MULTIDISCIPLINARY DESIGN
PROJECT REVIEW

A Mobile Distributed System for Personal Security

BATCH-4

- AAKRITI SHARMA (RA1511003030129)
- SHUBHAM KHURANA (RA1511003030132)
- DHANASHREE KURUP (RA1511003030151)
- AYUSH GUPTA (RA1511003030170)
- SHIVANGI GULATI (RA1511003030178)

INTRODUCTION

The goal of this system is to provide a location-based security system through which customers can be helped if they are lost or stuck into emergency situations. The system is divided into two parts, the client side and the server side. The client side works on the mobile phone platforms and is implemented by Java ME. The server side is a cluster running on Linux platform and implement by Java EE technology. The communication between client/server and server/server uses HTTP (Hypertext Transfer Protocol). Before starting the system analysis, a problem which occurred in the system needed to be solved. The problem is how will the system get the user's location if the mobile phone does not have GPS device or signal.

SYSTEM MODEL

2

The **Scrum** software development methodology is applied in our project, which divides the whole project into several sprints, each sprint is typically a 2–3 weeks period (depending on the task) and implements some use cases, and Documentation is also included in the sprint.

•• 2- 3 weeks of background study
•• Familiarity with programming languages
•• The draft database structure/schema
•• 8-10 weeks of design and implementation
•• Design protocol between client/server
•• Design user interface
•• Client and server implementation
•• Redesign and implementation
•• 3-4 weeks of testing, debugging and report writing

PROBLEM ANALYSIS

3

Positioning

Mobile positioning determines the user's actual location information by mobile terminals and mobile networks, such as text messages or mobile signals to complete a series of location information services. The mobile positioning method can be broadly classified into three types:

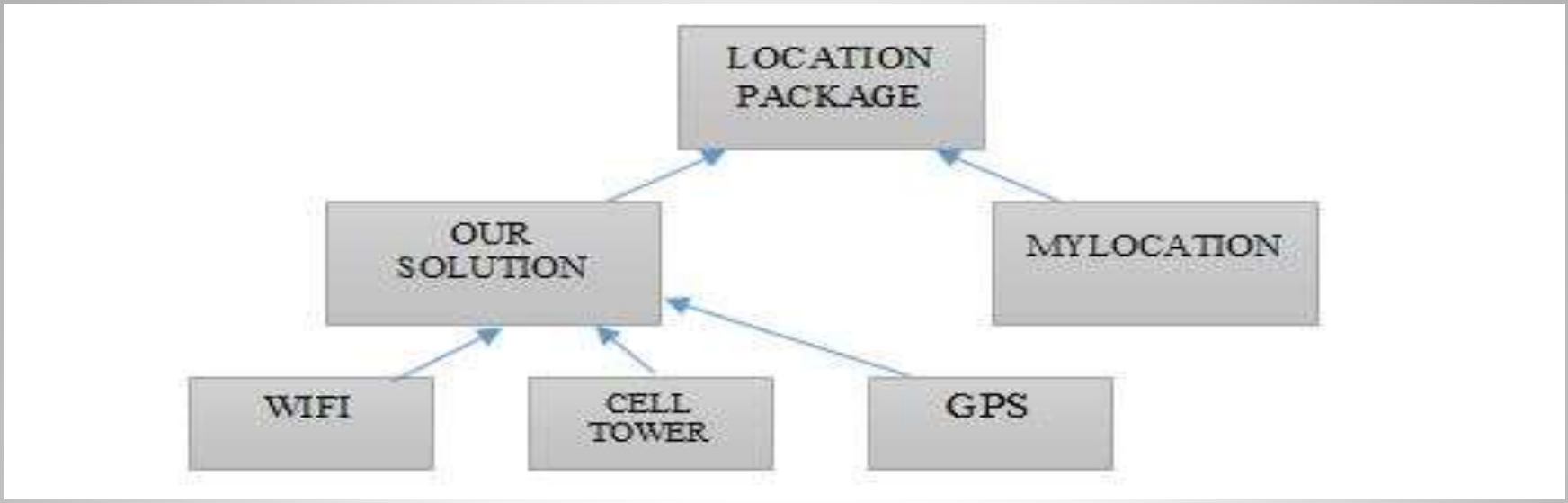
Positioning based on triangulation: This approach is most widely used in mobile positioning technology. It uses the geometric calculation of the triangular relationship to measure the target object's location.

Positioning based on environment analysis: This approach uses the characteristics of the environment surrounding the target object to measure its location.

Positioning based on the relationship: This approach uses the relationship between the target object and one or more existing objects to measure the target object's location.

Our goal is to overcome the limitations of existing product. For example, the solution from Mylocation is primarily used on the mobile phone platform, but the service area of Nordic countries is not reliable enough.

PROBLEM SOLUTION



The mobile phone may receive cell signal or Wi-Fi signal from different cell towers or access points. Therefore we would apply the triangulation theory in our solution to calculate approximate location. However, it cannot work so precisely as GPS. Therefore, the best way to make up for each other's deficiencies is to combine them together. GPS has the best accuracy which the program will try to open first. If the cell phone does not have a GPS device or there is no GPS signal, the system will then scans and try to access wireless which is the next best because of the communication speed. If the system cannot find wireless access point then will load the cell phone signal information.

REQUIREMENTS

Functional requirements

- The system should be able to
  - Find location information at GPS mode
  - Find location information without GPS
- The system should be able to send notification
  - Send notification by SMS (Short Message Service)
  - The notification includes location information
  - The notification could show where you are on a map
  - The notification could show your track on a map
- The system should be able to
  - Record location information with time stamp
  - Save last location information on phone for the user
- Statistic in server side
  - Back office for statistic
  - Log

REQUIREMENTS CONT.

Non-functional requirements

- The system is based on client-server mode
- The client works on iPhone (primarily)
- The client works on another kind of cell phone (maybe in future)
- The system should be able to support multiple servers
  - The server may be deployed in a different place
  - The user may access any server
  - The data must be synchronized

HARDWARE REQUIREMENTS

The hardware on which our software works is any smartphone or java supported phones (ex: Symbian supported phones) which may have digital LED display (not digital mono color display). Phone must support calling and messaging features. Magnetometer sensor and maps support are optional requirements.

