

NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY

**MILITARY COLLEGE OF SIGNALS**



**Object Oriented Programming (OOP)**  
**(CS-212)**

**( PROJECT DOCUMENTATION )**

- **Submitted by:** MUHAMMAD AHMAD SULTAN
  
- **RANK:** NC                                      ○ **CMS ID:** 408709
- **COURSE:** BESE-28                      ○ **SECTION:** C
  
- **DEPARTMENT:** COMPUTER SOFTWARE ENGINEERING
  
- **Submitted to:** Dr. Nauman Ali Khan
- **DATED:** 29-05-2023



## **TITLE OF THE PROJECT**



➤ **Object-Oriented Ping Pong  
Game in Java**

➤ **COLLEGE MANAGEMENT  
SYSTEM**

➤ **OBJECT ORIENTED  
APPLICATION: *TALK BUDDY***

# About me

Allow me to introduce myself. I am *Muhammad Ahmad Sultan*, a highly motivated first-year student at the ***Military College of Signals***, currently pursuing a **Bachelor's degree in Software Engineering**. With unwavering determination, I am committed to achieving my goals and making significant strides in my academic journey.



**Muhammad Ahmad Sultan**

Developer

# **Object-Oriented Ping Pong Game in Java**

## Table of Contents

✚ Introduction (Abstract & History) .....	5
✚ Project Overview .....	6
✚ Objective of the Project .....	7
✚ Project Scope .....	8
✚ Project Plan .....	12
✚ Project Timeline .....	13
✚ Project Budget .....	13
✚ Project Team .....	13
✚ Functionality & Implementation .....	14
✚ Module Specific Requirements .....	14
✚ Input Specifications .....	15
✚ Output Specifications .....	16
✚ Algorithm Details .....	18
✚ Procedure (Algorithm) .....	19
✚ Requirements of the Project .....	20
✚ Tools/Platforms & Languages used for Project .....	21
✚ Execution .....	23
✚ FlowChart of Pong Game .....	24
✚ Java Classes .....	26
✚ OOP Principles & Concepts .....	28
✚ Display .....	30
✚ Potential Questions .....	31
✚ Conclusion .....	32

# **College Management System**

## Table of Contents

✚ Introduction (Abstract) .....	34
✚ Project Overview .....	34
✚ Objective of the Project .....	35
✚ Project Scope .....	37
✚ Project Plan .....	40
✚ Project Timeline .....	41
✚ Project Budget .....	41
✚ Project Team .....	41
✚ Functionality & Implementation .....	42
✚ Input Specifications .....	42
✚ Output Specifications .....	43
✚ Requirements of the Project .....	44
✚ Overview of Front-End .....	45
✚ Overview of Back-End .....	45
✚ Execution .....	46
✚ FlowChart of CMS .....	46
✚ Java Classes .....	47
✚ DataBase Queries .....	47
✚ OOP Principles & Concepts .....	48
✚ Display .....	60
✚ Potential Questions .....	59
✚ Conclusion .....	60
✚ Reference Citations .....	60

# **Object-Oriented Application: *Talk Buddy***

## Table of Contents

✚ Introduction (Abstract) .....	62
✚ Project History .....	62
✚ Objective of the Project .....	63
✚ Project Scope .....	64
✚ Project Plan.....	67
✚ Project Timeline.....	68
✚ Project Budget .....	68
✚ Project Team .....	68
✚ Functionality & Implementation .....	69
✚ Input Specifications .....	69
✚ Output Specifications.....	69
✚ Requirements of the Project.....	70
✚ Socket Programming.....	71
✚ Execution .....	72
✚ Java Classes.....	72
✚ OOP Principles & Concepts .....	74
✚ Display .....	78
✚ Potential Questions .....	79
✚ Conclusion .....	80
✚ Reference Citations .....	80

# Introduction:

## *Abstract:*

The **Object-Oriented Ping Pong Game in Java** is a project aimed at developing a two-player game that simulates the classic game of ping pong. Using the **Java programming language and object-oriented programming concepts**, this project provides a virtual platform for players to enjoy the game from the comfort of their own homes. The game incorporates features such as a user-friendly interface, scoring system, and game-over screen to engage and entertain players. **By implementing OOP concepts**, the code is structured and easier to maintain. This project not only offers a fun and interactive way to play ping pong but also serves as an opportunity for developers to practice and enhance their object-oriented programming skills in Java.

## *History:*

Pong is a classic arcade video game that was first released in **1972** and became an instant hit, paving the way for the video game industry. Pong was developed by **Atari, Inc.**, a video game company founded by **Nolan Bushnell and Ted Dabney in 1972**. Ping pong is a popular game that is played by people of all ages. It is a game that requires quick reflexes and excellent hand-eye coordination.



## Project Overview:

The objective of this project is to create an object-oriented Ping Pong game in Java. The game will be designed for two players and will involve moving paddles to hit a ball back and forth across a screen. The game will be developed using Java programming language and will incorporate object-oriented programming concepts such as encapsulation, inheritance, and polymorphism.



The game will consist of **a playing field with two paddles, a ball, and a score board**. The paddles will be controlled by the players using the keyboard arrow keys. The ball will move in a straight line at the start of each round and will change direction when it hits a paddle or a wall.

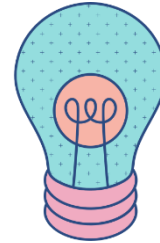
The game will have a scoring system that keeps track of the score for each player. The score board will be displayed on the screen during the game. When a player reaches the winning score, the game will end and a game-over screen will be displayed.

**To achieve the goal of developing an object-oriented Ping Pong game in Java,** the project will implement various object-oriented programming concepts. Encapsulation will be used to hide the implementation details of the game objects from other objects. Inheritance will be used to create subclasses of the game objects with specialized functionality. Polymorphism will be used to allow objects to take on different forms depending on the context in which they are used.

By creating a Ping Pong game using object-oriented programming concepts in Java, the project will provide **the developer** with a better understanding of how to apply these concepts in practice. **The end result will be an engaging and interactive game that can be enjoyed by players of all skill levels.**



## **Objective (Aim):**



## **Game Objective:**

The aim of the ping pong game is for players to score points by hitting the ball with their paddle and preventing the ball from hitting their side of the playing area. The player with the highest score at the end of the game wins.

## **Project Objective:**

The main objective of this project is to develop an object-oriented ping pong game in Java that implements key concepts such as encapsulation, inheritance, and polymorphism. Specifically, the project aims to:

- Ensure that the gameplay is easy to understand and intuitive for players of all skill levels
- Design the game to be visually appealing with appropriate graphics and animations
- Allow players to customize game settings such as game speed and paddle size
- Implement error handling to prevent crashes and improve the stability of the game
- Optimize the game's performance to ensure smooth gameplay even on slower machines
- Provide players with the option to save their high scores and view a leaderboard
- Include a pause and resume feature to allow players to take breaks during gameplay.
- Create a user-friendly interface with clear instructions
- Allow two players to play simultaneously
- Develop a paddle that can be moved up and down by the players
- Implement a ball that bounces off the walls and paddles
- Create a scoring system that keeps track of the score for each player
- Design a game-over screen that displays the winner

# **Project Scope:**



The project will include the following features:

- **Designing the User-friendly Interface:**

Creating a user-friendly interface involves designing visually appealing and intuitive screens for the start, game, and game-over screens. This includes selecting appropriate fonts, colors, and graphical elements to provide clear instructions, display the game elements (e.g., paddles, ball, score board), and present the game-over information in an aesthetically pleasing manner. The interface should be easy to navigate and understand for players of all skill levels.

- **Implementing Multiplayer Mode:**

This task involves creating the necessary logic and components to support multiplayer mode in the game. It includes setting up input handling for both players, synchronizing paddle movement, and managing interactions between the players' paddles and the ball. The goal is to ensure that both players can control their own paddles independently and compete against each other on the same screen.

- **Handling Paddle Movement:**

In this task, the focus is on implementing the logic to handle paddle movement based on player input. The keyboard arrow keys will be used to move the paddles up and down. The task involves capturing the keyboard input, updating the position of the corresponding paddle, and ensuring that the paddle stays within the boundaries of the playing field. The movement should be smooth and responsive to provide a seamless gameplay experience.

- **Handling Game Over:**

When a player reaches the winning score, the game needs to handle the end condition and transition to the game-over screen. This involves detecting when a player reaches the winning score, stopping the game, displaying the game-over screen with the winner's information, and providing an option to play again. Additionally, any necessary clean-up or reset operations should be performed to prepare for a new game.

- **Refactoring & Optimization:**

This task involves reviewing and improving the existing codebase to enhance its structure, efficiency, and maintainability. It may include identifying and eliminating code duplication, improving algorithm efficiency, optimizing resource usage, and applying design patterns or best practices. The goal is to make the codebase more organized, readable, and performant while reducing potential issues or bugs.

- **Managing Ball Movement:**

This task focuses on implementing the logic for ball movement within the game. The ball should move in a straight line initially and change direction upon colliding with the paddles or walls. The collision detection and response algorithms need to be implemented to ensure accurate ball movement and realistic bouncing behavior. The goal is to provide a smooth and challenging gameplay experience.

- **Documentation:**

Creating comprehensive documentation is essential for the project. It involves writing clear and concise explanations of the code, design decisions, and implementation details. The documentation should include instructions on how to set up and run the game, explanations of the game's features and mechanics, and any additional information that would help users or future developers understand and work with the codebase effectively.

- **Creating a Scoring System:**

The scoring system task involves implementing the logic to track and update the scores for each player. Points will be awarded when the opponent fails to hit the ball. The score board should be updated and displayed on the screen during the game, showing the current scores for both players. The task includes designing the score board UI, updating the scores based on the game events, and handling the conditions for reaching the winning score and ending the game.

## **Not in Scope:**

- **Tournament Mode:**

Implementing a tournament mode would involve creating a structured tournament system where players can compete in a series of matches to determine an overall winner. This feature goes beyond the scope of a basic object-oriented Ping Pong game as it requires additional functionalities such as managing multiple rounds, organizing brackets or groups, and tracking tournament progress.

- **AI Opponent:**

Introducing an AI opponent would involve developing an intelligent computer-controlled player that can compete against human players. This feature would require implementing artificial intelligence algorithms to control the movements and decision-making of the AI opponent. While it adds an interesting challenge to the game, it is a more advanced feature that goes beyond the initial scope of the project.

- **Networking and Database Integration:**

Adding networking capabilities would enable players to connect and play the game with others over a network. This would involve implementing network communication protocols, establishing connections between players, and synchronizing game state between different instances of the game running on different machines. Database integration could be utilized to store player profiles, high scores, or other relevant information. Both networking and database integration involve more advanced concepts and require additional infrastructure and development beyond a basic object-oriented Ping Pong game.

- **Online Multiplayer:**

Enabling online multiplayer would allow players to compete against each other over the internet. This would require implementing server-client architecture, handling matchmaking, managing game sessions, and ensuring smooth gameplay experience with low latency. Online multiplayer is a complex feature that involves not only game development but also server infrastructure, network programming, and scalability considerations.

- **Power-ups and Special Effects:**

Adding power-ups and special effects would introduce additional gameplay elements to enhance the overall experience. Power-ups could include temporary enhancements to the paddles or ball, such as increased speed or size. Special effects could involve visual and audio effects triggered by certain events or actions in the game. While these features can add excitement and variety to the gameplay, they are considered as extra enhancements and are not fundamental to a basic object-oriented Ping Pong game.

**Note:**

It's important to note that while these features are interesting and may enhance the game, they require additional development time, resources, and expertise. Therefore, if you're working on a project with a specific scope and timeline, it's important to prioritize and focus on the core features that align with the project's objectives

# **Project Plan:**

The project will be divided into the following phases:

## **Design Phase:**

In the design phase, the game's features and requirements will be defined, and the game's overall design will be planned. The following tasks will be completed in this phase:

- Identify the game's key features and requirements.
- Develop a game design that incorporates object-oriented programming concepts such as encapsulation, inheritance, and polymorphism.
- Create user interface (UI) wireframes that outline the game's interface, including menus, screens, and gameplay elements.

## **Implementation Phase:**

In the implementation phase, the game will be developed using Java programming language. The following tasks will be completed in this phase:

- Develop the game's UI using Java Swing or JavaFX.
- Implement the game's logic, including the ball and paddle physics, collision detection, and game rules.
- Integrate sound effects and game music.
- Implement game settings such as difficulty levels, paddle size, and ball speed.
- Create a game-over screen that displays the winner.

## **Testing Phase:**

In the testing phase, the game will be tested for any bugs or issues. The following tasks will be completed in this phase:

- Conduct thorough testing of the game to identify and fix any bugs or issues.
- Test the game on different operating systems and hardware configurations.
- Ensure that the game meets the requirements defined in the design phase.
- Optimize the game's performance to ensure smooth gameplay even on slower machines.

## **Project Category:**

This project falls under the category of Object Oriented Programming (OOP). OOP is a programming paradigm that emphasizes the use of objects and classes to organize and structure code. Java is an example of an OOP language, where all programs are written using OOP concepts.

## **Project Timeline:**

The project timeline is as follows:

- ❖ Design phase: 1 week
- ❖ Implementation phase: 2 weeks
- ❖ Testing phase: 1 week
- ❖ Total project duration: 4 weeks

## **Project Budget:**

This project will be developed using open-source tools and software, therefore there will be no cost involved.

