**A Mini Project on**

**HEALTH MONITORING ON SOCIAL MEDIA OVER TIME**

A Project Report submitted in partial fulfilment of the degree of the

Bachelor of Technology in

**COMPUTER SCIENCE AND ENGINEERING**

By

**MANTHRI VAMSHI 21C41A0562**

**ODELA SHIVAKRISHA 22C45A0506**

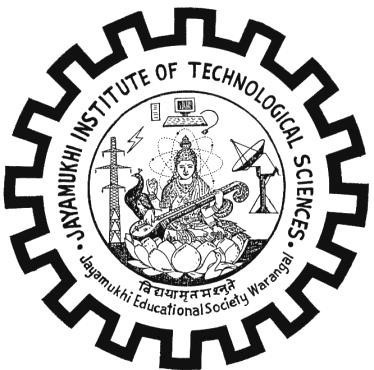
**BODDU PRANAY 21C41A0519**

**SHANAM GOPIKRISHNA 21C41A0537**

Under the Guidance of

**MR.K.CHAITANYA KUMAR**

**Assistant Professor**



Department of Computer Science and Engineering

**JAYAMUKHI INSTITUTE OF TECHNOLOGICAL SCIENCES**

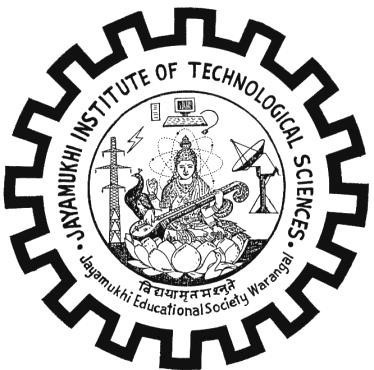
(Autonomous, Affiliated to JNTUH & Accredited by NAAC with ‘A’ Grade) NARSAMPET, WARANGAL-506 332

2024

**JAYAMUKHI INSTITUTE OF TECHNOLOGICAL SCIENCES**

(Autonomous, Affiliated to JNTUH & Accredited by NAAC with ‘A’ Grade)

NARSAMPET, WARANGAL – 506 332



**CERTIFICATE**

This is to certifty that the project report entitled “**HEALTH MONITORING ON SOCIAL MEDIA OVER TIME”** is a bonafide work of the students **M.VAMSHI, O.SHIVAKRISHNA, B.PRANAY, S.GOPIKRISHNA** of Roll No. **21C41A0562, 22C45A0506, 21C41A0519, 21C41A0537** Submitted in partial fullfillment of the requirements for the award of degree of Bachelor of Technology in **CSE** during the academic year **2024-2025.**

**Guide Head of the Department**

**Principal**

**ACKNOWLEDGEMENT**

We express our gratitude and sincere thanks to **Prof.V.Janaki**, Principal, Jayamukhi Institute of Technological Sciences for her constant motivation and support.

We express our gratitude and sincere thanks to **Dr. G.Josemary** , Head of the Department CSE, Jayamukhi Institute of Technological Sciences for her constant motivation and support.

We are grateful to our project guide **Mr.K.Chaitanya Kumar**, Assistant Professor for his valuable guidance and giving us the opportunity to do our work in a better manner.

We are also thankful to our management for providing all the facilities in completing the project.

**MANTHRI VAMSHI 21C41A0562**

**ODELA SHIVAKRISHA 22C45A0506**

**BODDU PRANAY 21C41A0519**

**SHANAM GOPIKRISHNA 21C41A0537**

**ABSTRACT**

Social media, especially Twitter, is an invaluable resource for monitoring public health trends in real-time at minimal cost. This work leverages the Ailment Topic Aspect Model (ATAM) for observing health transitions. We define two primary tasks: **health transition detection** and **health transition prediction**. For detection, we introduce **TM–ATAM**, a temporal extension of ATAM designed to model health-related topic transitions across different times and locations. For prediction, we propose **T–ATAM**, which treats time as a variable to capture latent health trends. Tests on an 8-month tweet corpus indicate that TM–ATAM outperforms TM–LDA in accurately detecting health trends across geographic regions.

**CONTENTS PAGENO**

1. **INTRODUCTION** .....................................................................1
2. **SYSTEM STUDY** ......................................................................3
3. **SYSTEM REQUIREMENTS**………………………………... 4
4. **J2EE SOFTWARE ENVIRONMENT** .....................................6
5. **MODULES** ........................................................................... 16
6. **UML DIAGRAMS** ................................................................ 17
7. **SOFTWARE ENVIRONMENT** .................................................24
8. **SYSTEM TESTING** ...................................................................35
9. **RESULT**……………………………………………………….. 39
10. **CONCLUSION** ............................................................................ 42
11. **REFERENCES** ........................................................................... 43

**INTRODUCTION**

Social media, especially Twitter, serves as a major data source for public health monitoring, allowing early detection of health trends in specific regions. Traditionally, health trends are monitored through sentinel surveillance, which is resource-intensive and lacks real-time capacity. Twitter provides a low-cost, near real-time solution for syndromic surveillance. However, analyzing this data involves challenges: (1) identifying health-related tweets, (2) detecting transitions in health discussions, and (3) distinguishing geographic-specific transitions.

This paper addresses two key problems in health monitoring on Twitter: **health transition detection** and **health transition prediction**.

1. **TM–ATAM Model for Detection**: This model combines the Ailment Topic Aspect Model (ATAM) with Temporal-LDA, capturing transitions in health discussions over time and space. It adapts based on factors like climate, which affects health patterns differently in various regions.
2. **T–ATAM Model for Prediction**: By treating time as a variable, this model predicts future health transitions, capturing subtle shifts in health discourse that may signal emerging trends.

