



CHRIST
(DEEMED TO BE UNIVERSITY)
B A N G A L O R E • I N D I A

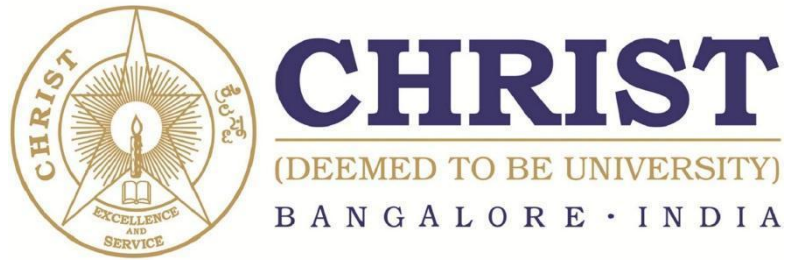
**DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING**

**(CS433P)
PROGRAMMING PARADIGMS**

***B. Tech – Computer Science and Engineering
(Artificial Intelligence and Machine Learning)***

**School of Engineering and Technology,
CHRIST (Deemed to be University),
Kumbalagodu, Bengaluru-560 074**

April 2023



TO-DO LIST

Submitted by

MEENAKSHI SA - 2162058

MEHWISH SULTANA -2162060

SANIYA THOMAS - 2162064

CHRIST (Deemed to-be University)

FACULTY OF ENGINEERING

Vision

Excellence and Service

Mission

CHRIST is a nurturing ground for an individual's holistic development to make an effective contribution to the society in a dynamic environment.

Department of

Computer Science and Engineering

Vision

To fortify Ethical Computational Excellence

Mission

- Imparts core and contemporary knowledge in the areas of Computation and Information Technology
 - Promotes the culture of research and facilitates higher studies
- Acquaints the students with the latest industrial practices, team building and entrepreneurship
- Sensitizes the students to serve for environmental, social & ethical needs of society through lifelong learning.

ABSTRACT

In today's fast-paced world, it can be challenging to keep track of all the tasks that need to be completed. To help individuals manage their daily tasks effectively, we propose creating a to-do list application in Java. This application will allow users to create, update, and manage their tasks in a simple and organized manner.

The to-do list application will be designed using several software tools, including PHP MyAdmin, MySQL, NetBeans, and Xampp. We will use PHP MyAdmin and MySQL to create a database that will store all the user's task data, such as the task description, due date, and status. We will use NetBeans, an integrated development environment (IDE), to create the Java code for our application. The user interface will be created using Java Swing, a graphical user interface toolkit, and will allow users to add, edit, and delete tasks. To retrieve data from the database, we will use SQL queries, and JDBC to connect to the MySQL database from our Java code.

The to-do list application will have a login screen where users can enter their username and password to access their task list. Once logged in, users will be directed to the main screen, where they can view their task list. The user interface will allow users to add new tasks, edit existing tasks, mark tasks as complete, update the due date, or delete the task. All the changes made by the user will be saved in the database.

Finally, we will use Xampp to deploy the application, allowing users to access the to-do list application from a web browser on their local machine. Xampp will enable us to create a WAR (Web ARchive) file of our Java project and deploy it on Xampp.

REQUIREMENT SPECIFICATION

Software Requirements:

1. **PHPMyAdmin:** PHPMyAdmin is a web-based tool used to manage MySQL databases. It provides an easy-to-use interface that allows developers to create, edit, and delete databases, tables, and records. PHPMyAdmin is written in PHP and is typically included in most web hosting packages. For a to-do list application, PHPMyAdmin can be used to create and manage the database that stores the list of tasks.
2. **MySQL:** MySQL is an open-source relational database management system (RDBMS) that is widely used for web-based applications. It is fast, reliable, and scalable, making it an ideal choice for applications that require the storage and retrieval of large amounts of data. In a to-do list application, MySQL can be used as the database engine to store the list of tasks.
3. **NetBeans:** NetBeans is an integrated development environment (IDE) for Java, PHP, and other programming languages. It provides a user-friendly interface for developing, testing, and debugging applications. NetBeans supports a wide range of features, including syntax highlighting, code completion, and debugging tools. For a to-do list application in PHP, NetBeans is used to write, test, and debug the PHP code.
4. **XAMPP:** XAMPP is an open-source software stack that includes Apache web server, MySQL, PHP, and other tools. It provides a convenient way to set up a web development environment on a local machine. XAMPP can be installed on Windows, Linux, or Mac OS, and provides a simple interface for starting and stopping the web server and database server. In a to-do list application, XAMPP can be used to set up a local development environment for testing and debugging the application before deploying it to a live server.

