

**A Project Report
On
“WEB-BASED TODO LIST APPLICATION”
Submitted to the
Department of MCA
In partial fulfillment of the
MASTER OF COMPUTER APPLICATIONS**

**Under the guidance of
Mrs. AMRUTHA KARTHIKEYAN**

Project Done by

**TONY JOHN
Reg No:223242210280**

**VISHNU SABU
Reg No:223242210288**



**DEPARTMENT OF MCA
UNION CHRISTIAN COLLEGE
ALUVA, KERALA
December 2023**

**UNION CHRISTIAN COLLEGE
ALUVA, KERALA**



CERTIFICATE

This is to certify that the project entitled “Web-Based To-Do List Application” has been successfully carried out by TONY JOHN (Reg No:223242210280), VISHNU SABU (Reg No:223242210288) in partial fulfillment of the Course Master of Computer Applications.

**Mrs. AMRUTHA KARTHIKEYAN
INTERNAL GUIDE**

**Mrs. SHERNA MOHAN
HEAD OF THE DEPARTMENT**

Date:

**UNION CHRISTIAN COLLEGE
ALUVA, KERALA**



CERTIFICATE

This is to certify that the project entitled “Web-Based ToDo List Application” has been successfully carried out by TONY JOHN (Reg No:223242210280), VISHNU SABU (Reg No:223242210288), in partial fulfillment of the Course Master of Computer Applications under my guidance.

Date:

**Mrs. AMRUTHA KARTHIKEYAN
INTERNAL GUIDE**

**UNION CHRISTIAN COLLEGE
ALUVA, KERALA**



DECLARATION

We TONY JOHN, VISHNU SABU hereby declare that the project work entitled “WEB-BASED TODO LIST APPLICATION” is an authenticated work carried out by us at UNION CHRISTIAN COLLEGE, ALUVA under the guidance of Mrs. AMRUTHA KARTHIKEYAN for the partial fulfillment of the course MASTER OF COMPUTER APPLICATIONS. This work has not been submitted for similar purpose anywhere else except to UNION CHRISTIAN COLLEGE, ALUVA.

We understand that detection of any such copying is liable to be punished in any way the school deems fit.

VISHNU SABU

TONY JOHN

**UNION CHRISTIAN COLLEGE
ALUVA, KERALA**



ACKNOWLEDGEMENT

First and foremost, of all we express our gratitude to God Almighty for giving us an opportunity to excel in my efforts to complete this project on time. We wish to express our deep sense of gratitude to **Dr. M.I Punnoose**, Principal of Union Christian College, Aluva for the support provided to us. We would like to express our gratitude and thanks to the Director of MCA, **Dr. AV Alex**, and Head of the Department **Ms. Sherna Mohan**, for providing us with the best facilities and atmosphere for completion of our project.

Our sincere thanks to our Internal Guide **Ms. Amrutha Karthikeyan** for the valuable guidance for project work that led us to complete the project in a determined way with in stipulated time.

Finally, we would like to express our sincere thanks to our family and our friends, for always being a source of inspiration, and for their undying support and encouragement without which this project would not have been a success.

Tony John

Vishnu Sabu

TABLE OF CONTENTS

ABSTRACT	1
1. INTRODUCTION	2
1.1 INTRODUCTION	2
1.2 PROBLEM STATEMENT	2
1.3 SCOPE AND RELEVANCE OF THE PROJECT	3
1.4 OBJECTIVES	3
2. SYSTEM ANALYSIS	4
2.1 INTRODUCTION	4
2.2 EXISTING SYSTEM	4
2.2.1 Limitations of Existing System	4
2.3 PROPOSED SYSTEM	4
2.3.1 Advantages of Proposed System	5
2.4 FEASIBILITY STUDY	5
2.4.1 Technical Feasibility	5
2.4.2 Operational Feasibility	5
2.4.3 Economic Feasibility	6
2.5 SOFTWARE ENGINEERING PARADIGM APPLIED	6
3. SYSTEM DESIGN	7
3.1 INTRODUCTION	7
3.2 DATABASE DESIGN	8
3.2.1 Entity Relationship Model	9
3.3 OBJECT ORIENTED DESIGN- UML DIAGRAMS	9
3.3.1 Activity Diagram	9
3.3.2 Sequence Diagram	16
3.3.3 Use Case Diagram	17
3.3.4 Class Diagram	18
3.4 INPUT DESIGN	19