Compared to standard topic modeling methods like LDA, ATAM is specifically designed for health-related Twitter data, detailing symptoms and treatments more accurately. Experiments on 500K tweets over eight months show that TM–ATAM outperforms TM–LDA in capturing temporal and regional health transitions. Results highlight two types of transitions: **Stable Topics** (e.g., constant mentions of ailments like migraines) and **One-Way Transitions** (e.g., discussions on smoking followed by respiratory issues).

Overall, these models offer a way to enhance public health campaigns by identifying and predicting health trends from social media in real-time.

**PRELIMINARY INVESTIGATION**

The project aims to develop a mail-enabled platform for a small firm with messaging, a search engine, an address book, and games, designed for use on a Local Area Network (LAN). This investigation has three parts:

1. **Request Clarification**: Identifies system requirements for a portal accessible within the organization.
2. **Feasibility Analysis**:
   * **Operational Feasibility**: Reduces admin workload and improves tracking.
   * **Economic Feasibility**: Low cost due to existing hardware; allows LAN access for multiple employees.
   * **Technical Feasibility**: Requires IBM-compatible machines with web browsers; platform-independent, developed using JSP, JavaScript, HTML, SQL Server, and WebLogic.
3. **Request Approval**: Approved projects are prioritized based on cost, completion time, and personnel needs.

**SYSTEM DESIGN AND DEVELOPMENT**

**Input Design**: Ensures accuracy by controlling errors in data entry through validations and user-friendly forms. Error messages guide users to correct mistakes, reducing invalid entries.

**Output Design**: Facilitates communication between project leaders, administrators, and clients. The system allows project leaders to manage client records, assign projects, and set folder access. Authentication ensures secure user access.

The application, running on a LAN, starts with server activation. The administrator's machine serves as the server, while other connected systems are clients, providing an intuitive, user-friendly experience.

**SYSTEM STUDY**

**FEASIBILITY STUDY**

In this phase, the feasibility of the project is evaluated with a business proposal, including a basic plan and cost estimates. This ensures that the system is practical and beneficial for the company. Key aspects analyzed include:

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

1. **Economic Feasibility**: Assesses the financial impact on the organization. The project remains within budget as most technologies used are free, with minimal purchases for custom components.
2. **Technical Feasibility**: Evaluates the technical requirements, ensuring the system doesn't overburden existing resources. The system has modest demands, with minimal adjustments needed for implementation.
3. **Social Feasibility**: Ensures user acceptance by providing adequate training and support. Users are guided to understand the system, fostering confidence and constructive feedback, as they are the end-users.

**EXISTING SYSTEM**

* In the existing system, the authors propose a method that learns changing word distributions of topics over time and in the system, the authors leverage the structure of a social network to learn how topics temporally evolve in a community. TM–ATAM and T–ATAM are however different from dynamic topic models such as [9] and [10], and from the work of Wang et al. [11], as they are designed to learn topic transition patterns from temporally-ordered posts, while dynamic topic models focus on changing word distributions of topics over time.
* TM–ATAM learns transition parameters that dictate the evolution of health-related topics by minimizing the prediction error on ailment distributions of consecutive periods at different temporal and geographic granularities. T–ATAM on the other hand discovers latent ailments in health tweets by treating time as a corpus-specific multinomial distribution.
* Classical approaches have been applied to mining topics for inferring citations. Other discriminative approaches have been applied to do an empirical study on topic modeling and time-based topic modeling respectively. None of those are directly applicable to health data.

**Disadvantages**

* + There is no Mapping Tweets to Documents.
  + There is Uncovering Health Topics with ATAM.

**PROPOSED SYSTEM**

* In the proposed system, the system formulates and solves two problems: the health transition detection problem and the health transition prediction problem. To address the detection problem, the system develops TM–ATAM that models temporal transitions of health-related topics. To address the prediction problem, we propose T–ATAM, a novel method which uncovers latent ailment inside tweets by treating time as a random variable natively inside ATAM.
* Treating time as a random variable is key to predicting the subtle change in health-related discourse on Twitter.

**Advantages**

* TM–ATAM, a model able to detect health-related tweets and their evolution over time and space. TM–ATAM learns, for a given region, transition parameters by minimizing the prediction error on ailment distributions of pre-determined time periods.
* T–ATAM, a new model able to predict health-related tweets by treating time as a variable whose values are drawn from a corpus-specific multinomial distribution.
* Extensive experiments that show the superiority of T–ATAM for predicting health transitions, when compared against TM–LDA and TM–ATAM, and its effectiveness against a ground truth.

**SYSTEM REQUIREMENTS**

**H/W System Configuration:-**

* Processor - Pentium –IV
* RAM - 4 GB (min)
* Hard Disk - 256 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**Software Requirements:**

* Operating System - Windows XP
* Coding Language - Java/J2EE(JSP,Servlet)
* Front End - J2EE
* Back End - MySq

#### **Client Server**

#### **Introduction**: The client-server model has become a core topic in computing, driving significant interest and adoption. Major players like IBM and DEC view client-server as a primary market focus, with its development tools expanding from $200 million in 1992 to over $1.2 billion by 1996. This model is complex in implementation but conceptually straightforward and powerful.

#### **Client-Server Model**: A client is a local application that can request database or service access from a remote server. Middleware software facilitates this interaction, allowing clients (often PCs or workstations) connected via network to powerful servers capable of handling multiple client requests. Servers can also act as clients and access additional servers when processing requests. The client-server model separates users from the data’s physical location, letting client applications access both local and remote databases seamlessly.

