpose:

Component diagrams can be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of

diagrams are used to represent the whole.

Before drawing a component diagram the following artifacts are to be identified clearly:

- Files used in the system.
- Libraries and other artifacts relevant to the application.
- Relationships among the artifacts.
- Now after identifying the artifacts the following points needs to be followed:

- Use a meaningful name to identify the component for which the diagram is to be drawn.
- Prepare a mental layout before producing using tools.
- Use notes for clarifying important points.

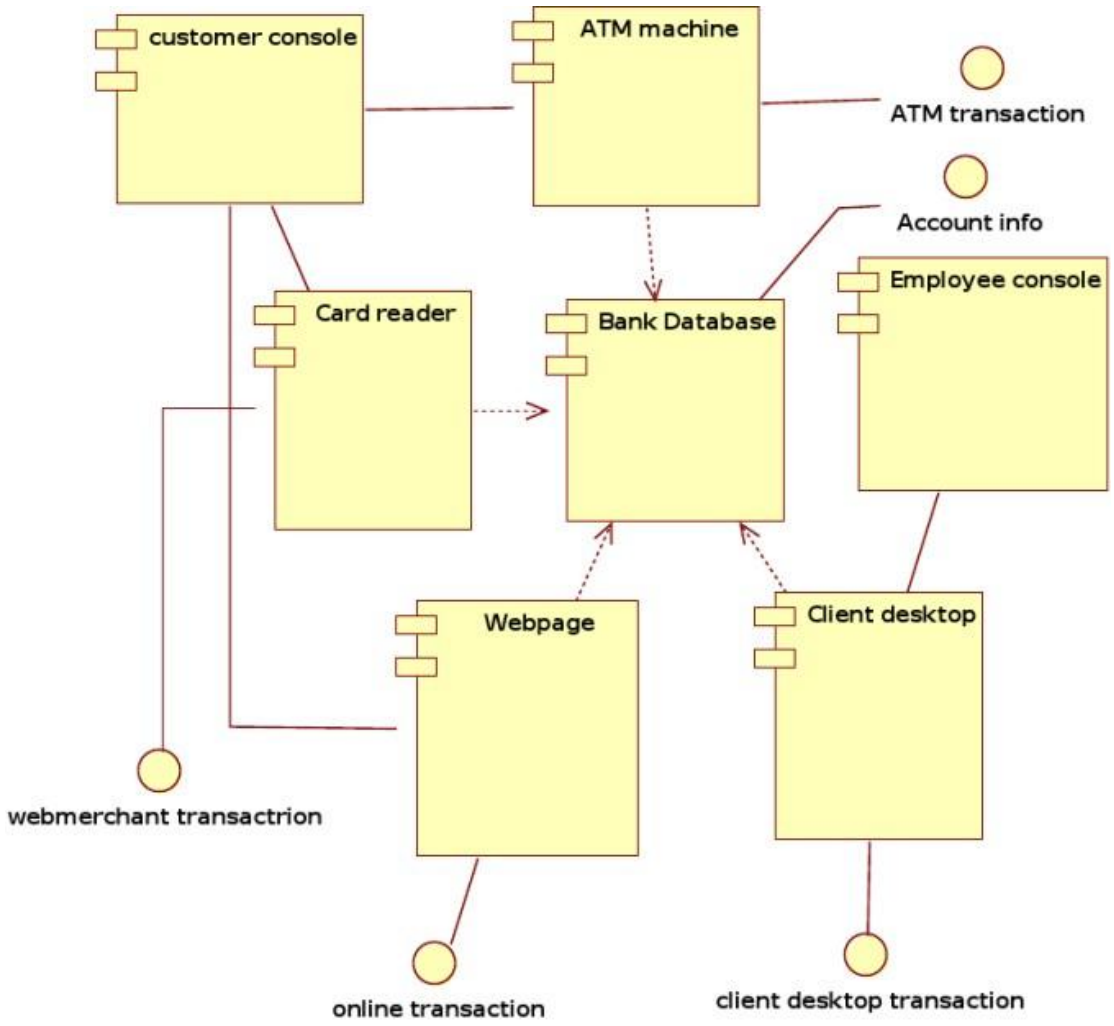
Now the usage of component diagrams can be described as:

1. Model the components of a system.
2. Model database schema.
3. Model executables of an application.
4. Model system's source code.

Contents

Components , Interfaces , Relationships

ALGORITHM/ FLOWCHART/PSEUDO CODE



RESULT:

Input: _____

Output:_____

Inferences:

1. Identify the states of the objects .
2. Model the state diagram for the system.

QUESTIONS:

1. What is a component diagram?
2. How components are used in component diagram ?
3. What are the common properties of components?
4. What are the contents of deployment diagram?

EXPERIMENT : 8

NAME OF EXPERIMENT: Deployment diagram for ATM System**.**

AIM: To design and implement Deployment diagram for ATM System.

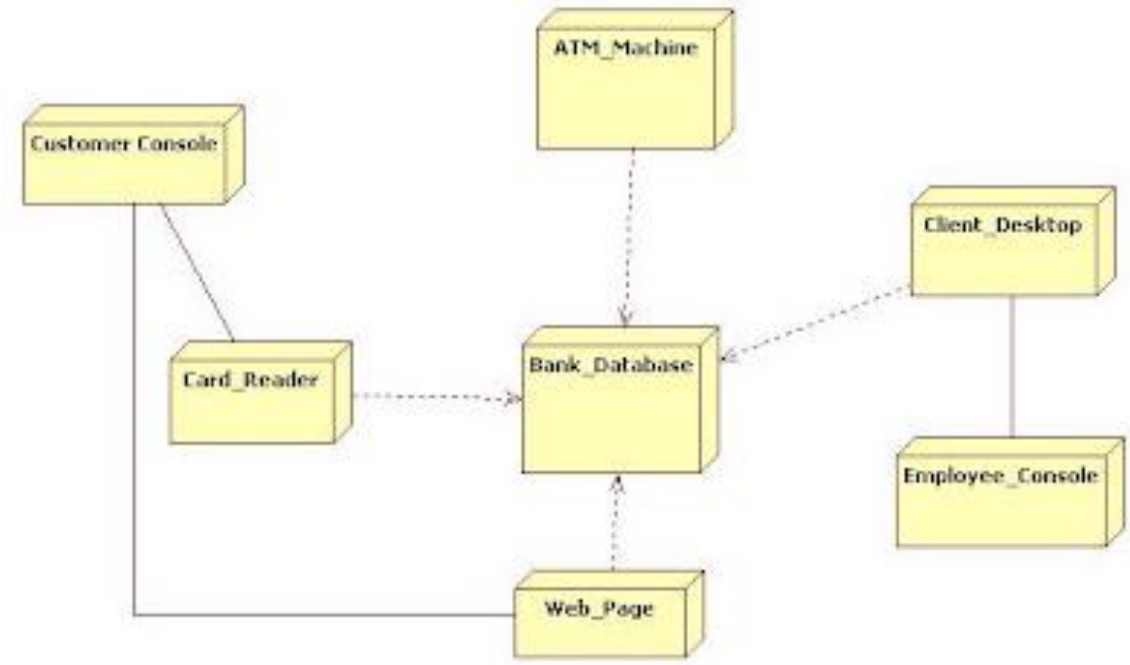
THEORY:

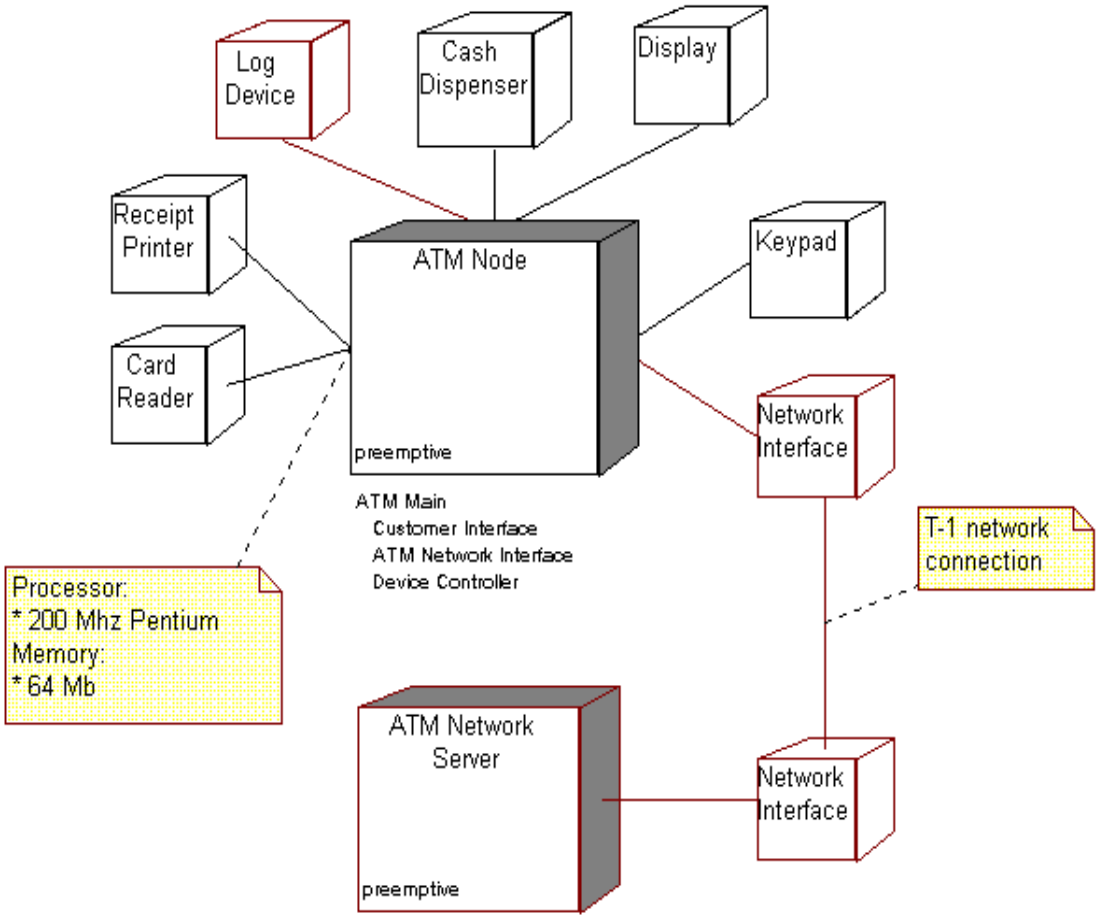
Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

Purpose:

The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related. Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.

Contents : Nodes , Dependency & Association relationships




Log Device
Cash Dispenser
Display
Receipt Printer
ATM Node
Keypad
Card Reader
Network Interface
preemptive
ATM Main
Customer Interface
ATM Network Interface
Device Controller
T-1 network connection
Processor:
* 200 Mhz Pentium
Memory:
* 64 Mb
ATM Network Server
Network Interface
preemptive

RESULT:

Input: _____

Output:_____

Inferences:

1. Identify the states of the objects .
2. Model the state diagram for the system.

QUESTIONS

5. What is a deployment diagram?
6. How components are used in deployment diagram ?
7. What are the common properties of components?
8. What are the contents of deployment diagram?
9. What are the relationships used in deployment diagram?
10. What is node?
11. How nodes are related to components?
12. What is the purpose of a note in deployment diagram?
13. Enumerate the steps for modeling deployment diagram?
14. How can you organize the nodes?

SOFTWARE TESTING

EXPERIMENT: 1

NAME OF THE EXPERIMENT: Write program in 'C' language to demonstrate the working of the following constructs

i.)do-while:

Syntax:

iteration-statement:

do statement while (expression) ;

Example:

```
#include<stdio.h>

#include<conio.h>

#include <stdio.h>

        main()
        {
         int i = 10;

           do{
            printf("Hello %d\n", i );
             i = i -1;
                    if( i == 6 )
                     {
                       break;
                     }
              }while ( i > 0 );

         }
```

ii.) while..do

```
while(condition)

{

Loop body

Increment or decrement;

   }
```

Example:

```
#include<stdio.h>

        int main()

      {
            int counter, howmuch;

            scanf("%d", &howmuch);
            counter = 0;
            while ( counter < howmuch)
            {
            counter++;
            printf("%d\n", counter);
            }
            return 0;
      }
```

iii) **if…else**

syntax:

```
if( condition 1 )
        statement1;
      else if( condition 2 )
        statement2;
      else if( condition 3 )
        statement3;
      else
        statement4;
```

example:

```
 #include<stdio.h>
int main(){
    int x,y;
    printf("Enter value for x :");
    scanf("%d",&x);
    printf("Enter value for y :");
    scanf("%d",&y);
    if ( x > y ){
        printf("X is large number - %d\n",x);
    }
    else{
        printf("Y is large number - %d\n",y);
    }
    return 0;
}
```

iv)switch

syntax:

```
 swithch(int/char const)
 {
Case const 1:stm1;
Break;
Case const 2:stmt2;
Break;
}
}
Default: stmt n;
Break;
}
```

Example:

```
#include <stdio.h>

int main() {
    int color = 1;
    printf("Please choose a color(1: red,2: green,3: blue):\n");
    scanf("%d", &color);

    switch (color) {
        case 1:
```

```
            printf("you chose red color\n");
            break;
        case 2:
            printf("you chose green color\n");
            break;
        case 3:
            printf("you chose blue color\n");
            break;
        default:
            printf("you did not choose any color\n");
    }
    return 0;
}
```

iv) for

syntax:

```
for(initialization;condition;increment/decrement)
{
//body of the loop
}
```

Example:

```
#include <stdio.h>

int main()
{
   int x;
   /* The loop goes while x < 10, and x increases by one every loop*/
   for ( x = 0; x < 10; x++ ) {
      /* Keep in mind that the loop condition checks
         the conditional statement before it loops again.
         consequently, when x equals 10 the loop breaks.
         x is updated before the condition is checked. */
      printf( "%d\n", x );
   }
   getchar();
}
```

Viva questions:

1. 1How to find entered number is EVEN or ODD without using conditional statement(not using if.. else,if.. , else if..,while, do... while...., for....)
2. Write a function to swap any two numbers?
3. if "condition"
   printf("Hello");
   else
   printf("World")

4. what should be the condition,
   so the output will be HelloWorld.
5. How can we find out prime numbers from 1 to 50?