3.5 OUTPUT DESIGN	21
4. SYSTEM ENVIRONMENT	23
4.1 INTRODUCTION	23
4.2 SOFTWARE REQUIREMENT SPECIFICATION	23
4.3 HARDWARE REQUIREMENT SPECIFICATION	23
4.4 TOOLS, PLATFORMS	23
4.4.1 Front End Tool	23
4.4.2 Back End Tool	24
4.4.3 Operating System	24
5. SYSTEM IMPLEMENTATION	25
5.1 INTRODUCTION	25
5.2 CODING	25
5.2.1 Sample Codes	25
5.2.2 Code Validation and Optimization	31
6. SYSTEM TESTING	32
6.1 INTRODUCTION	32
6.2 UNIT TESTING	32
6.3 INTEGRATION TESTING	32
6.4 SYSTEM TESTING	33
6.4.1 Test Plan	33
6.4.2 Test Cases	33
7. SYSTEM MAINTENANCE	35
7.1 INTRODUCTION	35
7.2 MAINTENANCE	36
8. FUTURE ENHANCEMENT AND SCOPE OF FURTHER DEVELOPMENT	37
8.1 INTRODUCTION	37
8.2 MERITS OF THE SYSTEM	37
8.3 LIMITATIONS OF THE SYSTEM	37
8.4 CONCLUSIONS AND FUTURE ENHANCEMENT OF THE SYSTEM	38

ABSTRACT

Develop a simple yet functional to-do list application using PHP and MySQL. The application will allow users to create, read, update, and delete (CRUD) tasks from a to-do list. Manage tasks with customizable due dates, reminders, recurring schedules. It will also allow users to mark tasks as complete or incomplete. Application will be implemented as a web application using PHP and MySQL. The application will use a simple three-tier architecture, with a presentation layer, a business logic layer, and a data access layer. The presentation layer will be responsible for displaying the user interface and handling user input. The business logic layer will be responsible for performing operations on the to-do list, such as adding and deleting tasks. The data access layer will be responsible for interacting with the MySQL database. To-do list application will be a valuable tool for anyone who wants to keep track of their tasks and stay organized. It will be easy to use and will have all of the features that users need to manage their to-do lists effectively.

1 INTRODUCTION

1.1 INTRODUCTION

In today's fast-paced world, it is more important than ever to stay organized and on top of your tasks. In this project “Todo web-based applications” can help you do just that by providing a convenient and easy-to-use platform to manage your to-do list.

Todo web-based applications typically offer a variety of features such as the ability to create and manage multiple to-do lists, the ability to add notes ,to edit the note, to delete the note and mark if the note are completed with pop-up notifications .

Manage tasks with customizable due dates, reminders, recurring schedules.

Todo web-based applications can be used by people of all ages and from all walks of life. Students can use them to keep track of their assignments . Professionals can use them to manage their workload. Homemakers can use them to keep track of their chores and shopping lists. And everyone can use them to stay organized and on top of their personal tasks.

1.2 PROBLEM STATEMENT

The current manual method of managing to-do lists is inefficient and time-consuming. It is difficult to keep track of all of one's tasks, and it can be easy to forget about important task. This can lead to missed opportunities and a sense of overwhelm. A web-based to-do list application would provide a more efficient and effective way to manage tasks. It would allow users to easily create, edit, and delete tasks.

It is difficult to keep track of all of one's tasks when they are scattered across different pieces of paper, sticky notes, and calendar apps. It is easy to forget about important deadlines when they are not centrally located. It can be time-consuming to manually update to-do lists and track progress.

A web-based to-do list application would address all of the specific problems above. The application would allow users to Create and manage to-do lists in one central location. The application would save users time by making it easier to manage to-do lists. The application would help users to stay organized and on top of their tasks.

The application would help users to remember about important task and avoid missed opportunities. A web-based to-do list application would be a valuable tool for users who need a more efficient and effective way to manage their tasks. The application would provide users with a number of benefits, including increased efficiency and productivity, reduced stress and overwhelm, improved ability to meet tasks, and increased sense of accomplishment.

In today's fast-paced world, individuals, professionals, and teams encounter numerous challenges in organizing and managing their tasks efficiently. The reliance on paper-based or non-integrated digital solutions . To address these issues, the development of a user-friendly, web-based to-do application aims to revolutionize task management by providing a cohesive and intuitive platform for individuals and groups to streamline their activities.