**Difference Between Client-Server and File-Server:**

* **File-Server:** Provides remote disk drive access, allowing LAN applications to access files one-by-one.
* **Client-Server:** Offers full relational database services, supporting SQL, data modification, and transactional integrity with robust middleware handling the client-server interaction.

#### **Why Client Server**

Client-server architecture addresses the need to efficiently distribute computing and data resources across departments and enterprises. Initially, mainframe systems centralized CPU and data resources, with users limited to batch requests. With time-sharing, remote terminals gained access to central data, and the role of relational databases grew, allowing non-programmers to query data directly. However, early client-server models followed a “slave-master” dynamic, with centralized control over data access and changes.

**Front-End Or User Interface Design**

#### The user interface is developed in a browser-specific environment using an Intranet-Based Architecture to support distributed functionality.

#### HTML standards are used for structure, while Java Server Pages (JSP) provide dynamic elements.

**Communication Or Database Connectivity Tier**

#### The communication architecture is based on Servlets and Enterprise Java Beans (EJB) for efficient processing. Database connectivity is achieved using Java Database Connectivity (JDBC), with a focus on a three-tier architecture to ensure high cohesion and minimal coupling for optimal operation.

#### **Features of The Language Used**

In my project, I have chosen *Java* language for developing the code.

#### **About Java**

Initially the language was called as “oak” but it was renamed as “Java” in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

* Java is a programmer’s language.
* Java is cohesive and consistent.
* Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.

Finally, Java is to Internet programming where C was to system programming.

#### **Importance of Java to the Internet**

Java has revolutionized the Internet by enabling dynamic, self-executing programs to be transmitted securely across networks, unlike traditional passive information. This breakthrough has introduced the applet, a small Java program that can be downloaded and executed by Java-compatible web browsers, enhancing interactivity and security.

#### **Java can be used to create two types of programs**

#### **Applications**: Standalone programs that run on a computer's operating system, similar to those in C or C++.

#### **Applets:** Small programs designed to be transmitted over the Internet and run in Java-compatible browsers, allowing interactive functionality.

#### **Features Of Java**

**Security**: Java's "firewall" design prevents viruses and safeguards private data. Users can safely download applets without risking malware.

**Portability**: Java's architecture enables applications to run on diverse platforms. This is achieved through its Bytecode system, which facilitates program portability.

**Bytecode And Java Virtual Machine (JVM)**

* **Bytecode**: Java compiles programs into Bytecode, an optimized instruction set that runs on the Java Virtual Machine (JVM), making it possible to execute on any system with a JVM.
* **Just-in-Time (JIT) Compiler**: Part of the JVM, JIT compiles bytecode into native code during execution, enhancing performance by compiling as needed.

**Java Virtual Machine (JVM)**

The JVM, a crucial part of Java, verifies bytecode to ensure security and proper functioning across platforms. Embedded in browsers or operating systems, the JVM's bytecode verification process ensures the accuracy and safety of the code before execution.

Overall **Description**

# Java Source

## Java byte code

# Java**VM**

Java

.Class

**Picture showing the development process of JAVA Program**

**Java Architecture And Compilation Process**

Java's architecture is designed to be portable, robust, and high-performing, making it adaptable to various platforms.

1. **Compilation to Bytecode**: Java source code, written in .java files, is compiled by the Java compiler (javac) into .class files containing bytecode. This bytecode is platform-independent, allowing it to be executed anywhere with a Java Virtual Machine (JVM).
2. **Execution via JVM**: The bytecode is then loaded and executed by the JVM, either locally or over a network. The JVM interprets the bytecode, enabling Java programs to run on different systems without needing recompilation, thus ensuring portability.
3. **Dynamic Code Loading**: Java’s architecture supports dynamic code loading, allowing it to pull code from any networked machine when needed, enhancing flexibility and reducing dependency on local resources.

**Compiling and interpreting Java Source Code**

**Java Bytecode and JVM**: Java source code is compiled into bytecode, which is platform-independent. The Java interpreter treats this bytecode as if it’s running on a Java Virtual Machine (JVM), enabling the code to execute on various systems, such as Windows, Solaris, or macOS.

**Source**

**Code**

**………..**

**………..**

**………..**

**…………**

# PC Compiler

**Macintosh**

**Compiler**

**SPARC**

###### Compiler

**Java**

**Byte code**

**(Platform**

**indepen**

**dent)**

**Java**

**Interpreter**

**(PC)**

**Java**

**Interpreter**

**(Macintosh)**

**Java**

**Interpreter**

**(Sparc)**

**Java’s Portability**: This flexibility allows Java programs, including applets, to be transferred across the internet and run seamlessly on any compatible system with a JVM.

**Features Of Java**

* **Simplicity**: Designed to be easy to learn and use, Java simplifies programming for professionals, especially those familiar with C/C++.
* **Object-Oriented**: Java’s object model is straightforward, balancing object use with non-object performance.
* **Robustness**: Java emphasizes reliability through strict typing, compile-time and runtime error checking, and automatic memory management, making it resilient across diverse platforms.

**JAVASCRIPT**

JavaScript, developed by Netscape, is a script-based language designed for web-based applications: **Client-Side Use**: Primarily used for client-side scripting, JavaScript enhances web pages by adding interactivity directly within browsers.

**Compatibility and Simplicity**: JavaScript is easy to learn, similar to HTML, and can be embedded directly in HTML documents. Its versatility makes it a preferred choice for client-side scripting across most modern browsers.