Hardware Requirements:

1. **Processor:** A modern processor with a clock speed of at least 1 GHz or higher should be sufficient for running a basic to-do list application.
2. **Memory (RAM):** The minimum recommended amount of RAM is 2 GB. However, depending on the size of the application and the number of tasks stored, the memory requirement may be higher.
3. **Storage:** A minimum of 100 MB of free storage space should be sufficient for the application and its associated data.
4. **Display:** The application should be designed to be compatible with a range of display sizes and resolutions. However, it is recommended to test the application on different displays to ensure that it is usable and legible.
5. **Operating System:** The application should be compatible with the operating system(s) that the target users are likely to use. This may include Windows, macOS, Linux, and mobile operating systems such as iOS and Android.
6. **Input Devices:** The application should be designed to be compatible with a range of input devices, including keyboard, mouse, touchpad, and touch screen.

7. Internet Connectivity: If the application requires internet connectivity, then the hardware should be able to connect to the internet via Ethernet, Wi-Fi, or cellular data.

Application requirement:

1. User Interface: The application should have a user-friendly interface that allows the user to create and manage tasks easily. The interface displays the tasks in a clear and organized manner, and allows the user to edit or delete tasks as needed.
2. Task Creation: The application allows the user to create tasks by entering a task name, due date, and any other relevant information.
3. Task Management: The application allows the user to view, edit, and delete tasks.
4. Data Persistence: The application should store the user's tasks and settings in a database or file so that the user can access them later.
5. Performance: The application is efficient and responsive, with fast load times and minimal lag.
6. Compatibility: The application is compatible with different operating systems and devices, such as desktops, laptops, and mobile devices.
7. Accessibility: The application is accessible to users with disabilities, such as those who use screen readers or have limited mobility.
8. Testing: The application should undergo rigorous testing to ensure that it works as intended and meets all requirements.

WORKING DESCRIPTION

To create a to-do list application in Java, we will be using several software tools including PHP MyAdmin, MySQL, NetBeans, and Xampp. The application will be designed to allow users to create and manage their daily tasks in an organized manner.

Database Creation:

First, we will create a MySQL database using PHP MyAdmin. This database will store all the user's task data, such as the task description, due date, and status. We will create a new database with a user table and a task table. The user table will store the user's information, such as username and password, while the task table will store the user's task data.

Java Code Creation:

We will use NetBeans, an integrated development environment (IDE), to create the Java code for our application. First, we will create a new Java project in NetBeans. Then, we will create a package for our application and add classes for the user interface, database connectivity, and task data.

User Interface:

Our to-do list application will have a graphical user interface (GUI) that will allow users to add, edit, and delete tasks. We will use Java Swing, a graphical user interface toolkit, to create the GUI. The user interface will consist of several components, including text fields, buttons, and lists.

We will create a login screen that will prompt the user to enter their username and password. Once the user logs in, they will be directed to the main screen, which will display their task list. The user will be able to add new tasks, edit existing tasks, mark tasks as complete, update the due date, or delete the task.

Database Connectivity:

To retrieve the user's task list from the database, we will use SQL queries to fetch the data from the MySQL database. We will create a class for database connectivity, which will include methods for connecting to the database, inserting new tasks, retrieving tasks, updating tasks, and deleting tasks.

We will use JDBC (Java Database Connectivity) to connect to the MySQL database from our Java code. We will write SQL queries to retrieve data from the database and populate the task list on the GUI.

Deployment:

Finally, we will use Xampp, a web server solution, to deploy the application. Xampp will allow users to access the to-do list application from a web browser on their local machine. We will create a WAR (Web ARchive) file of our Java project and deploy it on Xampp.

Exception Handling:

Exception handling is an important aspect of developing any software application, including a to-do list application in Java. Exception handling is the process of handling unexpected errors or exceptions that occur during the execution of the application. These exceptions can be caused by various factors such as invalid user input, network errors, or hardware failures.

In Java, exceptions are objects that represent errors or exceptional conditions that can occur during program execution. Exceptions are thrown when an error occurs, and they can be caught and handled by the application to prevent the application from crashing or terminating unexpectedly.