The development of a web-based to-do application represents a response to the pressing need for a comprehensive, user-friendly task management system. By leveraging the power of web technologies, the application seeks to simplify task handling, encourage collaboration, and provide a unified platform for individuals and teams to enhance productivity and efficiency. This problem statement lays the groundwork for the creation of a web-based to-do application that aims to address the challenges faced by users in managing tasks efficiently and collaboratively.

1.3 SCOPE AND RELEVANCE OF THE PROJECT

The scope of the to-do web-based application project is to develop a web-based application that allows users to create, manage, and share to-do lists. The application will provide users with the features Create and manage to-do lists, Add, edit, and delete to-do items, Mark to-do items as complete, View and edit shared to-do lists.

To-do web-based applications are a popular and useful tool for individuals and teams to manage their tasks and stay organized. There is a large and growing market for to-do web-based applications and the development of a new to-do web-based application is a relevant and timely project.

The to-do web-based application developed in this project will be relevant to a wide range of users including Individuals who use to-do lists to manage their personal tasks, Teams who use to-do lists to collaborate on projects, Businesses who use to-do lists to manage their workload.

1.4 OBJECTIVES

The main objective of the Project is to Create a user-friendly interface that allows users to easily add, edit, and delete tasks. The objective of a web-based TODO application is to provide users with an easy and efficient way to manage their tasks and to-do list. It should be easy to use even for users with limited technical experience.

The application should be easy to use for users of all skill levels. It should have a clear and intuitive user interface, and it should be easy to add, edit, and delete tasks.

2. SYSTEM ANALYSIS

The system analysis of a web-based to-do application involves a comprehensive assessment of its requirements, functionalities, and user interactions. It begins by defining the scope, objectives, and stakeholders involved. Understanding user needs through interviews, surveys, and market research helps in identifying key features such as task creation, task completed mark and task deletion. Analysis also delves into the technological aspects, determining the required hardware, software, and databases to support the application.

2.1 INTRODUCTION

Consideration of security measures, data encryption, user authentication, and authorization mechanisms is vital. Additionally, the system analysis phase involves mapping out the user interface design, navigation flow, and user experience to ensure ease of use. Scalability, performance, and potential integration with other systems are also key considerations during this phase, ensuring the application's adaptability to evolving user needs and technological advancements.

2.2 . EXISTING SYSTEM

Many people used to keep track of their to-do lists, such as paper notebooks, sticky notes. These methods can be effective but they also have their drawbacks. Paper to-do lists can be easily lost or misplaced, and sticky notes can be cluttered and difficult to read and they do not offer the same features and functionality as a web-based to-do application.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- Traditional note-taking systems often lack robust organization and search features, making it difficult to find specific notes or information when needed. This can be particularly challenging for users with large collections of notes
- Many existing note-taking systems restrict users to a single note format, such as plain text or bulleted lists. This can limit creativity and flexibility in capturing ideas and information.
- Collaboration is often an essential aspect of todo web-based application projects. However, many existing note-taking systems lack built-in collaboration features, making it difficult for team members to share notes and work together effectively
- Integrating note-taking systems with other project management tools can be challenging, leading to silos of information and inefficiencies.
- Some note taking systems may not be accessible to users with disabilities, making it difficult for them to use the system effectively.

2.3 PROPOSED SYSTEM

A web-based to-do application that allows users to create and manage their to-do lists t. These applications are easy to use and offer a variety features such as ability to create tasks add notes, mark todo items as complete and remove tasks from lists. It is a simple text file to store todo items. Manage tasks with customizable due dates, reminders,

recurring schedules, Todo web-based applications typically offer a variety of features such as the ability to create and manage multiple to-do lists, the ability to add notes ,to edit the note, to delete the note and mark if the note are completed with pop-up notifications .

2.3.1 ADVANTAGES OF PROPOSED SYSTEM

A todo web-based application can help users to be more productive by providing a centralized location to store and manage their tasks. This can help to reduce the amount of time that users , and it can also help to ensure that tasks are not forgotten.odo web-based applications are also more flexible than traditional paper-based to-do lists. This is because they can be easily updated and changed as needed. For example, users can add new tasks, delete tasks. The proposed system has a user-friendly interface that makes it easy to add, edit, and delete tasks.The proposed system can send users notifications when they have completed tasks, which can help them to feel a sense of accomplishment.