<SCRIPTS>..</SCRIPT>.

<SCRIPT LANGUAGE = “JavaScript”>

JavaScript statements

</SCRIPT>

**JAVASCRIPT VS JAVA**

* **Distinct Languages:**
  + Java Applets: Typically displayed in a designated area within a web document.
  + JavaScript: Can modify any part of the web document dynamically.
* **Application Complexity:**
  + Java: Suited for complex applications across various platforms.
  + JavaScript: Best for simpler applications and enhancing web interactivity.

**ADVANTAGES OF JAVASCRIPT**

* Versatile Scripting: Can be utilized for both server-side and client-side scripting.
* Flexibility: More adaptable than VBScript.
* Browser Compatibility: Supported by all major web browsers, making it the default client-side scripting language.

**HYPERTEXT MARKUP LANGUAGE (HTML)**

* **Definition**: HTML is the standard language for creating web pages that incorporate text, graphics, and hyperlinks.
* **Not a Programming Language**: Based on SGML, HTML is a markup language that structures content for the web, allowing non-linear navigation through hyperlinks.
* **Features:**
  + **Versatility:** Works across various platforms and can display documents from remote servers.
  + **Tag System:** Uses tags (special codes) to format content, which are case-insensitive. Tags help create visually appealing documents with diverse graphics, fonts, sizes, and colors.
* **Navigation:** Hyperlinks enable users to jump to different sections or documents based on their interests.

**Basic HTML Tags :**

**<!-- -->** Specifies comments

**<A>……….</A>** Creates hypertext links

**<B>……….</B>** Formats text as bold

**<BIG>……….</BIG>** Formats text in large font.

**<BODY>…</BODY>** Contains all tags and text in the HTML document

**<CENTER>...</CENTER>** Creates text

**<DD>…</DD>** Definition of a term

**<DL>...</DL>**  Creates definition list

**<FONT>…</FONT>** Formats text with a particular font

**<FORM>...</FORM>** Encloses a fill-out form

**<FRAME>...</FRAME>** Defines a particular frame in a set of frames

**<H#>…</H#>** Creates headings of different levels

**<HEAD>...</HEAD>** Contains tags that specify information about a document

**<HR>...</HR>** Creates a horizontal rule

**<HTML>…</HTML>** Contains all other HTML tags

**<META>...</META>** Provides meta-information about a document

**<SCRIPT>…</SCRIPT>** Contains client-side or server-side script

**<TABLE>…</TABLE>**  Creates a table

**<TD>…</TD>** Indicates table data in a table

**<TR>…</TR>** Designates a table row

**<TH>…</TH>** Creates a heading in a table

**ADVANTAGES**

* A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.
* HTML is platform independent.
* HTML tags are not case-sensitive.

**Java Database Connectivity (JDBC)**

**What is JDBC?**

* JDBC is a Java API that facilitates the execution of SQL statements, allowing for database access through a set of classes and interfaces. It enables developers to write database applications in pure Java, ensuring cross-platform compatibility ("write once, run anywhere").

**Functions of JDBC:**

1. Establishing a connection with a database.
2. Sending SQL statements.
3. Processing the results returned from the database.

**JDBC vs ODBC:**

* **ODBC** (Open Database Connectivity) is a widely used API for accessing relational databases, but it’s not optimal for Java due to its C interface, which introduces security and portability issues.
* JDBC provides a more secure, object-oriented alternative tailored for Java programmers, making it easier to learn and use while maintaining a pure Java solution.

**Database Access Models:**

* **Two-Tier Model**: A Java application communicates directly with the database, using a JDBC driver for SQL statements and results in a client-server configuration.
* **Three-Tier Model**: Commands are sent to a "middle tier" that interacts with the database, allowing for better control and performance management. This model can be implemented in Java for improved robustness and security.

**JDBC Driver Types:**

1. JDBC-ODBC Bridge plus ODBC driver.
2. Native-API partly-Java driver.
3. JDBC-Net pure Java driver.
4. Native-protocol pure Java driver.

**Java Server Pages (JSP)**

**Overview:**

* JSP is a technology for creating dynamic web pages by separating content generation from presentation, enabling web designers and application developers to work independently.

**Features:**

* **Portability**: JSP files can run on any web server or application server that supports them.
* **Reusable Components**: Supports embedding Java Beans and Servlets within JSP files.
* **Processing**: JSP files are essentially HTML documents with embedded JSP tags, processed into Servlets on the server side.

**Execution Steps of a JSP Application:**

1. The client requests a JSP file from the web server.
2. The server passes the request to the JSP engine.
3. The JSP engine converts JSP tags into a Servlet, which is executed and returns HTML to the client.

**JDBC Connectivity in JSP:**

* JDBC allows JSP applications to connect to various databases, manage transactions, execute SQL statements, and handle stored procedures.

**TOMCAT 6.0 WEB SERVER**

**Overview:**

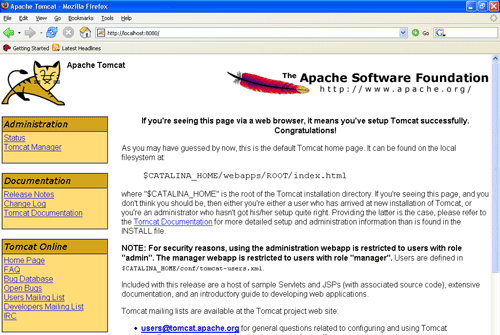
* Apache Tomcat is an open-source web server and servlet container for Java Servlet and JSP technologies.
* It provides support for web components and acts as a reference implementation for Java standards.