EXPERIMENT: 2

NAME OF THE EXPERIMENT: Write a C program that uses functions to perform the following:

i) Multiplication of Two Matrices*/

```
#include<stdio.h>
#include<conio.h>
Void main()
{
        int a[10][10], b[10][10], m[10][10], I, j, p, q, r, s, k;
        Clrscr ();
        Printf("enter the size of a");
        Scanf("%d %d", &p, &q);
        Printf("Enter the size of b");
        Scanf("%d %d ", &r, &s);
        if(q=r)
        {
           Printf("Enter the elements of matrix a:\n");
           for(i=0; i<p; i++)
             {
                 for(j=0; j<q; j++)
                 {
                   Scanf("%d", &a[i][j]);
                 }
             }
           Printf("Enter the elements of matrix b:\n");
           for(i=0; i<r; i++)
            {
               for(j=0; j<s; j++)
                  {
                    Scanf("%d", &b[i][j]);
                  }
            }
               For (i=0; i<p; i++)
                  {
                     for(j=0; j<s; j++)
                       {
                            m[i][j]=0;
                            for(k=0; k<q; k++)
                            {
                            m[i][j] =m[i][j] +a[i][j] * b[k][j];
                            }
                  }
        }
        printf("matrix multiplication is:\n");
        for(i=0; i<p; i++)
         {
```

```
            for(j=0; j<s; j++)
             {
                   Printf("%d\t", m[i][j]);
             }
          ` Printf("\n");
          }
  }
Else
printf("matrix multiplication is not possible");
getch();
}
```

FAILURE CASES:

output:

1. Enter the size of a: 2 3
   Enter the size of b: 2 3

Matrix multiplication is not possible.

Reason to fail: to do multiplication of matrices the number of columns in matrix "a" should be equal to number of rows in matrix "b".

2. Enter the size of a: p q
   Enter the size of b: q s

Matrix multiplication is not possible.

Reason to fail: to do multiplication of matrices the number of columns in matrix "a" should be equal to number of rows in matrix "b", and rows & columns should be integer values.

3. Enter the size of a: 1.5 2
   Enter the size of b: 2 3

Matrix multiplication is not possible.

Reason to fail: to do multiplication of matrices the number of columns in matrix "a" should be equal to number of rows in matrix "b", and rows & columns should be integer values.

4. Enter the size of a: 350 480
   Enter the size of b: 480 620

Matrix multiplication is not possible.

Reason to fail: size of buffer will be not be sufficient to handle this multiplication.

5. Enter the size of a: -1 -2
   Enter the size of b: -2 3

Matrix multiplication is not possible.

Reason to fail: to do multiplication of matrices the number of columns in matrix "a" should be equal to number of rows in matrix "b", and rows & columns should be positive integer values.

viva questions:

2. syntax for multiplication
3. syntax for matrix multiplication
4. what the logic for matrix multiplication?

EXPERIMENT: 3

NAME OF THE EXPERIMENT: ATM system specifications and report the various bugs

Purpose:

This document describes the software requirements and specification (SRS) for an automated teller machine (ATM) network. The document is intended for the customer and the developer (designers, testers, maintainers). The reader is assumed to have basic knowledge of banking accounts and account services. Knowledge
and understanding of Unified Modeling Language (UML) diagrams is also required.

Scope:

The software supports a computerized banking network called 'Bank24'. The network enables customers to complete simple bank account services via automated teller machines (ATMs) that may be located off premise and that need not be owned and operated by the customer's bank. The ATM identifies a customer by a cash card and password. It collects information about a simple account transaction (e.g., deposit, withdrawal, transfer, bill payment), communicates the transaction information to the customer's bank, and dispenses cash to the customer. The banks provide their own software for their own computers. The 'Bank24' software requires appropriate record keeping and security provisions. The software must handle concurrent accesses to the same account correctly.

Intended Audience:

The intended audience of this SRS consists of:

- Software designers
- Systems engineers
- Software developers
- Software testers
- Customers

The actors of the system are:

1.User
2. ATM Machine
3. Bank

Product Perspective:

An automated teller machine (ATM) is a computerized telecommunications device that provides the customers of a financial institution with access to financial transactions in a public space without the need for a human clerk or bank teller. On most modern ATMs, the customer is identified by inserting a plastic ATM card with a magnetic stripe or a plastic smartcard with a chip, that contains a unique card number and some security information, such as an expiration date or CVC (CVV). Security is provided by the customer entering a personal identification number (PIN).

Product functions:

Using an ATM, customers can access their bank accounts in order to make cash withdrawals (or credit
card cash advances) and check their account balances.
The functions of the system are:
1. Login
2. Get Balance Information
3. Withdraw Cash
4. Transfer Funds

Operating Environments:

The hardware,software and technology used should have following specifications:

- Ability to read the ATM card.
- Ability to count the currency notes.
- Touch screen for convenience.
- Keypad(in case touchpad fails)
- Continuous power supply.
- Ability to connect to bank's network.
- Ability to validate user.

Design/implementation constraints:

Login:
Validate Bank Card

- Validate for Card Expiration Date
- Validate that the card's expiration date is later than today's date
- If card is expired, prompt error message "Card is expired"

Validate for Stolen or Lost Card

- Validate that the card is not reported lost or stolen
- If card is lost, prompt error message, "Card has been reported lost"
- If card is stolen, prompt error message, "Card has been reported stolen"

Validate for Disabled Card

- Validate that the card is not disabled
- If card is disabled, prompt error message, "Card has been disabled as of <expiration date>"

Validate for Locked Account

Validate that the account is not locked

- If account is locked, prompt error message "Account is locked"

Validate PIN

- Validate that the password is not blank
- If PIN is blank, prompt error message "Please provide PIN"
- Validate that the password entered matches the password on file
- If password does not match, prompt error message "Password is Incorrect"

Lock Account

- If number of consecutive unsuccessful logins exceeds three attempts, lock account

Maintain Consecutive Unsuccessful Login Counter
Increment Login Counter
For every consecutive Login attempt, increment logic counter by 1.
Reset login counter to 0 after login is successful.
Get Balance Information
Withdraw Cash
Transfer Funds

Assumptions and Dependencies:

- Hardware never fails
- ATM casing is impenetrable
- Limited number of transactions per day (sufficient paper for receipts)
- Limited amount of money withdrawn per day (sufficient money)

External Interface Requirements
User interfaces
The customer user interface should be intuitive, such that 99.9% of all new ATM users are able to
complete their banking transactions without any assistance.

Hardware interfaces

The hardware should have following specifications:

- Ability to read the ATM card
- Ability to count the currency notes
- Touch screen for convenience
- Keypad (in case touchpad fails)
- Continuous power supply
- Ability to connect to bank's network
- Ability to take input from user
- Ability to validate user

Software interfaces

The software interfaces are specific to the target banking software systems. At present, two known
banking systems will participate in the ATM network.

- State Bank
- Indian Overseas Bank

Safety requirements:

- Must be safe kept in physical aspects, say in a cabin
- Must be bolted to floor to prevent any kind of theft
- Must have an emergency phone outside the cabin
- There must be an emergency phone just outside the cabin
- The cabin door must have an ATM card swipe slot
- The cabin door will always be locked, which will open only when user swipes his/her ATM card in the slot & is validated as genuine

Security requirements:

- Users accessibility is censured in all the ways
- Users are advised to change their PIN on first use
- Users are advised not to tell their PIN to anyone
- The maximum number of attempts to enter PIN will be three

Some of the possible Bugs on ATM machine?

1.successful insertion of ATM card

2.unsuccessful operation due to insert card in wrong angle

3.unsuccessful operation due to invalid account Ex: other bank card or time expired card

4.successful entry of PIN number

5.un successful operation due to enter wrong PIN number 3times

6.successful selection of language

7.successful selection of account type

8.unsuccessful operation due to invalid account type

10.successful selection of withdraw operation

11.successful selection of amount to be withdrawl

12. successful withdraw operation

13. unsuccessful withdraw operation due to wrong denominations

14. unsuccessful withdraw operation due to amount is greater than day limit

15.unsuccessful withdraw operation due to lack of money in ATM

16. unsuccessful withdraw operation due to amount is greater than possible balance

17.unsuccessful withdraw operation due to transactions is greater than day limit

18. unsuccessful withdraw operation due to click cancel after insert card

19.unsuccessful withdraw operation due to click cancel after insert card & pin number

20.unsuccessful withdraw operation due to click cancel after insert card , pin number & language

21.unsuccessful withdraw operation due to click cancel after insert card , pin number , language &account type

22.unsuccessful withdrawl operation due to click cancel after insert card , pin number , language ,account type & withdrawl operation

23.unsuccessful withdrawl operation due to click cancel after insert card , pin number , language ,account type ,withdrawl operation &amount to be withdraw.

EXPERIMENT: 4

NAME OF THE EXPERIMENT. Test cases for Banking applications

Banking applications are considered to be one of the most complex applications in today's software development and testing industry. **What makes Banking application so complex?** What approach should be followed in order to test the complex workflows involved? In this article we will be highlighting different stages and techniques involved in testing Banking applications. **The characteristics of a Banking application are as follows:**

- Multi tier functionality to support thousands of concurrent user sessions
- Large scale Integration , typically a banking application integrates with numerous other applications such as Bill Pay utility and Trading accounts
- Complex Business workflows
- Real Time and Batch processing
- High rate of Transactions per seconds
- Secure Transactions
- Robust Reporting section to keep track of day to day transactions
- Strong Auditing to troubleshoot customer issues
- Massive storage system
- Disaster Management.

The above listed ten points are the **most important characteristics of a Banking application.**

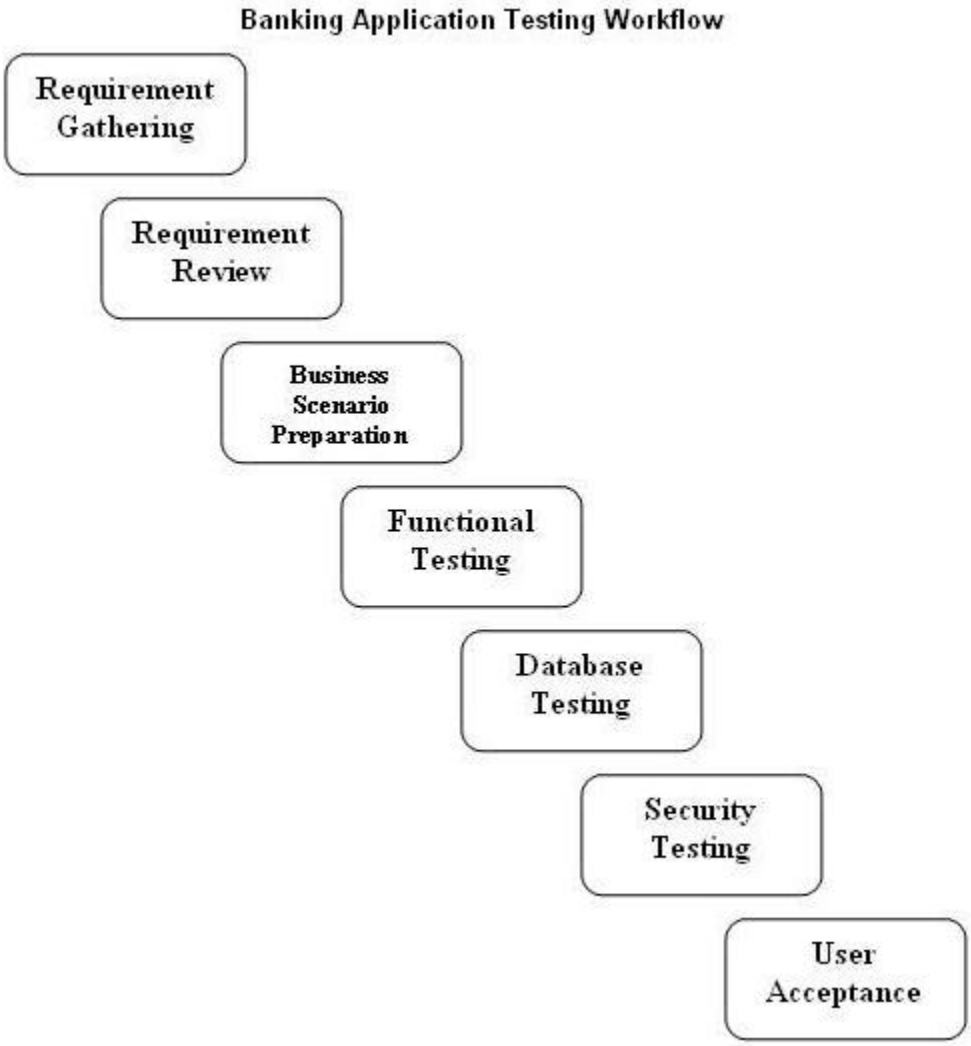
Banking applications have multiple tiers involved in performing an operation. For Example, a **banking application may have:**

1. Web Server to interact with end users via Browser
2. Middle Tier to validate the input and output for web server
3. Data Base to store data and procedures
4. Transaction Processor which could be a large capacity Mainframe or any other Legacy system to carry out Trillions of transactions per second.

If we talk about testing banking applications it requires an **end to end testing methodology involving multiple software testing techniques to ensure:**

- Total coverage of all banking workflows and Business Requirements
- Functional aspect of the application
- Security aspect of the application
- Data Integrity
- Concurrency
- User Experience

Typical stages involved in testing Banking Applications are shown in below workflow which we will be discussing individually.



1) Requirement Gathering:

Requirement gathering phase involves documentation of requirements either as Functional Specifications or Use Cases. Requirements are gathered as per customer needs and documented by Banking Experts or Business Analyst. To write requirements on more than one subject experts are involved as banking itself has multiple sub domains and one full fledge banking application will be the integration of all. For Example: A banking application may have separate modules for Transfers, Credit Cards, Reports, Loan Accounts, Bill Payments, Trading Etc.

2) Requirement Review:

The deliverable of Requirement Gathering is reviewed by all the stakeholders such as QA Engineers, Development leads and Peer Business Analysts. They cross check that neither existing business workflows nor new workflows are violated.