In a to-do list application, exception handling can be used to prevent the application from crashing when unexpected errors occur. For example, if a user enters an invalid date format while creating a new task, an exception can be thrown and caught by the application to provide the user with an error message indicating that the date format is invalid.

Java provides various constructs for handling exceptions, such as try-catch blocks, throw statements, and finally blocks. A try-catch block is used to catch and handle exceptions that occur during program execution. If an exception is thrown within the try block, it is caught by the catch block, which can handle the exception or rethrow it to be caught by an outer try-catch block. A throw statement can be used to explicitly throw an exception, and a finally block is used to execute code that needs to be executed regardless of whether an exception is thrown or not.

When developing a to-do list application in Java, it is important to consider all the possible exceptions that can occur and handle them appropriately. Some common exceptions that can occur in a to-do list application include database connection errors, file I/O errors, and input validation errors.

By implementing robust exception handling mechanisms, developers can ensure that their to-do list application is more reliable and resilient to unexpected errors. This can improve the user experience, as users will not have to deal with unexpected crashes or error messages, and can continue to use the application without interruption.

source code:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    //GEN-FIRST:event_jButton1ActionPerformed

        String Data = null;

        int[] row = jTable1.getSelectedRows();

        int[] columns = jTable1.getSelectedColumns();

        for (int i = 0; i < row.length; i++) {

            for (int j = 0; j < columns.length; j++) {

                Data = (String) jTable1.getValueAt(row[i],
columns[j]);

            }

        }

        //exception handling

        try{

            PreparedStatement pst = con.prepareStatement("DELETE
FROM todo WHERE work=?");

            pst.setString(1, Data);

            pst.execute();

            list_frame lst = new list_frame();

            lst.setBounds(300,300,500,500);

            lst.setVisible(true);

        }
    }
}
```

```
        dispose() ;  
  
    }  
  
    catch (Exception e) {  
  
        System.out.println(e) ;  
  
    }  
  
}
```

Event Handling:

Event handling is a critical component of developing a to-do list application in Java. Event handling is the process of responding to user input or other events that occur within the application, such as mouse clicks or button presses. Event handling allows the application to respond to user actions and update the interface accordingly.

In Java, event handling is implemented using event listeners and event objects. Event listeners are objects that listen for events, and event objects are objects that represent the events that have occurred. When an event occurs, the event listener receives the event object and performs the appropriate action based on the event.

In a to-do list application, event handling can be used to respond to user input, such as creating a new task, editing an existing task, or deleting a task. For example, when the user clicks on the "Add Task" button, an event listener can be triggered, which can create a new task object and add it to the to-do list database. Similarly, when the user clicks on the "Edit Task" button, an event listener can be triggered, which can display a dialog box allowing the user to edit the task details.

Java provides various constructs for implementing event handling, such as event listeners, action listeners, and mouse listeners. Event listeners are the most general-purpose listeners and can handle all types of events, while action listeners are used for handling button clicks and other actions that generate events. Mouse listeners are used for handling mouse events such as clicks, drags, and drops.

When developing a to-do list application in Java, it is important to design the user interface with event handling in mind. The user interface should be designed to be intuitive and easy to use, with clear indications of the actions that can be performed. The event handling code should be structured in a modular and reusable way, to make it easier to maintain and extend the application in the future.

Overall, event handling is a critical aspect of developing a to-do list application in Java. By implementing robust event handling mechanisms, developers can ensure that their application is more responsive to user input and provides a better user experience.

Source Code:

```
//exception handling

try{

    PreparedStatement pst = con.prepareStatement("DELETE
FROM todo WHERE work=?");

    pst.setString(1, Data);

    pst.execute();

    list_frame lst = new list_frame();

    lst.setBounds(300,300,500,500);

    lst.setVisible(true);

    dispose();

}

catch(Exception e){

    System.out.println(e);