**Web Server vs Application Server:**

* **Web Server**: Only supports web components (e.g., Tomcat).
* **Application Server**: Supports both web components and business logic (e.g., BEA WebLogic).

**Development Requirement:**

* To develop web applications using JSP and Servlets, a web server like Tomcat must be installed to run the applications.



**IMPLEMENTATION**

* **Admin**

In this module, the Admin has to login by using valid user name and password. After login successful he can perform some operations such as View All Users And Authorize, View All Friend Request and Response, Add Health Filter, View All Health Tweets with Discussion Comments, Capture and View Different Health Monitoring for different geographic regions, Capture and View Different Health Monitoring Based On Disease, View Number of Same Disease in Chart, View Health Tweet Scores in Chart

**Friend Request & Response**

In this module, the admin can view all the friend requests and responses. Here all the requests and responses will be displayed with their tags such as Id, requested user photo, requested user name, user name request to, status and time & date. If the user accepts the request then the status will be changed to accepted or else the status will remains as waiting.

* **User**

In this module, there are n numbers of users are present. User should register before performing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Verify finger print and Login Once Login is successful user can perform some operations like My Profile, Search Friend Track and Find Friend Request, View All My Friends, Create Your Health Tweet, View All My Health Tweets, View and Monitor All My Friends Health Tweets.

**Searching Users to make friends**

In this module, the user searches for users in Same Network and in the Networks and sends friend requests to them. The user can search for users in other Networks to make friends only if they have permission.

**Architecture Diagram**

Admin

View All Users And Authorize

View All Friend Request and Response

Add Health Fileter

View All Health Tweets with Discussion Comments

Capture and View Different Health Monitoring for different geographic regions

Capture and View Different Health Monitoring Based On Disease

View Number of Same Disease in Chart

View Health Tweet Scores in Chart

Accepting all user Information

**Admin**

View user data details

Authorize the Admin

Process all User queries

**Store and retrievals**

**WEB Database**

Registering the User

**User**

My Profile

Search Friend Track and Find Friend Request

View All My Friends

Create Your Health Tweet

View All My Health Tweets

View and Monitor All My Friends Health Tweets

* **Data Flow Diagram** :

Admin

Response

System

Request

User

.

* **Class Diagram :**

**Admin**

View All Users And Authorize, View All Friend Request and Response, Add Health Filter, View All Health Tweets with Discussion Comments, Capture and View Different Health Monitoring for different geographic regions, Capture and View Different Health Monitoring Based On Disease,View Number of Same Disease in Chart, View Health Tweet Scores in Chart

Health Tweet Name, Select Tweet image, Health Details Description, Tweet Owner, Tweet Data and Time, Retweet

Methods

Members

**Login**

**Register**

Login, Register

User Name, Password

Register (), Reset ()

User Name, Password, E-mail, Mobile, Address, DOB, Gender, Pin code, Image

Login (), Reset (), Register ().

User Name, Password.

Methods

Methods

Members

Members

**User**

My Profile, Search Friend Track and Find Friend Request, View All My Friends, Create Your Health Tweet, View All My Health Tweets, View and Monitor All My Friends Health Tweets

Health Tweet Name, Select Tweet image, Health Details Description, Tweet Owner, Tweet Data and Time, Retweet

Methods

Members

* **Sequence Diagram**

User

Admin

Web Server

View all Users and authorize

View all Users and authorize

View all Users and authorize

Register and Login, View Profile

Search Friends, Track and Find Friend Request

Add Health Filter

Create Your Health Tweet

View All Health Tweets with Discussion Comments

View Revisited product pages



Capture and View Different Health Monitoring for different geographic regions

View All My Health Tweets

View and Monitor All My Friends Health Tweets

Capture and View Different Health Monitoring Based On Disease

View Number of Same Disease

* **Flow Chart : User**

User Register

Start

Login

Yes No

Username & Password Wrong

View users Profile

Search Friend Track and Find Friend Request

View All My Friends

Create Your Health Tweet

Logout

View All My Health Tweets

View and Monitor All My Friends Health Tweets

**Flow Chart : Admin**

**Start**

**Admin Login**

**Login**

Yes No

View all authorized users

**Username & Password Wrong**

View all friend and request and response

**Log Out**

Add Health Filter

View All Health Tweets with Discussion Comments

Capture and View Different Health Monitoring for different geographic regions

Capture and View Different Health Monitoring Based On Disease

View Number of Same Disease in Chart

View Number of Same Disease in Chart

**Software Environment**

## **Java Technology**

Java technology is both a programming language and a platform.

### The Java Programming Language

### **The Java programming language is a high-level language that can be characterized by all of the following buzzwords:**

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



Java bytecodes serve as the machine code instructions for the Java Virtual Machine (JVM). This enables the "write once, run anywhere" capability of Java. By compiling a program into bytecodes on any platform with a Java compiler, these bytecodes can then be executed on any JVM implementation. As a result, programs written in Java can run on various operating systems—such as Windows, Solaris, or macOS—as long as a compatible JVM is present.



## The Java platform is unique as it operates as a software-only environment that runs on top of existing hardware-based platforms, such as Windows, Linux, Solaris, and macOS. It consists of two primary components:

## **Java Virtual Machine (JVM):** This is the core of the Java platform, enabling Java programs to be executed on any hardware by providing a virtual environment.

## **Java Application Programming Interface (Java API):** This is a comprehensive collection of pre-built software components that offer a wide range of functionalities, including graphical user interface (GUI) widgets. The API is organized into libraries, known as packages, which group related classes and interfaces.