SYSTEM ARCHITECTURE

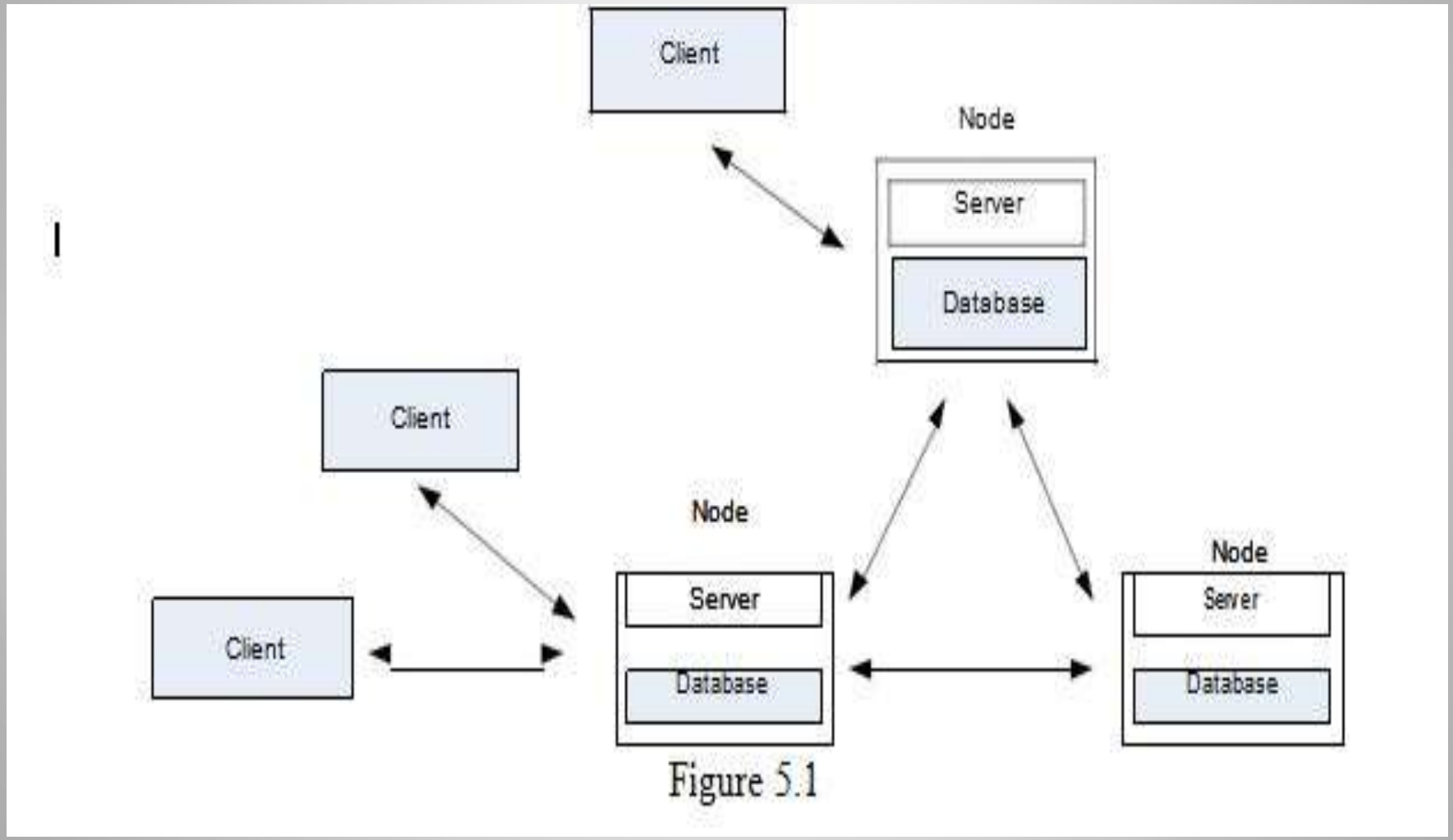
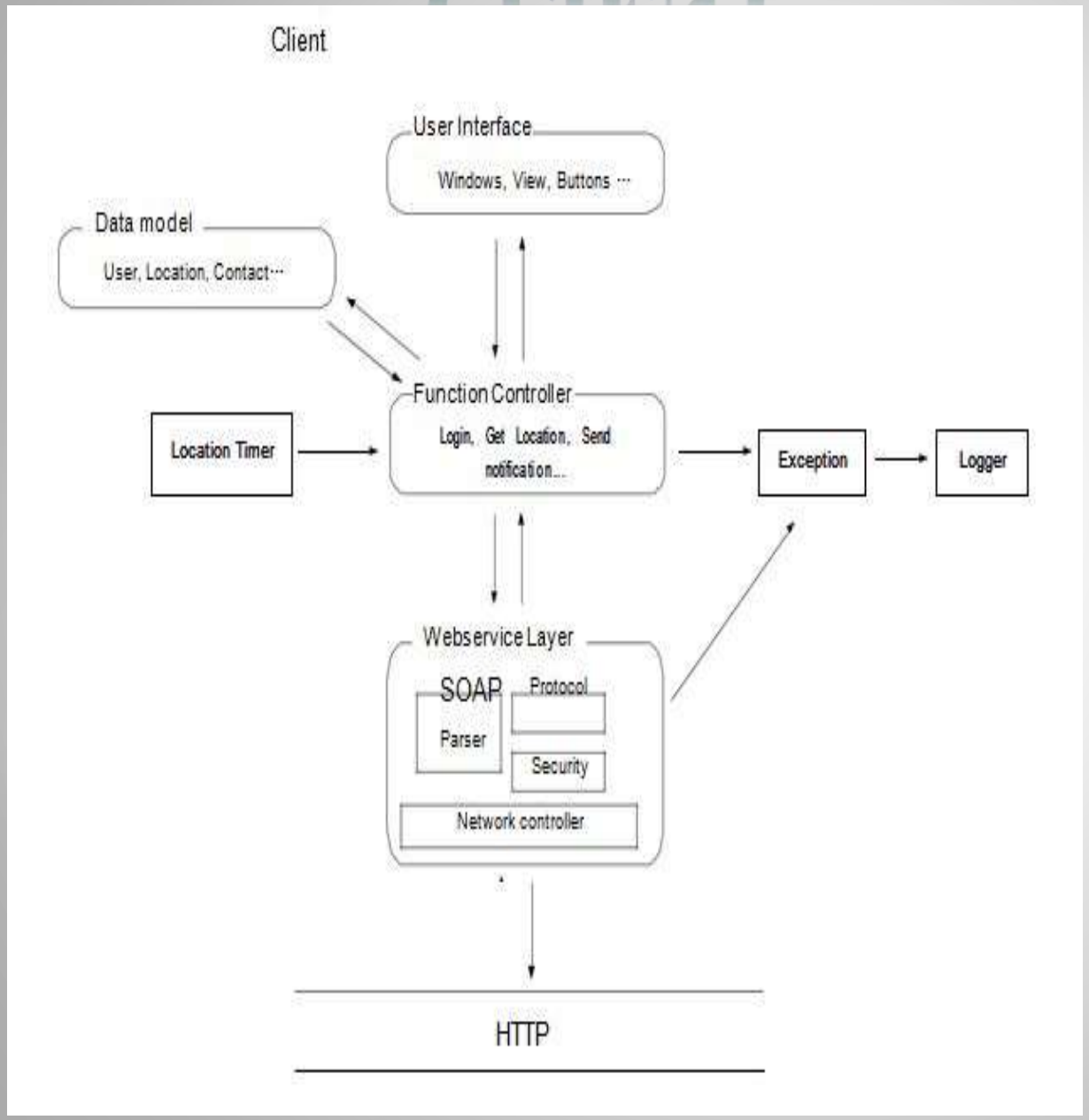