2.4 FEASIBILITY STUDY

The purpose of this feasibility study is to assess the viability of developing a web-based todo application. The study considers the technical, financial, and operational aspects of the project. The findings of the study indicate that the project is feasible and has the potential to be successful .A to-do web application is a software program that allows users to create, read, update, delete.

2.4.1 Technical Feasibility

The development of a todo web-based application using PHP and MySQL is technically feasible. PHP is a widely used server-side scripting language that is well-suited for web development. MySQL is a popular open-source relational database management system (RDBMS) that is easy to use and integrate with PHP. The resources available online for developing PHP and MySQL applications The combination of these two technologies provides a powerful and flexible platform for building web applications.PHP and MySQL are both mature and well-supported technologies with a large community of developers. There are numerous resources available for learning and using these technologies, and there are many pre-built libraries and frameworks that can simplify the development process. With careful planning and design, a web-based todo application can be developed using PHP and MySQL that is scalable, secure, and easy to maintain.

2.4.2 Operational Feasibility

The operational feasibility of a web-based todo application refers to the practicality of running and maintaining the application over time. This includes evaluating the resources required to operate the application, such as hardware, software, and personnel, as well as the ongoing costs associated with maintaining and updating the application.The web-based todo application is a viable and operationally feasible solution for managing tasks and organizing personal productivity. The application's ease of use, accessibility, and low maintenance requirements make it a suitable choice for individuals, teams, and small businesses.

2.4.3 Economic Feasibility

The economic feasibility of a web-based todo application depends on several factors, including the target market, the features of the application, and the pricing strategy. However, there are several reasons to believe that such an application could be profitable. First, there is a large and growing demand for productivity tools, as people increasingly seek ways to manage their time and tasks more effectively. Second, there are a number of potential revenue streams for a web-based todo application. The application could be offered as a freemium service, with a basic version that is free to use and a premium version that offers additional features. The application could also be monetized through advertising or by selling premium features to businesses. Finally, the development costs for a web-based todo application are relatively low. The application could be developed using open-source software and could be hosted on a cloud-based platform. This would make the application more affordable to develop and maintain than a traditional software application. Overall, the economic feasibility of a web-based todo application is promising. With a large and growing market, several potential revenue streams, and relatively low development costs, there is a good chance that such an application could be profitable.

2.5 SOFTWARE ENGINEERING PARADIGM APPLIED

The agile model is used for the entire development of the Web-Based Todo List Application. Agile methodology represents a flexible and iterative approach to software development, where tasks are tackled in short, incremental cycles known as sprints. This model emphasizes collaboration, adaptability, and the delivery of a minimum viable product at the end of each iteration. Agile is particularly suited for projects where requirements may evolve, and there is a need for frequent feedback and adjustments. In the case of the Web-Based Todo List Application, the agile model allows for continuous refinement based on user feedback and changing priorities. Reasons to opt for the agile model for the Web-Based Todo List Application include its ability to accommodate evolving requirements and allowing for adjustments based on user experiences and changing task management needs.

The agile model is well-suited for the Web-Based Todo List Application due to its iterative nature. Each sprint focuses on delivering a set of prioritized features, ensuring that the application evolves incrementally based on user needs and feedback. This approach is beneficial for a dynamic project like a todo list application where user requirements might evolve during the development process. Collaboration is crucial in agile development, fostering communication among team members, stakeholders, and end-users. This is particularly advantageous for a Web-Based Todo List Application, where user interface and experience play a pivotal role in the application's success. The agile model's adaptability allows the development team to respond promptly to changing requirements or emerging trends in task management, ensuring that the Web-Based Todo List Application remains relevant and user-friendly.

3 SYSTEM DESIGN

3.1 INTRODUCTION

In the design of a web-based to-do application using PHP and MySQL, a comprehensive system architecture is crucial for its effective functionality and scalability. At the heart of the design lies the client-server model, where the client, typically a web browser, interacts with the server housing the PHP scripts and MySQL database. The client-side, responsible for the user interface, employs HTML, CSS to create a seamless and responsive task management experience. PHP, as the server-side scripting language, processes user requests, manages session data, and interacts with the MySQL database for task storage and retrieval.