Together, the Java API and JVM create an abstraction layer that allows Java programs to run independently of the underlying hardware, ensuring portability and ease of use across different systems.



Native code is specific to a particular hardware platform and, once compiled, runs directly on that platform. In contrast, Java's platform-independent nature may result in slower performance compared to native code. However, advancements such as smart compilers and just-in-time (JIT) bytecode compilers help optimize performance, making it competitive with native code while maintaining portability.

**What Can Java Technology Do?**

Java technology supports various types of programs, primarily through **applets** and **applications**:

* **Applets**: These are small programs that run in a Java-enabled browser, designed to enhance web pages with interactive features.
* **Applications**: These are standalone programs that operate directly on the Java platform. They can include servers, such as web, mail, and print servers, which provide services to clients over a network.
* **Servlets**: These are server-side counterparts to applets. Servlets extend web applications by running on a server, providing dynamic content, and managing requests without the overhead of CGI scripts.

**API Support**

The Java API is essential for enabling diverse program functionalities through various packages, which provide features like:

* **Core functionalities**: Objects, strings, threads, numbers, I/O operations, data structures, and system properties.
* **Applet conventions**: Specific rules that govern how applets operate within a browser.
* **Networking**: Tools for working with URLs, TCP/UDP sockets, and IP addresses.
* **Internationalization**: Support for adapting programs to different languages and locales.
* **Security**: Comprehensive security features, including electronic signatures and access control.
* **JavaBeans**: Reusable software components that can integrate with existing architectures.
* **Object Serialization**: Allows data persistence and communication using Remote Method Invocation (RMI).
* **Java Database Connectivity (JDBC)**: Facilitates uniform access to various relational databases.

Additionally, the Java platform includes APIs for graphics, accessibility, collaboration, telephony, and more, making it a versatile environment for developers.



**How Will Java Technology Change My Life?**

Learning the Java programming language can enhance your programming experience in several ways:

* **Quick Start**: Java's object-oriented nature makes it easy to learn, especially for those familiar with C or C++.
* **Reduced Code Size**: Programs written in Java can be significantly smaller—up to four times less code than equivalent programs in C++.
* **Better Code Quality**: Java promotes good coding practices and includes garbage collection, reducing the likelihood of memory leaks. Its component architecture and extensive API enable code reuse and lower the chances of introducing bugs.
* **Faster Development**: Java can cut development time in half compared to C++, due to simpler syntax and fewer lines of code.
* **Platform Independence**: Java supports "write once, run anywhere" through machine-independent bytecode, allowing programs to run consistently across different Java platforms.
* **Easier Software Distribution**: Applets can be upgraded easily from a central server, allowing new classes to be loaded dynamically without recompiling the entire program.

**ODBC**

**Open Database Connectivity (ODBC)** is a standard programming interface that allows application developers to connect with various database systems without worrying about the underlying database syntax. This simplifies database interactions and allows applications to switch between databases easily.

Key Features:

* **Data Source Management**: ODBC Administrator enables configuration of data sources, linking specific databases to applications.
* **Database Independence**: Developers can write applications that work across different databases (like Oracle or SQL Server) without changing the source code.
* **Ease of Use**: The loading of ODBC drivers is seamless, handling many network-related issues for the programmer.

**Drawback**: While ODBC provides many advantages, it can be slower than direct database connections. However, advancements in driver quality have improved performance.

**JDBC**

**Java Database Connectivity (JDBC)** is an API developed by Sun Microsystems to standardize database access in Java applications. It offers a consistent interface for connecting to various relational databases.

**JDBC Goals:**

1. **SQL Level API**: Defines a SQL interface for Java, enabling the creation of higher-level tools and APIs.
2. **SQL Conformance**: Allows passing various SQL queries to accommodate differences among database vendors.
3. **Integration with Existing APIs**: JDBC sits on top of other common SQL APIs, enabling the use of existing ODBC drivers.
4. **Consistency with Java**: Maintains alignment with the overall Java system design.
5. **Simplicity**: Features a straightforward design to minimize confusion.
6. **Strong Typing**: Utilizes strong static typing for better compile-time error checking.
7. **Ease of Common Use Cases**: Simplifies common SQL operations (e.g., SELECT, INSERT, DELETE) while still supporting complex queries.

For our project, we will implement Java Networking and utilize an MS Access database for dynamically updating the cache table.

**Overview of Java**

Java encompasses two key components: a programming language and a platform.

Characteristics of Java:

* Simple: Designed to be easy to learn and use.
* Object-oriented: Supports principles of encapsulation, inheritance, and polymorphism.
* Architecture-neutral: Java programs can run on any device with a Java Virtual Machine (JVM).
* Portable: Write once, run anywhere due to platform-independent bytecode.
* Distributed: Facilitates distributed computing across networks.
* High-performance: Optimized through techniques like Just-In-Time (JIT) compilation.
* Interpreted: The bytecode is interpreted by the JVM, allowing for flexibility and ease of debugging.
* Multithreaded: Supports concurrent execution of threads for efficient program execution.
* Robust: Strong memory management and error-handling features enhance reliability.
* Dynamic: Can adapt to changing environments and features dynamic linking of libraries.
* Secure: Built-in security features protect against threats and malicious code.

Compilation and Interpretation

Java's unique execution model involves both compilation and interpretation:

* Compilation: The Java source code is translated into an intermediate format known as Java bytecode, which is platform-independent. This step occurs only once.
* Interpretation: Each time the program runs, the JVM interprets the bytecode and executes it. This allows for flexibility and ease of running Java applications on different platforms.