3) Business Scenario Preparations:

In this stage QA Engineers derive Business Scenarios from the requirement documents (Functions Specs or Use Cases); Business Scenarios are derived in such a way that all Business Requirements are covered. Business Scenarios are high level scenarios without any detailed steps, further these Business Scenarios are reviewed by Business Analyst to ensure all of Business Requirements are met and **its easier for BAs to review high level scenarios than reviewing low level detailed Test Cases.**

4) Functional Testing:

In this stage functional testing is performed and the usual software testing activities are performed such as:

Test Case Preparation:
In this stage Test Cases are derived from Business Scenarios, one Business Scenario leads to several positive test cases and negative test cases. Generally tools used during this stage are Microsoft Excel, Test Director or Quality Center.
Test Case Review:
Reviews by peer QA Engineers
Test Case Execution:
Test Case Execution could be either manual or automatic involving tools like QC, QTP or any other.

5) **Database Testing:**

Banking Application involves complex transaction which are performed both at UI level and Database level, Therefore Database testing is as important as functional testing. Database in itself is an entirely separate layer hence it is carried out by database specialists and it uses techniques like

- Data loading
- Database Migration
- Testing DB Schema and Data types
- Rules Testing
- Testing Stored Procedures and Functions

- Testing Triggers
- Data Integrity

6) Security Testing:

Security Testing is usually the last stage in the testing cycle as completing functional and non functional are entry criteria to commence Security testing. Security testing is one of the major stages in the entire Application testing cycle as this stage ensures that application complies with Federal and Industry standards. Security testing cycle makes sure the application does not have any web vulnerability which may expose sensitive data to an intruder or an attacker and complies with standards like OWASP.

In this stage the major task involves in the whole application scan which is carried out using tools like IBM Appscan or HP WebInspect (2 Most popular tools).

Once the Scan is complete the Scan Report is published out of which False Positives are filtered out and rest of the vulnerability are reported to Development team for fixing depending on the Severity.

Other **Manual tools for Security Testing** used are: Paros Proxy, Http Watch, Burp Suite, Fortify tools Etc.

Apart from the above stages there might be different stages involved like Integration Testing and Performance Testing.

In today's scenario **majority of Banking Projects are using**: Agile/Scrum, RUP and Continuous Integration methodologies, and Tools packages like Microsoft's VSTS and Rational Tools.

As we mentioned RUP above, RUP stands for Rational Unified Process, which is an iterative software development methodology introduced by IBM which comprises of four phases in which development and testing activities are carried out.

Four phases are:

i) Inception

ii) Collaboration

iii) Construction and

iv) Transition

RUP widely involves IBM Rational tools.

In this article we discussed **how complex a Banking application could be** and what are the **typical phases involved in testing the application**. Apart from that we also discussed current trends followed by IT industries including software development methodologies and tools.

Testcases for opening bank account

1. Input parameters checking

-Name
-Date of Birth
-Photo
-Address Proof
-Identity proof
-Introducers (if applicable)
-PAN card
-Initial deposit
-Whether checkbook / ATM card / Online banking facilities are needed or not
-Customer signature

Type of account

-Savings account
-Salary account
-Joint account
-Current account
-Secondary account
-RD account
-Account for a company

Test cases

-Checking mandatory input parameters
-Checking optional input parameters
-Check whether able to create account entity.
-Check whether you are able to deposit an amount in the newly created account (and thus updating the balance)
-Check whether you are able to withdraw an amount in the newly created account (after deposit) (and thus updating the balance)

-Check whether company name and its pan number and other details are provided in case of salary account
-Check whether primary account number is provided in case of secondary account
-Check whether company details are provided in cases of company's current account
-Check whether proofs for joint account is provided in case of joint account
-Check whether you are able deposit an account in the name of either of the person in an joint account.
-Check whether you are able withdraw an account in the name of either of the person in an joint account.
-Check whether you are able to maintain zero balance in salary account
-Check whether you are not able to maintain zero balance (or mini balance) in non-salary account.

viva questions

1. Can you explain boundary value analysis?

2. Can you explain equivalence partitioning?

3. Can you explain random/monkey testing?

4. What are semi-random test cases?

5.What is negative and positive testing?

6.How did you define severity ratings in your project?

EXPERIMENT: 5

NAME OF THE EXPERIMENT: Test plan document for library application

The Library Management System is an online application for assisting a librarian in managing a book library in a University. The system would provide basic set of features to add/update clients, add/update books, search for books, and manage check-in / checkout processes. Our test group tested the system based on the requirement specification.

INTRODUCTION

This test report is the result for testing in the LMS. It mainly focuses on two problems: what we will test and how we will test.

Result

GUI test

Pass criteria: librarians could use this GUI to interface with the backend library database without any difficulties

Result: pass

Database test

Pass criteria: Results of all basic and advanced operations are normal (refer to section 4)

Result: pass

Basic function test

Add a student

Pass criteria:

- Each customer/student should have following attributes: Student ID/SSN (unique), Name, Address and Phone number.

  Result: pass

- The retrieved customer information by viewing customer detail should contain the four attributes.

  Result: pass

Update/delete student

Pass criteria:

- The record would be selected using the student ID

  Result: pass

- Updates can be made on full. Items only: Name, Address, Phone number

  Result: pass

- The record can be deleted if there are no books issued by user.

  Result: Partially pass. When no books issued by user, he can be deleted. But when there are books

  Issued by this user, he was also deleted. It is wrong.

- The updated values would be reflected if the same customer's ID/SSN is called for.

  Result: pass

- If customer were deleted, it would not appear in further search queries.

  Result: pass

Add a book

Pass criteria:

- Each book shall have following attributes: Call Number, ISBN, Title, Author name.

  Result: pass

- The retrieved book information should contain the four attributes.

  Result: pass

Update/delete book

Pass criteria:

- The book item can be retrieved using the call number

  Result: did not pass. Can not retrive using the call number

- The data items which can be updated are: ISBN, Title, Author name

  Result: pass

- The book can be deleted only if no user has issued it.

  Result: partially pass. When no user has issued it, pass. When there are user having issued it,

  did not pass

- The updated values would be reflected if the same call number is called for

  Result: pass

- If book were deleted, it would not appear in further search queries.

  Result: pass

Search for book

Pass criteria:

- The product shall let Librarian query books' detail information by their ISBN number or Author or Title.

  Result: pass

- The search results would produce a list of books, which match the search parameters with following Details: Call number, ISBN number, Title, Author

  Result: pass

- The display would also provide the number of copies which is available for issue

  Result: pass

- The display shall provide a means to select one or more rows to a user-list

  Result: pass

- A detailed view of each book should provide information about check-in/check out status, with the borrower's information.

  Result: pass

- The search display will be restricted to 20 results per page and there would be means to navigate from sets of search results.

  Result: pass

- The user can perform multiple searches before finally selecting a set of books for check in or checkout. These should be stored across searches.

  Result: pass

- A book may have more than one copy. But every copy with the same ISBN number should have same detail information.

  Result: pass

- The borrower's list should agree with the data in students' account

Result: pass

Check-in book

Pass criteria:

- Librarians can check in a book using its call number

  Result: pass

- The check-in can be initiated from a previous search operation where user has selected a set of books.

  Result: pass

- The return date would automatically reflect the current system date.

  Result: did not pass.

- Any late fees would be computed as difference between due date and return date at rate of 10 cents a day.

  Result: did not pass

- A book, which has been checked in once, should not be checked in again

  Result: pass

Check-out book

Pass criteria:

- Librarians can check out a book using its call number

  Result: pass

- The checkout can be initiated from a previous search operation where user has selected a set of books.

  Result: pass

- The student ID who is issuing the book would be entered

  Result: pass

- The issue date would automatically reflect the current system date.

  Result: did not pass

- The due date would automatically be stamped as 5 days from current date.

  Result: did not pass

- A book, which has been checked out once, should not be checked out again

  Result: pass

- A student who has books due should not be allowed to check out any books

  Result: did not pass

- The max. No of books that can be issued to a customer would be 10. The system should not allow checkout of books beyond this limit.

  Result: pass

View book detail

Pass criteria:

- This view would display details about a selected book from search operation

  Result: pass

- The details to be displayed are: Call number, IBN, Title, Author, Issue status (In library or checked out), If book is checked out it would display, User ID & Name, Checkout date, Due date

  Result: for checkout date and due date, did not pass

- Books checked in should not display user summary

  Result: pass

- Books checked out should display correct user details.

  Result: pass

View student detail

Pass criteria:

- Librarians can select a user record for detailed view

  Result: pass

- The detail view should show:

  a.User name, ID, Address & Phone number

  Result: pass

  b. The books issued by user with issue date, due date, call number, title

  Result: did not pass

  c. Late fees & Fines summary and total

Result: did not pass

- The display should match existing user profile

  Result: pass

- The books checked out should have their statuses marked

  Result: pass

- The book search query should show the user id correctly.

  Result: pass

Network test

Pass criteria: Results of operations (ping, ftp and ODBC connectivity check) are normal

Result: did not test this item, because no enough machines and no available envirenment.

viva questions

1. how to Create a test plan document for Library Management System?

2. what is objectrepositry

3. How many test cases can u write 1) File - open dialog box in notepad please write 5 if software failed in customer environment what we called a)error b)fault c)defect d)failure

4. What test plan should contains?

5. What is test strategy?

6. Define test Plan? What is the difference between Master Test Plan and Test Plan?

EXPERIMENT: 6

NAME OF THE EXPERIMENT: Study of Any Testing Tool(WinRunner)

WinRunner is a program that is responsible for the automated testing of software. WinRunner is a Mercury Interactive's enterprise functional testing tool for Microsoft windows applications.

Importance of Automated Testing:

1. Reduced testing time
2. Consistent test procedures – ensure process repeatability and resource independence. Eliminates errors of manual testing
3. Reduces QA cost – Upfront cost of automated testing is easily recovered over the lifetime of the product
4. Improved testing productivity – test suites can be run earlier and more often
5. Proof of adequate testing
6. For doing Tedious work – test team members can focus on quality areas.

WinRunner Uses:

1. With WinRunner sophisticated automated tests can be created and run on an application.
2. A series of wizards will be provided to the user, and these wizards can create tests in an automated manner.
3. Another impressive aspect of WinRunner is the ability to record various interactions, and transform them into scripts. WinRunner is designed for testing graphical user interfaces.
4. When the user make an interaction with the GUI, this interaction can be recorded. Recording the interactions allows to determine various bugs that need to be fixed.
5. When the test is completed, WinRunner will provide with detailed information regarding the results. It will show the errors that were found, and it will also give

important information about them. The good news about these tests is that they can be reused many times.

6. WinRunner will test the computer program in a way that is very similar to normal user interactions. This is important, because it ensures a high level of accuracy and realism. Even if an engineer is not physically present, the Recover manager will troubleshoot any problems that may occur, and this will allow the tests to be completed without errors.
7. The Recover Manager is a powerful tool that can assist users with various scenarios. This is important, especially when important data needs to be recovered.

The goal of WinRunner is to make sure business processes are properly carried out. WinRunner uses TSL, or Test Script Language.

WinRunner Testing Modes

Context Sensitive

Context Sensitive mode records your actions on the application being tested in terms of the GUI objects you select (such as windows, lists, and buttons), while ignoring the physical location of the object on the screen. Every time you perform an operation on the application being tested, a TSL statement describing the object selected and the action performed is generated in the test script. As you record, WinRunner writes a unique description of each selected object to a GUI map.

The GUI map consists of files maintained separately from your test scripts. If the user interface of your application changes, you have to update only the GUI map, instead of hundreds of tests. This allows you to easily reuse your Context Sensitive test scripts on future versions of your application.

To run a test, you simply play back the test script. WinRunner emulates a user by moving the mouse pointer over your application, selecting objects, and entering keyboard input. WinRunner reads the object descriptions in the GUI map and then searches in the application being tested for objects matching these descriptions. It can locate objects in a window even if their placement has changed.

Analog

Analog mode records mouse clicks, keyboard input, and the exact x- and y-coordinates traveled by the mouse. When the test is run, WinRunner retraces the mouse tracks. Use Analog mode when exact mouse coordinates are important to your test, such as when testing a drawing application.

The WinRunner Testing Process

Testing with *WinRunner* involves six main stages:

1. Create the GUI Map

The first stage is to create the GUI map so WinRunner can recognize the GUI objects in the application being tested. Use the RapidTest Script wizard to review the user interface of your application and systematically add descriptions of every GUI object to the GUI map. Alternatively, you can add descriptions of individual objects to the GUI map by clicking objects while recording a test.

2. Create Tests

Next is creation of test scripts by recording, programming, or a combination

of both. While recording tests, insert checkpoints where we want to check the response of the application being tested. We can insert checkpoints that check GUI objects, bitmaps, and databases. During this process, WinRunner captures data and saves it as *expected results*—the expected response of the application being tested.

3. Debug Tests

Rrun tests in Debug mode to make sure they run smoothly. One can set breakpoints, monitor variables, and control how tests are run to identify and isolate defects. Test results are saved in the debug folder, which can be discarded once debugging is finished.

When WinRunner runs a test, it checks each script line for basic syntax errors, like incorrect syntax or missing elements in **If**, **While**, **Switch**, and **For** statements. We can use the **Syntax**

Check options (**Tools >Syntax Check**) to check for these types of syntax errors before running your test.

4. Run Tests

Tests can be run in Verify mode to test the application. Each time WinRunner

encounters a checkpoint in the test script, it compares the current data of the application being tested to the expected data captured earlier. If any mismatches are found, WinRunner captures them as *actual results*.

5. View Results

Following each test run, WinRunner displays the results in a report. The report details all the major events that occurred during the run, such as checkpoints, error messages, system messages, or user messages.

If mismatches are detected at checkpoints during the test run, we can view the expected results and the actual results from the Test Results window. In cases of bitmap mismatches, one can also view a bitmap that displays only the difference between the expected and actual results.

We can view results in the standard WinRunner report view or in the Unified report view. The WinRunner report view displays the test results in a Windows-style viewer. The Unified report view displays the results in an HTML-style viewer (identical to the style used for QuickTest Professional test results).

6. Report Defects

If a test run fails due to a defect in the application being tested, one can report information about the defect directly from the Test Results window.