}

}
```

Swing:

Swing is a graphical user interface (GUI) toolkit for Java that allows developers to create sophisticated and cross-platform desktop applications. Swing provides a rich set of components and widgets that can be used to create complex user interfaces for applications such as to-do lists, media players, and image editors.

Swing components are lightweight and have a consistent look and feel across different platforms, which makes them ideal for creating cross-platform applications. Swing also provides a powerful layout manager system that allows developers to design complex layouts for their applications, without having to deal with the low-level details of positioning and sizing individual components.

In a to-do list application, Swing can be used to create a user-friendly interface that allows the user to manage their tasks efficiently. For example, the application can use Swing components such as labels, text fields, buttons, and checkboxes to create a list of tasks that the user can interact with. The application can also use Swing layout managers to arrange the components in a logical and aesthetically pleasing way.

Swing also provides advanced components such as tables, trees, and list boxes, which can be used to create more sophisticated user interfaces. For example, the application can use a table to display a list of tasks along with their due dates, priorities, and other relevant information. The user can then sort the tasks based on various criteria, such as due date or priority, by clicking on the table headers.

Swing also provides support for event handling, which allows the application to respond to user input and other events. For example, the application can use event handling to respond to button clicks, mouse events, and keyboard events, allowing the user to interact with the application in a natural and intuitive way.

Overall, Swing is a powerful and versatile toolkit that can be used to create sophisticated and cross-platform user interfaces for a variety of applications, including to-do lists. By leveraging the rich set of components and layout managers provided by Swing, developers can create user-friendly and aesthetically pleasing interfaces that enhance the user experience and improve the productivity of the application.

Source code:

```
private void initComponents() {  
  
    jScrollPane1 = new javax.swing.JScrollPane();  
  
    jTable1 = new javax.swing.JTable();  
  
    jButton1 = new javax.swing.JButton();  
  
    jButton3 = new javax.swing.JButton();  
  
    jLabel1 = new javax.swing.JLabel();  
}
```

```
jLabel2 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

setBackground(new java.awt.Color(51, 255, 204));

jTable1.setModel(new javax.swing.table.DefaultTableModel(

    new Object [][] {

        },

    new String [] {

        "Task", "Date"

    }

) {

    Class[] types = new Class [] {

        java.lang.String.class, java.lang.String.class

    };

    public Class getColumnClass(int columnIndex) {

        return types [columnIndex];

    }

});

jScrollPane1.setViewportViewView(jTable1);

jButton1.setText("Remove");
```

```
jButton1.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt)
    {
        jButton1ActionPerformed(evt);
    }
});

jButton3.setText("Add");

jButton3.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt)
    {
        jButton3ActionPerformed(evt);
    }
});

jLabel1.setFont(new java.awt.Font("Verdana", 0, 24)); // NOI18N
jLabel1.setText("Stuff to do");

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)

        .addGroup(layout.createSequentialGroup()

            .addGap(132, 132, 132)

            .addComponent(jButton3)

            .addGap(132, 132, 132)

            .addComponent(jButton1))

        .addGroup(layout.createSequentialGroup()

            .addGap(108, 108, 108)

            .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 330,
javax.swing.GroupLayout.PREFERRED_SIZE)))

        .addContainerGap(104, Short.MAX_VALUE))

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

            .addGap(0, 0, Short.MAX_VALUE)

            .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 161,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(40, 40, 40)

            .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 94,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(47, 47, 47))

        );

        layout.setVerticalGroup(

layout.createSequentialGroup(javax.swing.GroupLayout.Alignment.LEADING)
```



```
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())

        .addContainerGap(28, Short.MAX_VALUE)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)

        .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 34,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(jLabel2))

        .addGap(18, 18, 18)

        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 290,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addGap(18, 18, 18)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)

        .addComponent(jButton1)

        .addComponent(jButton3))

        .addGap(34, 34, 34))

    );

    pack();

}
```

JDBC connection to mysql database:

```
package to_do_list;

import java.sql.*;

import com.mysql.jdbc.ResultSetMetaData;

public class db {

    Connection con = null;

    java.sql.PreparedStatement pst;

    public static Connection dbconnect()

    {

        try{

            Class.forName("com.mysql.jdbc.Driver");

            Connection

conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/to_do_lis
t","root","");

            return conn;

        }

        catch(Exception e) {

            System.out.println(e);

            return null;