## **Project Team:**

The project will be developed by a team of one developer only.

# Functionality & Implementation:

## **Module-Specific Requirements:**

### **User Interface Module:**

- Design a user-friendly interface for the game.
- Provide clear instructions on how to play the game.
- Allow players to start a new game and exit the game.

### **Game Logic Module:**

- Create a paddle that can be moved up and down by the players.
- Implement a ball that bounces off the walls and paddles.
- Create a scoring system that keeps track of the score for each player.
- Design a game-over screen that displays the winner.

### **Sound Module (Optional):**

- Integrate sound effects for when the ball hits the paddle or the wall.
- Add background music to the game.

### **Network Module (optional):**

- Allow players to play against each other over a network.
- Implement a chat feature for communication between players.



## **Input Specifications:**

- **User Input**

- Keyboard inputs for player movement (up and down)
- Mouse inputs for menu selection

- **Game Input**

- Ball speed and trajectory
- Paddle speed and position
- Score tracking

- **Sound Input (optional)**

- Sound effects for ball bouncing, paddle hitting, and game over
- Background music

- **Network Input (optional)**

- Data transmission between networked players
- Chat messages between players

By defining the input requirements for the ping pong game, we can ensure that the game is able to receive and process the necessary user, game, sound, and network inputs. This will make the game more interactive and engaging for the users.

## **Output Specifications:**

- **User Interface**

The user interface for the Ping Pong Game includes a start menu, a game screen, and a game-over screen.

- **Start Menu**

The start menu allows players to start a new game or exit the game. It includes clear instructions on how to play the game.

- **Game Screen**

The game screen displays the two paddles, the ball, and the current score. The paddles can be moved up and down using the arrow keys. The ball bounces off the walls and paddles and updates the score as it hits the scoring area.

- **Game-Over Screen**

The game-over screen displays the winner and allows players to start a new game or exit the game.

- **Game Logic**

The game logic for the Ping Pong Game includes the movement of the paddles and ball, collision detection, and score keeping.

- **Paddle Movement**

The paddles can be moved up and down using the arrow keys. The movement of the paddles is limited to the top and bottom of the screen.

- **Ball Movement**

The ball bounces off the walls and paddles and updates the score as it hits the scoring area. The ball moves at a constant speed and changes direction when it hits an object.

- **Collision Detection**

Collision detection is used to determine when the ball hits a paddle or the wall. When the ball hits a paddle, it changes direction and increases in speed. When the ball hits the wall, it bounces off the wall and continues in the same direction.

- **Score Keeping**

The score is updated whenever the ball hits the scoring area. The game ends when one player reaches a specified score.

- **Sound Effects (optional)**

The Ping Pong Game includes sound effects for when the ball hits the paddle or the wall. Additionally, background music is played during the game.

- **Network (optional)**

The Ping Pong Game can be played over a network, allowing players to play against each other remotely. The network module includes the ability to connect to the network, send and receive game data, and chat messaging.

By implementing all of these features, the Ping Pong Game provides a fun and engaging user experience for players of all skill levels.

# Algorithm Details:

Here are the main algorithms which will be used :

- **Game Loop Algorithm:** The game loop algorithm is used to keep the game running continuously. It calculates the time between the current and last frame, updates the game state, and then renders the game screen. This algorithm is implemented in the `run()` method of the `GamePanel` class.
- **Collision Detection Algorithm:** This algorithm is used to detect collisions between the ball and the paddles, as well as between the ball and the top/bottom window edges. If a collision is detected, the ball's velocity is updated accordingly. This algorithm is implemented in the `checkCollision()` method of the `GamePanel` class.
- **Movement Algorithm:** This algorithm is used to update the position of the paddles and the ball. It is called in the `run()` method of the `GamePanel` class.
- **Draw Algorithm:** This algorithm is used to render the game screen. It draws the paddles, ball, and score onto the screen. This algorithm is implemented in the `paint()` and `draw()` methods of the `GamePanel` class.
- **Keyboard Input Algorithm:** This algorithm is used to detect when the player presses the up/down arrow keys to move their paddle. It is implemented in the `keyPressed()` and `keyReleased()` methods of the `AL` class (which implements the `KeyListener` interface).
- **Randomization Algorithm:** This algorithm is used to randomly generate the ball's starting position and velocity. It is implemented using the `java.util.Random` class, in the `newBall()` method of the `GamePanel` class.
- **Score Keeping Algorithm:** This algorithm is used to keep track of the score for each player. It is implemented in the `Score` class, which keeps track of each player's score as an integer value. The score is incremented when a player scores a point, and the score is displayed on the game screen using the `draw()` method of the `Score` class.

## Procedure (Algorithm):

First, the **PongGame** class is defined with a main method that creates a **GameFrame** object.

In the **GameFrame** class, a **GamePanel** object is created, added to the frame, and then the frame is initialized with various properties such as its size, title, etc.

In the **GamePanel** class, some constant values are defined such as the game's width and height, the ball diameter, and the paddle width and height. A **Thread** object is also created for game loop execution. The **newBall**, **newPaddles**, and **Score** methods are called to initialize the game objects, set the focus on the game panel, and add a key listener for handling paddle movement.

In the **newBall** method, a **Random** object is created and a **Ball** object is initialized with the ball's initial position and diameter, using the random object to generate a random initial y position.

In the **newPaddles** method, two **Paddle** objects are initialized, one at the left of the game screen and the other at the right.

In the **paint** method, an image is created and then the draw method is called to draw the game objects on the image. Finally, the image is drawn on the panel's graphics object.

In the **draw** method, the draw method of each game object is called to draw the object on the graphics object. The **Toolkit.getDefaultToolkit().sync()** method is called to synchronize the drawing on the screen.

In the **move** method, the move method of each game object is called to update their positions.

In the **checkCollision** method, the ball's collision with the window edges, paddles, and score is checked. If the ball hits the top or bottom of the screen, its vertical velocity is reversed. If it hits

a paddle, its horizontal and vertical velocities are adjusted based on which paddle it hit. If a paddle hits the edge of the screen, its position is adjusted to prevent it from going off screen. If the ball hits the left or right edge of the screen, the score is updated and new paddles and a ball are created.

In the **run method**, the game loop is executed. The loop uses a delta time approach to ensure that the game runs at a consistent frame rate, moving, checking collisions, and repainting the screen for each frame.

Overall, this code implements a simple Pong game using Java Swing, with objects such as paddles and a ball that can be moved and interact with the game's window edges and each other. The game loop ensures smooth game play, and the score is updated as each player scores a point.

## Requirements of Pong Game:

### **Hardware Requirements:**

- **Processor:** Intel Core i3 or higher
- **RAM:** 4 GB or more
- **Storage:** 500 MB or more
- **Graphics card:** Any modern graphics card that supports OpenGL 2.0 or higher
- **Monitor:** 1280x800 or higher resolution display
- **Input devices:** Keyboard and mouse or game controller

## Software Requirements:

- **Operating System:** Windows 10 or any Linux distribution that supports Java
- **Java Development Kit (JDK)** version 11 or higher
- **Integrated Development Environment (IDE):** Eclipse, NetBeans, or IntelliJ IDEA
- **OpenGL 2.0** or higher graphics library

## Game Libraries:

- **JBox2D:** a 2D physics engine for games in Java
- **LWJGL:** Lightweight Java Game Library for OpenGL

## Tools/Platform, Languages To Be Used for this project?

For an OOP Java Ping Pong game project, the following tools/platforms, and languages may be used:

**Java:** Java is a widely used programming language, especially in game development. It is an object-oriented language and is well-suited for building complex games.

**Java Swing:** Java Swing is a GUI toolkit for Java. It provides a set of GUI components for building desktop applications, including games. It is widely used for developing games in Java.

**Eclipse or IntelliJ IDEA:** These are Integrated Development Environments (IDEs) that provide powerful tools for Java development. They offer features such as code completion, debugging, and refactoring, making it easier to develop games.

**Git:** Git is a version control system that allows developers to collaborate on a project and track changes made to the codebase. It is widely used in software development, including game development.

**GitHub:** GitHub is a web-based platform for hosting and sharing code repositories. It provides features such as issue tracking, code review, and collaboration tools, making it easier for developers to work together on a project.

**Sound libraries:** For sound effects and background music, libraries like Java Sound API or JLayer can be used.

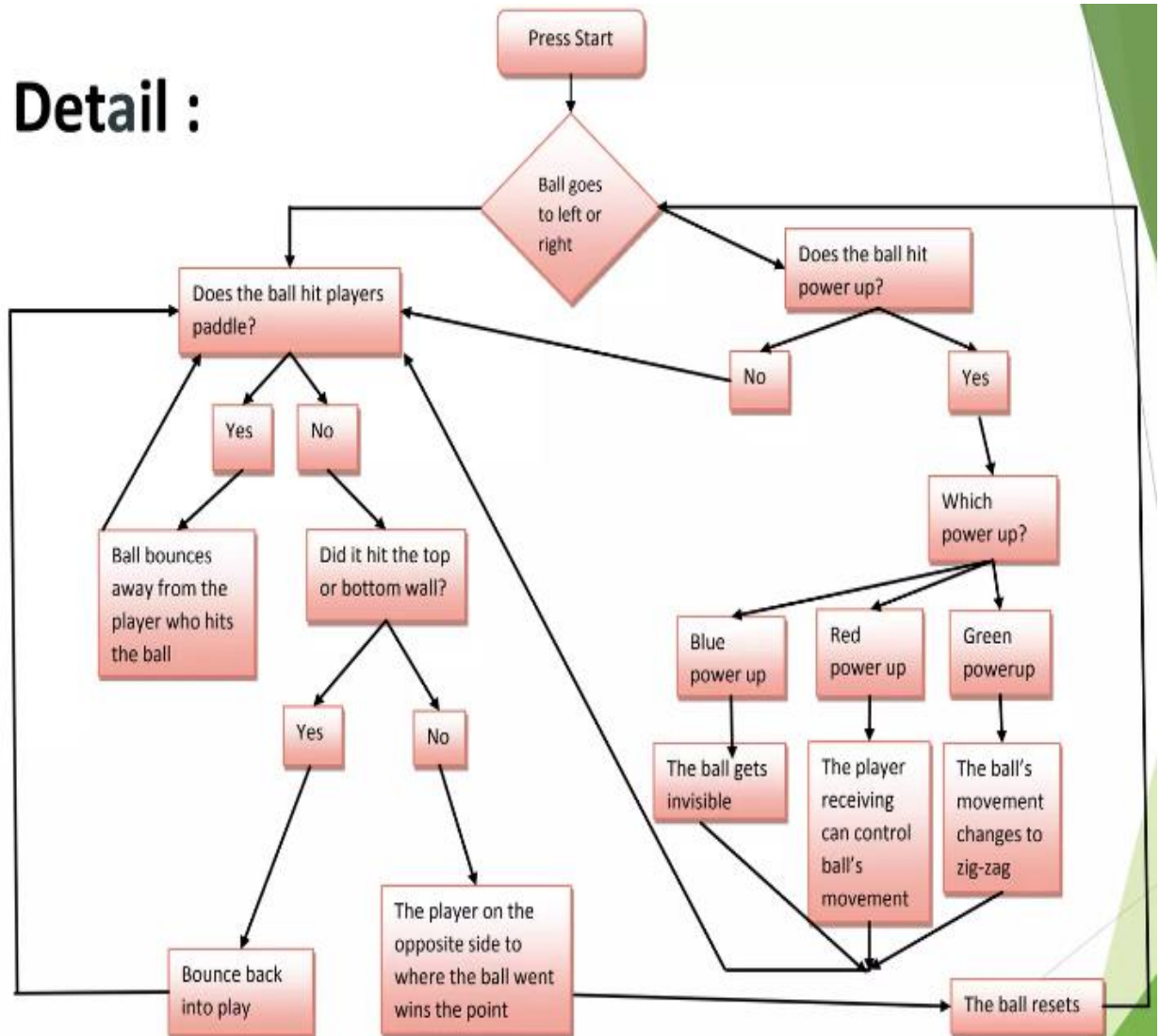
**Overall, using Java with Java Swing for GUI, and an IDE like Eclipse or IntelliJ IDEA for development and Git/GitHub for version control and collaboration would be a good choice for developing an OOP Java Ping Pong game.**



# Execution

## FlowChart of Pong Game:

### Design Detail :



Flowchart of Pong Game

# Java Classes:

- MainWindow.java
- PongGame.java
- Paddle.java
- GamePanel.java
- GameFrame.java
- Ball.java
- Score.java

## MainWindow.java

This class extends JFrame and implements the Runnable interface. It represents the main window of the game. It creates a window with an image, starts a new thread, and handles the animation for the initial appearance of the window. After a certain time, it makes the window invisible and opens a new GameFrame window.

```
ImageIcon i1 = new ImageIcon(ClassLoader.getResource("icons/pingpong.png"));  
Image i2 = i1.getImage().getScaledInstance(1000, 700, Image.SCALE_DEFAULT);
```

```
int x = 1;  
for (int i = 2; i <= 600; i+=4, x+=1) {  
    setLocation(600 - ((i + x)/2), 350 - (i/2));  
    setSize(i + 3*x, i + x/2);
```

## PongGame.java

The PongGame class is the entry point of the pong game. It creates an instance of the GameFrame class, which represents the game window and handles the game's logic. The program execution continues within the GameFrame class until the game is closed or completed.