The MySQL database plays a pivotal role in storing and organizing task-related information. The 'tasks' table, for instance, holds records with fields such as 'id' for unique identification, 'task' for the task description, and 'completed' to track the status of completion. Proper indexing and normalization enhance database efficiency and ensure faster query processing. The PHP scripts responsible for interacting with the database implement secure practices, such as parameterized queries, to mitigate the risk of SQL injection. Additionally, the system can be optimized by caching frequently accessed data and employing connection pooling for efficient handling of database connections. The system design not only focuses on meeting current requirements but also anticipates future scalability, ensuring the web-based to-do application remains robust and adaptable to evolving user needs.

In the realm of web application development, the design of a system plays a pivotal role in determining its functionality, efficiency, and user experience. The web-based todo application, developed using PHP and MySQL, aims to provide users with a seamless and intuitive platform for managing tasks and increasing productivity. This system design encompasses various components, including the front-end interface, back-end logic, and database management.

3.2 DATABASE DESIGN

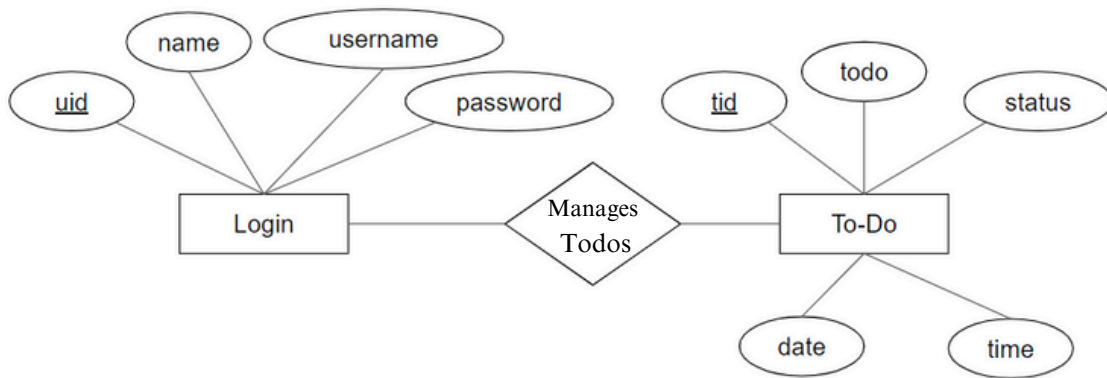
Login Table

Field Name	Field Type	Constraints	Description
uid	int	primary key	User login id
name	varchar	NOT NULL	Name of user
user name	varchar	NOT NULL	User name for login
password	varchar	NOT NULL	Password for login

Todo Table

Field Name	Field Type	Constraints	Description
tid	int	primary key	todo item id
uid	int	foreign key	User login id
todo	varchar	NOT NULL	Todo task item
status	tinyint	NOT NULL	Status for todo complete
date	date	NOT NULL	Due Date
time	time	NOT NULL	Due Time