This dual approach enhances Java's portability and adaptability across various computing environments.

**Java Program**

**Compilers**

**Interpreter**

**My Program**

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

## **Networking**

### TCP/IP stack

The TCP/IP stack is shorter than the OSI one:



TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

### **IP Datagrams and Protocols**

### **IP Layer**

### The **IP (Internet Protocol)** layer provides a **connectionless** and **unreliable** delivery system, treating each datagram independently.

### It includes:

### **Checksum**: Ensures data integrity for the header.

### **Source and Destination Addresses**: Required for routing.

### The IP layer handles:

### **Routing**: Directing datagrams through the Internet.

### **Fragmentation**: Breaking large datagrams into smaller packets for transmission and reassembling them at the destination.

### **UDP (User Datagram Protocol)**

### **Connectionless**: No established connection before sending data.

### **Unreliable**: Does not guarantee delivery of datagrams.

### Adds:

### **Checksum**: Validates the contents of the datagram.

### **Port Numbers**: Supports a client/server model by identifying specific processes.

### **TCP (Transmission Control Protocol)**

### Provides a **reliable**, **connection-oriented** communication above the IP layer.

### Establishes a **virtual circuit** between two processes, ensuring data is delivered accurately and in order.

### **Internet Addressing**

### The Internet uses a **32-bit address** scheme for locating machines, known as the **IP address**.

### The address consists of:

### **Network ID**: Identifies the network.

### **Host Address**: Identifies the specific device within that network.

### **Address Classes**

### **Class A**: 8 bits for the network address, 24 bits for other addressing (large networks).

### **Class B**: 16 bits for the network address (medium-sized networks).

### **Class C**: 24 bits for the network address (small networks).

### **Class D**: Uses all 32 bits, typically for multicast addresses.

### **Subnet Addressing**

### Networks can be divided into **subnets** for better organization.

### Example: A UNIX network may use **10-bit addressing** for subnets, allowing for **1024 different hosts**.

### **Host Addressing**

### **8 bits** are used for host addresses within each subnet, limiting the number of machines to **256** per subnet.

### **Total address**



The 32 bit address is usually written as 4 integers separated by dots.

**Port Addresses**

* A **port address** is a **16-bit number** that identifies a specific service on a host.
* When sending a message to a server, it is directed to the port associated with that service on the host, which is not location transparent.
* Certain port numbers are designated as **"well known"** ports, reserved for specific services (e.g., HTTP on port 80, HTTPS on port 443).

**Sockets**

* A **socket** is a system-managed data structure used to handle network connections.
* Sockets are created using the socket() system call, which returns an integer akin to a file descriptor.

**JFreeChart**

JFreeChart is a free, open-source Java chart library that enables developers to create professional-quality charts for applications. Its key features include:

* Consistent and Well-Documented API: Supports various chart types.
* Flexible Design: Easily extendable for server-side and client-side applications.
* Output Support: Can generate Swing components, image files (PNG, JPEG), and vector graphics (PDF, EPS, SVG).
* Licensing: Distributed under the GNU Lesser General Public License (LGPL), allowing use in proprietary applications.

Project Ideas:

1. Map Visualizations: Create charts related to geographical data (e.g., population density, income per capita).
   * Tasks include sourcing vector outlines, creating datasets, and integrating with the existing XYPlot class in JFreeChart.
2. Time Series Chart Interactivity: Develop interactive time series charts featuring a control that shows a small version of all data with a sliding view rectangle for selecting data subsets.
3. Dashboards: Design a dashboard mechanism that supports various JFreeChart types (dials, pies, thermometers, bars, lines) for easy delivery via Java Web Start and applets.
4. Property Editors: Extend the property editor mechanism to allow greater end-user control over chart appearance.

**SYSTEM TESTING**

Purpose of Testing:  
The primary goal of testing is to identify and discover errors within a software product. It involves a systematic process to assess the functionality of various components, subassemblies, assemblies, or the final product, ensuring that the software meets specified requirements and user expectations while avoiding unacceptable failures.

**Types of Tests**

1. **Unit Testing:**
   * **Definition:** Validates that individual software units function correctly and that inputs yield valid outputs.
   * **Focus**: Tests all decision branches and internal code flows of the application.
   * **Characteristics:** Conducted after completing a unit, it is structural and invasive, requiring knowledge of the unit's construction.
   * **Objective:** Ensure each unique business process path performs as documented with clear inputs and expected results.
2. **White Box Testing:**
   * Definition: Testing where the tester has knowledge of the internal structure and logic of the software.
   * Purpose: Targets areas that cannot be assessed from a black-box perspective.
3. **Black Box Testing:**
   * Definition: Testing conducted without knowledge of the internal workings of the module.
   * Focus: Provides inputs and observes outputs, treating

**SYSTEM TESTING**

### TESTING METHODOLOGIES

The following are the Testing Methodologies:

* **Unit Testing.**
* **Integration Testing.**
* **User Acceptance Testing.**
* **Output Testing.**
* **Validation Testing.**

**Unit Testing**

Unit testing verifies the functionality of individual modules by checking their internal logic and ensuring they produce valid outputs. It focuses on:

* Individual Module Verification: Testing each module in isolation.
* Interface Consistency: Ensuring module interfaces align with design specifications.
* Path Testing: Validating all significant processing and error-handling paths.