## Paddle.java

The Paddle class represents the paddle in the game. It extends the Rectangle class and contains properties and methods related to the paddle's movement, such as handling key events for moving the paddle up and down. The Paddle class is used to create and control the paddles in the game.

## Ball.java

The Ball class represents a ball object in a ping pong game. It extends the Rectangle class and has properties for the ball's velocities and random generator. It provides methods for setting directions, moving the ball, and drawing it on a Graphics object. The class encapsulates the behavior and attributes of a ball in the game.

## GamePanel.java

This class extends JPanel and implements the Runnable interface. It represents the game panel where the game objects are drawn and animated. It contains methods for creating and moving the ball, creating and controlling the paddles, handling collision detection, and drawing the game objects. The GamePanel class is the core component responsible for the game's logic and rendering.

## GameFrame.java

This class extends JFrame, a class in the Swing library, and is responsible for creating the main window of the game. It creates a GamePanel object and adds it to the JFrame. It sets some of the JFrame's attributes, such as its size, background color, and title, and makes it visible.

## Score.java

The Score class represents the score display in the Pong game. It extends the Rectangle class and has properties for player scores.

The draw method is responsible for drawing the scores on the screen using the Graphics object.

Getter and setter methods are provided to access and modify the player scores.

The Score class handles the visual representation and manipulation of scores in the Pong game

## The OOP Principles & Concepts

All OOP languages have features that facilitate the implementation of the object-oriented model. In Java, these features are:

### 1. Encapsulation

**Encapsulation** is a mechanism that binds data and code together, protecting them from external interference and misuse. It is achieved through the use of access modifiers such as public, private, and protected, which control the visibility of data and methods in a class.

### 2. Inheritance

**Inheritance** is the process by which one class (called the subclass) acquires the properties and behavior of another class (called the superclass). This allows for the creation of hierarchical classifications of objects, where subclasses inherit the characteristics of their superclasses and add additional functionality.

### 3. Polymorphism

**Polymorphism** is a feature that allows objects of different classes to be treated as if they belong to a common class. This means that a single interface or method can be used to perform operations on objects of different types. This is achieved through the use of method overloading and overriding, as well as through the implementation of interfaces and abstract classes.

#### Usage of OOP Concepts in Project

- **Classes and Objects:** The code defines multiple classes, such as GameFrame, Score, and Ball, which are used to create objects and encapsulate related properties and behaviors.
- **Inheritance:** The GameFrame class extends the JFrame class, inheriting its properties and behaviors. Similarly, the Score class extends the Rectangle class, and the Ball class extends the Rectangle class as well.
- **Encapsulation:** The classes encapsulate their properties and methods, allowing for data hiding and providing controlled access through public and private access modifiers. For example, the Score class encapsulates the player scores and provides public methods to get and set the scores.
- **Method Overriding:** The draw method in the Score and Ball classes overrides the draw method from the Rectangle class. The overridden methods provide custom implementation to draw the score and ball on the screen.
- **Constructor:** The classes define constructors, such as the GameFrame constructor, which are used to initialize the objects when they are created. Constructors have the same name as the class and are called when an object is instantiated.

- **Polymorphism:** Polymorphism is a fundamental concept in OOP. It allows objects of different classes to be treated as objects of a common superclass. Polymorphism enables code reusability and flexibility by allowing different objects to be used interchangeably through inheritance and method overriding.

**By utilizing these OOP principles, this project will create a well-structured and maintainable codebase for the ping pong game.**

## **Future Enhancements & Works**

### **▪ Advanced AI Opponent:**

This enhancement focuses on implementing an intelligent and challenging AI opponent in the game. The AI opponent should have the ability to adapt its gameplay strategy, adjust paddle movement, and react to the ball's trajectory. It should provide a competitive experience for single-player mode, making the game engaging and enjoyable even when playing against the computer.

### **▪ Online Multiplayer:**

Adding online multiplayer functionality to the game allows players to compete against each other over the internet. This enhancement involves implementing network communication protocols and creating a server-client architecture to facilitate real-time gameplay between players from different locations. It may also involve incorporating features such as matchmaking, leaderboards, and chat functionalities to enhance the multiplayer experience.

### **▪ Power-ups and Special Effects:**

Introducing power-ups and special effects adds an extra layer of excitement and strategic depth to the game. Power-ups could include items that temporarily enhance a player's paddle speed, increase ball speed, or enable special shots. Special effects can include visual and audio effects

triggered by specific game events, such as scoring a point or activating a power-up. This enhancement aims to make the game more dynamic and entertaining for players.

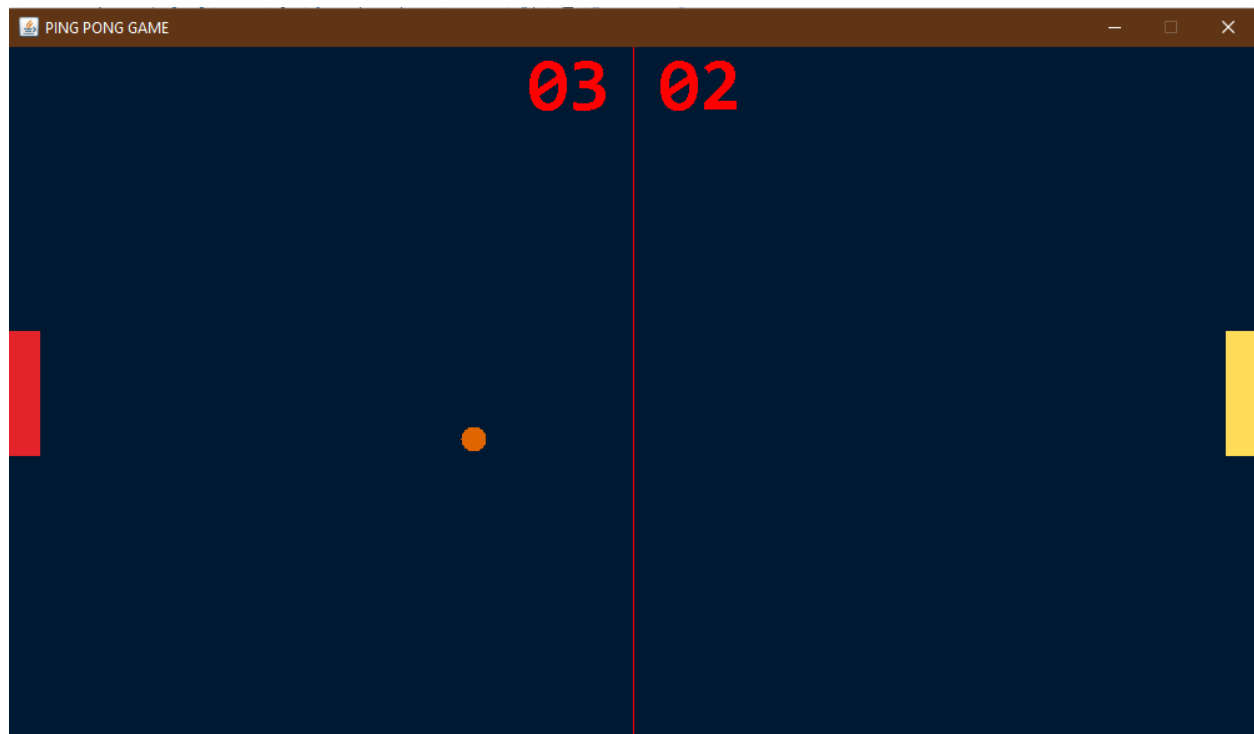
- **Tournament Mode:**

Implementing a tournament mode allows players to participate in a series of matches and compete for the championship. This enhancement involves designing a tournament structure, managing match progressions, and keeping track of player standings. It may include features such as customizable tournament settings, multiple brackets, and AI opponents of varying difficulties. The tournament mode adds a competitive and immersive experience to the game.

- **Customization Options:**

Introducing customization options enables players to personalize their gameplay experience. This enhancement can include features such as customizable paddle designs, ball appearances, backgrounds, or soundtracks. It allows players to express their individuality and preferences, enhancing the overall immersion and enjoyment of the game. Additionally, customization options may extend to gameplay settings, such as paddle sensitivity or ball physics, allowing players to tailor the game to their preferred playstyle.

## Display:



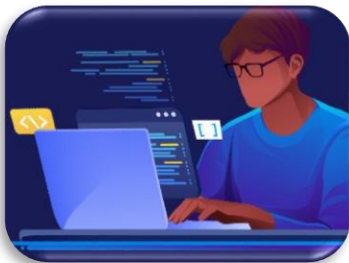


## **Potential Questions:**

Q) How can the AI opponent in the Pong game be enhanced to provide different difficulty levels, allowing players to choose between easy, medium, and hard opponents?

Q) What changes would be necessary in the Score class to implement a leaderboard feature that keeps track of high scores in the Pong game?

Q) How could the GameFrame class be updated to support customizable game settings, such as paddle size, ball speed, or background themes, in the Pong game?



## **Conclusion:**

In a nutshell, *the proposed OOP Java Object-Oriented Ping Pong game* is a promising project that can have a wide range of applications. The game's primary objective is **to introduce programming concepts to students, but it also has the potential to be a recreational tool for game enthusiasts, a skill development tool for players, and a research tool for researchers studying the effects of games on human psychology.**

**The game's features**, such as its interactive gameplay, sound effects, and dynamic user interface, make it an engaging and exciting way to learn programming concepts. Moreover, the project's use of object-oriented programming principles demonstrates **the development team's expertise in software engineering and game development.**

The project proposal's strengths include its clear objectives, comprehensive project plan, and potential for future applications. Additionally, the proposed tools and platforms, such as Java, Git, and Eclipse, are industry-standard and widely used in game development.

*In conclusion*, **the proposed OOP Java Object-Oriented Ping Pong game is a well-designed project with a clear purpose and scope.** The project's potential for future applications, combined with the development team's expertise, make it a valuable contribution to the field of game development and software engineering.

## **Reference Citations:**

<https://www.w3schools.com/>

<https://www.codecademy.com/learn/learn-java>

<https://stackify.com/java-tutorials//>

<https://www.learnjavaonline.org/>

<https://www.youtube.com/>

# **Introduction:**

## ***Abstract:***

The **College Management System** is a comprehensive and efficient software solution designed to streamline and enhance the administrative and academic operations of a college or educational institution. **Built using object-oriented programming principles in Java**, this project aims to provide a user-friendly interface and automate various tasks involved in managing a college. Overall, the College Management System is a powerful tool that brings efficiency, accuracy, and transparency to the administrative and academic processes of a college. By automating routine tasks and providing reliable data management, it empowers stakeholders to focus on delivering high-quality education and fostering a conducive learning environment.

## ***Overview:***

The College Management System has been developed using Java as the core programming language, leveraging the power of Java Swing and AWT toolkits for creating a user-friendly graphical interface. The MySQL database serves as the robust data management system, seamlessly integrated with the project using JDBC for efficient data handling. Following the principles of object-oriented programming, the project's code is structured for modularity and reusability. An IDE like Eclipse, IntelliJ IDEA, or NetBeans has been utilized for coding, debugging, and project management, while version control, such as Git, ensures effective collaboration and source code management.



## **Objective (Aim):**

- **Automation of Administrative Tasks:** The College Management System automates repetitive administrative tasks, such as student enrollment, course registration, and fee management. This reduces manual effort, improves efficiency, and minimizes errors in data entry and processing.
- **Streamlined Data Management:** The system provides a centralized platform for managing student, faculty, and administrative data. It ensures efficient data storage, retrieval, and updates, enabling quick access to accurate information whenever needed.
- **Enhanced Communication and Collaboration:** The system facilitates seamless communication and collaboration among students, faculty, staff, and parents/guardians. It offers features like messaging systems, notifications, and portals to enable timely and effective communication, fostering better coordination and engagement.
- **Improved Decision-Making:** With access to comprehensive data and analytics, the system empowers administrators and educators to make informed decisions. It provides insights into student performance, attendance patterns, and other relevant metrics, enabling proactive interventions and strategic planning.
- **Accuracy and Data Integrity:** The system ensures data accuracy and integrity by implementing data validation checks, security measures, and user access controls. This safeguards sensitive information and maintains the reliability of data throughout the system.
- **User-Friendly Interface:** The system offers a user-friendly interface, making it intuitive and easy to navigate for all stakeholders. It simplifies tasks, reduces learning curves, and enhances user satisfaction.
- **Scalability and Future Growth:** The system is designed to accommodate the growing needs of the educational institution. It can handle increasing volumes of data,

user accounts, and system functionalities, ensuring scalability and adaptability as the institution expands.

- **Integration Possibilities:** The system allows integration with other systems and platforms, such as financial management software, learning management systems, or external APIs. This facilitates seamless data exchange and interoperability, enabling a connected ecosystem of applications.
- **Security and Privacy:** The system prioritizes data security and privacy by implementing robust encryption, access controls, and compliance with privacy regulations. It safeguards sensitive information and ensures confidentiality for all users.
- **Cost and Time Efficiency:** The system reduces administrative costs and saves time by automating tasks, streamlining processes, and minimizing manual paperwork. It optimizes resource utilization, freeing up staff to focus on higher-value activities within the institution.