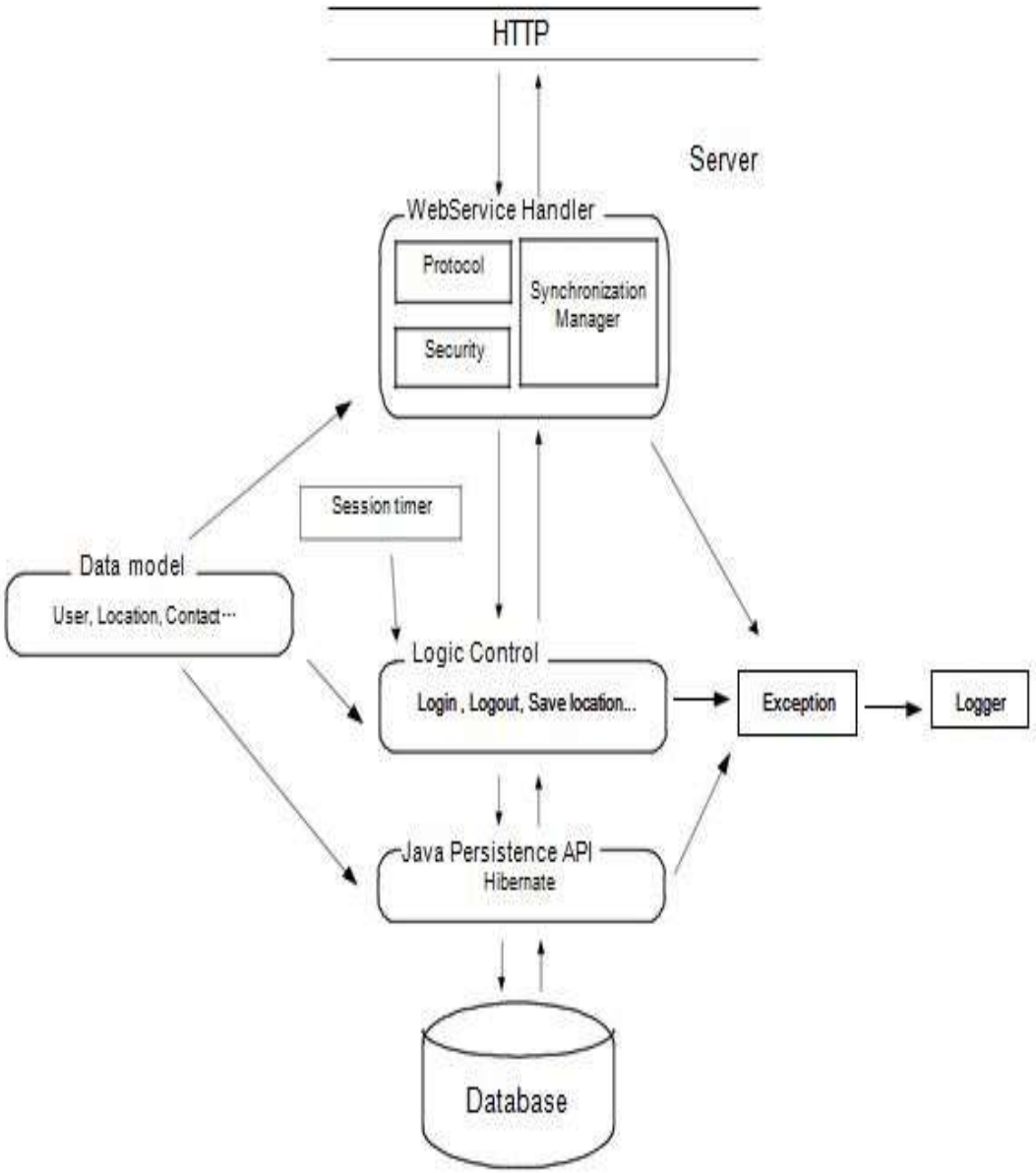
Figure 5.1

CLIENT



- MVC Design Pattern
- Web Services:
  - SOAP
  - Protocol
  - Security
  - Network Control
- Function Controller
- User Interface
- Data Model

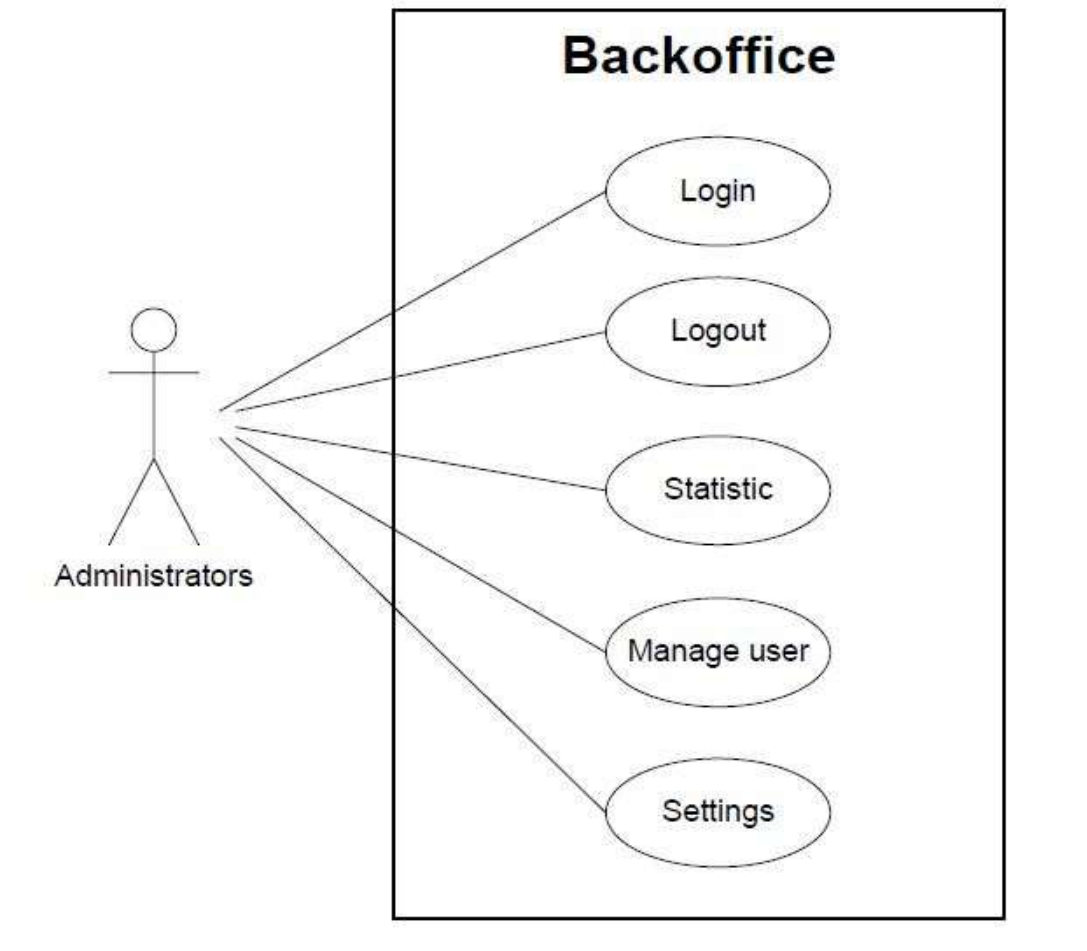
SERVER



- Web Handler
  - Protocol
  - Security
  - Sync
- Logic Control:
  - Login
  - Logout
- API:

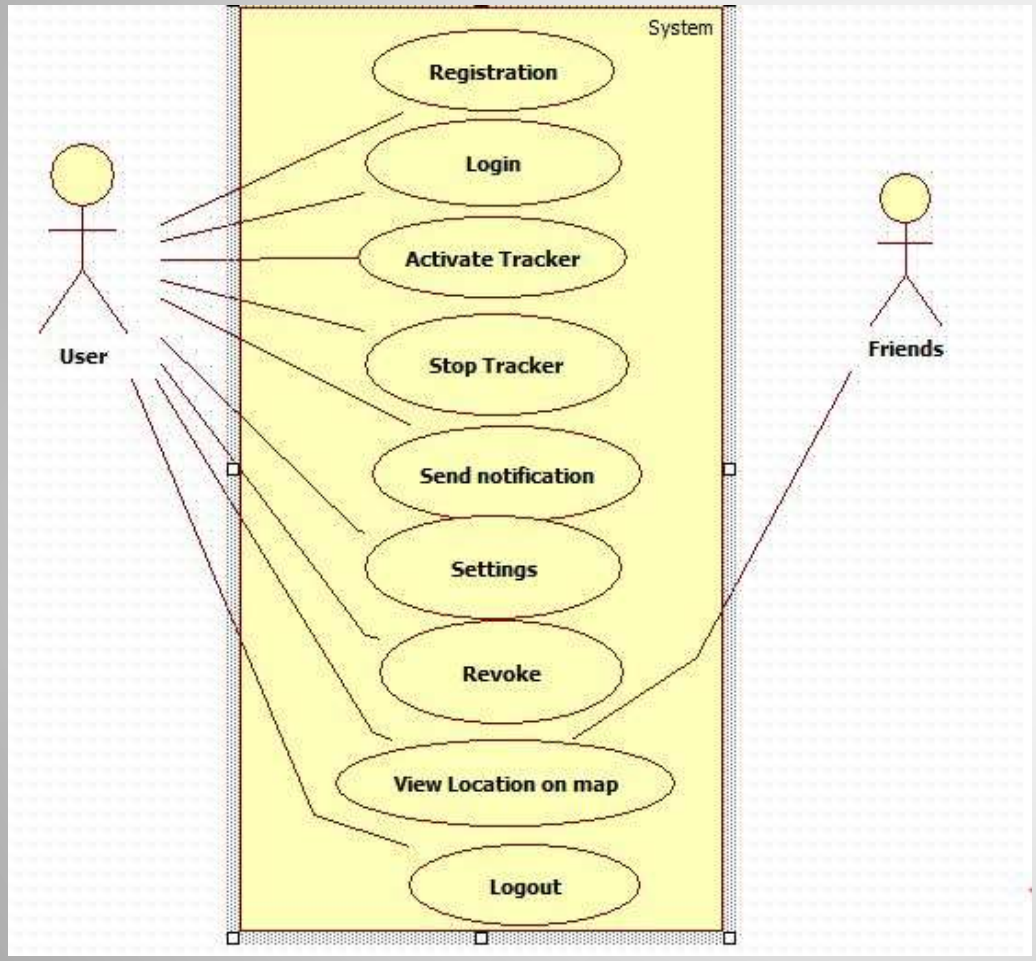
Java Persistence
Hibernate

Use Case Diagram-Backoffice



The back office part is used to make statistics of the user's data, server management
and server configuration that is mainly interactive with the administrators.
There are five use cases here:
1.Login
2.Logout
3.Statistics
4.Manage User
5.Settings