This information is sent via e-mail to the quality assurance manager, who tracks the defect until it is fixed.

Using Winrunner Window

Before you begin creating tests, you should familiarize yourself with the WinRunner main window.

1.4.1. To start WinRunner:

Choose **Programs** > **WinRunner** > **WinRunner** on the **Start** menu.

The first time you start WinRunner, the Welcome to WinRunner window and the "What's New in WinRunner" help open. From the Welcome window you can create a new test, open an existing test, or view an overview of WinRunner in your default browser.

If you do not want this window to appear the next time you start WinRunner, clear the **Show on Startup** check box. To show the **Welcome to WinRunner** window upon startup from within WinRunner, choose **Settings > General Options**, click the **Environment** tab, and select the **Show Welcome screen** check box.

1.4.2. The Main WinRunner Window

The main WinRunner window contains the following key elements:

- *WinRunner title bar*
- *Menu bar,* with drop-down menus of WinRunner commands
- *Standard toolbar,* with buttons of commands commonly used when running a test
- *User toolbar,* with commands commonly used while creating a test
- *Status bar,* with information on the current command, the line number of the insertion point and the name of the current results folder

The *Standard toolbar* provides easy access to frequently performed tasks, such as opening, executing, and saving tests, and viewing test results.

Standard Toolbar

The *User toolbar* displays the tools you frequently use to create test scripts. By default, the User toolbar is hidden. To display the User toolbar, choose **Window** > **User Toolbar**. When you create tests, you can minimize the WinRunner window and work exclusively from the toolbar.

The User toolbar is customizable. You choose to add or remove buttons using the **Settings > Customize User Toolbar** menu option. When you re-open WinRunner, the User toolbar appears as it was when you last closed it.

The commands on the Standard toolbar and the User toolbar are described in detail in subsequent lessons.

Note that you can also execute many commands using *softkeys*. Softkeys are keyboard shortcuts for carrying out menu commands. You can configure the softkey combinations for your keyboard using the Softkey Configuration utility in your WinRunner program group. For more information, see the "WinRunner at a Glance" chapter in your *WinRunner User's Guide*.

Now that you are familiar with the main WinRunner window, take a few minutes to explore these window components before proceeding to the next lesson.

The Test Window

You create and run WinRunner tests in the test window. It contains the following

key elements:

- *Test window title bar*, with the name of the open test
- *Test script*, with statements generated by recording and/or programming in TSL, Mercury Interactive's Test Script Language
- *Execution arrow*, which indicates the line of the test script being executed during a test run, or the line that will next run if you select the Run from arrow option
- *Insertion point*, which indicates where you can insert or edit text

viva questions

1.What is the purpose of set_window command?

2. Why don't we normally load the GUI maps through start up scripts?

3. How do you unload the GUI map?

4. How do you configure GUI map?

5.What is the purpose of GUI spy?

6.What are the virtual objects and how do you learn them?

7.What is the use of Virtual Object Wizard and how it is used?

EXPERIMENT: 7

NAME OF THE EXPERIMENT: Study of any web testing tool (e.g. Selenium)

7.1What is Selenium?

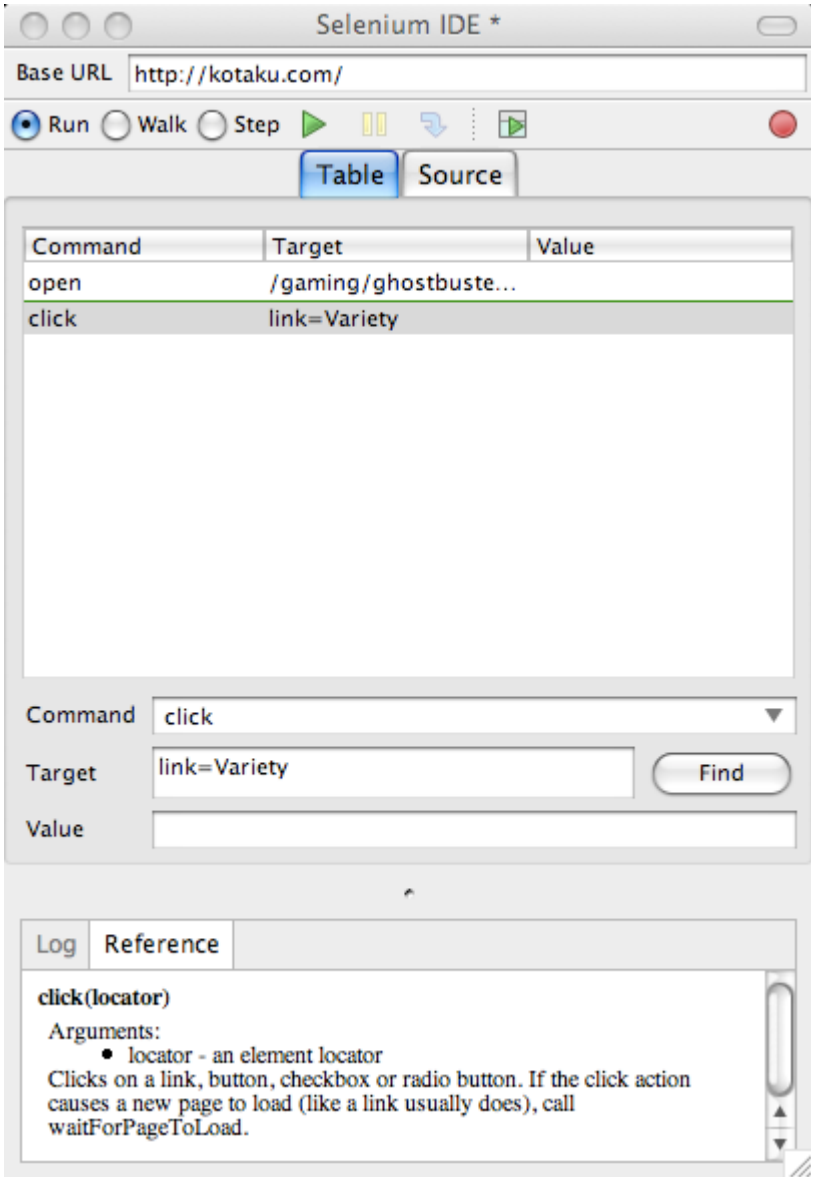
Javascript framework that runs in your webbrowser Works anywhere Javascript is supported Hooks for many other languages Java, Ruby, Python Can simulate a user navigating through pages and then assert for specific marks on the pages All you need to really know is HTML to start using it right away

Where to get it?

You can use Selenium-Core and customize everything But it is easier to just get a Firefox plug-in "Selenium-IDE" that helps you "record" test cases You can record how an app is being used and then play back those recordings followed by asserts Get everything at: www.openqa.org/selenium/

7.2 Selenium IDE

The root of web application you want to test The list of actions in the actual test case to execute The log of the events that were executed, including any errors or warning that may have occurred

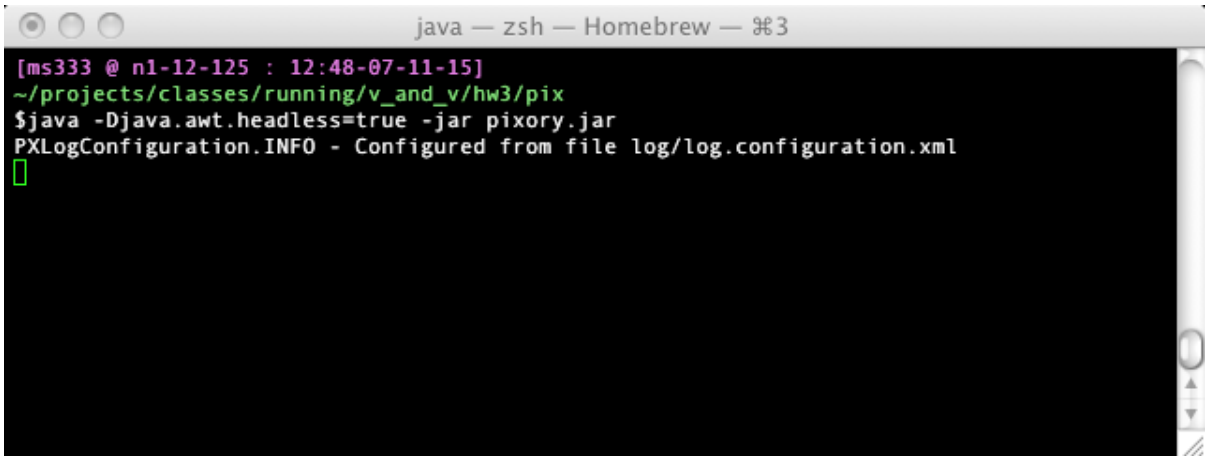


Execution Commands Try the test in the Web based TestRunner Reference of the currently selected command Record test actions Specify commands, including asserts

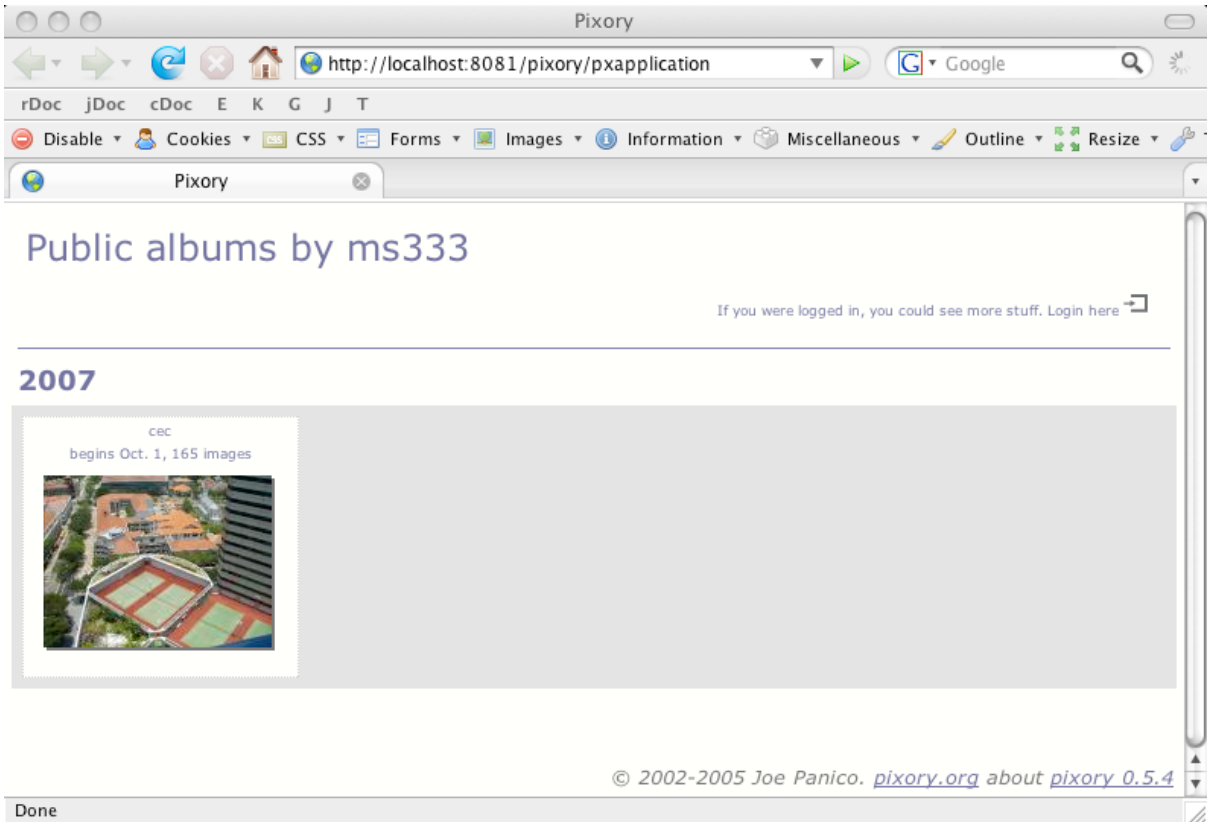
Test Creation Demo

• Create test case to log into the gallery

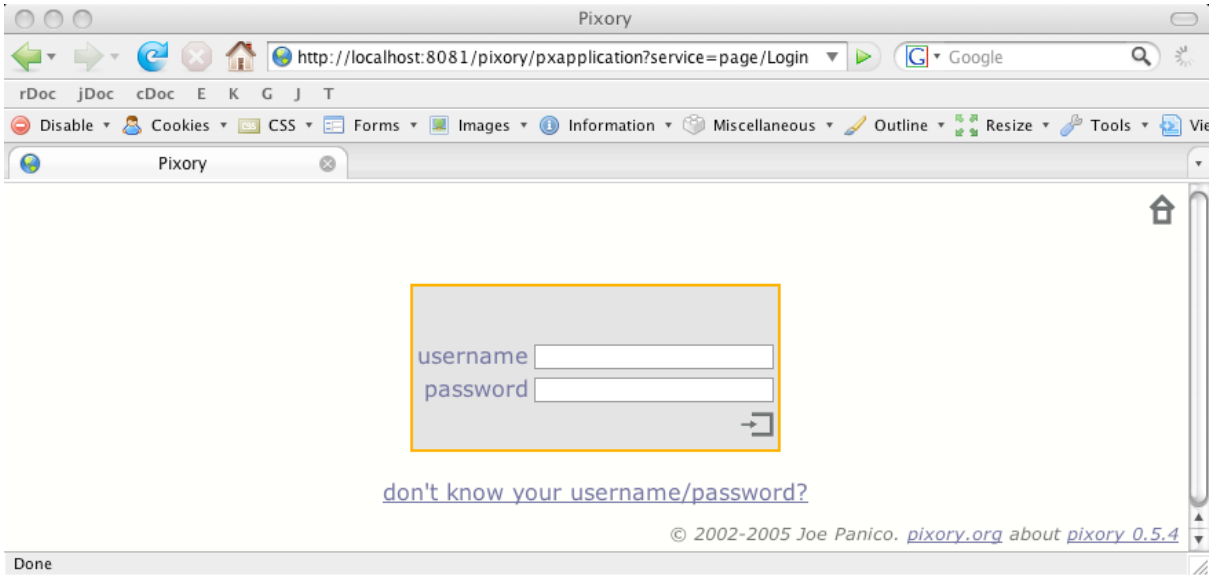
• Create test case to log out of the gallery