## **Project Scope:**



The project will include the following features:

- **Student Management:** This module focuses on managing student-related information, including enrollment, admission, attendance, and academic records. It allows for efficient student data management and provides functionalities such as student registration, profile management, and tracking of academic progress.
- **Faculty Management:** The faculty management module handles faculty-related tasks, such as maintaining faculty profiles, assigning courses, managing schedules, and tracking attendance. It streamlines faculty information management and supports efficient communication between faculty members and administrators.
- **Leave Management:** This feature enables students and faculty to apply for leaves and tracks their leave history. It simplifies the leave application process, automates approval workflows, and provides visibility into leave statuses, ensuring smooth leave management within the institution.
- **Designing the User-friendly Interface:** The project focuses on developing an intuitive and user-friendly interface for easy navigation and interaction. It emphasizes visual design, usability, and responsiveness to enhance the user experience and ensure efficient usage of the system.
- **User Roles and Authentication:** This aspect involves implementing a role-based access control system that defines different user roles, such as administrators, faculty, and students. It ensures appropriate access and permissions for each role, safeguarding sensitive information and maintaining data security.
- **Fee Management:** The fee management module handles student fee-related processes, including fee collection, tracking, and generating fee reports. It provides functionalities to manage fee payments, generate receipts, and monitor outstanding balances, streamlining the fee management process.

- **Result Management:** This module focuses on managing and processing student examination results. It allows for result entry, calculation, and generation of result reports or cards. It provides an efficient and accurate system for recording and sharing student performance information.
- **Examination Management:** The examination management module encompasses activities related to scheduling and conducting examinations. It includes functionalities such as exam timetable creation, room allocation, and exam invigilation management, ensuring smooth and organized examination processes within the institution.

## **Not in Scope:**

- **Library Management:** This module focuses on efficient management of library resources, including book cataloging, circulation, and inventory management. It allows users to search and reserve books, tracks borrowing and return activities, and provides tools for library staff to manage acquisitions, subscriptions, and inter-library loan services.
- **Hostel Management:** The Hostel Management module streamlines the management of hostel facilities within the educational institution. It includes features for room allocation, maintenance requests, and tracking hostel fees and payments. It enables administrators to manage room availability, assign roommates, and handle check-in and check-out processes effectively.
- **Alumni Management:** The Alumni Management module is designed to maintain a strong connection with the institution's alumni. It helps in tracking and managing alumni records, including their contact information, employment details, and contributions to the institution. It facilitates alumni engagement activities, such as organizing reunions, alumni events, and networking opportunities.

- **Online Learning and Course Management:** This module provides a platform for delivering online courses, managing course materials, assignments, and assessments. It supports features like virtual classrooms, discussion forums, and multimedia content. It enables students to access learning materials anytime, anywhere, and provides tools for instructors to track student progress and deliver interactive online learning experiences.
- **Financial Management:** The Financial Management module focuses on the institution's financial operations, including budgeting, accounting, and financial reporting. It helps in tracking income and expenses, managing accounts payable and receivable, and generating financial statements. It enables administrators to make informed financial decisions, monitor financial health, and ensure compliance with financial regulations.

### **Note:**

It's important to note that while these features are interesting and may enhance the college management system, they require additional development time, resources, and expertise. Therefore, if you're working on a project with a specific scope and timeline, it's important to prioritize and focus on the core features that align with the project's objectives



# **Project Plan:**

The project will be divided into the following phases:

## **Design Phase:**

In the design phase, the college management system's features and requirements will be defined, and the college management system's overall design will be planned. The following tasks will be completed in this phase:

- Identify the college management system's key features and requirements.
- Develop a college management system design that incorporates object-oriented programming concepts such as encapsulation, inheritance, and polymorphism.
- Create user interface (UI) wireframes that outline the college management system's interface, including menus, screens, and college management system elements.

## **Implementation Phase:**

In the implementation phase, the college management system will be developed using Java programming language. The following tasks will be completed in this phase:

- Develop the college management system's UI using Java Swing or JavaFX.
- Implement the college management system's logic and rules
- Implement college management system settings

## **Testing Phase:**

In the testing phase, the college management system will be tested for any bugs or issues. The following tasks will be completed in this phase:

- Conduct thorough testing of the college management system to identify and fix any bugs or issues.
- Test the college management system on different operating systems and hardware configurations.
- Ensure that the college management system meets the requirements defined in the design phase.
- Optimize the college management system's performance to ensure smooth college management system play even on slower machines.

## **Project Category:**

This project falls under the category of Object Oriented Programming (OOP). OOP is a programming paradigm that emphasizes the use of objects and classes to organize and structure code. Java is an example of an OOP language, where all programs are written using OOP concepts.

## **Project Timeline:**

The project timeline is as follows:

- ❖ Design phase: 3 week
- ❖ Implementation phase: 3 weeks
- ❖ Testing phase: 1 week
- ❖ Total project duration: 7 weeks

## **Project Budget:**

This project will be developed using open-source tools and software, therefore there will be no cost involved.

## **Project Team:**

The project will be developed by a team of one developer only.

# Functionality & Implementation:

## Input Specifications:

- **Student Details:**

Input student information such as name, address, contact details, date of birth, and enrollment number.

- **Faculty Details:**

Input faculty member information, including name, contact details, designation, and employee ID.

- **Leave Application:**

Accept leave requests from students and faculty members, requiring inputs such as leave type, start date, end date, and reason.

- **Examination Marks:**

Input examination marks for students, including subject-wise scores and overall grades.

- **Fee Details:**

Input fee-related information, including tuition fees, additional charges, discounts, and payment deadlines.

- **Course Registration:**

Accept student course preferences for each semester, including course codes, timings, and faculty assignments.

By defining the input requirements for the college management system, we can ensure that college management system will be more interactive and engaging for the users.

## **Output Specifications:**

- **Student and Faculty Details:**

Display the stored information for students and faculty members, including personal details, contact information, and academic records.

- **Leave Status:**

Provide the status of leave requests, indicating whether they have been approved or rejected.

- **Result Card:**

Generate and print result cards for students, displaying subject-wise marks, grades, and overall performance.

- **Fee Form:**

Generate and print fee forms for students, presenting the amount due, payment deadlines, and payment methods.

- **Reports and Analytics:**

Generate reports and analytics based on various parameters such as student performance, attendance, fee payment status, and other relevant metrics.

- **Fee Structure:**

Display the fee structure, including tuition fees, additional charges, and any applicable discounts

By implementing & defining the output requirements for the college management system, we can ensure that college management system will be more interactive and engaging for the users.

# **Requirements of College Management System:**

## **Hardware Requirements:**

- Processor Brand : Intel
- Processor Type : Core i3
- Processor Speed : 2 GHz
- Processor Count : 1
- RAM Size : 2 GB
- Memory Technology : DDR3
- Computer Memory Type : DDR3 SDRAM
- Hard Drive Size : 160 GB

## **Software Requirements:**

- Operating system : Windows 10
- Application server : JAVA (NetBeans)
- Front end : JAVA
- Connectivity: JDBC Driver
- Database connectivity : WAMP (MYSQL Console)

## Overview of Front-End

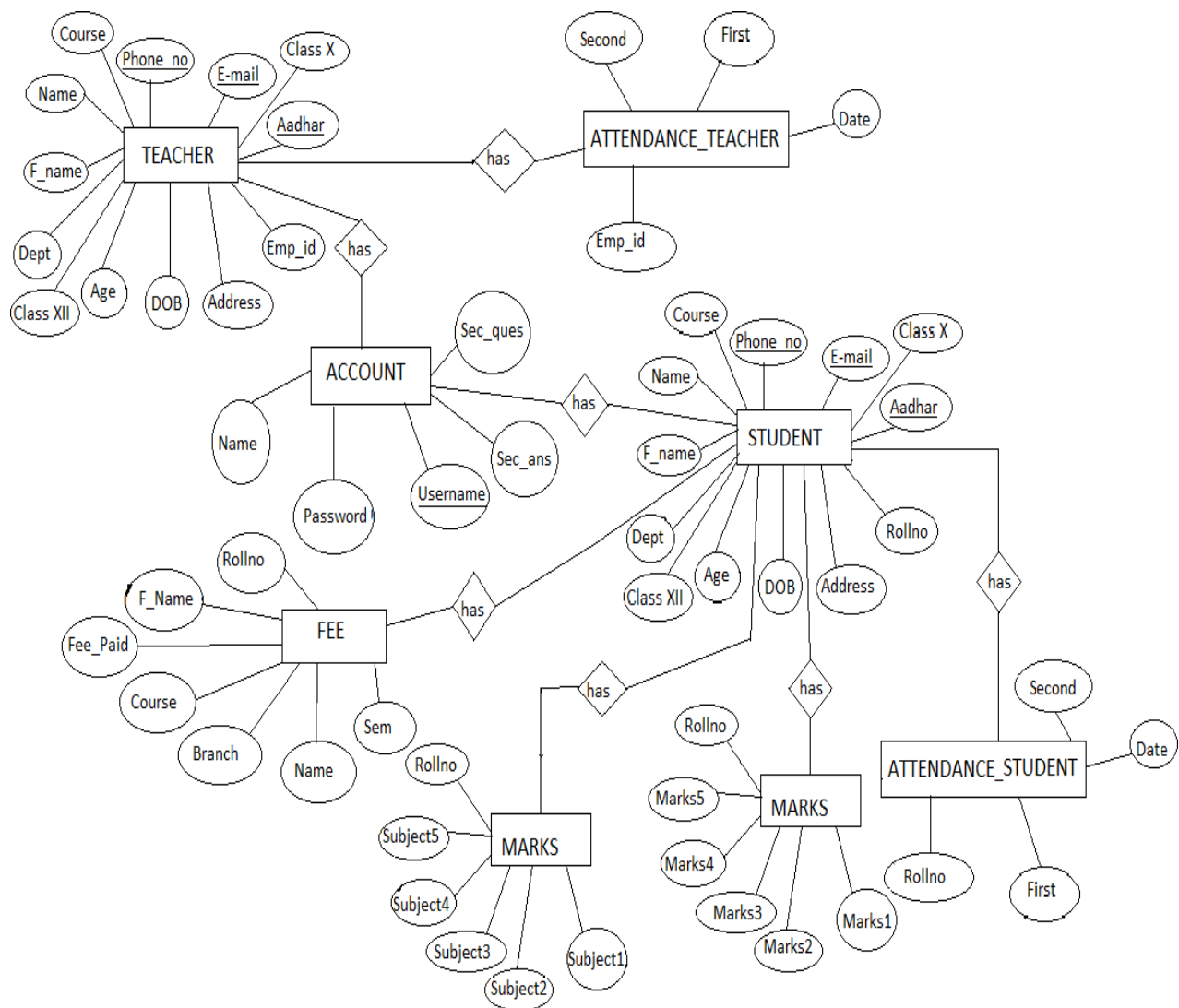
By selecting Java as the front-end technology, you can leverage its features such as platform independence, scalability, flexibility, and event-driven programming. **Java Swing and AWT** toolkits can be utilized for creating the **graphical user interface (GUI)** components, allowing for the development of user-friendly interfaces that meet the needs of employees from non-IT backgrounds. The Java language's rich ecosystem and libraries provide excellent support for reporting features and printing capabilities. Furthermore, Java's popularity and widespread adoption make it a suitable choice for long-term development and maintenance. It offers strong debugging and testing tools, facilitating the identification and resolution of issues during the development process.

## Overview of Back-End

After careful consideration of various factors, we chose **MySQL** as the back-end technology for our project. **MySQL** is a popular and efficient relational database management system (RDBMS) that offers scalability, security features, and robust data handling capabilities. It provides inherent support for multiple users, efficient data retrieval, and maintenance, and compatibility with different operating systems. With its widespread popularity, strong community support, and availability of drivers, MySQL seamlessly integrates with the chosen front-end technology. Its open-source nature and compatibility with various programming languages make it a reliable choice for our project's back-end needs, ensuring smooth data management and optimal performance.

# Execution

## FlowChart of College Management System:



## **Java Classes:**

- MainWindow.java
- Login.java
- Project.java
- AddStudent.java
- AddTeacher.java
- dbConnectivity.java
- feeStructure.java
- UpdateStudent.java
- UpdateTeacher.java
- Leave.java
- EnterMarks.java
- LeaveDetails.java
- ExaminationDetails.java
- About.java

## **Database Queries**

- Create database
- Use database you just created
- Create login table
- Insert some values in the login table
- Create student table
- Create teacher table
- Create student leave table



- Create teacher leave table
- Create table to store subjects
- Create table to store marks
- Create table for fee structure
- Insert some values in the table
- Create table to store student fee details

## **The OOP Principles & Concepts**

All OOP languages have features that facilitate the implementation of the object-oriented model. In Java, these features are:

### **4. Encapsulation**

**Encapsulation** is a mechanism that binds data and code together, protecting them from external interference and misuse. It is achieved through the use of access modifiers such as public, private, and protected, which control the visibility of data and methods in a class.