Use Case Diagram-Client side



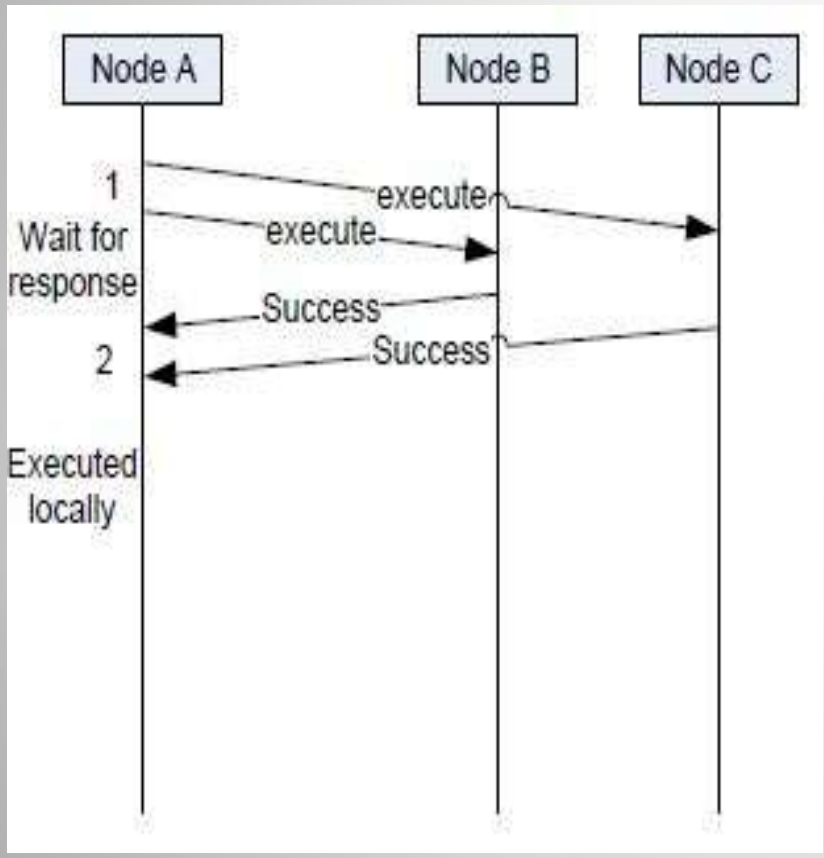
The other part contains the client side which is interactive with user and the server
side which provides services to the user. There are nine use cases in this part:

1.Registration
2.Login
3.Activate Tracker
4.Stop Tracker
5.Send Notification
6.Settings
7.Revoke
8.View Location on map
9.Logout

SYNCHRONIZATION ANALYSIS

- NEED: To increase reliability, load balance and localization among 3 servers
- Sync plays a vital role in decreasing data inconsistency
- METHODS: Database centralization and Database decentralization
- MEMBERS: Coordinator and Participant
- Two CASES: Either it will execute or not
- FAILURE: At coordinator side/ No Failure/ At participant side

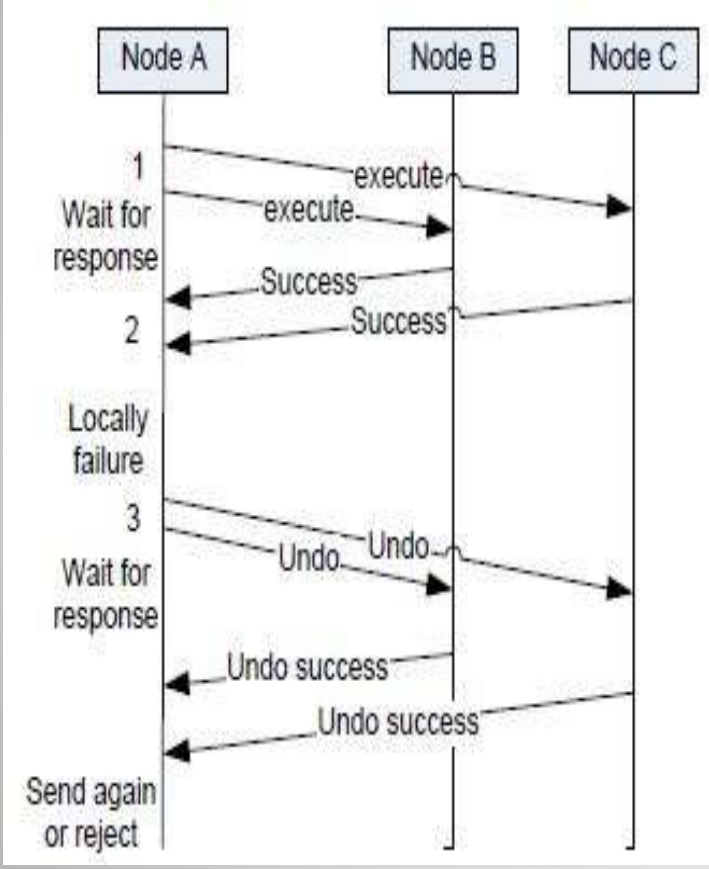
Sequence Diagram-No failure



If there is no problem at the coordinator or the participants side execute statement will generate a “success” command immediately.

DEF: Sequence diagram shows object interactions arranged in time sequence.

Sequence Diagram-Failure at Coordinator

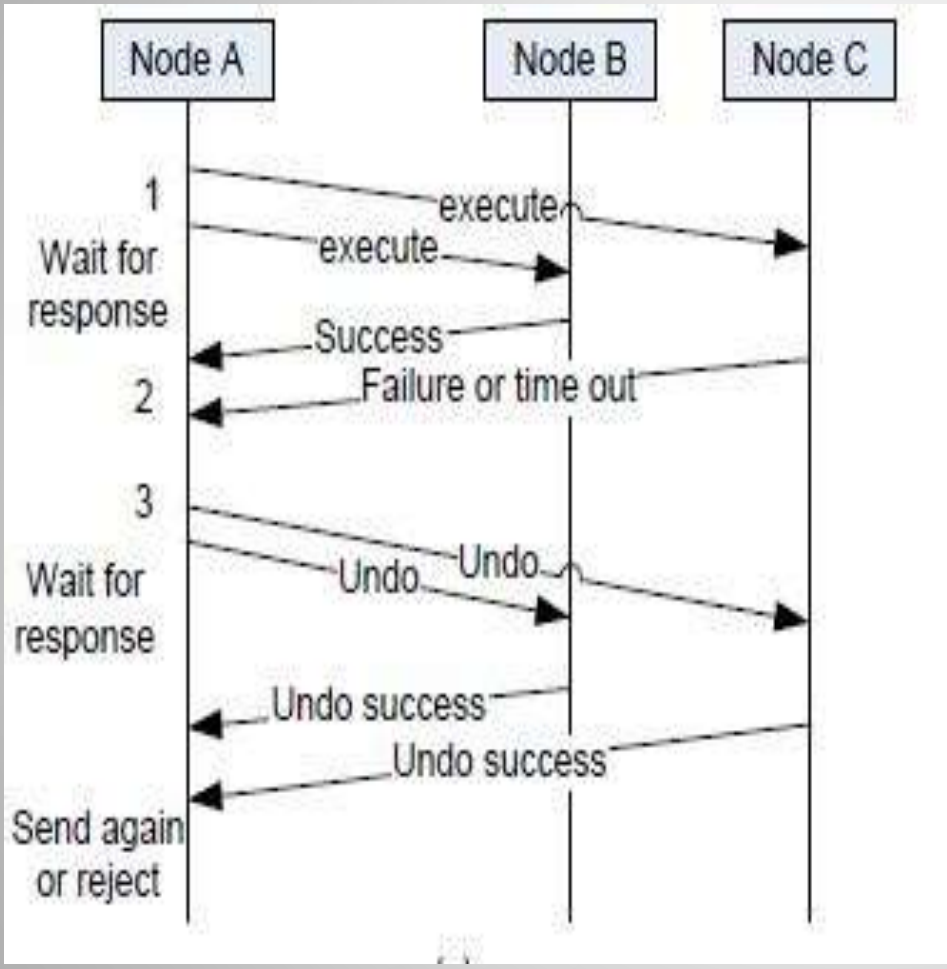


The following diagram shows sequence of problems in three situations:

•**Fault at the coordinator side**:

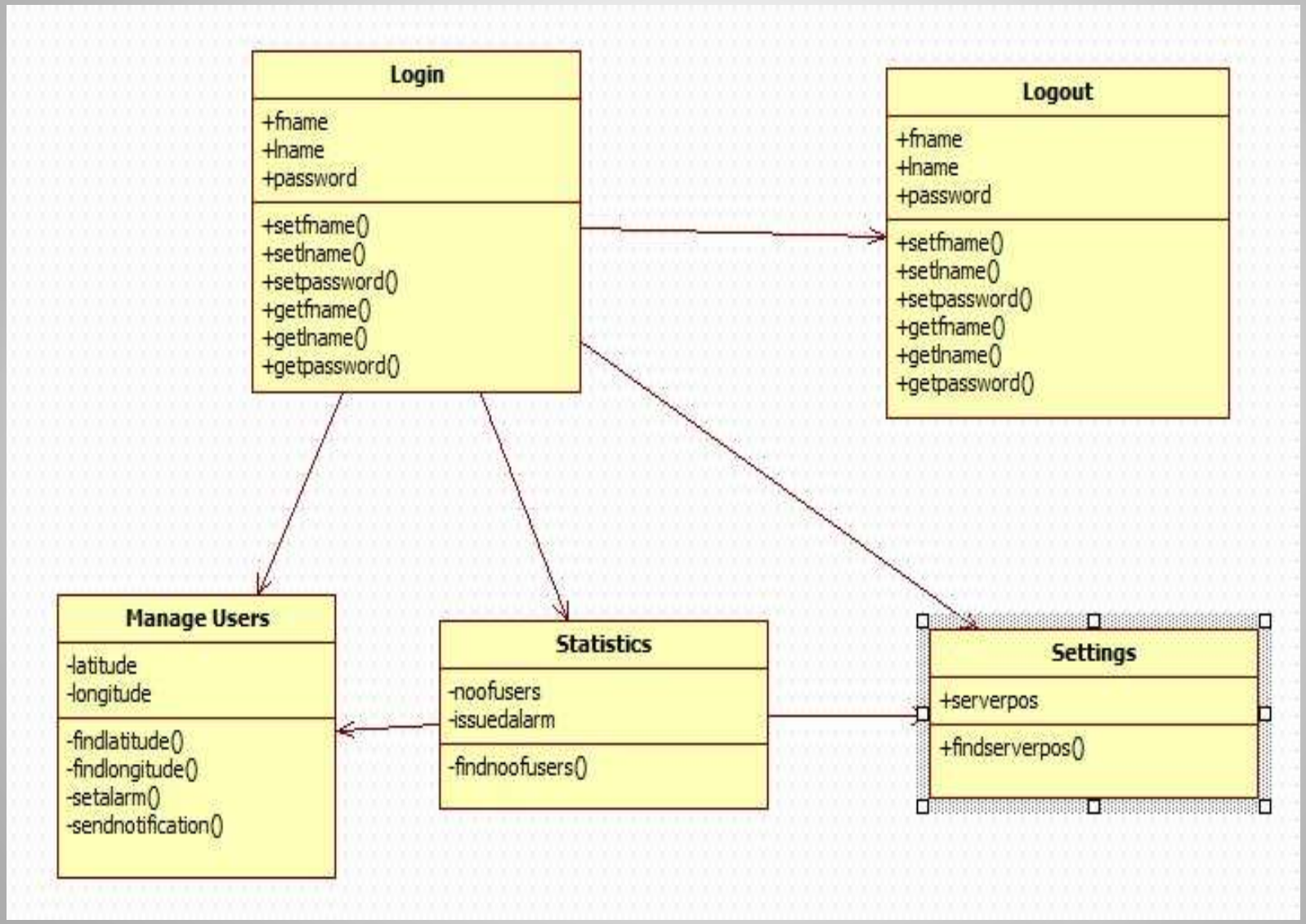
A particular node sends message to both nodes and then waits for response. If all nodes respond successfully then execute the statement otherwise send the “undo” command to all participants and wait for ”undo success” command.

Failure comes from participants side



In participant side, wait for command from coordinator then copy that commands to cache. If the command is “execute”, execute the command locally, and then send “success” message to coordinator. If the command is “undo”
,execute “undo” command locally, and then send “undo success”
message to coordinator.

Class Diagram



Code Snippet

Find and save current location service:

```
<findlocation>
  <session></session>
  <locationset>
    <location>
      </lot>
      </lat>
      </dis>
    </location>    .....
  </locationset>
</findlocation>
<findlocation-return>
 <state>true|false</state>
 <info></info>
</findlocation-return>
```

This service is used for recording the user's current location, as well as calculating and saving their current location if the mobile phone client can only get cell tower or access point information. The tag "locationset" includes the location information list (longitude, latitude, cover distance) of the cell towers or access points. These locations will be used for calculation.

Code Snippet

Save current location service:

<location> <session></session>
<lot></lot>
<lat></lat>
</location>

<location-return>
<state>true|false</state> <info></info>
</location-return>

This service is used for saving the user's current location if the location can be found by mobile phone built in location service. The session id, longitude and latitude information should be provided to the server.

SOFTWARE TESTING

Software testing is a process of executing a program or application with the intent of finding the **software bugs**.
It can also be stated as the **process of validating and verifying** that a software program or application or product:
•Meets the business and technical requirements that guided it's design and development
•Works as expected
•Can be implemented with the same characteristic.

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.
Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

METHODOLOGIES

Black Box Testing

A software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

White Box Testing

A software testing method in which the internal structure/design/implementation of the item being tested is known to the tester.

Gray Box Testing

A software testing method which is a combination of Black Box Testing method and White Box Testing method.

Agile Testing

A method of software testing that follows the principles of agile software development.

Ad Hoc Testing

A method of software testing without any planning and documentation.

Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

Advantages	Disadvantages
• Well suited and efficient for large code segments. • Code access is not required. • Clearly separates user's perspective from the developer's perspective through visibly defined roles. • Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems.	• Limited coverage, since only a selected number of test scenarios is actually performed. • Inefficient testing, due to the fact that the tester only has limited knowledge about an application. • Blind coverage, since the tester cannot target specific code segments or error-prone areas. • The test cases are difficult to design.

White-Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open-box testing**. In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

Advantages	Disadvantages
• As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively. • It helps in optimizing the code. • Extra lines of code can be removed which can bring in hidden defects. • Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.	• Due to the fact that a skilled tester is needed to perform white-box testing, the costs are increased. • Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems, as many paths will go untested. • It is difficult to maintain white-box testing, as it requires specialized tools like code analyzers and debugging tools.

Differences Between Black Box Testing and White Box Testing

Criteria	Black Box Testing	White Box Testing
Definition	Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester	White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
Levels Applicable To	Mainly applicable to higher levels of testing:Acceptance Testing System Testing	Mainly applicable to lower levels of testing:Unit Testing Integration Testing
Responsibility	Generally, independent Software Testers	Generally, Software Developers
Programming Knowledge	Not Required	Required
Implementation Knowledge	Not Required	Required
Basis for Test Cases	Requirement Specifications	Detail Design

What is Cyclomatic Complexity?