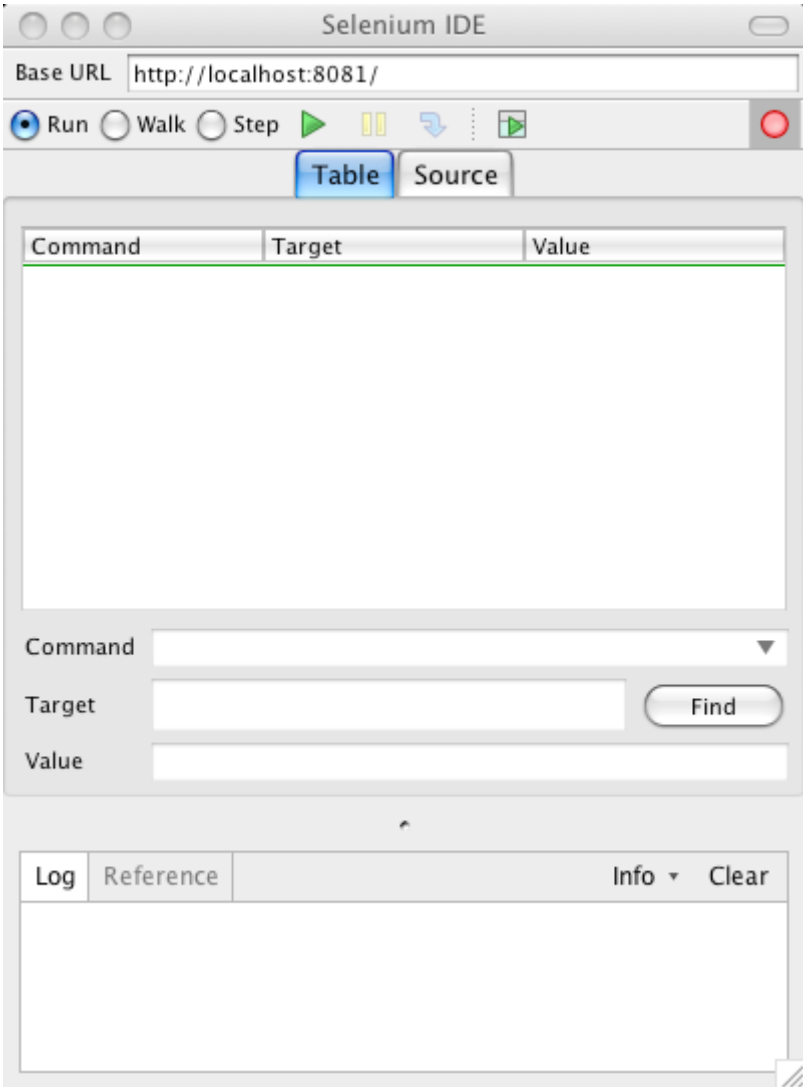
Start pixory



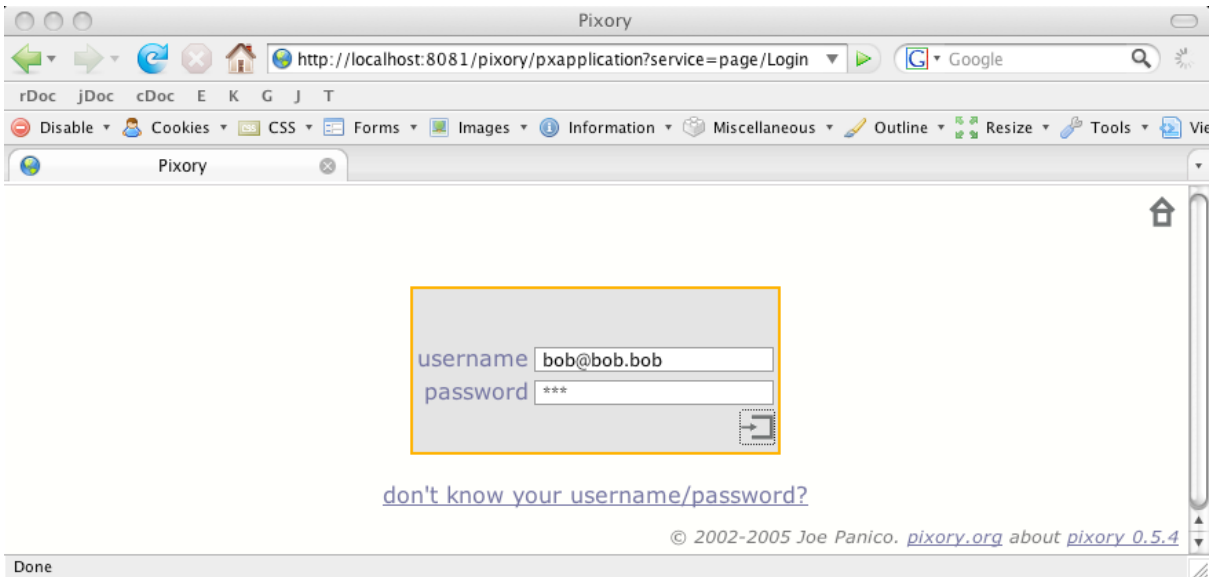
Connect to the server



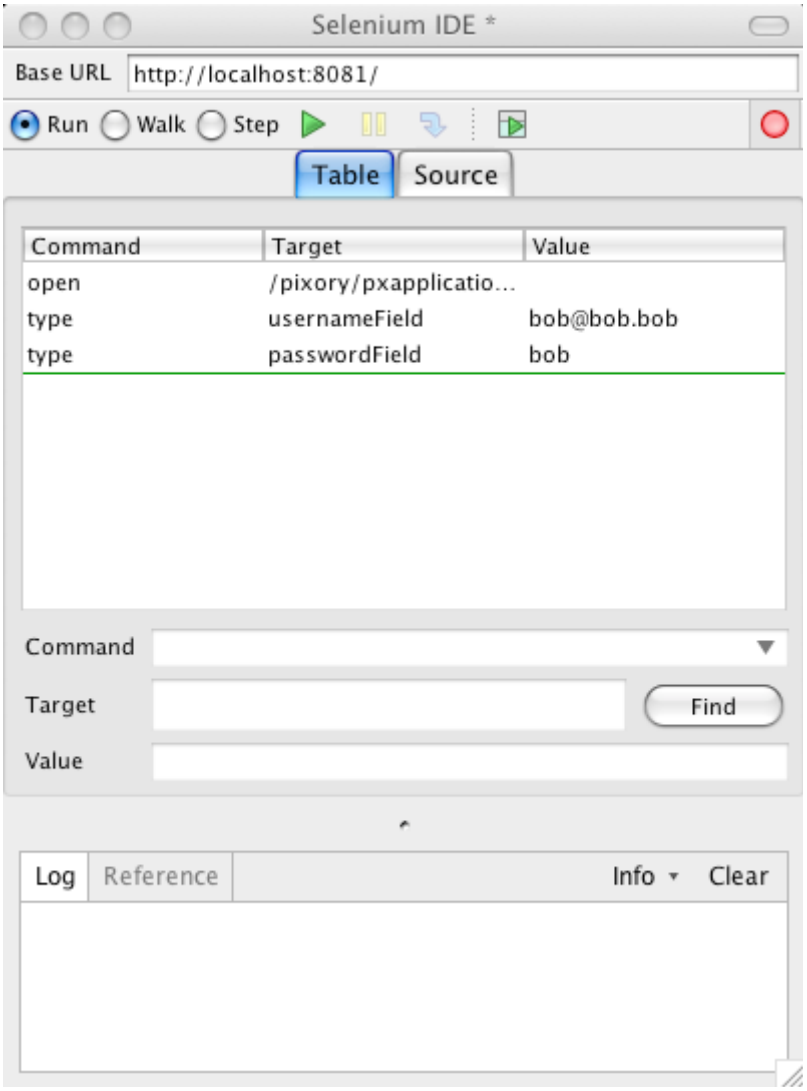
Go to the Login Screen



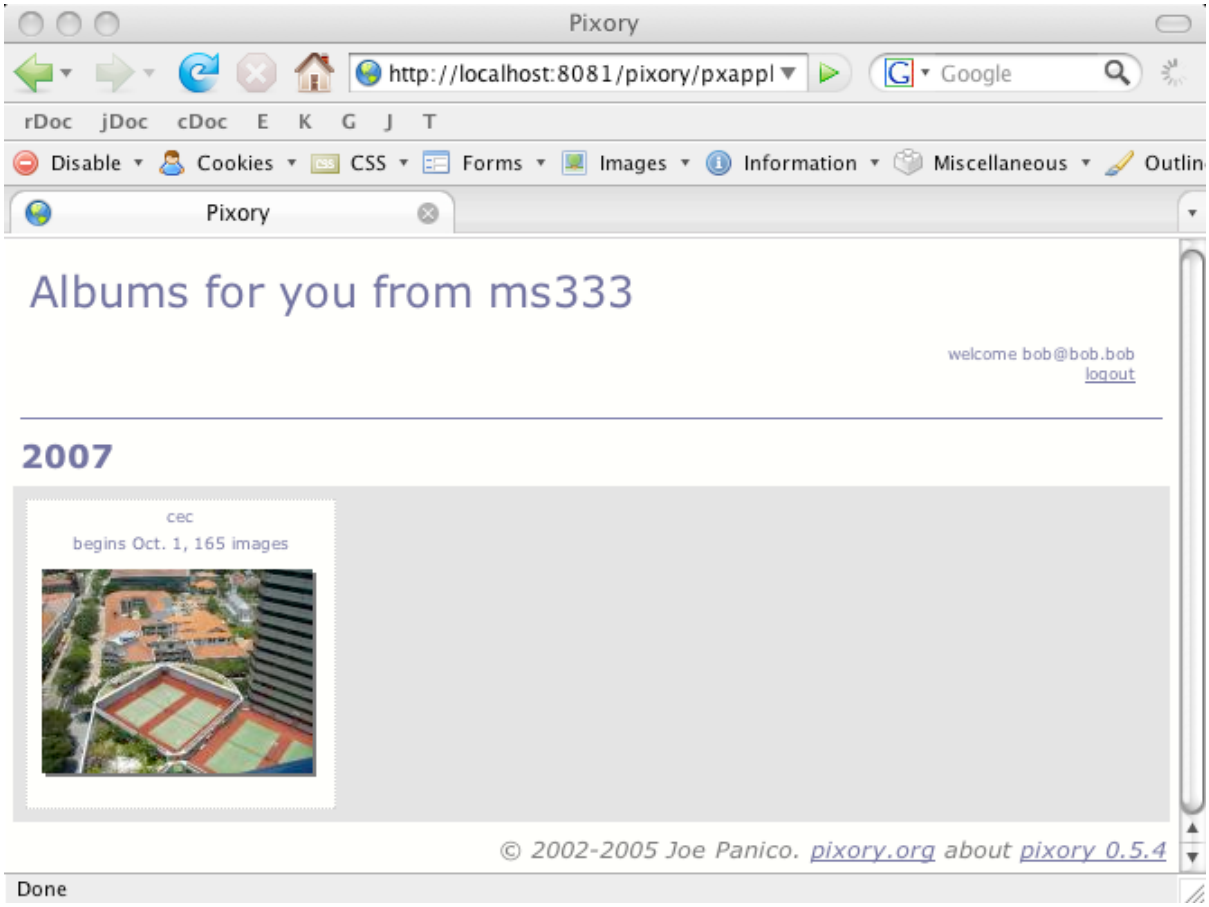
Hit the Record Button



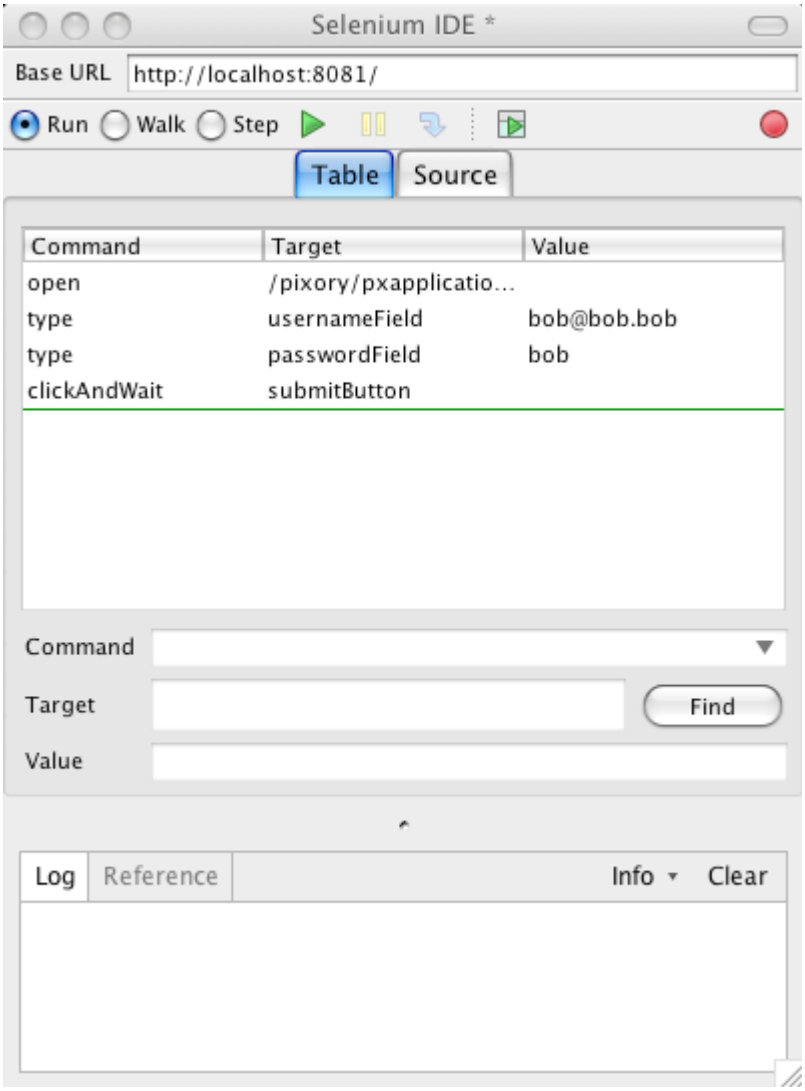
Type in Username and Password



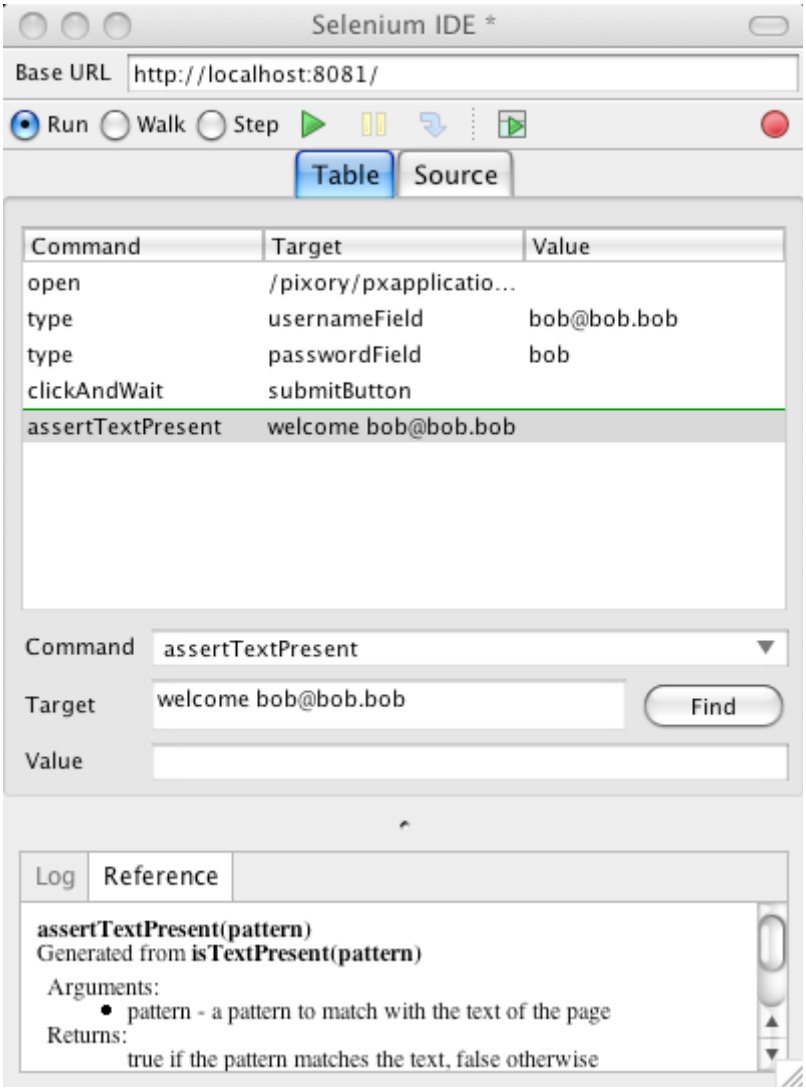
IDE should update


Pixory
http://localhost:8081/pixory/pxappl
Google
rDoc jDoc cDoc E K G J T
Disable Cookies CSS Forms Images Information Miscellaneous
Pixory
Albums for you from ms333
welcome bob@bob.bob
logout
2007
cec
begins Oct. 1, 165 images
© 2002-2005 Joe Panico. pixory.org about pixory 0.5.4
Done

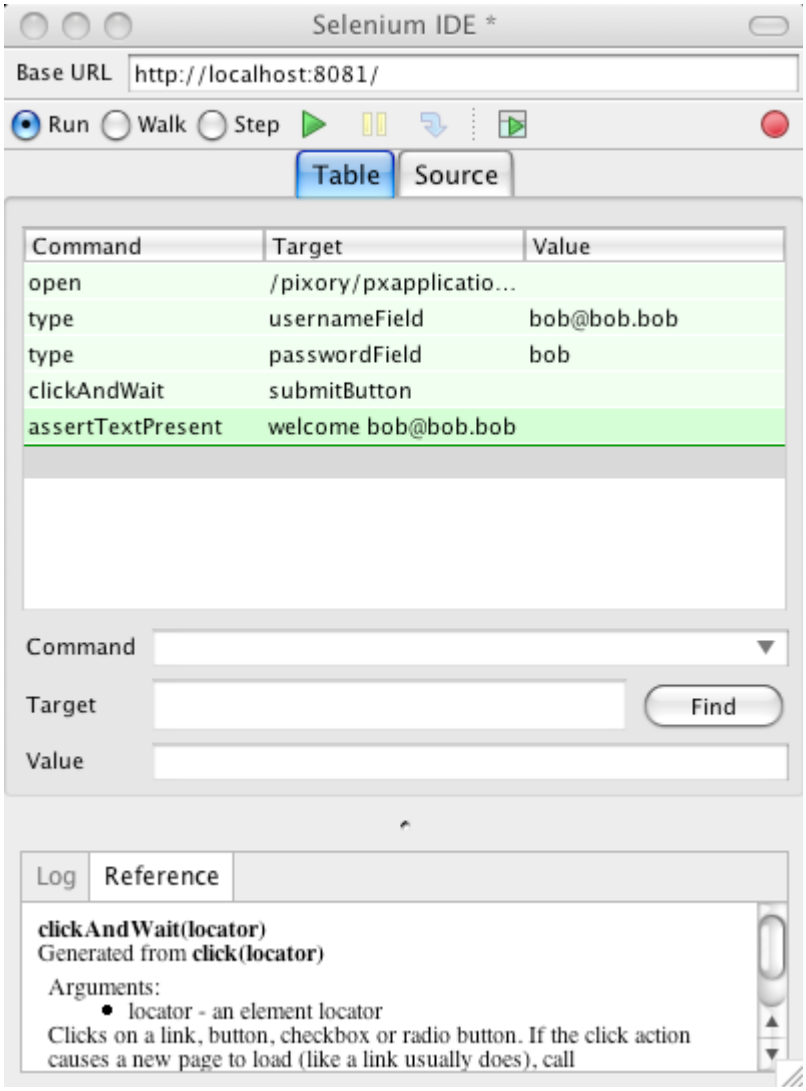
Hit submit



Hit record again to stop



Add assert Text Present and type the text to search for



Hit play to make sure your test is successful

7.3 Creating a Test Suite:

```
<html>
<head>
<meta content="text/html; charset=ISO-8859-1"
http-equiv="content-type">
<title>Demo Test Suite</title>
</head>
<body>
<table id="suiteTable">
<tbody>
<tr><td>
<b>Demo Test Suite</b>
</td></tr>
```

```
<tr><td>
<a href="./testLogin.html">TestLogin</a>
</td></tr>
<tr><td>
<a href="./testLogout.html">TestLogout</a>
</td></tr>
</tbody>
</table>
</body>
</html>
```

A Test Suite in Selenium is just an HTML file that contains a table of links to tests

Executing the Test Suite

• Selenium Core is a collection of Javascript and HTML with iFrames

• Due to security concerns Core must be deployed within the same server as the application being hosted

• The simplest way to run Pixory is to just run the Java application and let it use its own server

• Problems using Core with Pixory

• Selenium IDE is a plug-in for Firefox and thus can go around these restrictions

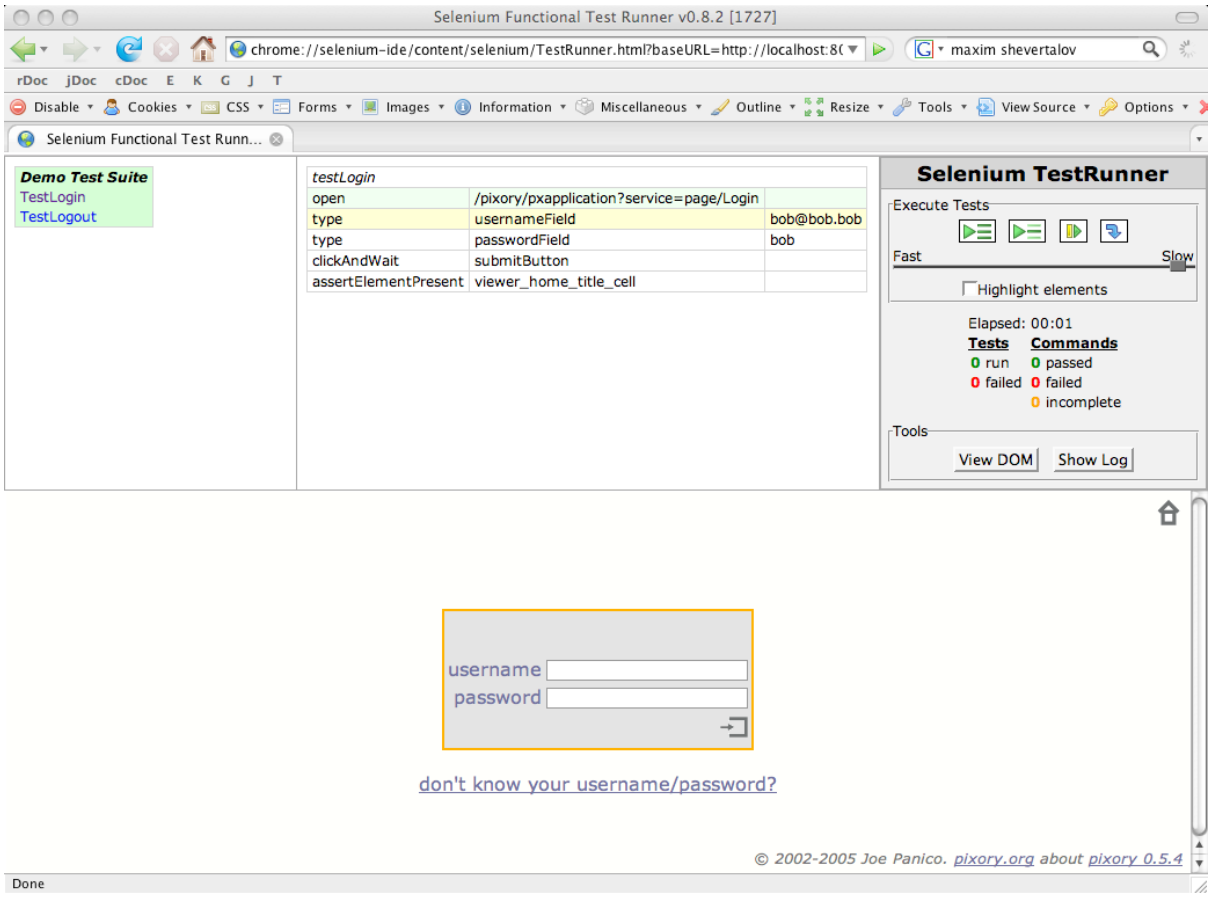
Running the Test Suite

• We basically want to execute the test suite using the Selenium IDE plug-in TestRunner.html

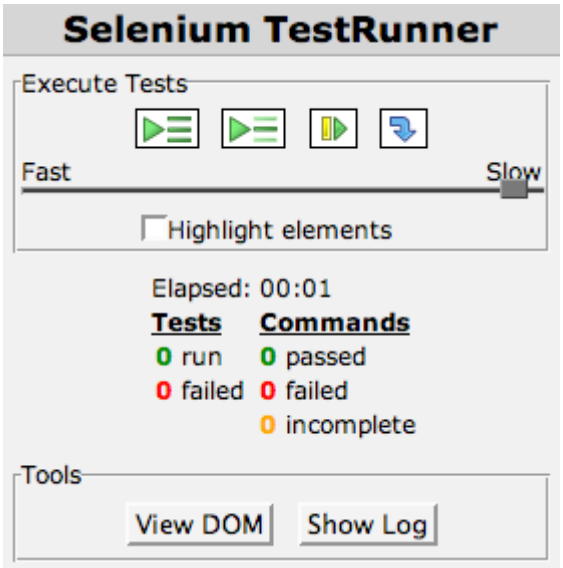
chrome://selenium-ide/content/selenium/TestRunner.html?
baseURL=**<BASE>**&test=file:///**<TEST SUITE FILE>**&auto=true

chrome://selenium-ide/content/selenium/TestRunner.html?
baseURL=**http://localhost:8081**&test=file:///**Users/ms333/**
projects/classes/running/v_and_v/hw3/selenium/test/
TestSuite.html&auto=true

Test Suite



Test Runner Control :