### **5. Inheritance**

**Inheritance** is the process by which one class (called the subclass) acquires the properties and behavior of another class (called the superclass). This allows for the creation of hierarchical classifications of objects, where subclasses inherit the characteristics of their superclasses and add additional functionality.

### **6. Polymorphism**

**Polymorphism** is a feature that allows objects of different classes to be treated as if they belong to a common class. This means that a single interface or method can be used to perform operations on objects of different types. This is achieved through the use of method overloading and overriding, as well as through the implementation of interfaces and abstract classes.

## Usage of OOP Concepts in Project

- **Classes and Objects:** Different classes can be created to represent various entities in the college management system, such as Student, Faculty, Course, Department, etc. Objects of these classes can then be instantiated to represent individual students, faculty members, courses, and departments.
- **Inheritance:** Inheritance can be used to establish relationships between classes in the college management system. For instance, a base class like Person can be created, which can be extended by subclasses like Student and Faculty to inherit common attributes and behaviors.
- **Encapsulation:** Encapsulation can be applied to the classes in the college management system to encapsulate data and functionality. For example, the Student class can encapsulate student-specific details like name, ID, and enrolled courses, while providing methods to access and modify this information securely.
- **Method Overriding:** Method overriding can be employed in the college management system when there is a need to provide custom implementations of inherited methods. For instance, the display Information() method in the Student class can override the same method in the Person class to display student-specific information.
- **Constructor:** Constructors can be utilized in various classes of the college management system to initialize object properties when they are created. For example, a constructor in the Course class can be used to set initial values such as course code, title, and credits.
- **Polymorphism:** Polymorphism can be utilized in the college management system to treat objects of different classes as objects of a common superclass or interface. This allows for flexibility and code reusability. For example, a method that processes students' attendance can accept both Student and Faculty objects, treating them as instances of a common Person superclass.

**By utilizing these OOP principles, this project will create a well-structured and maintainable codebase for the college management system.**

## Future Enhancements & Works

- **Library Management:** This module focuses on efficient management of library resources, including book cataloging, circulation, and inventory management. It allows users to search and reserve books, tracks borrowing and return activities, and provides tools for library staff to manage acquisitions, subscriptions, and inter-library loan services.
- **Hostel Management:** The Hostel Management module streamlines the management of hostel facilities within the educational institution. It includes features for room allocation, maintenance requests, and tracking hostel fees and payments. It enables administrators to manage room availability, assign roommates, and handle check-in and check-out processes effectively.
- **Alumni Management:** The Alumni Management module is designed to maintain a strong connection with the institution's alumni. It helps in tracking and managing alumni records, including their contact information, employment details, and contributions to the institution. It facilitates alumni engagement activities, such as organizing reunions, alumni events, and networking opportunities.
- **Online Learning and Course Management:** This module provides a platform for delivering online courses, managing course materials, assignments, and assessments. It supports features like virtual classrooms, discussion forums, and multimedia content. It enables students to access learning materials anytime, anywhere, and provides tools for instructors to track student progress and deliver interactive online learning experiences.
- **Financial Management:** The Financial Management module focuses on the institution's financial operations, including budgeting, accounting, and financial reporting. It helps in tracking income and expenses, managing accounts payable and receivable, and generating financial statements. It enables administrators to make informed financial decisions, monitor financial health, and ensure compliance with financial regulations.

## **Display:**

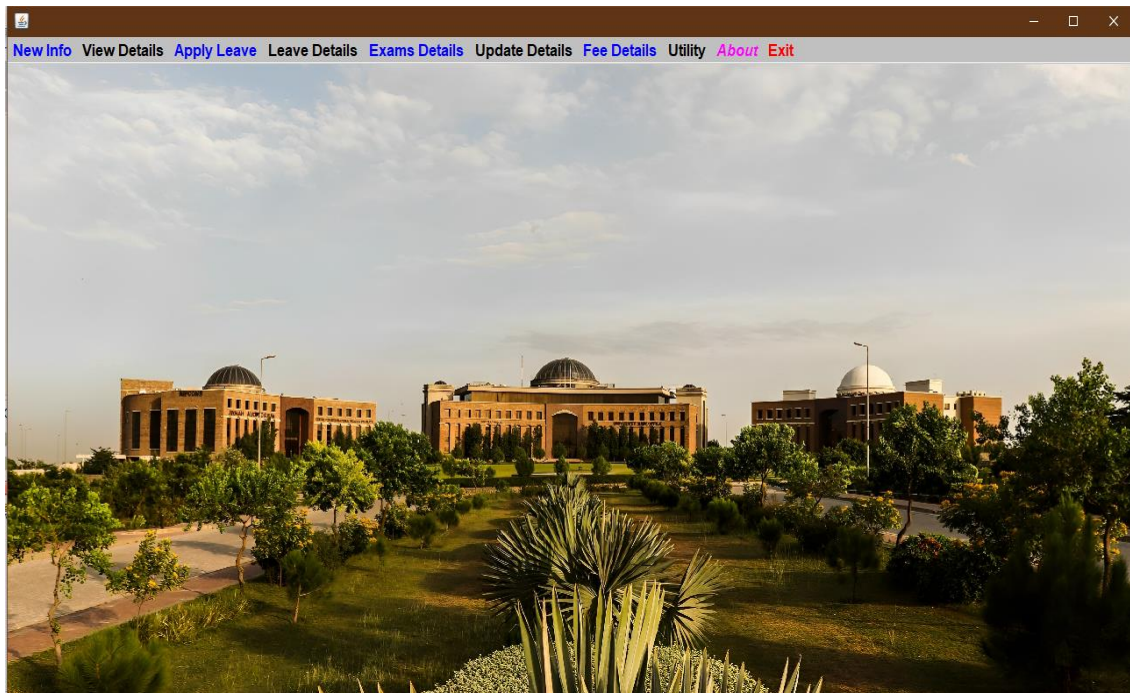
- **OUTPUT (SCREENSHOT) FOR MAIN WINDOW**



- **OUTPUT (SCREENSHOT) FOR LOGIN WINDOW**



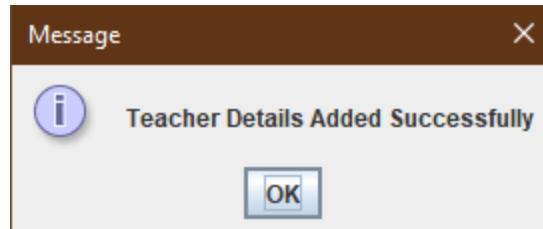
## ■ OUTPUT (SCREENSHOT) FOR LOGIN WINDOW



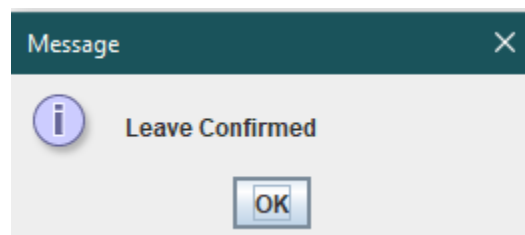
## ■ OUTPUT (SCREENSHOT) FOR ADD NEW TEACHER WINDOW

**New Teacher (Faculty) Details**

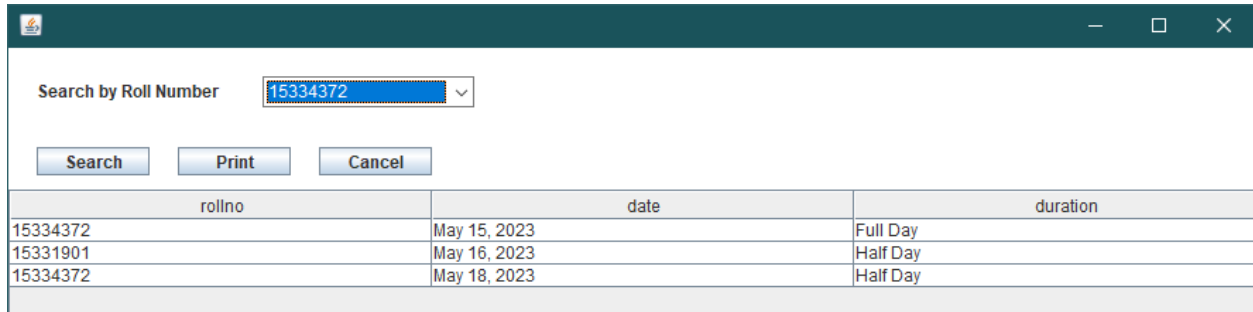
Name	<input type="text" value="Dr.Kamran Khan"/>	Father's Name	<input type="text" value="Nauman Khan"/>
Employee Id	<input type="text" value="1014640"/>	Date of Birth	<input type="text" value="Sep 16, 1970"/>
Address	<input type="text" value="5,Sector E/11-3,Islamabad"/>	Phone	<input type="text" value="03413421401"/>
Email Id	<input type="text" value="kkhan@gmail.com"/>	MATRIC '%'	<input type="text" value="85"/>
FSc '%'	<input type="text" value="76"/>	CNIC Number	<input type="text" value="3460192345621"/>
Qualification	<input type="text" value="PhD"/>	Department	<input type="text" value="Computer Science"/>



- **OUTPUT (SCREENSHOT) FOR STUDENT LEAVE WINDOW**

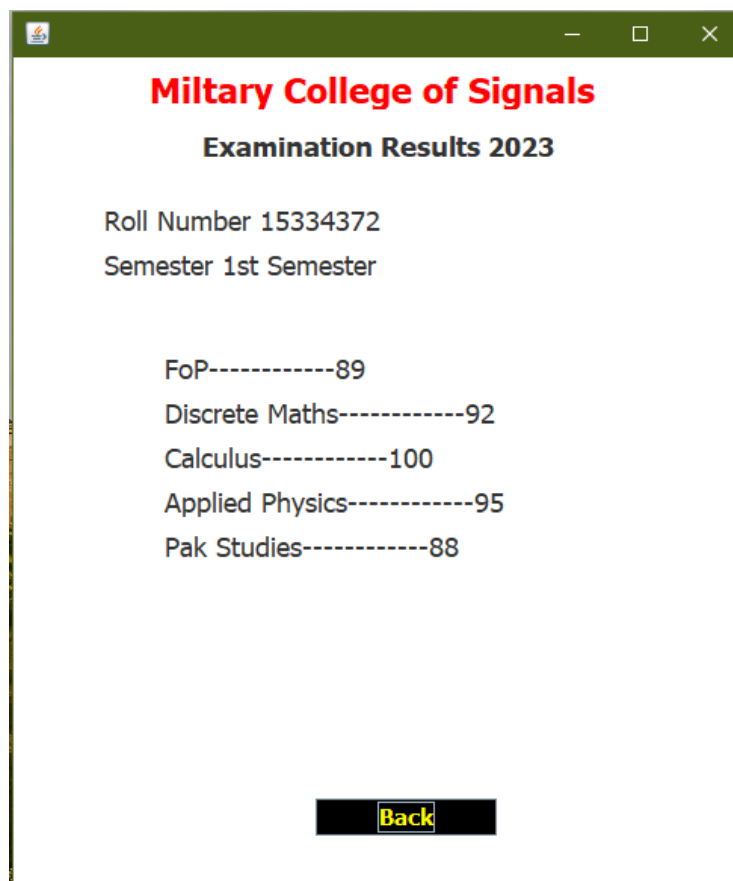
A screenshot of a web application window titled 'Apply Leave (Student)'. The window has a dark teal header bar with standard window controls (minimize, maximize, close). The main content area is white. It contains the following elements: a heading 'Apply Leave (Student)', a label 'Search by Roll Number' above a dropdown menu showing '15334372', a label 'Date' above a date input field showing 'May 18, 2023' with a calendar icon, a label 'Time Duration' above a dropdown menu showing 'Half Day', and two buttons at the bottom: 'Submit' (green text on a black button) and 'Cancel' (red text on a black button).