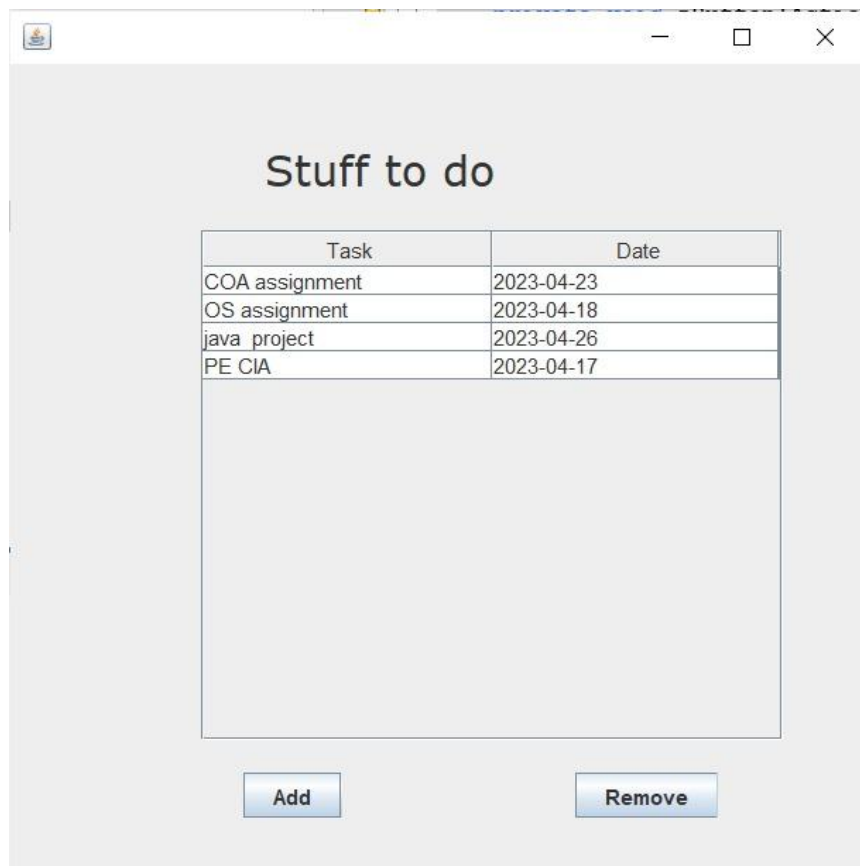
        }

    }

}
```

SNAPSHOTS

List Framework:

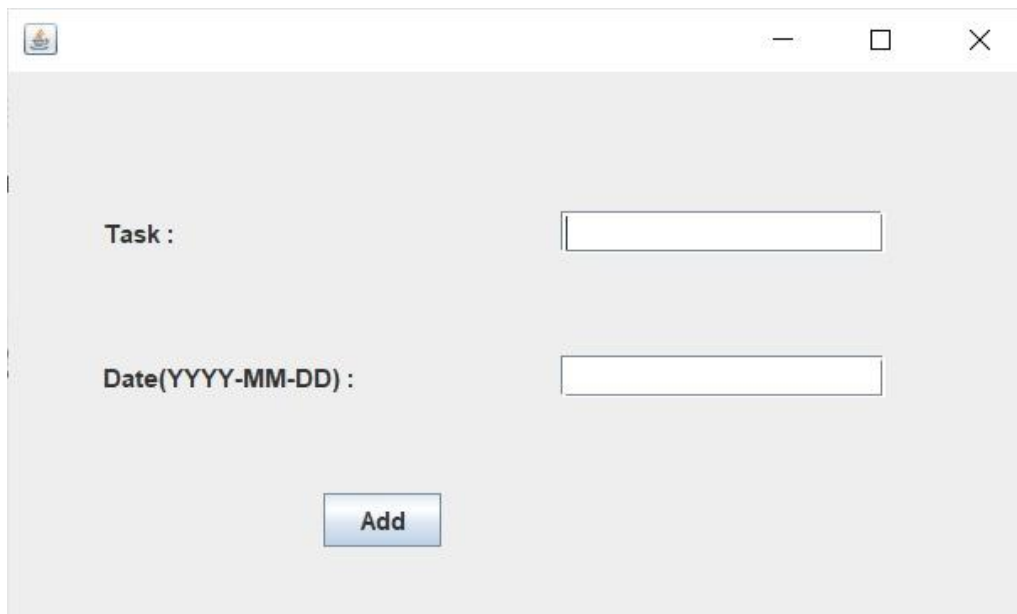


The screenshot shows a window titled "Stuff to do" with a table of tasks and their due dates. The table has two columns: "Task" and "Date". The tasks listed are "COA assignment", "OS assignment", "java project", and "PE CIA". The dates are "2023-04-23", "2023-04-18", "2023-04-26", and "2023-04-17" respectively. Below the table are two buttons: "Add" and "Remove".

Task	Date
COA assignment	2023-04-23
OS assignment	2023-04-18
java project	2023-04-26
PE CIA	2023-04-17

Buttons: Add, Remove

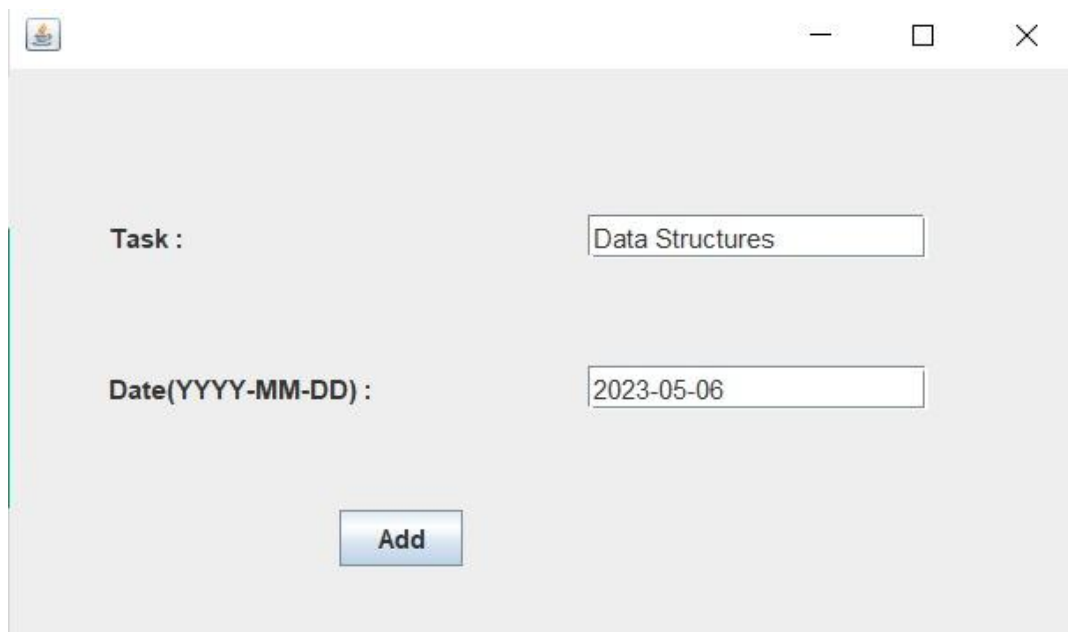
Add Framework



Task :

Date(YYYY-MM-DD) :

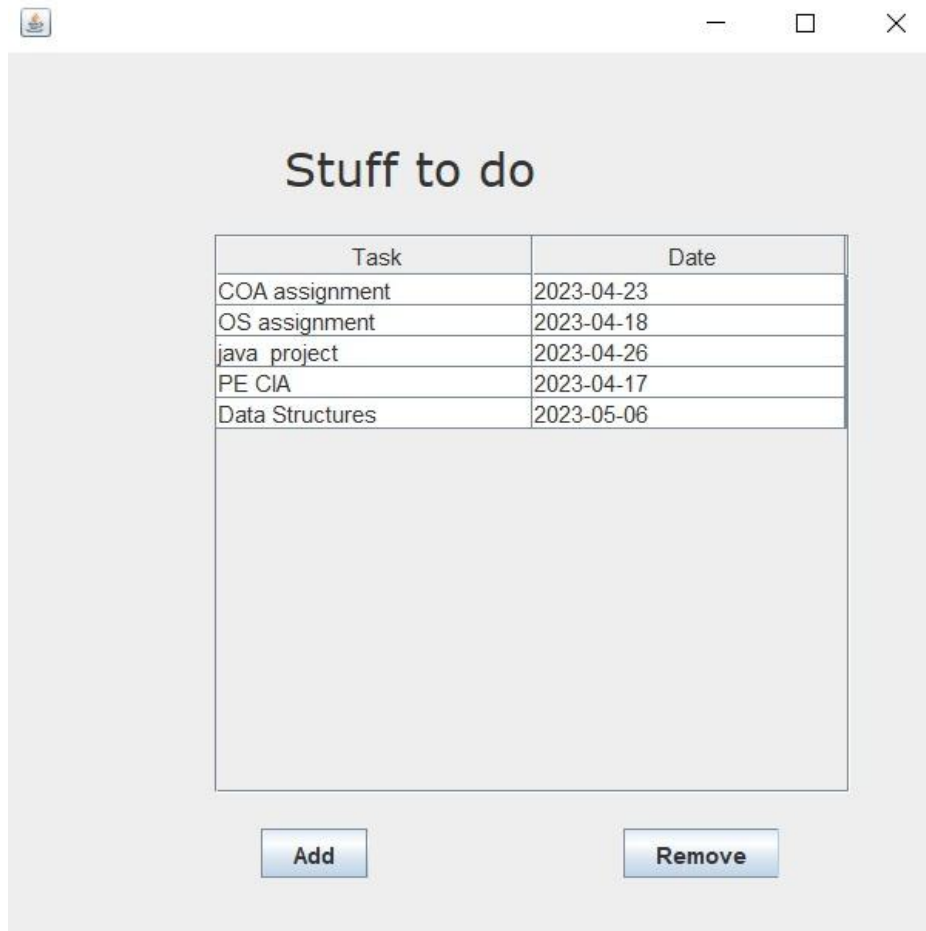
Adding Task