Cyclomatic complexity is a source code complexity measurement that is being correlated to a number of coding errors. It is calculated by developing a Control Flow Graph of the code that measures the number of linearly-independent paths through a program module. The cyclomatic complexity of a section of source code is the number of linearly independent paths within it.

Lower the Program's cyclomatic complexity, lower the risk to modify and easier to understand.

It can be represented using the below formula:

Cyclomatic complexity = E - N + 2*P

where,

E = number of edges in the flow graph.

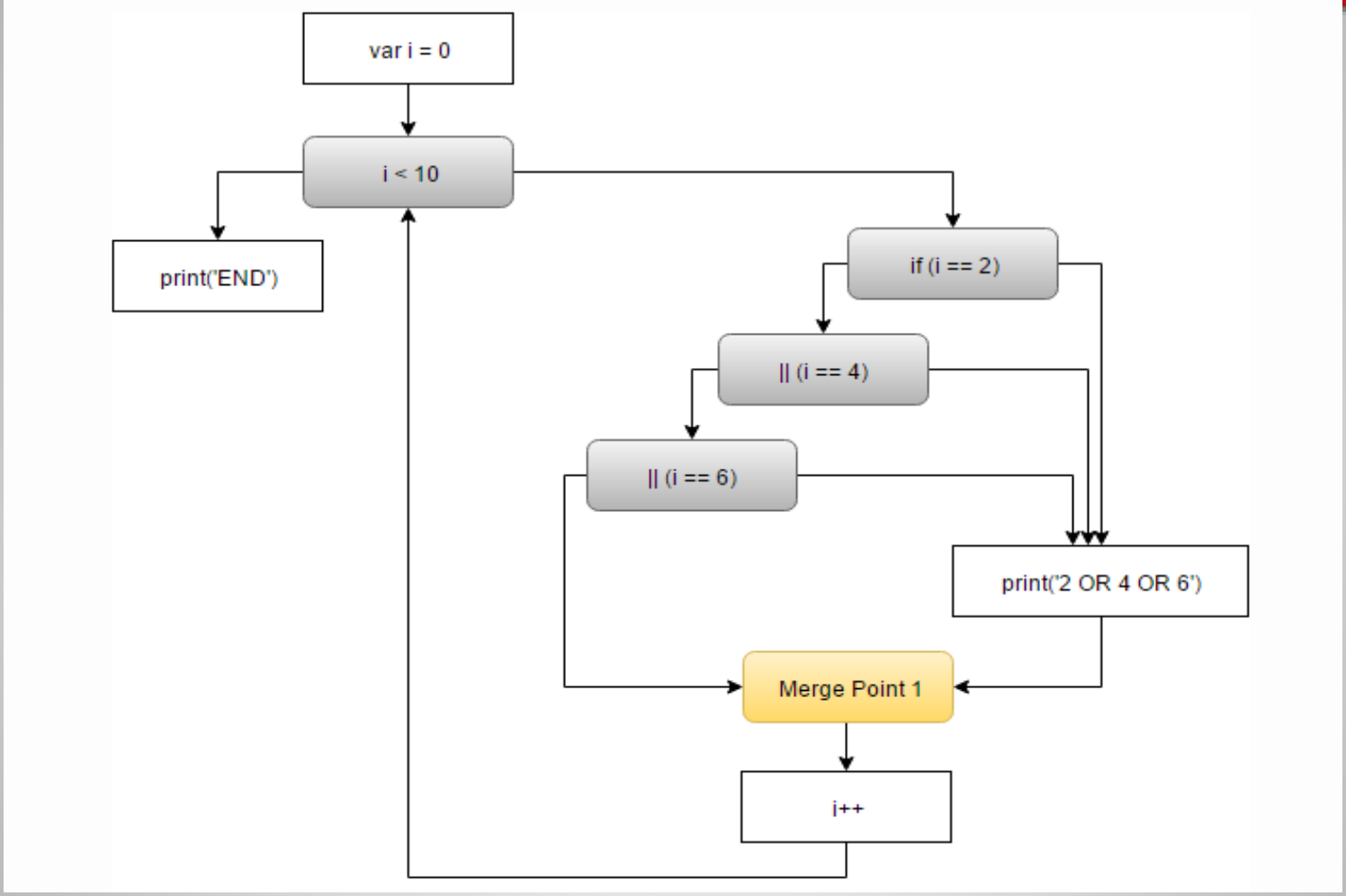
N = number of nodes in the flow graph.

P = number of nodes that have exit points

EXAMPLE OF CYCLOMATIC COMPLEXITY USING FOLLOWING CODE

```
1  for (var i = 0; i < 10; i++) {
2    if (i == 2 || i == 4 || i == 6) {
3      print('2 OR 4 OR 6');
4    }
5  }
6  print('END');
```

CONTROL FLOW GRAPH



There are 9 nodes and 12 edges, so the complexity is 5 (12 - 9 + 2).

Cost Estimation

In our project, we are using basic COCOMO model.

- Basic COCOMO computes software development effort (and cost) as a function of program size. Program size is expressed in estimated thousands of source lines of code (SLOC,KLOC).
- COCOMO applies to three classes of software projects:

1) Organic projects - "small" teams with "good" experience working with "less than rigid" requirements
2) Semi-detached projects - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
3) Embedded projects - developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects.(hardware, software, operational, ...)

Basic Cocomo Model

- Basic COCOMO Model The basic COCOMO model gives an approximate estimate of the project parameters. The basic COCOMO estimation model is given by the following expressions:

1) Effort = **a1 x (KLOC)a 2 PM**

2) Tdev = **b1 x (Effort)b 2 Months**

Where ,

KLOC is the estimated size of the software product expressed in Kilo Lines of Code.

a1, a2, b1, b2 are constants for each category of software products,

Tdev is the estimated time to develop the software, expressed in months,

Effort is the total effort required to develop the software product, expressed in person months (PMs)

- Estimation of development effort For the three classes of software products, the formulas for estimating the effort based on the code size are shown below:

Organic : **Effort = 2.4(KLOC) 1.05 PM**

- Estimation of development time For the three classes of software products, the formulas for estimating the development time based on the effort are given below:

Organic : Tdev = 2.5(Effort) 0.38 Months

Our project is divided into 2 categories:

1)Client Side:

From the basic COCOMO estimation formula for organic software for client side,

The approximate cost can be calculated as:

Estimated line of code:3000;

Cost of an engineer(approx):18000 PM

Effort = 2.4(3)^1.05 = 8 PM

Nominal development time = 2.5(8)^0.38 = 5 months

Cost required to develop the project = 5*18000 = Rs.90,000/-

2)Server Side:

From the basic COCOMO estimation formula for organic software for client side,

The approximate cost can be calculated as:

Estimated line of code:5000;