▪ **OUTPUT (SCREENSHOT) FOR STUDENT  
LEAVE DETAILS WINDOW**



rollno	date	duration
15334372	May 15, 2023	Full Day
15331901	May 16, 2023	Half Day
15334372	May 18, 2023	Half Day

▪ **OUTPUT (SCREENSHOT) FOR EXAMINATION  
DETAILS WINDOW**



**Military College of Signals**

**Examination Results 2023**

Roll Number 15334372

Semester 1st Semester

FoP-----89

Discrete Maths-----92

Calculus-----100

Applied Physics-----95

Pak Studies-----88

**Back**

▪ **OUTPUT (SCREENSHOT) FOR ENTER MARKS WINDOW**

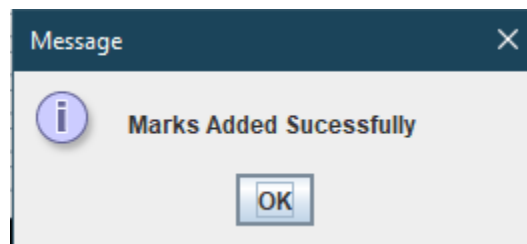
**Enter Marks of Student**

Select Roll Number: 15337361

Select Semester: 2nd Semester

Enter Subject	Enter Marks
OOP	90
CA&DLD	94
Linear Algebra	89
Islamiat	85
OHS	99

Submit Back



▪ **OUTPUT (SCREENSHOT) FOR UPDATE TEACHER WINDOW**

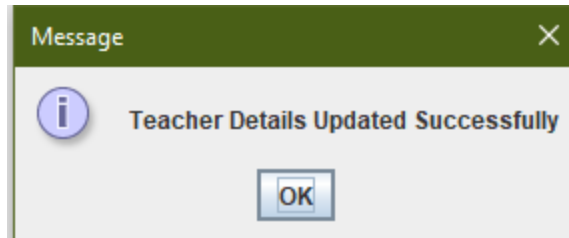
**Update Teacher Details**

Select Employee Id: 1014640

Name	Dr.Kamran Khan	Father's Name	Nauman Khan
Employee Id	1014640	Date of Birth	Sep 16, 1970
Address	5, Sector E/11-3, Islamabad	Phone	03413421401
Email Id	kkhan@gmail.com	MATRIC '%'	85
FSc '%'	76	CNIC Number	3460192345621
Education	PhD	Department	Telecom

Update Cancel



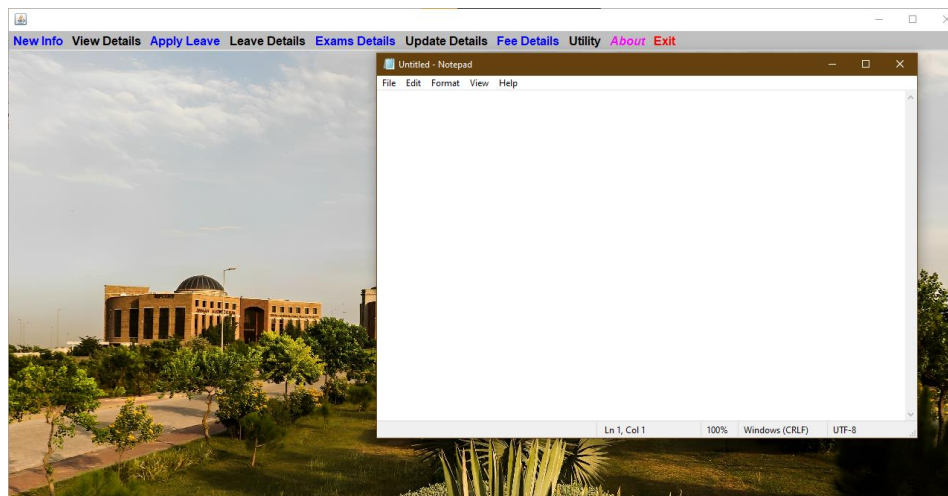


- **OUTPUT (SCREENSHOT) FOR STUDENT FEE FORM WINDOW**

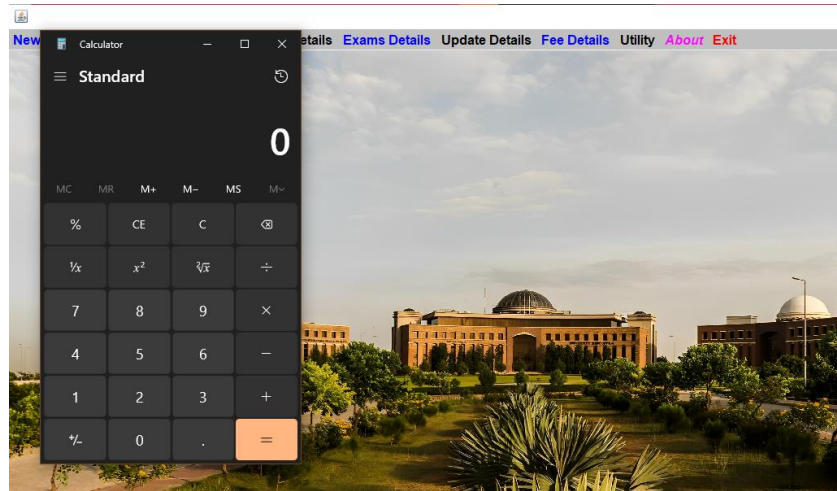
Select Roll No	15334372
Name	Ali Akbar
Father's Name	Akbar Ahmad
Course	Bachelors
Branch	Software Engineering
Semester	Semester1
Total Payable	175000

[Update](#) [Pay Fee](#) [Back](#)

- **OUTPUT (SCREENSHOT) FOR NOTEPAD WINDOW**



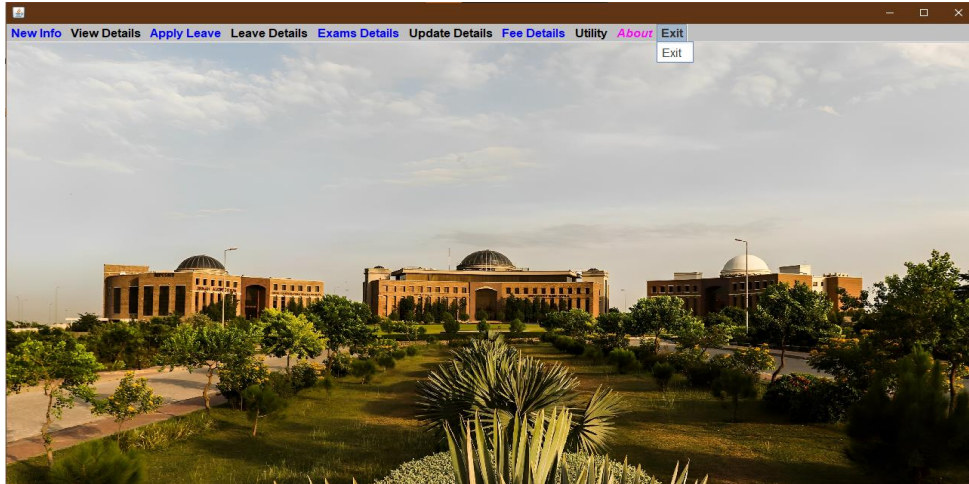
- OUTPUT (SCREENSHOT) FOR CALCULATOR WINDOW



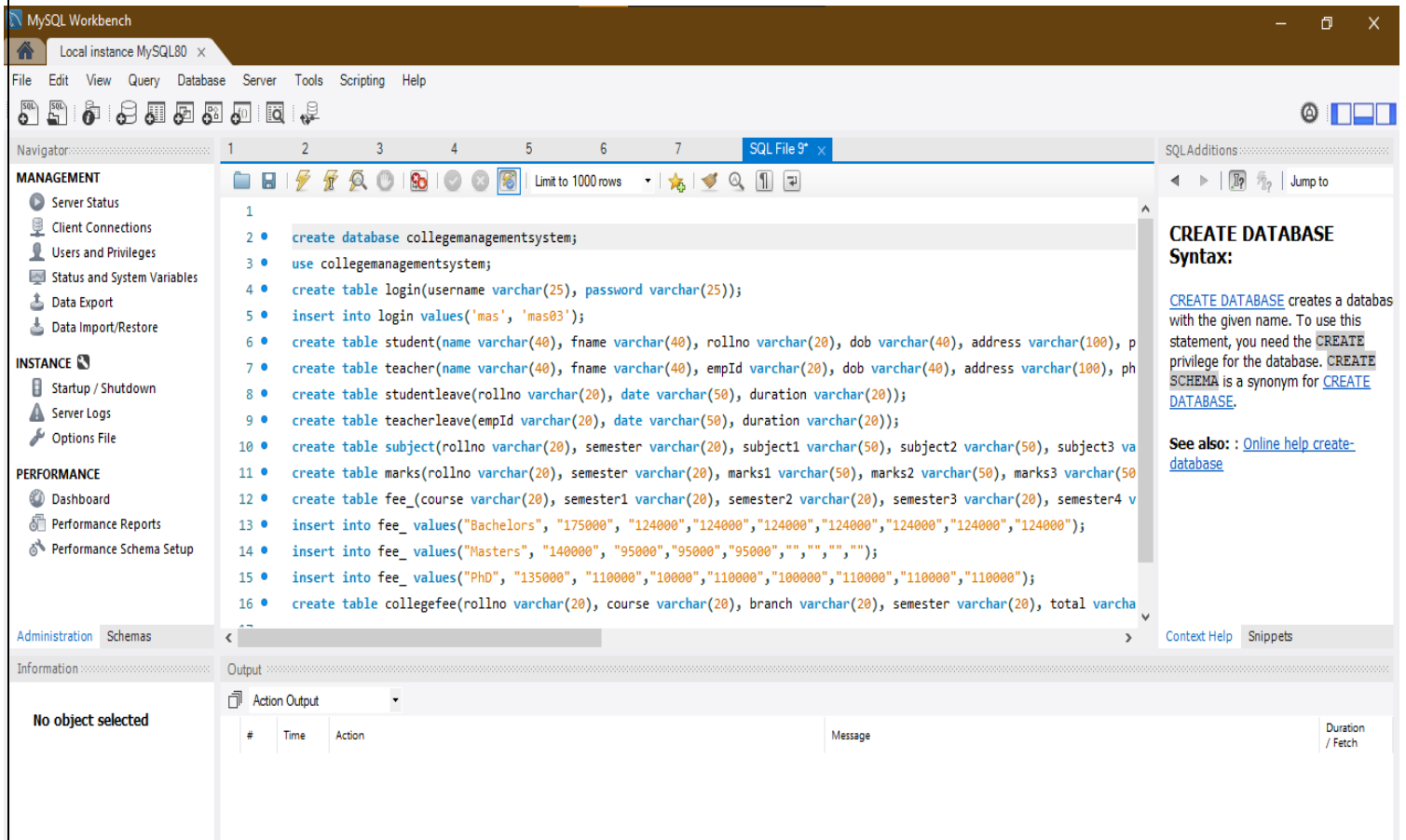
- OUTPUT (SCREENSHOT) FOR ABOUT DEV WINDOW



## ■ OUTPUT (SCREENSHOT) FOR EXIT WINDOW



## OUTPUT (SCREENSHOT) FOR DATABASE MYSQL CONNECTIVITY WINDOW

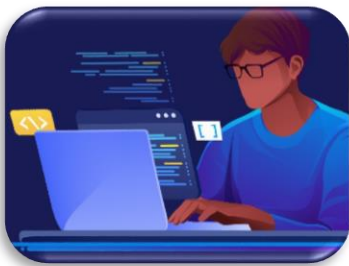


## **Potential Questions:**

Q) How does the College Management System ensure data security and privacy for sensitive student and faculty information?

Q) How does the College Management System leverage AI algorithms or machine learning techniques to analyze student performance data and provide personalized

Q) How does the College Management System facilitate effective communication and collaboration between parents/guardians and the educational institution to ensure transparent and timely updates on student progress and involvement?



## **Conclusion:**

In a nutshell, the *College Management System project* is a comprehensive and efficient solution that addresses the diverse needs of educational institutions. With its range of modules and functionalities, the system streamlines administrative tasks, enhances communication and collaboration, and provides valuable insights for informed decision-making. The **integration of Java, MySQL**, and other technologies showcases the team's proficiency in leveraging industry-standard tools for developing robust software solutions.

The project's strengths lie in its ability to effectively manage student and faculty information, automate processes, and provide a user-friendly interface. The system's scalability and extensibility ensure that it can accommodate the evolving needs of educational institutions of varying sizes.

While the project's scope is focused on core functionalities, future enhancements such as integrating AI for data analysis or adding modules for library management and alumni engagement can further enhance its capabilities.

Overall, the College Management System project demonstrates a deep understanding of the challenges faced by educational institutions and provides a reliable and efficient solution. With its implementation, educational institutions can streamline their operations, foster better communication, and ultimately create an environment conducive to academic success.

## **Reference Citations:**

<https://www.w3schools.com/>

<https://www.codecademy.com/learn/learn-java>

<https://stackify.com/java-tutorials/>

<https://www.learnjavaonline.org/>

<https://www.youtube.com/>



# Introduction:

## *Abstract:*

The **Object-Oriented Application: Talk Buddy** presents a simple chat application implemented using **Java Swing and socket programming**. The application consists of a **server-side component and a client-side component** that enable users to exchange messages in real-time. The server component creates a **graphical user interface (GUI)** to display the chat window and establishes a server socket to listen for incoming client connections. The client component also creates a GUI and establishes a socket connection with the server. The application allows users to send and receive messages, which are displayed in the respective GUIs using a formatted layout. Overall, this project demonstrates the implementation of a basic chat application **using socket programming**, providing a foundation for further development and enhancements in real-time communication systems

## *History:*

Late 1960s and early 1970s: Development of computer networks like ARPANET leads to the creation of the first online chat systems.

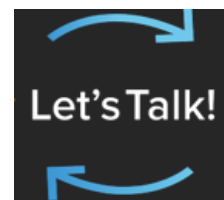
1971: Douglas Engelbart develops "On-Line System" (NLS) with message group chat feature.

1990s: Internet Relay Chat (IRC) gains popularity as a real-time communication protocol.

Late 1990s: Instant messaging applications such as ICQ, AIM, and MSN Messenger revolutionize online communication.

Early 2000s: Introduction of voice and video chat capabilities with platforms like Skype and Google Talk.

Dominance of platforms like WhatsApp, Facebook Messenger, and WeChat in the mobile chat app market.