Test Runner Demo:

• Execute Tests created inside the Firefox TestRunner

viva questions

1.What is Selenium?

2.What are the main components of Selenium testing tools?

3.What is Selenium IDE?

4.What is the use of context menu in Selenium IDE?

5.Can tests recorded using Selenium IDE be run in other browsers?

6.What are the advantage and features of Selenium IDE?

7.What are the disadvantage of Selenium IDE tool?

8.What is Selenium Grid?

9.How Selenium Grid works?

EXPERIMENT: 8

NAME OF THE EXPERIMENT: Study of Any Bug Tracking Tool (Bugzilla)

8.1 Introduction

Bugzilla is a popular bug-tracking system that allows teammates and developers to track outstanding bug, Assign, track, and resolve error fixes and conflict resolutions using the Bugzilla database Communicate with teammates, Submit and review patches and Manage quality assurance (QA).

Bugzilla is one example of a class of programs called "Defect Tracking Systems", or, more commonly, "Bug-Tracking Systems". Defect Tracking Systems allow individual or groups of developers to keep track of outstanding bugs in their product effectively. Bugzilla was originally written by Terry Weissman in a programming language called "TCL", to replace a crappy bug-tracking database used internally for Netscape Communications. Terry later ported Bugzilla to Perl from TCL, and in Perl it remains to this day. Most commercial defect-tracking software vendors at the time charged enormous licensing fees, and Bugzilla quickly became a favorite of the open-source crowd. It is now the de-facto standard defect-tracking system against which all others are measured.

8.2Features

Bugzilla has matured immensely, and now boasts many advanced features. These include:

- integrated, product-based granular security schema
- inter-bug dependencies and dependency graphing
- advanced reporting capabilities
- a robust, stable RDBMS back-end
- extensive configurability
- a very well-understood and well-thought-out natural bug resolution protocol
- email, XML, console, and HTTP APIs
- available integration with automated software configuration management systems, including Perforce and CVS
- too many more features to list

8.3 Anatomy of a Bug

The core of Bugzilla is the screen which displays a particular bug. It's a good place to explain some Bugzilla concepts.