Cost of an engineer(approx):18000 PM

Effort = 2.4(5)^1.05 = 13 PM

Nominal development time = 2.5(13)^0.38 = 7 months

Cost required to develop the project = 5*18000 = Rs.1,26,000/-

- **Total cost of project**: 2,16,000

Test Scenarios

- **What is a Test Scenario?**

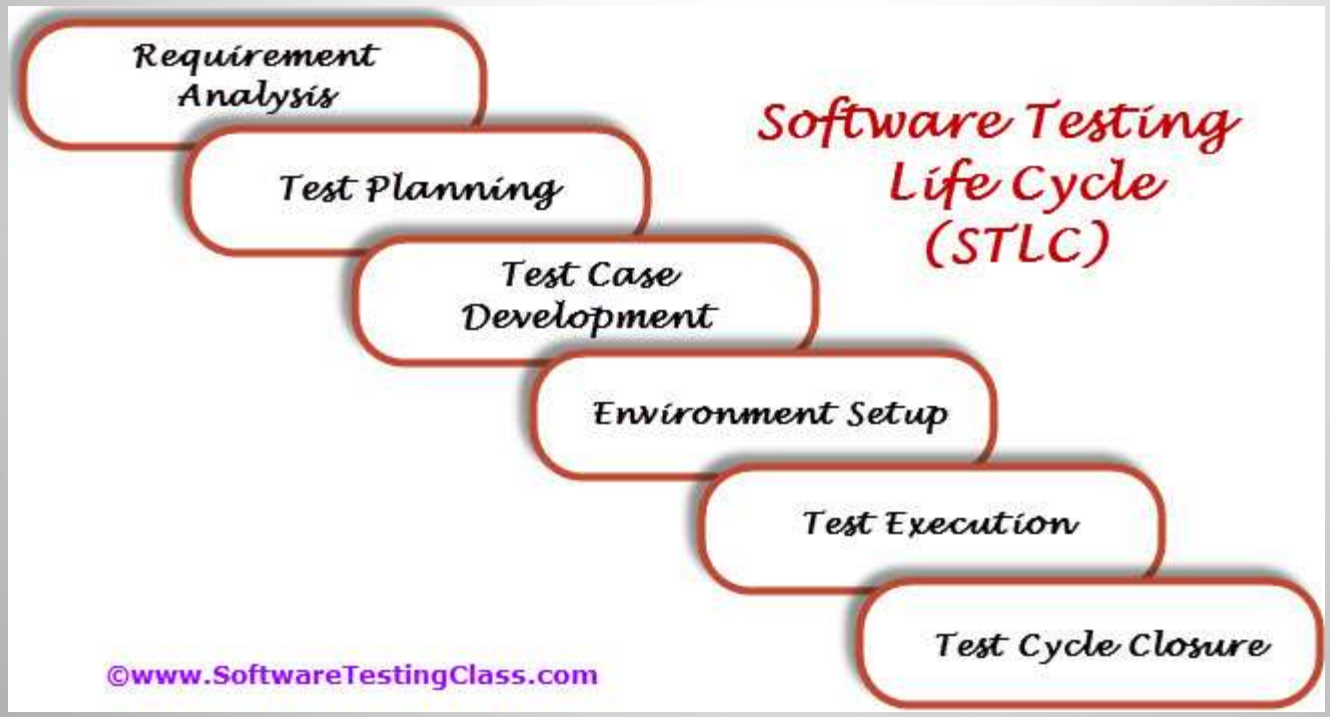
A Test Scenario is any functionality that can be tested. It is also called **Test Condition or Test Possibility**. As a tester, you may put yourself in the end user's shoes and figure out the real-world scenarios and use cases of the Application Under Test.

- **What is Scenario Testing?**

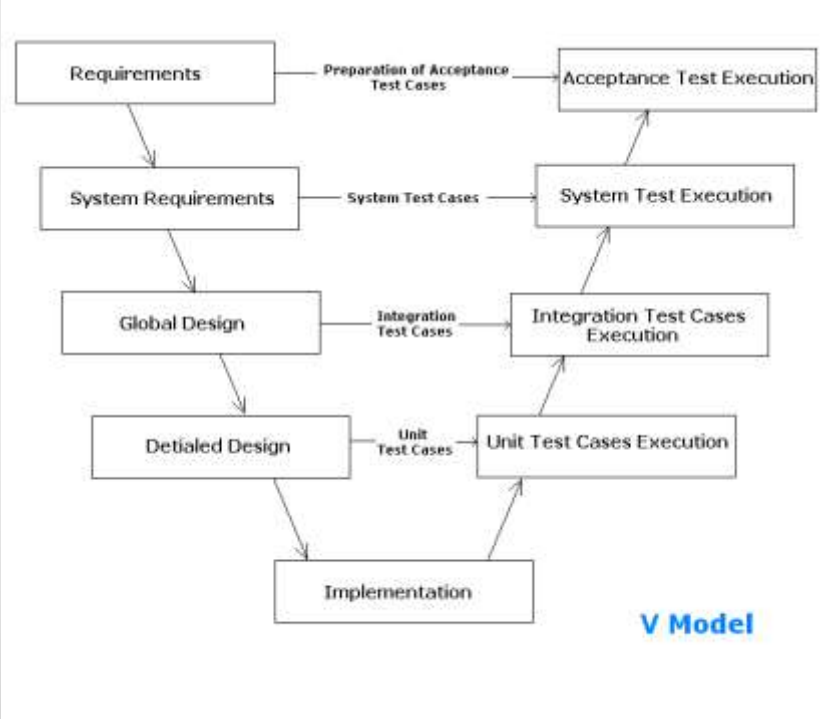
Scenario Testing is a variant of Software Testing where Scenarios are Used for Testing. Scenarios help in an Easier Way of Testing of the more complicated Systems

Software Testing Life Cycle

SOFTWARE TESTING LIFE CYCLE (STLC) *IS THE TESTING PROCESS WHICH IS EXECUTED IN SYSTEMATIC AND PLANNED MANNER. IN STLC PROCESS, DIFFERENT ACTIVITIES ARE CARRIED OUT TO IMPROVE THE QUALITY OF THE PRODUCT.*

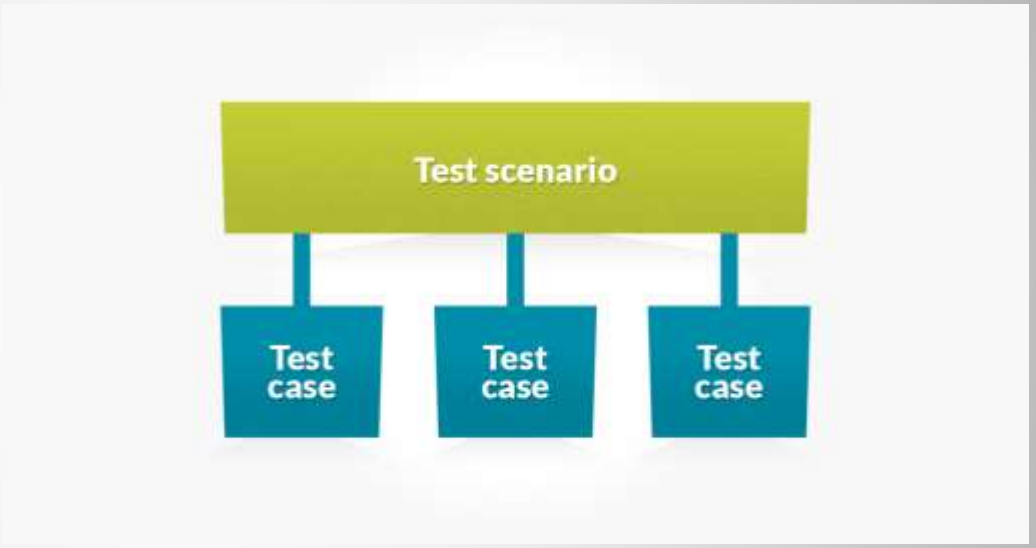


V-Model of STLC



Test scenario :

- MODULE NAME
- TESTING CONDITION
- TESTING PROCESS
- TESTING DATA
- EXPECTED OUTPUT
- ACTUAL OUTPUT
- TEST RESULTS



Test cases :

- LOGIN PAGE
- CURRENT LOCATION
- ALARM SERVICES
- PRECISION TESTING
- SYSTEM FUNCTIONALITY TESTING

LOGIN PAGE TEST CASES

1

- Module Name: Login Page
- Testing Condition: Invalid username
- Testing Process: Checks if username has been correctly input
- Test case: Again loads the input page
- Expected Output: Enter the valid Login details
- Actual Output: Wrong input credentials
- Test Results: Fail

2

- Module Name: Login Page
- Testing Condition: Invalid password
- Testing Process: Checks if password has been correctly input
- Test case: Again loads the input page
- Expected Output: Enter the valid Login details
- Actual Output: Wrong input credentials
- Test Results: Fail

3

- Module Name: Login Page
- Testing Condition: Invalid format for username and password
- Testing Process: Checks if the format of the username name and password is correct. Password must be alphanumeric
- Test case: Password is below minimum required
- Expected Output: Enter the valid format of password
- Actual Output: Wrong input credentials
- Test Results: Fail