## **Objective (Aim):**

- **Enable Real-time Communication:** Implement mechanisms that allow users to exchange messages in real-time, ensuring minimal latency and delay. This can involve using technologies such as web sockets or long polling to establish a persistent connection between the client and server, enabling instantaneous message transmission.
- **Develop a Chatting Application:** Create a dedicated application that enables users to engage in conversations with each other. This involves designing the user interface, implementing the necessary backend functionality, and providing features like user authentication, contact management, and message handling.
- **Provide User-to-User Messaging:** Enable direct communication between users by establishing a messaging channel where they can exchange messages privately. This can involve creating a messaging system that identifies users, handles message routing, and ensures secure and reliable delivery.
- **Ensure Message Delivery and Receipt:** Implement mechanisms to track and confirm the successful delivery of messages. This may include features such as message acknowledgment, delivery status notifications, and read receipts to provide users with information about the status of their messages.
- **Support Text-based Communication:** Enable users to exchange text-based messages as the primary means of communication. This can involve implementing text input fields, message composition features (e.g., emojis, attachments), and ensuring proper rendering and display of text messages in the user interface.
- **Create a User-Friendly Interface:** Design an intuitive and visually appealing interface that enhances the user experience. This includes considerations such as organizing conversations, displaying message threads, providing a clear indication of user presence or activity, and incorporating features like search, filters, and notifications for easy navigation and interaction.

- **Implement Client-Server Architecture:** Develop a server-side component that handles message routing, user authentication, and storage of messages and user data. The client-side application interacts with the server to send and receive messages, update user information, and synchronize chat history across multiple devices.
- **Utilize Socket Programming for Communication:** Use socket programming techniques to establish and manage the network connections between the client and server. This enables bidirectional communication, allowing messages to be sent and received in real-time.
- **Ensure Message Formatting and Display:** Implement proper formatting and rendering of messages to enhance readability and user experience. This includes handling line breaks, special characters, emojis, and formatting styles (e.g., bold, italic) in messages and displaying them appropriately in the user interface.

## Project Scope:

The project will include the following features:



- **Client-Server Communication:** This involves establishing a communication channel between the chat client and the server. It includes defining protocols and APIs for sending and receiving messages, handling client-server interactions, and ensuring secure and efficient data transmission.
- **Message Sending and Receiving:** This focuses on implementing the functionality to send and receive messages within the chat client. It involves handling user input, validating and formatting messages, and implementing the necessary logic for sending messages to other users or groups, as well as receiving and displaying incoming messages.



- **Message History and Logging:** This feature involves storing and managing message history and logs within the chat client. It includes designing a database or data storage mechanism to store messages, implementing mechanisms for retrieving and displaying message history, and ensuring efficient storage and retrieval of messages.
- **Refactoring & Optimization:** Refactoring involves restructuring and improving the existing codebase to enhance readability, maintainability, and performance. It may include identifying and resolving bugs, optimizing algorithms, and improving the overall efficiency and scalability of the chat client.
- **Graphical User Interface (GUI) Development:** GUI development focuses on creating an intuitive and user-friendly interface for the chat client. It involves designing and implementing the visual components, such as chat windows, message input fields, user lists, and notification mechanisms, to provide a seamless and engaging user experience.
- **Online Status and Presence Management:** This feature deals with managing and displaying the online status and presence of users in the chat client. It includes implementing mechanisms for users to indicate their online/offline status, tracking user activity and availability, and displaying real-time updates regarding the presence of other users.
- **Testing and Quality Assurance:** This involves implementing testing strategies to ensure the reliability and stability of the chat client. It includes writing test cases, conducting unit testing, integration testing, and system testing to identify and address any functional or performance issues. Quality assurance also involves implementing error handling and logging mechanisms to capture and resolve any runtime errors or exceptions.
- **User Profile Management:** This feature focuses on managing user profiles within the chat client. It includes implementing functionality for users to create and edit their profiles, manage profile pictures or avatars, update personal information, and configure privacy settings.

## **Not in Scope:**

- **Audio and Video Calls:** This feature allows users to have real-time audio and video communication within the chat client. Users can initiate voice or video calls with their contacts, enabling more interactive and immersive conversations.
- **Artificial Intelligence-based Features:** By leveraging AI, the chat client can offer advanced features such as smart replies, chatbot integration, intelligent chat moderation, and automated content filtering. AI algorithms can analyze conversations, understand user preferences, and provide personalized recommendations or automated actions.
- **Voice Messaging:** Voice messaging enables users to send and receive recorded audio messages within the chat client. It provides an alternative communication method, allowing users to express themselves using their own voice and adding a more personal touch to the conversation.
- **Multiple Language Support:** This feature enables users to communicate in different languages within the chat client. It includes language translation capabilities that can automatically translate messages between different languages, facilitating communication between users who speak different languages.
- **File Transfer:** File transfer functionality allows users to share files (e.g., documents, images, videos) with their contacts directly through the chat client. It provides a convenient way to exchange files without the need for external file-sharing platforms or email attachments, enhancing collaboration and productivity.

### **Note:**

It's important to note that while these features are interesting and may enhance the chatting application, they require additional development time, resources, and expertise. Therefore, if you're working on a project with a specific scope and timeline, it's important to prioritize and focus on the core features that align with the project's objectives

# **Project Plan:**

The project will be divided into the following phases:

## **Design Phase:**

In the design phase, the chatting application's features and requirements will be defined, and the chatting application's overall design will be planned. The following tasks will be completed in this phase:

- Identify the chatting application's key features and requirements.
- Develop a chatting application design that incorporates object-oriented programming concepts such as encapsulation, inheritance, and polymorphism.
- Create user interface (UI) wireframes that outline the chatting application's interface, including menus, screens, and chatting application elements.

## **Implementation Phase:**

In the implementation phase, the chatting application will be developed using Java programming language. The following tasks will be completed in this phase:

- Develop the chatting application's UI using Java Swing or JavaFX.
- Implement the chatting application's logic and rules
- Implement chatting application settings

## **Testing Phase:**

In the testing phase, the chatting application will be tested for any bugs or issues. The following tasks will be completed in this phase:

- Conduct thorough testing of the chatting application to identify and fix any bugs or issues.
- Test the chatting application on different operating systems and hardware configurations.
- Ensure that the chatting application meets the requirements defined in the design phase.
- Optimize the chatting application's performance to ensure smooth chatting applicationplay even on slower machines.

## **Project Category:**

This project falls under the category of Object Oriented Programming (OOP). OOP is a programming paradigm that emphasizes the use of objects and classes to organize and structure code. Java is an example of an OOP language, where all programs are written using OOP concepts.

## **Project Timeline:**

The project timeline is as follows:

- ❖ Design phase: 1 week
- ❖ Implementation phase: 3 weeks
- ❖ Testing phase: 1 week
- ❖ Total project duration: 5 weeks

## **Project Budget:**

This project will be developed using open-source tools and software, therefore there will be no cost involved.

## **Project Team:**

The project will be developed by a team of one developer only.

# Functionality & Implementation:

## Input Specifications:

### ☐ **User Profile Settings:**

Allowing users to update their profile information, including profile picture, status message, and notification preferences.

### ☐ **Text Input:**

Messages entered by the user for sending to the recipient

By defining the input requirements for the chatting application, we can ensure that chatting application will be more interactive and engaging for the users.

## Output Specifications:

- **Chat History:** Displaying the conversation history between the user and the recipient.
- **Received Messages:** Displaying the incoming messages from the recipient.
- **Sent Messages:** Displaying the messages sent by the user.
- **User Status:** Indicating the online/offline status of the user

By implementing & defining the output requirements for the chatting application, we can ensure that chatting application will be more interactive and engaging for the users.

# **Requirements of College Management System:**

## **Hardware Requirements:**

- Processor: Intel Core i3 or higher
- RAM: 4 GB or more
- Storage: 500 MB or more
- Monitor: 1280x800 or higher resolution display
- Input devices: Keyboard and mouse or game controller
- Reliable Internet connection

## **Software Requirements:**

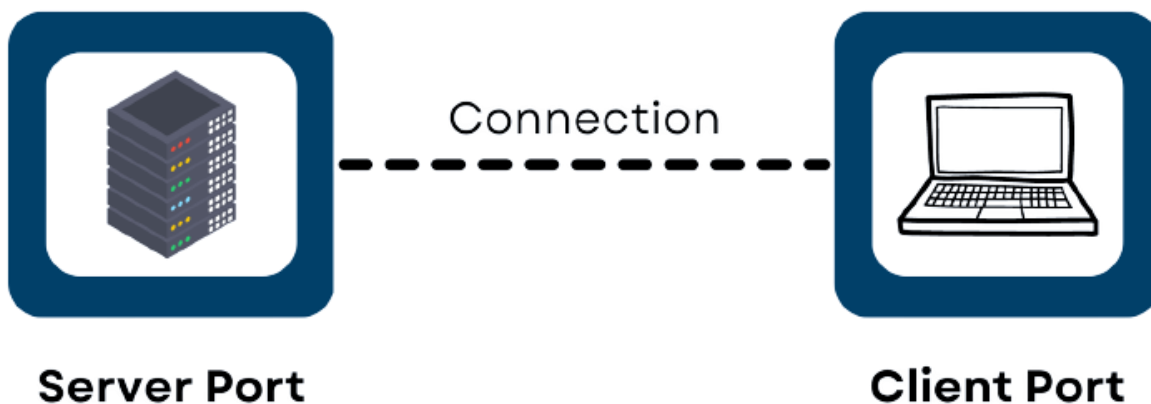
- Operating System (e.g., Windows, macOS, Linux)
- Java Development Kit (JDK) installed
- Integrated Development Environment (IDE) such as Eclipse or IntelliJ IDEA
- Swing and AWT libraries for graphical user interface (GUI) development
- Understanding of socket programming concepts for establishing network connections and data transmission
- Networking libraries for handling network communication and socket programming

# Socket Programming

Socket programming is a crucial component of the chatting application project as it enables communication between the client and server. By using sockets, the application can establish connections, send and receive messages, handle data transmission, and provide real-time updates to users. It allows processes on different machines to connect and exchange data seamlessly, ensuring reliable and instant communication in the chatting application.

## ➤ *Use in Project*

- Establishing Connection
- Sending and Receiving Messages
- Handling Data Transmission
- Enabling Real-time Updates



# *Execution*

## Java Classes:

- MainWindow.java
- Server.java
- Client.java

### **MainWindow.java**

This class extends JFrame and implements the Runnable interface. It represents the main window of the application. It creates a window with an image, starts a new thread, and handles the animation for the initial appearance of the window. After a certain time, it makes the window invisible and opens new Server & Client window.

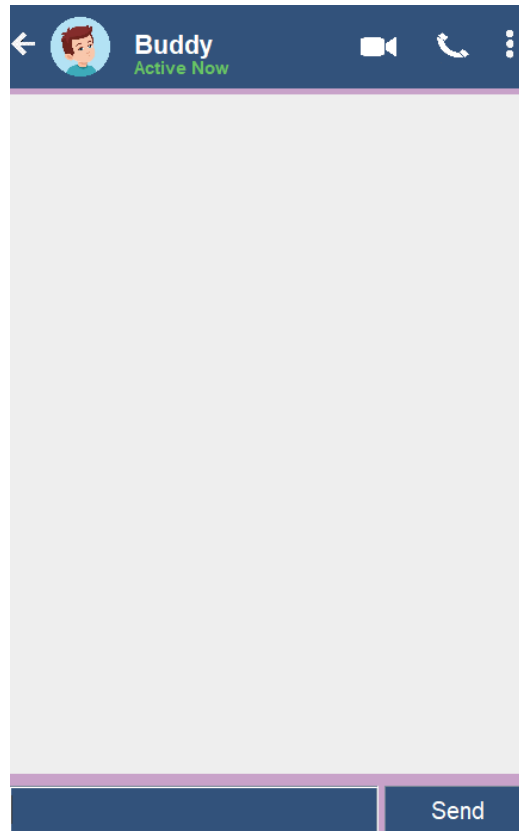
```
ImageIcon i1 = new ImageIcon(ClassLoader.getResource("icons/TalkBuddy.png"));  
Image i2 = i1.getImage().getScaledInstance(1000, 700, Image.SCALE_DEFAULT);
```

```
int x = 1;  
for (int i = 2; i <= 600; i+=4, x+=1) {  
    setLocation(600 - ((i + x)/2), 350 - (i/2));  
    setSize(i + 3*x, i + x/2);  
}
```

### **Client.java**

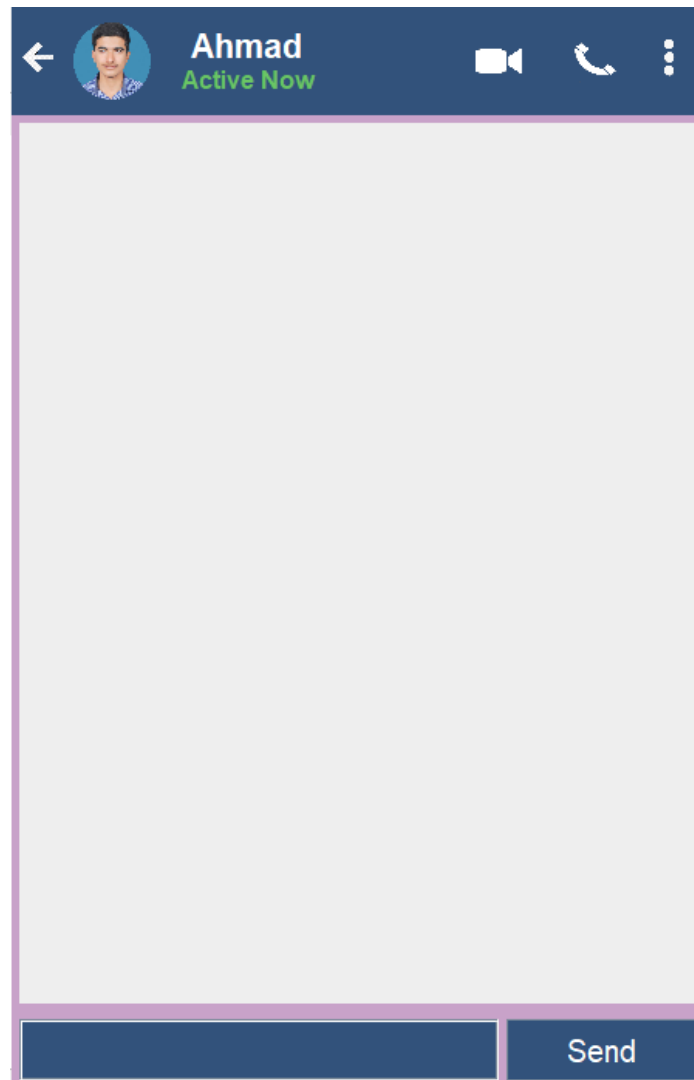
The client interface consists of a chat window where users can send and receive messages. The program establishes a socket connection with a server running on the local host and listens for incoming messages. Outgoing messages are sent to the server using a DataOutputStream. The GUI features various components such as profile icons, status indicators, and a text input field for sending messages. Messages are displayed in a panel with timestamps, and the panel is updated dynamically as new messages arrive. Overall, the code implements a basic chat client functionality.