1. Product and Component: Bugs are divided up by Product and Component, with a Product having one or more Components in it. For example, bugzilla.mozilla.org's "Bugzilla" Product is composed of several Components:

- Administration: Administration of a Bugzilla installation.
- Bugzilla-General: Anything that doesn't fit in the other components, or spans multiple components.
- Creating/Changing Bugs: Creating, changing, and viewing bugs.
- Documentation: The Bugzilla documentation, including The Bugzilla Guide.
- Email: Anything to do with email sent by Bugzilla.
- Installation: The installation process of Bugzilla.
- Query/Bug list: Anything to do with searching for bugs and viewing the bug lists.
- Reporting/Charting: Getting reports from Bugzilla.
- User Accounts: Anything about managing a user account from the user's perspective. Saved queries, creating accounts, changing
- User Interface: General issues having to do with the user interface cosmetics (not functionality) including cosmetic issues, HTML

2. Status and Resolution: These define exactly what state the bug is in - from not even being confirmed as a bug, through to being fixed and the fix confirmed by Quality Assurance. The different possible values for Status and Resolution on your installation should be documented in the context-sensitive help for those items.

3. Assigned To: The person responsible for fixing the bug.

4. *QA Contact: The person responsible for quality assurance on this bug.

5. *URL: A URL associated with the bug, if any.

6. Summary: A one-sentence summary of the problem.

7. *Status Whiteboard: A free-form text area for adding short notes and tags to a bug.

8. *Keywords: The administrator can define keywords which you can use to tag and categories bugs - e.g. The Mozilla Project has keywords like crash and regression.

9. Platform and OS: These indicate the computing environment where the bug was found.

10. Version: The "Version" field is usually used for versions of a product which have been released, and is set to indicate which versions of a Component have the particular problem the

bug report is about.

11. Priority: The bug assignee uses this field to prioritize his or her bugs. It's a good idea not to change this on other people's bugs.

12. Severity: This indicates how severe the problem is - from blocker ("application unusable") to trivial ("minor cosmetic issue"). You can also use this field to indicate whether a bug is an enhancement request.

13. *Target: A future version by which the bug is to be fixed. e.g. The Bugzilla Project's milestones for future Bugzilla versions are 2.18, 2.20, 3.0, etc. Milestones are not restricted to numbers, thought - you can use any text strings, such as dates.

14. Reporter: The person who filed the bug.

15. CC list: A list of people who get mail when the bug changes.

16. *Time Tracking: This form can be used for time tracking. To use this feature, you have to be blessed group membership specified by the "time tracking group" parameter.

- Orig. Est.: This field shows the original estimated time.
- Current Est.: This field shows the current estimated time. This number is calculated from "Hours Worked" and "Hours Left".
- Hours Worked: This field shows the number of hours worked.
- Hours Left: This field shows the "Current Est." - "Hours Worked". This value + "Hours Worked" will become the new Current Est.
- %Complete: This field shows what percentage of the task is complete.
- Gain: This field shows the number of hours that the bug is ahead of the "Orig. Est.".
- Deadline: This field shows the deadline for this bug.

17. Attachments: You can attach files (e.g. test cases or patches) to bugs. If there are any attachments, they are listed in this section. Attachments are normally stored in the Bugzilla database, unless they are marked as Big Files, which are stored directly on disk.

18. *Dependencies: If this bug cannot be fixed unless other bugs are fixed (depends on), or this bug stops other bugs being fixed (blocks), their numbers are recorded here.

19. *Votes: Whether this bug has any votes.

20. Additional Comments: You can add your two cents to the bug discussion here, if you have something worthwhile to say.

8.4 Life Cycle of a Bug

The life cycle of a bug, also known as workflow, is customizable to match the needs of organization. Below Figure-1 contains a graphical representation of the default workflow using the default bug statuses.

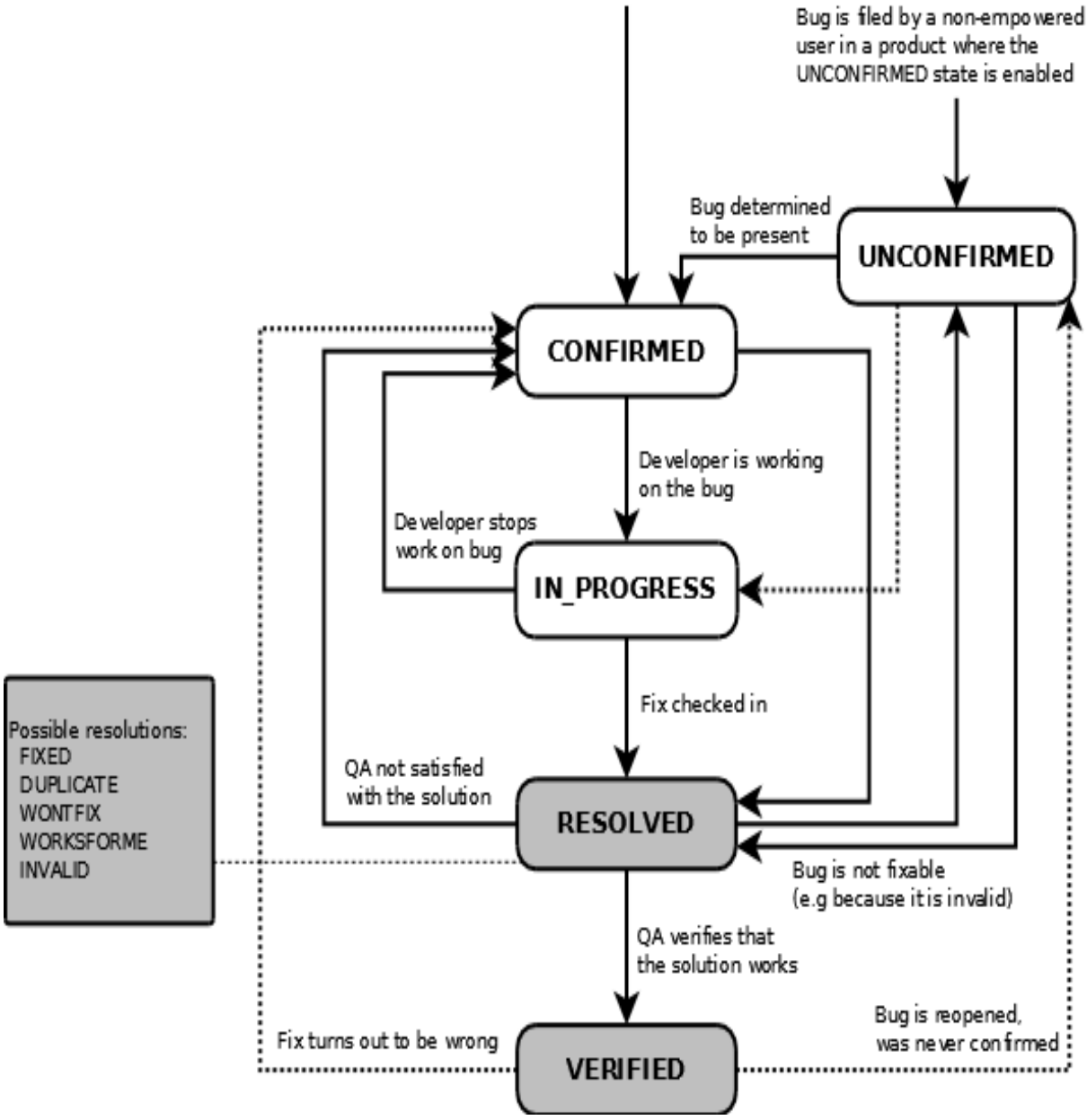


Figure-1. Lifecycle of a Bugzilla Bug

8.5 Searching for Bugs

The Bugzilla Search page is the interface where we can find any bug report, comment, or patch currently in the Bugzilla system. The Search page has controls for selecting different possible values for all of the fields in a bug. For some fields, multiple values can be selected. In those cases, Bugzilla returns bugs where the content of the field matches any one of the selected values. If none is selected, then the field can take any value. After a search is run, we can save it as a Saved Search, which will appear in the page footer. If we are in the group defined by the "query share group" parameter, we may share your queries with other users; see Saved Searches for more details.

8.5.1. Boolean Charts

Highly advanced querying is done using Boolean Charts. The Boolean charts further restrict the set of results returned by a query. There are three fields in each row of a boolean search.

- Field: the items being searched
- Operator: the comparison operator
- Value: the value to which the field is being compared

    1. Pronoun Substitution
    2. Negation
    3. Multiple Charts

8.5.2. Quick search

Quick search is a single-text-box query tool which uses meta characters to indicate what is to be searched. For example, typing "foot bar" into Quick search would search for "foo" or "bar" in the summary and status whiteboard of a bug; adding ": Baz Product" would search only in that product. we can use it to find a bug by its number or its alias, too.

8.5.3. Case Sensitivity in Searches

Bugzilla queries are case-insensitive and accent-insensitive, when used with either MySQL or Oracle databases. When using Bugzilla with PostgreSQL, however, some queries are case-sensitive. This is due to the way PostgreSQL handles case and accent sensitivity.

8.5.4. Bug Lists

If we run a search, a list of matching bugs will be returned. The format of the list is configurable. For example, it can be sorted by clicking the column headings. Other useful features can be accessed using the links at the bottom of the list:

- Long Format: this gives a large page with a non-editable summary of the fields of each bug.

- XML: get the bug list in the XML format.
- CSV: get the bug list as comma-separated values, for import into e.g. a spreadsheet.
- Feed: get the bug list as an Atom feed. Copy this link into our favorite feed reader. If we are using Firefox, you can also save the list
- iCalendar: Get the bug list as an iCalendar file. Each bug is represented as a to-do item in the imported calendar.
- Change Columns: change the bug attributes which appear in the list.
- Change several bugs at once: If our account is sufficiently empowered, and more than one bug appear in the bug list, this link is displayed
- Send mail to bug assignees: If more than one bug appear in the bug list and there are at least two distinct bug assignees, this links is displayed
- Edit Search: If we didn't get exactly the results you were looking for, we can return to the Query page through this link and make small Remember
- Search As: we can give a search a name and remember it; a link will appear in our page footer giving you quick access to

8.5.5. Adding/removing tags to/from bugs

We can add and remove tags from individual bugs, which let you find and manage them more easily. Creating a new tag automatically generates a saved search - whose name is the name of the tag - which lists bugs with this tag. This saved search will be displayed in the footer of pages by default, as all other saved searches.

8.6. Filing Bugs

8.6.1. Reporting a New Bug

The procedure for filing a bug is as follows:

1. Click the "New" link available in the footer of pages, or the "Enter a new bug report" link displayed on the home page of the Bugzilla installation.
2. We first have to select the product in which we found a bug.
3. We now see a form where you can specify the component. the version of the program we were using, the Operating System and platform our program is running on and the severity of the bug.
4. We now have to give a short but descriptive summary of the bug you found.
5. As we file the bug, we can also attach a document (test case, patch, or screenshot of the problem).

6. Depending on the Bugzilla installation we are using and the product in which we are filing the bug, we can also request developers to consider our bug in different ways.
7. Now is a good time to read our bug report again. Remove all misspellings, otherwise our bug may not be found by developers running queries for some specific words, and so our bug would not get any attention. Also make sure we didn't forget any important information developers should know in order to reproduce the problem, and make sure our description of the problem is explicit and clear enough.

8.6.2. Clone an Existing Bug

Starting with version 2.20, Bugzilla has a feature that allows us to clone an existing bug. The newly created bug will inherit most settings from the old bug. This allows us to track more easily similar concerns in a new bug. To use this, go to the bug that we want to clone, then click the "Clone This Bug" link on the bug page. This will take us to the "Enter Bug" page that is filled with the values that the old bug has. we can change those values and/or texts if needed.

8.7 Disadvantages

Despite its current robustness and popularity, Bugzilla faces some near-term challenges, such as

- reliance on a single database,
- a lack of abstraction of the user interface and program logic,
- verbose email bug notifications,
- a powerful but daunting query interface,
- little reporting configurability,
- problems with extremely large queries,
- some unsupportable bug resolution options,
- little internationalization (although non-US character sets are accepted for comments), and
- Dependence on some nonstandard libraries.

Despite these small problems, Bugzilla is very hard to beat. It is under *very* active development to address the current issues, and continually gains new features.