**Integration Testing**

Integration testing assesses the interaction between combined modules after unit testing. It can be performed using:

1. Top-Down Integration: Begins with the main module and integrates subordinate modules incrementally, using stubs for untested components.
2. Bottom-Up Integration: Starts with low-level modules, which are tested first, eliminating the need for stubs. A driver program coordinates input and output during testing.

**User Acceptance Testing (UAT)**

UAT involves end-user participation to ensure the system meets their needs. It emphasizes user-friendly interfaces and incorporates feedback for improvements.

**Output Testing**

Output testing verifies the system produces required outputs in specified formats by engaging users to understand their expectations for both screen and printed outputs.

**Validation Checking**

Validation checks ensure:

* Text Fields: Limit character counts and types.
* Numeric Fields: Accept only valid numbers.

Modules are tested for accuracy using sample data to identify defects and validate outputs.

Preparation of Test Data

Test data preparation involves:

* Collecting Varied Data: To cover different scenarios.
* Executing Tests: Using the data to uncover errors.
* Documenting Errors: Noting issues for correction and future reference.

Successful tests confirm that the system operates correctly and meets user requirements.

**Using Live Test Data**

Live test data are real data extracted from organizational files, often collected from users during their normal activities. This method provides realistic insights into how the system will function. However, it has limitations:

* Limited Availability: It can be challenging to obtain enough live data for thorough testing.
* Typical Bias: Live data may not cover all combinations or formats, potentially overlooking edge cases that could lead to system failures.

**Using Artificial Test Data**

Artificial test data are specifically generated for testing purposes, allowing comprehensive coverage of all possible input combinations. Benefits include:

* Thorough Testing: Artificial data can be designed to explore all program paths and edge cases.
* Independent Generation: It’s more effective when created by testers who weren’t involved in writing the code, leading to unbiased testing.

**User Training**

User training is essential for new systems to ensure users understand how to operate the system effectively. Demonstrations were conducted to show the project's functionality, emphasizing its ease of use, especially for users with computer knowledge.

**Maintenance**

Maintenance encompasses various activities, including fixing code and design errors. To minimize future maintenance needs, user requirements were clearly defined during development. The system is designed to be simple and adaptable, allowing for potential feature enhancements as technology evolves. This approach will facilitate easier maintenance in the long run.

**Testing Strategy**

The testing strategy for a system integrates test cases and design techniques into a structured process aimed at successfully constructing software. Key components include:

* Planning: Establishing a comprehensive test plan that outlines the scope, resources, and timeline for testing.
* Test Case Design: Developing detailed test cases that address both low-level tests for specific code segments and high-level tests for major system functions.
* Test Execution: Carrying out the tests according to the plan, ensuring that all functionalities are thoroughly evaluated.

**RESULTS**

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a device

Description automatically generated**

**A screenshot of a yellow grid

Description automatically generated**

**A screenshot of a graph

Description automatically generated**

**CONCLUSION**

We develop methods to uncover ailments over time from social media. We formulated health transition detection and prediction problems and proposed two models to solve them. Detection is addressed with TM–ATAM, a granularity-based model to conduct region-specific analysis that leads to the identification of time periods and characterizing homogeneous disease discourse, per region. Prediction is addressed with T–ATAM,that treats time natively as a random variable whose values are drawn from a multinomial distribution. The fine-grained nature of T–ATAM results insignificant improvements in modeling and predicting transitions of health-related tweets. We believe our approach is applicable to other domains with time-sensitive topics such as disaster management and national security matters.

**REFERENCES**

[1] L. Manikonda and M. D. Choudhury, “Modeling and understanding visual attributes of mental health disclosures in social media,”in Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017., 2017, pp.170–181.

[2] S. R. Chowdhury, M. Imran, M. R. Asghar, S. Amer-Yahia, and C. Castillo, “Tweet4act: Using incident-specific profiles for classifying crisis-related messages,” in 10th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Baden-Baden, Germany, May 12-15, 2013., 2013.

[3] T. Davidson, D. Warmsley, M. W. Macy, and I. Weber, “Automated hate speech detection and the problem of offensive language,” in Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017.,2017, pp. 512–515.

[4] M. J. Paul and M. Dredze, “You Are What You Tweet: Analyzing Twitter for Public Health,” in ICWSM’11, 2011.

[5] T. Hofmann, “Probabilistic Latent Semantic Indexing,” in SIGIR’99, 1999, pp. 50–57.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet Allocation,”Journal of Machine Learning, vol. 3, pp. 993–1022, 2003.

[7] Y. Wang, E. Agichtein, and M. Benzi, “TM-LDA: Efficient Online Modeling of Latent Topic Transitions in Social Media,” in KDD’12,2012, pp. 123–131.

[8] S. Sidana, S. Mishra, S. Amer-Yahia, M. Clausel, and M. Amini,“Health monitoring on social media over time,” in Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July17-21, 2016, 2016, pp. 849–852.

[9] D. M. Blei and J. D. Lafferty, “Dynamic Topic Models,” in ICML’06,2006, pp. 113–120.

[10] C. X. Lin, Q. Mei, J. Han, Y. Jiang, and M. Danilevsky, “The Joint Inference of Topic Diffusion and Evolution in Social Communities,”in ICDM’11, 2011, pp. 378–387.

[11] X. Wang and A. McCallum, “Topics Over Time: A Non-Markov Continuous-time Model of Topical Trends,” in KDD’06, 2006, pp.424–433.

[12] K. W. Prier, M. S. Smith, C. Giraud-Carrier, and C. L. Hanson,“Identifying Health-related Topics On Twitter,” in Social computing, behavioral-cultural modeling and prediction. Springer, 2011, pp.18–25.