## Server.java

The execution process starts by creating an instance of the **Server** class, which sets up the **graphical user interface (GUI)** for the chat application. The GUI includes components such as text fields, buttons, and panels for displaying messages. The code then establishes a **server socket on port 6008** to listen for incoming connections. Once a client connects, the server creates input and output streams for communication with the client. The server continuously listens for messages from the client using a loop, and each received message is displayed in the GUI by formatting it into a panel. The process continues indefinitely, allowing for real-time chat communication between the server and connected clients.



## The OOP Principles & Concepts

All OOP languages have features that facilitate the implementation of the object-oriented model. In Java, these features are:

### 7. Encapsulation

**Encapsulation** is a mechanism that binds data and code together, protecting them from external interference and misuse. It is achieved through the use of access modifiers such as public, private, and protected, which control the visibility of data and methods in a class.

## 8. Inheritance

**Inheritance** is the process by which one class (called the subclass) acquires the properties and behavior of another class (called the superclass). This allows for the creation of hierarchical classifications of objects, where subclasses inherit the characteristics of their superclasses and add additional functionality.

## 9. Polymorphism

**Polymorphism** is a feature that allows objects of different classes to be treated as if they belong to a common class. This means that a single interface or method can be used to perform operations on objects of different types. This is achieved through the use of method overloading and overriding, as well as through the implementation of interfaces and abstract classes.

## Usage of OOP Concepts in Project

- **Classes and Objects:** The code defines a class called Client which represents the client application for a chat program. Objects of this class are created in the main method to run the chat client.
- **Encapsulation:** The class Client encapsulates the implementation details of the chat client, including the GUI components, event handling, and network communication. The instance variables and methods are appropriately encapsulated to control access and ensure data integrity.
- **Inheritance:** The code does not explicitly demonstrate inheritance, as the Client class does not extend any other class. However, inheritance could be used if the Client class were to inherit from a more generic ChatClient class, which could provide common functionalities to multiple types of chat clients.

- **Abstraction:** The Client class abstracts the complexities of the chat client implementation by providing high-level methods and GUI components. The underlying details, such as network communication, are hidden from the user.
- **Polymorphism:** Polymorphism is not explicitly demonstrated in the provided code. However, it could be utilized if there were multiple types of clients, each implementing a common interface or extending an abstract class, but with different implementations of certain methods.
- **Method Overriding:** The actionPerformed method in the Client class overrides the corresponding method defined in the ActionListener interface. This allows the custom implementation of the method to handle button click events.
- **Method Overloading:** Method overloading is not demonstrated in the provided code. It allows defining multiple methods with the same name but different parameter lists, enabling flexibility in method invocation.

**By utilizing these OOP principles & concepts, this project will create a well-structured and maintainable codebase for the chatting application: TalkBuddy**

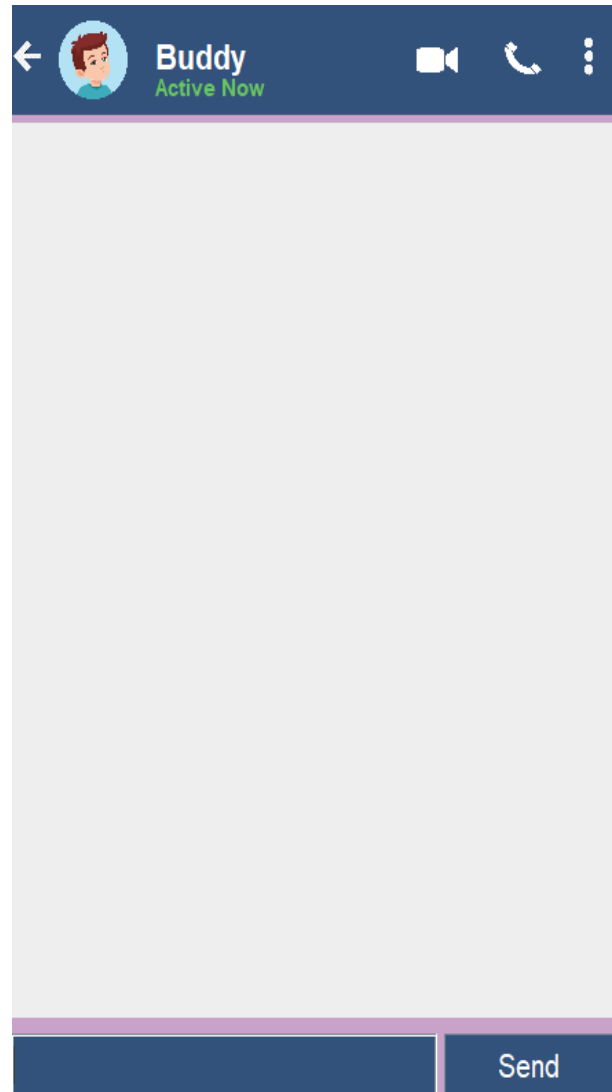
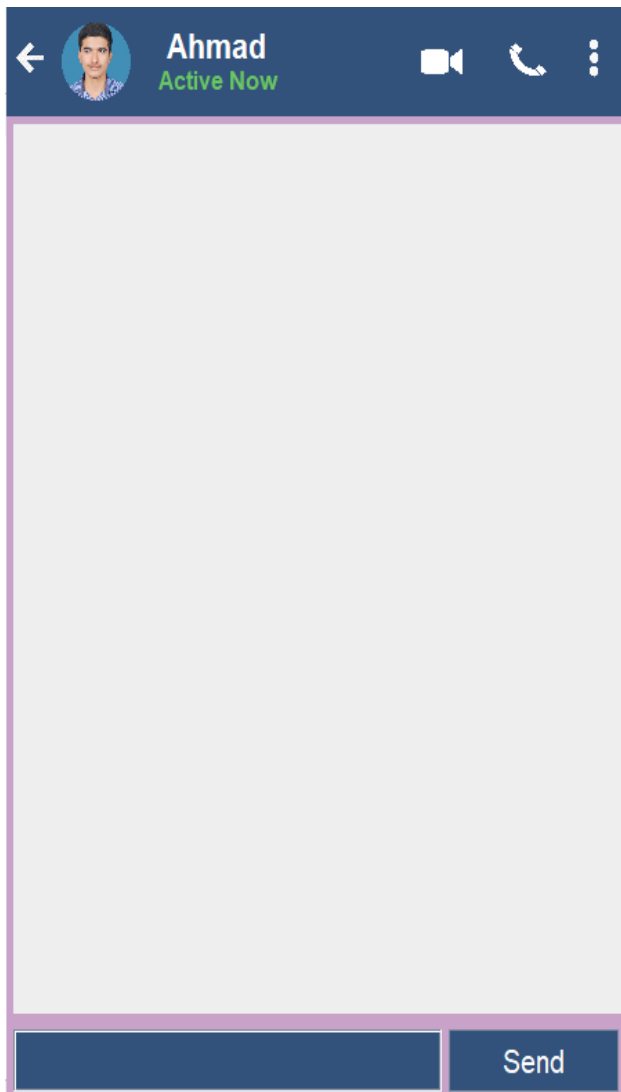
## **Future Enhancements & Works**

- ❖ **Multi-User Support:** This feature involves enabling multiple users to participate in a chat session simultaneously. It includes implementing user management functionalities such as creating user accounts, managing user profiles, and allowing users to join or leave chat rooms or groups. It enhances the collaboration and communication capabilities of the chat client by facilitating interactions among a larger number of participants.
- ❖ **Audio and Video Calls:** Integrating audio and video call functionalities into the chat client allows users to have real-time voice and video conversations. This feature typically involves establishing peer-to-peer or server-mediated connections for voice and video streaming, implementing call controls such as mute, speaker options, and video on/off

toggles, and ensuring high-quality audio and video transmission to enhance the communication experience.

- ❖ **Artificial Intelligence-based Features:** Leveraging artificial intelligence (AI) in the chat client can enhance its functionality in various ways. This could include implementing AI-powered chatbots that can provide automated responses and assistance to users' queries, utilizing natural language processing (NLP) techniques to enable intelligent message analysis and sentiment analysis, or incorporating machine learning algorithms to personalize the user experience and make intelligent suggestions.
- ❖ **Voice Messaging:** Voice messaging allows users to send and receive recorded voice messages within the chat client. It involves implementing audio recording and playback functionalities, providing controls for recording, sending, and playing voice messages, and managing voice message storage and retrieval. Voice messaging adds a convenient and expressive communication option for users who prefer voice-based interactions.
- ❖ **File Transfer:** Enabling file transfer capabilities in the chat client allows users to share files with each other. This feature includes implementing file selection and upload functionalities, managing file storage and retrieval, and ensuring secure and efficient file transmission. File transfer enhances collaboration and information sharing by enabling users to exchange documents, images, and other files directly within the chat client.
- ❖ **Emoticons and Rich Text:** Emoticons and rich text support allow users to express themselves creatively and enhance the visual appeal of their messages. It involves implementing a range of emoticons or emojis that users can select and insert into their messages, as well as supporting rich text formatting options such as bold, italics, underline, and different font styles. Emoticons and rich text features add a touch of personalization and enhance the user experience in the chat client.
- ❖ **User Authentication and Authorization:** User authentication and authorization are essential for ensuring the security and privacy of the chat client. This involves implementing secure user authentication mechanisms, such as username/password authentication or integration with third-party authentication services, to verify user identities. User authorization controls can be implemented to define access rights and permissions for different user roles, ensuring that users can access and interact with appropriate chat features based on their privileges.

## Display:

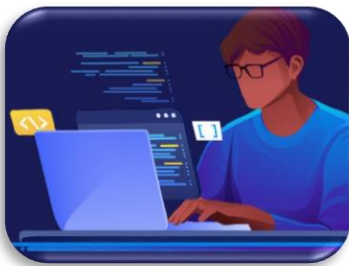


## **Potential Questions:**

Q) How can the chat client's security be further strengthened to protect user privacy and prevent unauthorized access or data breaches?

Q) How can the group chat feature be developed to improve collaboration and communication among participants? recommend ations for academic support or interventions?

Q) How can artificial intelligence and natural language processing techniques be leveraged to improve the chat client's functionality, such as providing automated suggestions, intelligent message filtering, or sentiment analysis?



## **Conclusion:**

**In a nut shell**, the talkBuddy chatting application is a comprehensive and user-friendly solution that caters to the communication needs of users. It provides a seamless and efficient platform for users to connect, chat, and collaborate. **The integration of various technologies, including Java, Socket Programming, and AI**, highlights the project's technological prowess and its ability to leverage cutting-edge tools for developing innovative software solutions.

The strengths of the talkBuddy application lie in its multi-user support, audio and video call capabilities, **AI-based features, voice messaging, file transfer, emoticons, and rich text**.

**In summary**, *talkBuddy* offers a robust and reliable chatting application that caters to the needs of users seeking seamless communication and collaboration. With its user-friendly interface and advanced features, talkBuddy aims to enhance the overall chatting experience and foster meaningful connections between users.

## **Reference Citations:**

<https://www.w3schools.com/>

<https://www.codecademy.com/learn/learn-java>

<https://stackify.com/java-tutorials//>

<https://www.learnjavaonline.org/>

<https://www.youtube.com/>

- **Signature of the student:** \_\_\_\_\_
- **Name of the Instructor:** **Dr. Nauman Ali Khan**
- **Signature of the Instructor :** \_\_\_\_\_

*THE END*