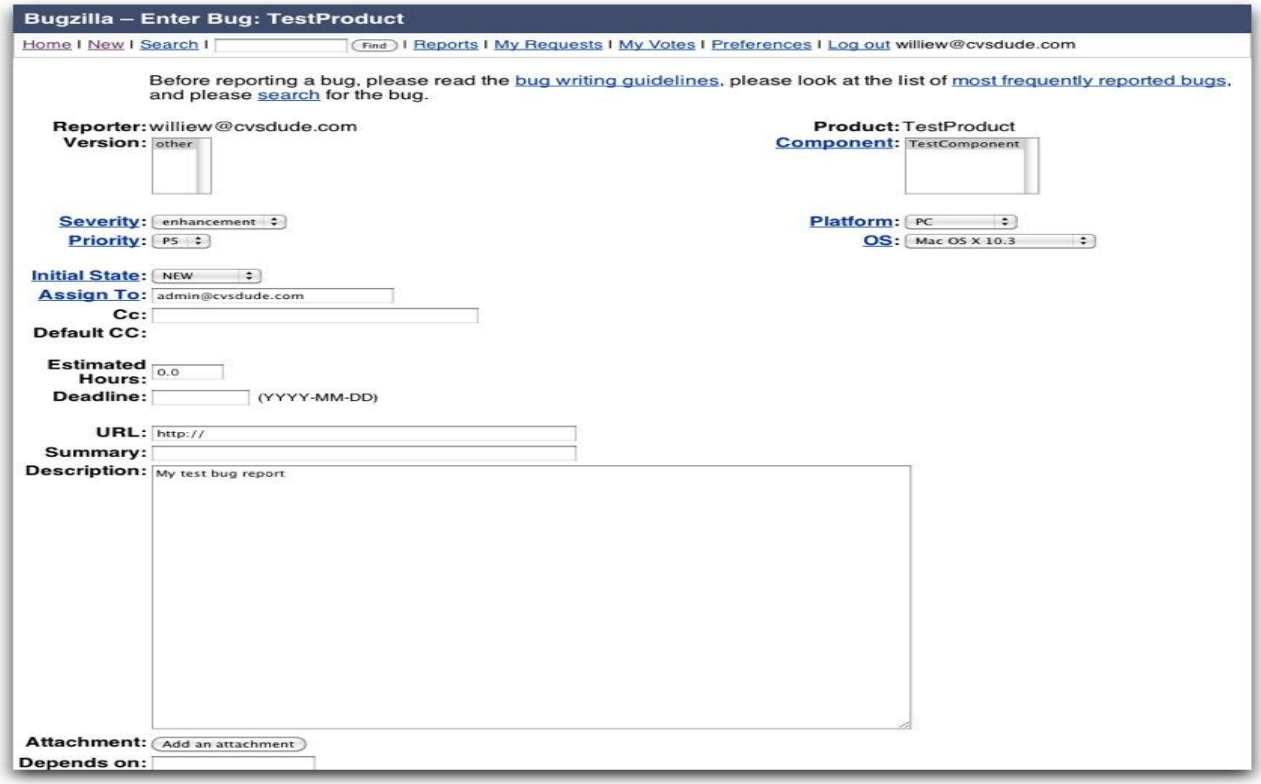


Figure-2: Bugzilla main screen

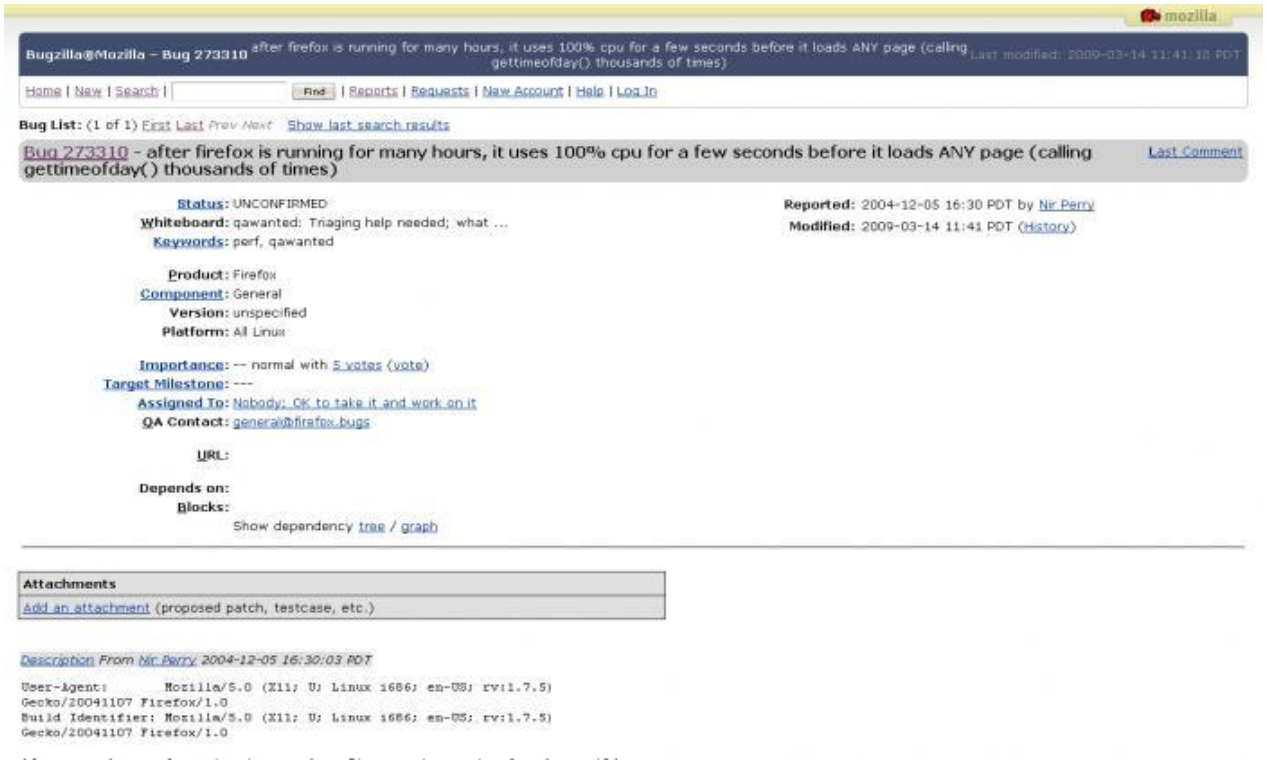


Figure-3: Bugzilla sample test report

.

viva questions

1.What is Bugzilla?

2.What is Bugzilla's features?

3.What are Bugzilla Components?

4.How do I change my user name in Bugzilla?

5.What is about the Bug List page?

6.How to Write a Useful Bug Report with Bugzilla

EXPERIMENT: 9

NAME OF THE EXPERIMENT: Study of Any Test Management Tool (TestDirector)

Test Director is a global test management solution which provides communication, organization, documentation and structure to the testing project.

Test Director is used for

- Mapping Requirements to User acceptance test cases
- Test Planning by placing all the test cases and scripts in it.
- Manual testing by defining test steps and procedures
- Test Execution status
- Defect Management

The TestDirector Testing Process

TestDirector offers an organized framework for testing applications before they are deployed. Since test plans evolve with new or modified application requirements, you need a central data repository for organizing and managing the testing process. TestDirector guides through the requirements specification, test planning, test execution, and defect tracking phases of the testing process.

The TestDirector testing process includes four phases:

Specifying Requirements

- Requirements are linked to tests and defects to provide complete traceability and aid the decision-making process
- See what percent of requirements are covered by tests
- Each requirement in the tree is described in detail, and can include any relevant attachments. The QA tester assigns the requirement a priority level which is taken into consideration when the test team creates the test plan

- Import from Microsoft Word or third party RM tool

Planning Tests

- The Test Plan Manager enables to divide application according to functionality. Application can be divided into units, or subjects, by creating a test plan tree.

- Define subjects according to:
    - Application functionality-such as editing, file operations, and reporting
    - Type of testing-such as functional, user interface, performance, and load

- As the tests are also linked to defects, this helps ensure compliance with testing requirements throughout the testing process

Running Tests

- As the application constantly changes, using test lab, run manual and automated tests in the project in order to locate defects and assess quality.

- By creating test sets and choosing which tests to include in each set, test suite can be created. A test set is a group of tests in a TestDirector project database designed to achieve specific testing goals.

- Tests can be run manually or scheduled to run automatically based on application dependencies.

Tracking Defects

Locating and repairing application defects efficiently is essential to the testing process. Defects can be detected and added during all stages of the testing process. In this phase you perform the following tasks:

- This tool features a sophisticated mechanism for tracking software defects, enabling Testing Team and the project Team to monitor defects closely from initial detection until resolution

- By linking TestDirector to e-mail system, defect tracking information can be shared by all Development and Management Teams, Testing and Wipro Software Quality Assurance personnel

System Requirements for TestDirector

Server System configuration : 128 MB of RAM , 500 MB of free disk space, Win NT server, Win 2K server, IIS 5.0, MSAccess/Oracle 7.x,8.x,9/MS SQL Server

Client System configuration : 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5 , Netscape 4.7

viva questions

1. Orchestrate Vs Datastage Parallel Extender?

2. What is an Exception ? What are types of Exception ?

3. What are Routines and where/how are they written and have you written any routines before?

4. Types of vies in Datastage Director?

5. What are the command line functions that import and export the DS jobs?

6. How many types of database triggers can be specified on a table ? What are they ? **7.** What is NLS in datastage? how we use NLS in Datastage ? what advantages in that ? at the time of ins. ..

8. What are types of Hashed File?

9. What are the datatypes a available in PL/SQL ?

EXPERIMENT: 10

NAME OF THE EXPERIMENT: Study of any open source testing tool (TestLink)

Testlink is an open source test management tool. It enables creation and organization of test cases and helps manage into test plan. Allows execution of test cases from test link itself. One can easily track test results dynamically, generate reports, generate test metrics,prioritize test cases and assign unfinished tasks.

Its a web based tool with GUI, which provides an ease to develop test cases, organize test cases into test plans, execute these test cases and generate reports.

Test link exposes API, written in PHP, can help generate quality assurance dashboards. The functions like AddTestCase ToTestPlan, Assign Requirements,Create TestCase etc. helps create and organize test cases per test plan. Functions like GetTestCasesForTestPlan, GetLastExecutionResult allows one to create quality assurance dashboard.

TestLink enables easily to create and manage Test cases as well as organize them into Test plans. These Test plans allow team members to execute Test cases and track test results dynamically, generate reports, trace software requirements, prioritize and assign tasks. Read more about implemented features and try demo pages.

Overall structure

There are three cornerstones: **Product**, **Test Plan** and **User**. All other data are relations or attributes for this base. First, definition of a couple of terms that are used throughout the documentation.

Products and Test Plans

Product: A Product is something that will exist forever in TestLink. Products will undergo many different versions throughout their life times. Product includes Test Specification with Test Cases and should be sorted via Keywords.

Test Plan: Test Plans are created when you'd like to execute test cases. Test plans can be made up of the test cases of one or many Products. Test Plan includes Builds, Test Case Suite and Test Results.

User: An User has a Role, that defines available TestLink features.

Test Case Categorization

TestLink breaks down the test case structure into three levels Components, Categories, and test cases. These levels are persisted throughout the application.

Component: Components are the parents of Categories. Each Component can have many Categories.

Category: Categories are the parents of test cases. Each Category can have many test cases.

Test Case: Test cases are the fundamental piece of TestLink.

Test Specification: All Components, Categories and test cases within Product.

Test Case Suite: All Components, Categories and test cases within Test Plan.

Test Specification

Creating Test Cases

Tester must follow this structure: Component, Category and test case. At first you create Component(s) for your Product. Component includes Categories. Category has the similar meaning but is second level of Test Specification and includes just Test Cases.

User can also copy or move Test Cases.

Test Cases has following parts:

• Title: could include either short description or abbreviation (e.g. TL-USER-LOGIN)

• Summary: should be really short; just for overview.

• Steps: describe test scenario (input actions); can also include precondition and cleanup information here.

• Expected results: describe checkpoints and expected behaviour a tested Product or system.

Deleting Test Cases

Test cases, Categories, and Components may be deleted from a test plan by users with lead permissions from the "delete test cases" screen. Deleting data may be useful when first creating a test plan since there are no results. However, Deleting test cases will cause the loss of all results associated with them. Therefore, extreme caution is recommended when using this functionality.

Requirements relation

Test cases could be related with software/system requirements as n to n. The functionality must be enabled for a Product. User can assign Test Cases and Requirements via link Assign Requirements in the main screen.

Test Plans

Test plan contains name, description, collection a chosen test cases, builds, test results, milestones, tester assignment and priority definition.

Creating a new Test Plan

Test Plans may be deleted from the "Create test plan" page (link "Create Test Plan") by users with lead privileges. Test plans are the basis for test case execution. Test plans are made up of test cases imported from Products at a specific point of time. Test plans can only be created by users with lead privileges. Test plans may be created from other test plans. This allows users to create test plans from test cases that at a desired point in time. This may be

necessary when creating a test plan for a patch. In order for a user to see a test plan they must have the propper rights. Rights may be assigned (by leads) in the define User/Project Rights section. This is an important thing to remember when users tell you they can't see the project they are working on.

Test Execution

Test execution is available when:

1. A Test Specification is written.

2. A Test Plan is created.

3. Test Case Suite (for the Test Plan) is defined.

4. A Build is created.

5. The Test plan is assigned to testers (otherwise they cannot navigate to this Test Plan).

Select a required Test Plan in main page and navigate to the 'Execute tests' link. Left pane

serves for navigation in Test Case Suite via tree menu, filtering and define a tested build.

Test Status

Execution is the process of assigning a result (pass, fail, blocked) to a test case for a specific build. 'Blocked' test case is not possible to test for some reason (e.g. a problem in configuration disallows to run a tested functionality).

Insert Test results

Test Results screen is shown via click on an appropriate Component, Category or test case in navigation pane. The title shows the current build and owner. The colored bar indicate status of the test case. Yellow box includes test scenario of the test case.

Updated Test Cases: If users have the proper rights they can go to the "Update modified test case" page through the link on main page. It is not necessary for users to update test cases if there has been a change (newer version or deleted).

Advantages:

1. Easy in tracking test cases(search with keyword, test case id, version etc)
2. We can add our custom fields to test cases.
3. Allocating the work either test case creation/execution any kind of documents is easy
4. when a test cases is updated the previous version also can be tracked
5. We can generate results build wise
6. Test plans are created for builds and work allocations can be done.
7. Report, is one of the awesome functionality present in the Test link, it generates reports in desired format like HTML/ CSV /Excel and we can create graphs too.
8. And the above all is done on the privileges based which is an art of the testlink and i liked this feature much

Example of TestLink workflow:

1. Administrator create a Product “Fast Food” and a user Adam with rights “leader” and Bela with rights “Senior tester”.

2. Adam imports Software Requirements and for part of these requirements generates empty Test cases.

3. Bela describe test sneario of these Test cases that are organized according to Components and Categories.

4. Adam creates Keyword: “Regression” and assignes this keyword to ten of these test cases.

5. Adam creates a Test Plan “Fish & Chips”, Build “Fish 0.1” and add Test Cases with keywords “Regression”.

6. Adam and Bela execute and record the testing with result: 5 passed, 1 failed and 4 are blocked.

7. Developers make a new build “Fish 0.2” and Bela tests the failed and blocked test cases only. Exceptionaly all these five Test cases passed.

8. Manager would like to see results. Administrator explain him that he can create account himself on the login page. Manager do it. He has “Guest” rights and could see results and

Test cases. He can see that everything passed in overal report and problems in build “Fish 0.1” in a report for particular Build. But he can change nothing.

viva questions

1.What is difference between Performance Testing, Load Testing and Stress Testing?

2. what are the advantages?

3. what are the test plans?

4. examples for test plan?