4

- Module Name: Login Page
- Testing Condition: Successful input
- Testing Process: Checks if username and password has been correctly input
- Test case: Loads the next page
- Expected Output: Login Successful
- Actual Output: Welcome to the Security Systems
- Test Results: Pass

Test Scenario & Test case Documents (Combination)

Test Scenario ID: TS001_Login

Test ID	Test Action&Test case	Expected Results	Pass/Fail	Approval
Login01	1. Input User: wararat Input Password: comsci 2. Click Log In	Show Message Welcome to wararat	Pass	Ok
Login02	1. Input User: Input Password: comsci 2. Click Log In	Login page is not display	Pass	Ok
Login03	1. Input User: wararat Input Password: 2. Click Log In	Login page is not display	Pass	Ok
Login04	1. Input User: wararat2 Input Password: wararat 2. Click Log In	Login page is not display	Pass	Ok

CURRENT LOCATION TEST CASES

1

- Module name: Current Location
- Testing Condition: Location cant be found
- Testing Process: Checks if the GPS and mobile services are on and user is in valid circle of Wi-Fi range
- Testing Data: Try loading last location traced
- Expected Output: The current location cant be traced.
- Actual Output: Last traced location is as given. Refresh for current location
- Test Results: Passed

2

- Module name: Current Location
- Testing Condition: Database not loaded
- Testing Process: Checks if database connection is still on
- Testing Data: Storing the last known location
- Expected Output: location not found
- Actual Output: Last traced location is as given. Refresh for current location
- Test Results: Passed

3

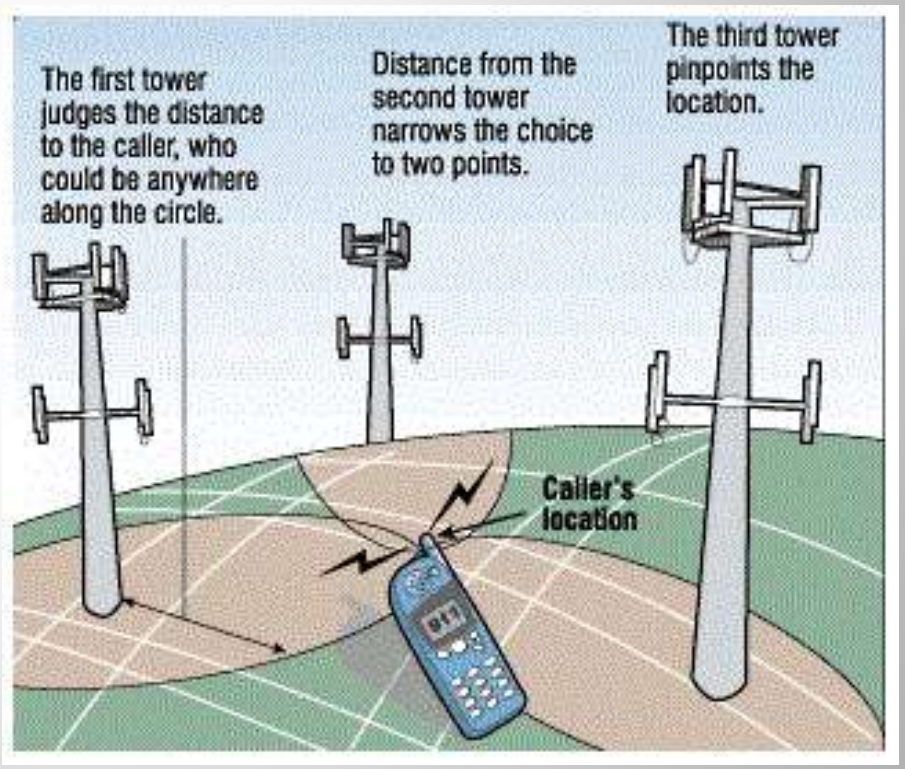
- Module name: Current Location
- Testing Condition: Session Expired
- Testing Process: The current session is not valid
- Testing Data: Time out occurs Login again
- Expected Output: Session Expired Login again to continue
- Actual Output: Session expired Login again to continue
- Test Results: Passed

4

- Module name: Current Location
- Testing Condition: Mobile turned off
- Testing Process: The client mobile is switched off
- Testing Data: Checks weather the location is synchronized
- Expected Output: No active connection
- Actual Output: Cant connect to database, no active connection
- Test Results: Passed

Current location :

- LOCATION NOT FOUND
- CAN’T CONNECT TO DATABASE
- SESSION EXPIRED
- MOBILE TURNED OFF



ALARM SERVICES TEST CASE:

1

- Module Name: Alarm Services
- Testing condition: Sender mobile switched off
- Testing process: Check if service works if mobile is switched off
- Test case: Try sending alarm
- Expected output: Not possible, Alarm service off
- Actual output: Alarm Service Off
- Result: Passed

2

- Module Name: Alarm Services
- Testing condition: Receiver gets notification
- Testing process: 1)Send notification by sender
  2)Check if receiver receives
- Test case: Check mobile of receiver
- Expected output: Notification received by receiver
- Actual output: Notification received by receiver
- Result: Passed

3

- Module Name: Alarm Services
- Testing condition: Blank message received
- Testing process: 1) Send message from sender's mobile
  2) Check the message received by the receiver
- Test case: Open the message of the receiver's phone and read the message received.
- Expected output: Blank message is not received
- Actual output: Blank message is not received
- Result: Passed

4

- Module Name: Alarm Services
- Testing condition: Check if carrier signal at receiver side
- Testing process: 1) Send a message from the sender's phone
  2) Check if message is received by receiver
- Test case: Check the phone by the receiver
- Expected output: Message is received which implies that carrier signal is available at the receiver's side
- Actual output: Message received at receiver's side
- Result: Passed

PRECISION TESTING TEST CASE:

1

- Module Name: Precision Testing
- Testing condition: Cell phone or Wi-Fi signal can be received
- Testing process: 1) Send message/ notification from sender's phone
  2) Check the nearby antennas if they received the notification signal.
- Test case: Check if any nearby antenna have received the message by typing a code
- Expected output: Output message by antenna is returned by the code
- Actual output: output message by antenna is returned by the code
- Result: Passed

2

- Module Name: Precision Testing
- Testing condition: Error in calculation
- Testing process: 1) Send message/ notification from sender's phone
  2) Check the nearby antennas if they received the notification signal.
- Test case: Check the location shown by the antenna
- Expected output: Location is right..(Because of wrong calculation)
- Actual output: Location is right
- Result: Passed

3

- Module Name: Precision Testing
- Testing condition: Accuracy testing
- Testing process: 1) Check the location shown by the antenna calculation
  2) Check the actual location by GPS
  3) Match the location
- Test case: Check the location if correct or not
- Expected output: Location should be accurate
- Actual output: Location should be accurate
- Result: Passed

System functionality testing :

- TRACKING USER LOCATION
- SUCCESSFULLY SENDING MESSAGE
- CURRENT LOCATION INFORMATION
- MAXIMUM ACCURATE LOCATION

SYSTEM FUNCTIONALITY TESTING:

1

- Module Name: System Functionality Testing
- Testing condition: System not working
- Testing process: 1) Database not Sync with server
  2) Cannot get the actual location by GPS
  3) Mobile switched off
- Test case: Check if the previous test cases are passed
- Expected output: System not functional
- Actual output: Can’t load, Refresh for more
- Result: Passed

2

- Module Name: System Functionality Testing
- Testing condition: Message cant be send
- Testing process: Checks if client mobile have enough balance to send message
- Test case: Checks if mobile is active
- Expected output: System not functional
- Actual output: Can’t send message, use emergency services
- Result: Passed

3

- Module Name: System Functionality Testing
- Testing condition: Location Accuracy
- Testing process: Check the nearby antennas if they received the notification signal.
- Test case: Check if any nearby antenna have received the message
- Expected output: Cant determine the exact location
- Actual output: Loading previous known location
- Result: Passed

4

- Module Name: System Functionality Testing
- Testing condition: Client and Server not in sync
- Testing process: Check the network connection for sync in both client and server
- Test case: checks the connectivity between client and server
- Expected output: Can’t the client and server
- Actual output: Reload again
- Result: Passed