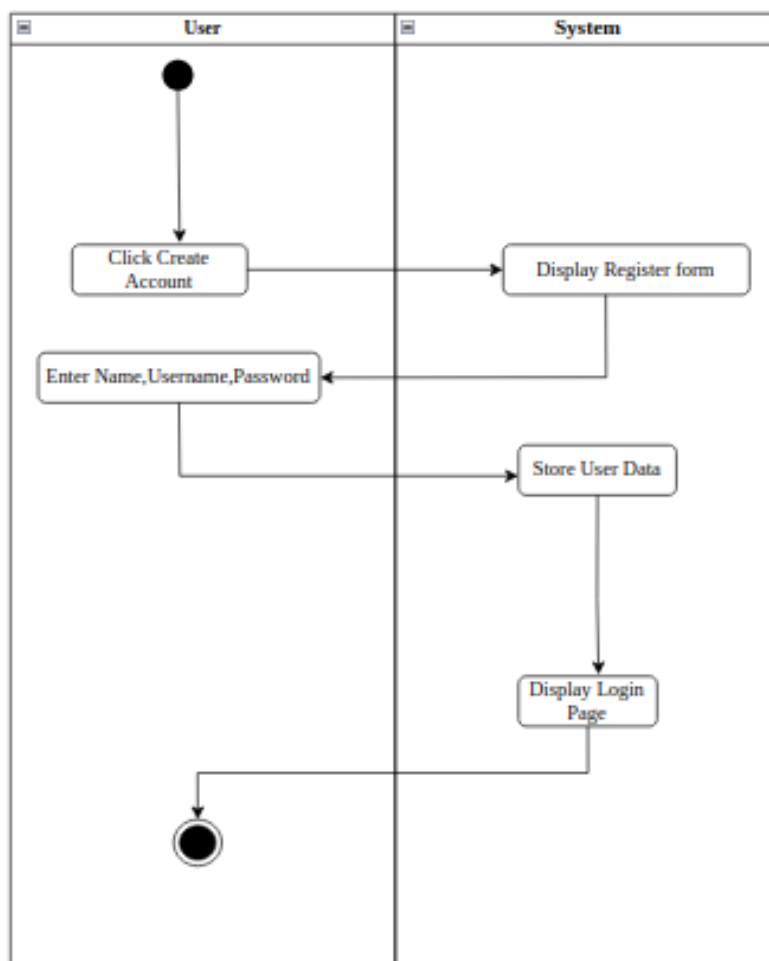
3.2.1 Entity Relationship Model



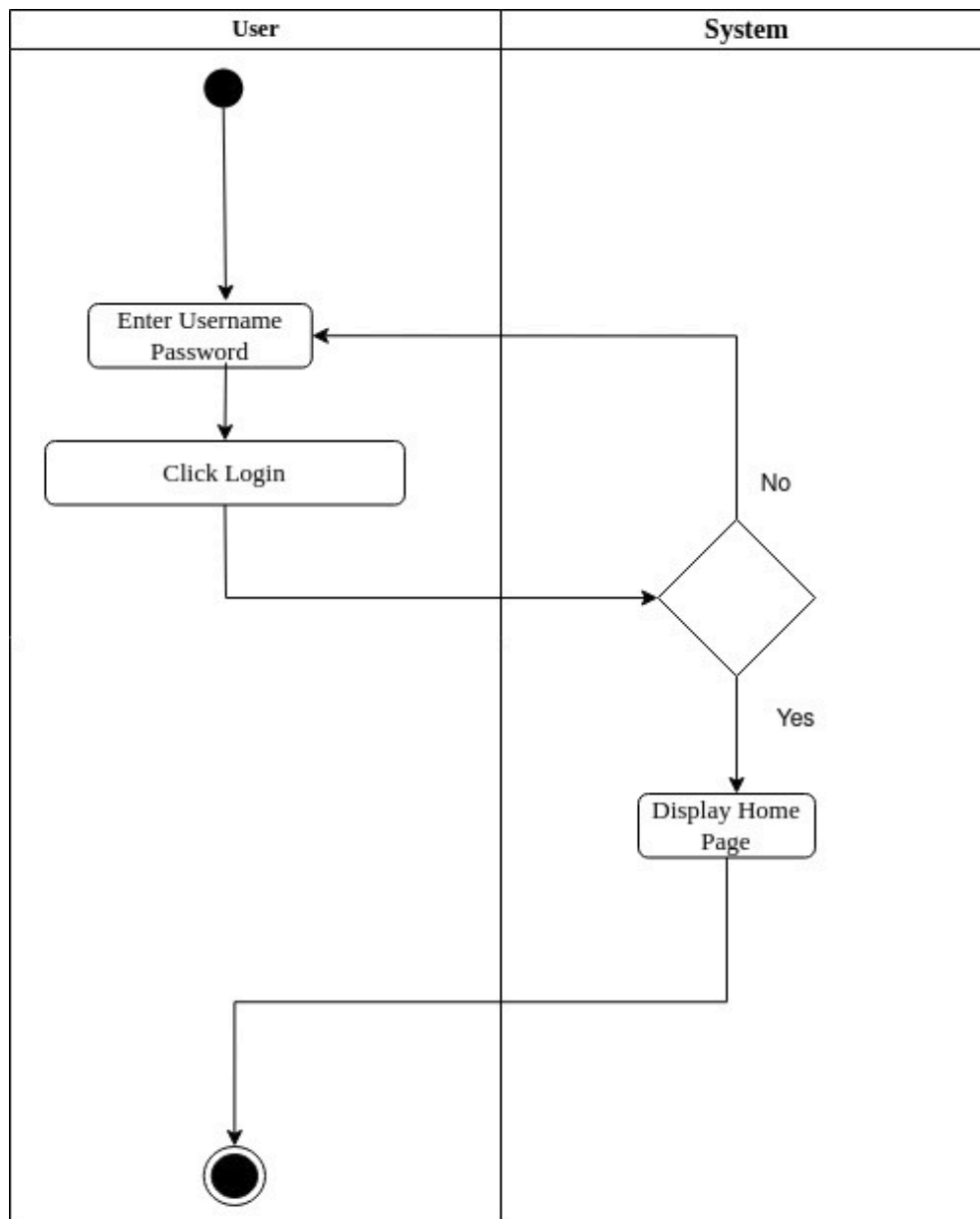
3.3 OBJECT ORIENTED DESIGN – UML DIAGRAMS