Task :

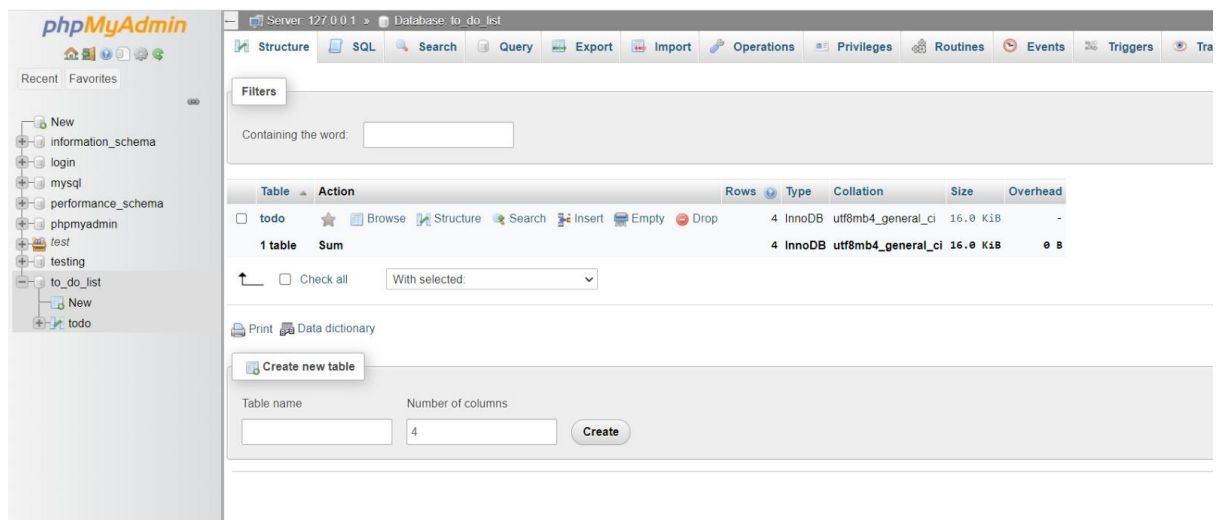
Date(YYYY-MM-DD) :

Task added to the list



Task	Date
COA assignment	2023-04-23
OS assignment	2023-04-18
java project	2023-04-26
PE CIA	2023-04-17
Data Structures	2023-05-06

Database



phpMyAdmin

Server: 127.0.0.1 - Database: to_do_list

Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Tra

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> todo	<input type="checkbox"/> Browse <input type="checkbox"/> Structure <input type="checkbox"/> Search <input type="checkbox"/> Insert <input type="checkbox"/> Empty <input type="checkbox"/> Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
1 table	Sum	4	InnoDB	utf8mb4_general_ci	16.0 KiB	0 B

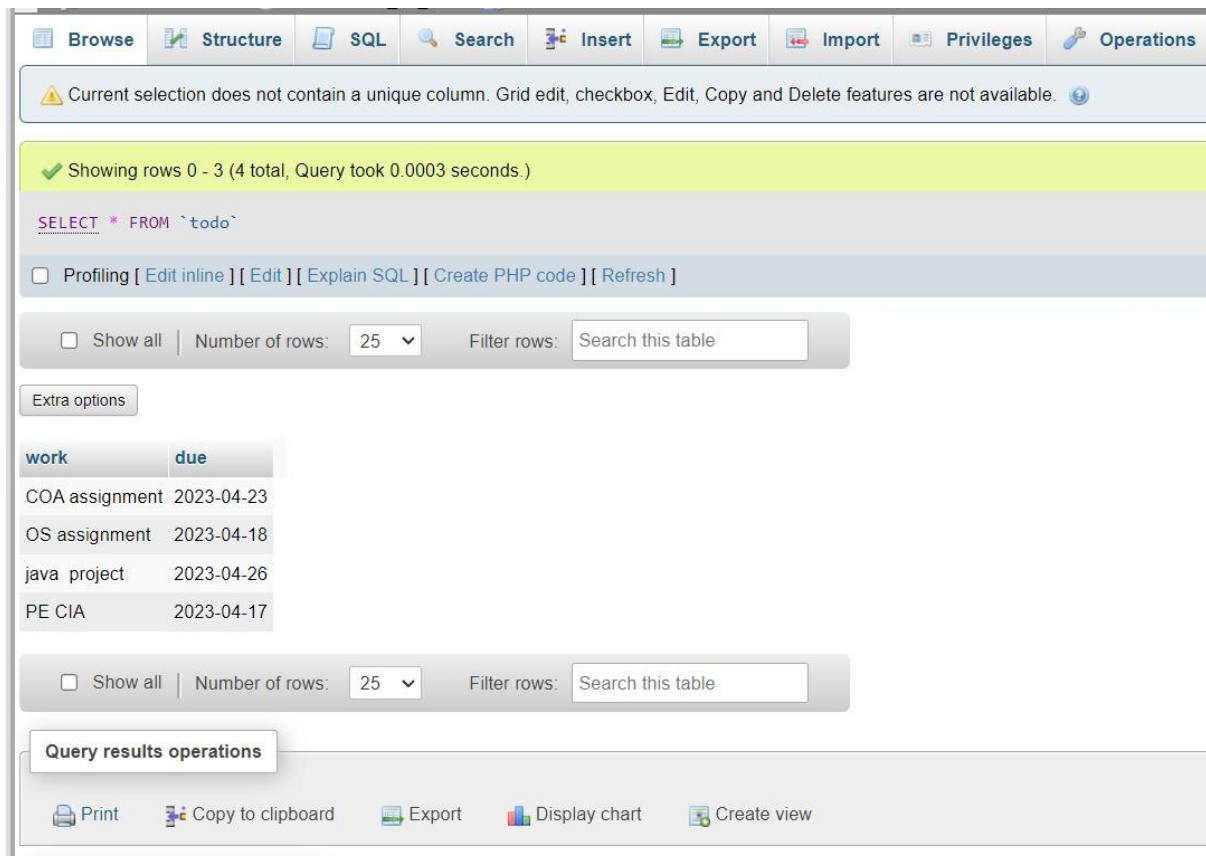
☐ Check all With selected:

Print Data dictionary

Create new table

Table name Number of columns

Table for todolist



CONCLUSION:

We have successfully developed a To-Do list application using java. With the help of PHP MyAdmin, NetBeans and Xampp we were able to build an application to help users to manage their daily tasks effectively. The application's user interface will be user-friendly, and users will be able to add, edit, and delete tasks with ease. The database will ensure that all the user's task data is stored securely and efficiently. The use of SQL queries and JDBC will enable us to retrieve data from the database and populate the task list on the GUI.

Overall, the to-do list application in Java is a simple yet powerful solution for managing daily tasks. It is ideal for individuals who want to stay organized and productive in today's fast-paced world. The application's use of industry-standard software tools and efficient code development ensures its robustness and scalability, making it an excellent choice for individuals and organizations alike.

REFERENCES

1. [Exception Handling in Java | Java Exceptions - javatpoint](#)
2. [Exceptions in Java - GeeksforGeeks](#)

3. [AWT Event Handling \(tutorialspoint.com\)](https://www.tutorialspoint.com/java/awt_event_handling.htm)
4. [www.programiz.com](https://www.programiz.com/java/tutorial/examples/awt)
5. Herbert Schildt, “Java The Complete Reference” , Ninth Edition, McGraw Hill Publishers 2014.
6. Paul Deitel and Harvey Deitel , “Java How to program”, Tenth Edition, Deitel, 2016.