3.3.1 Activity Diagram

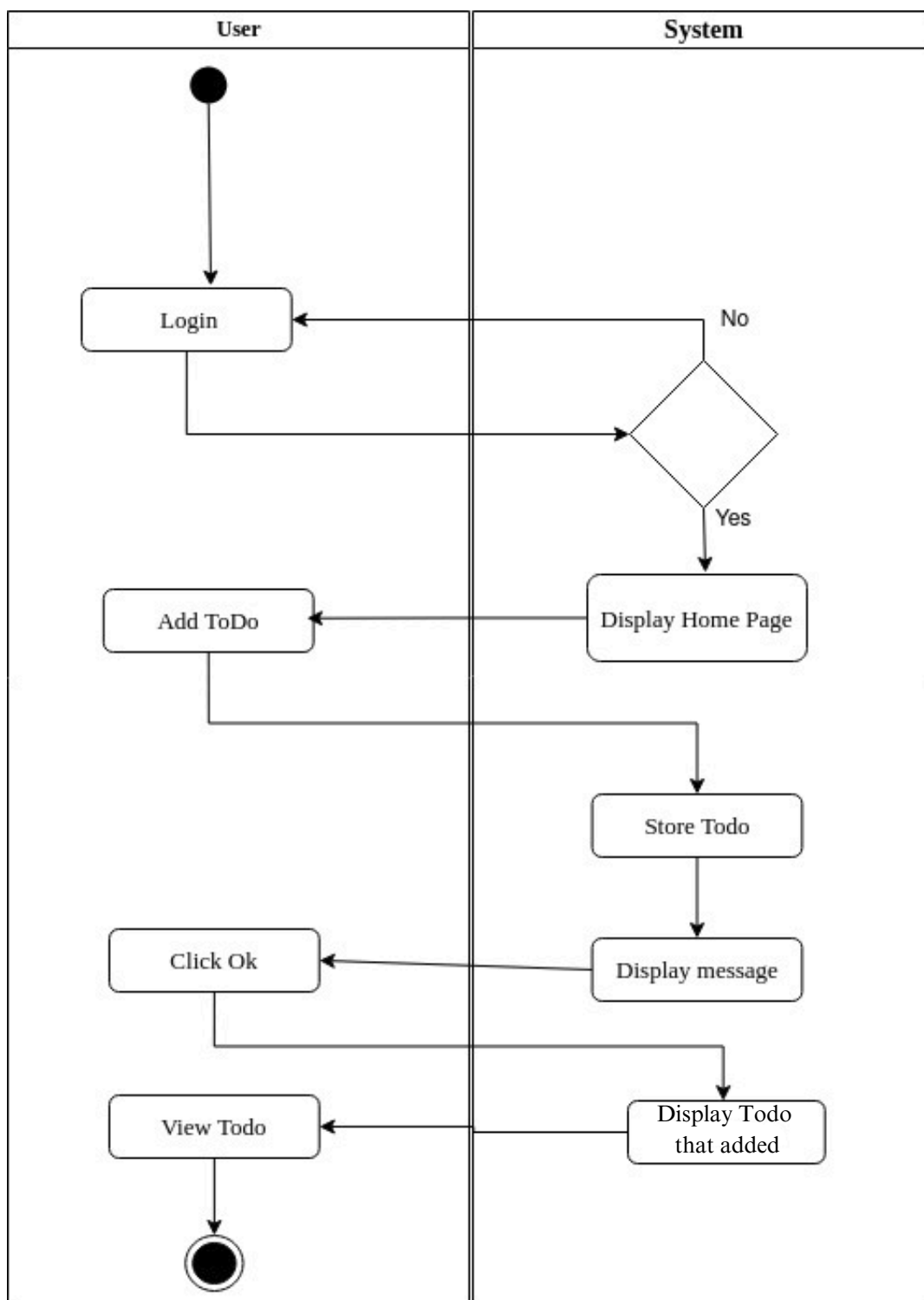
. REGISTRATION



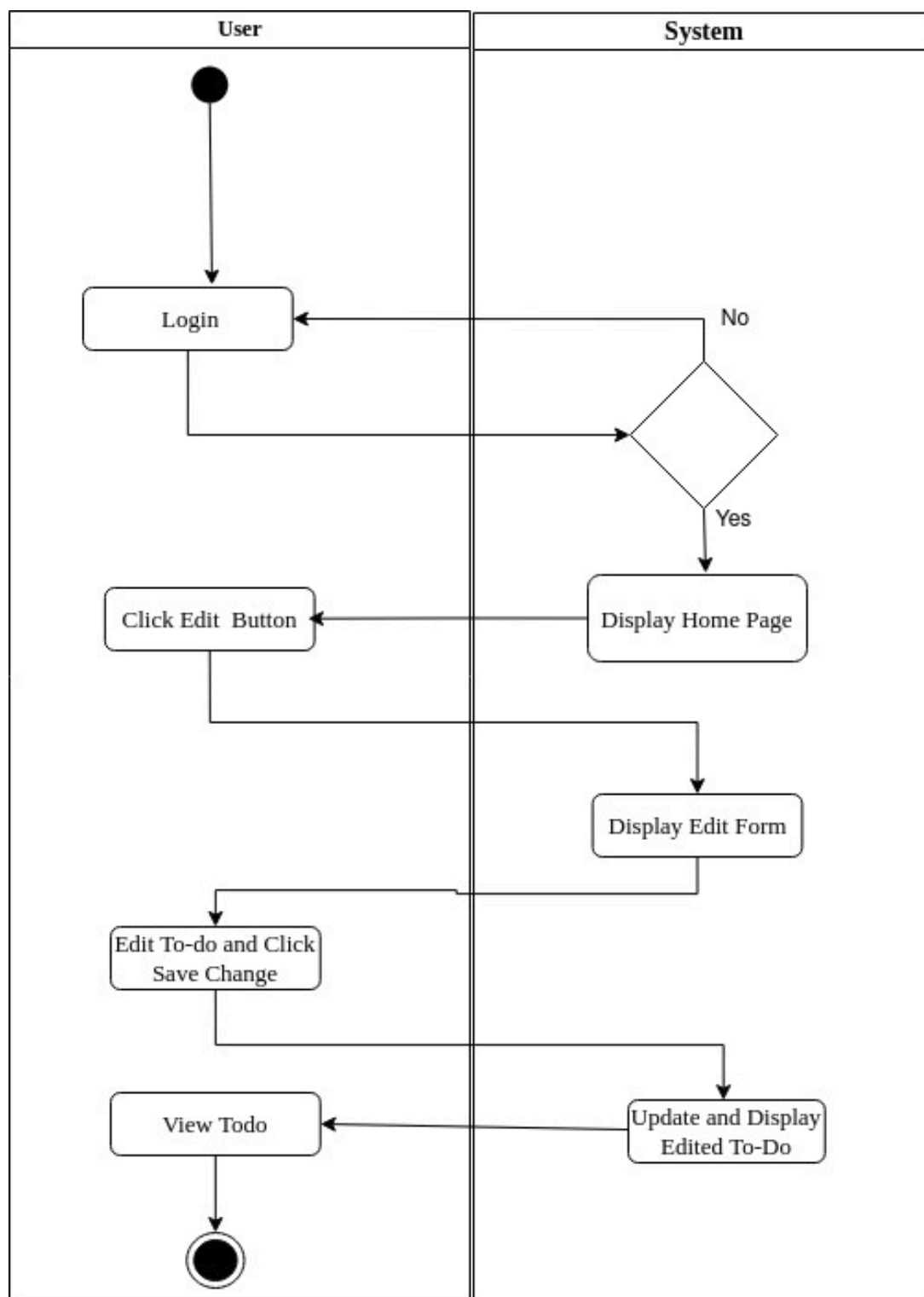
• LOGIN



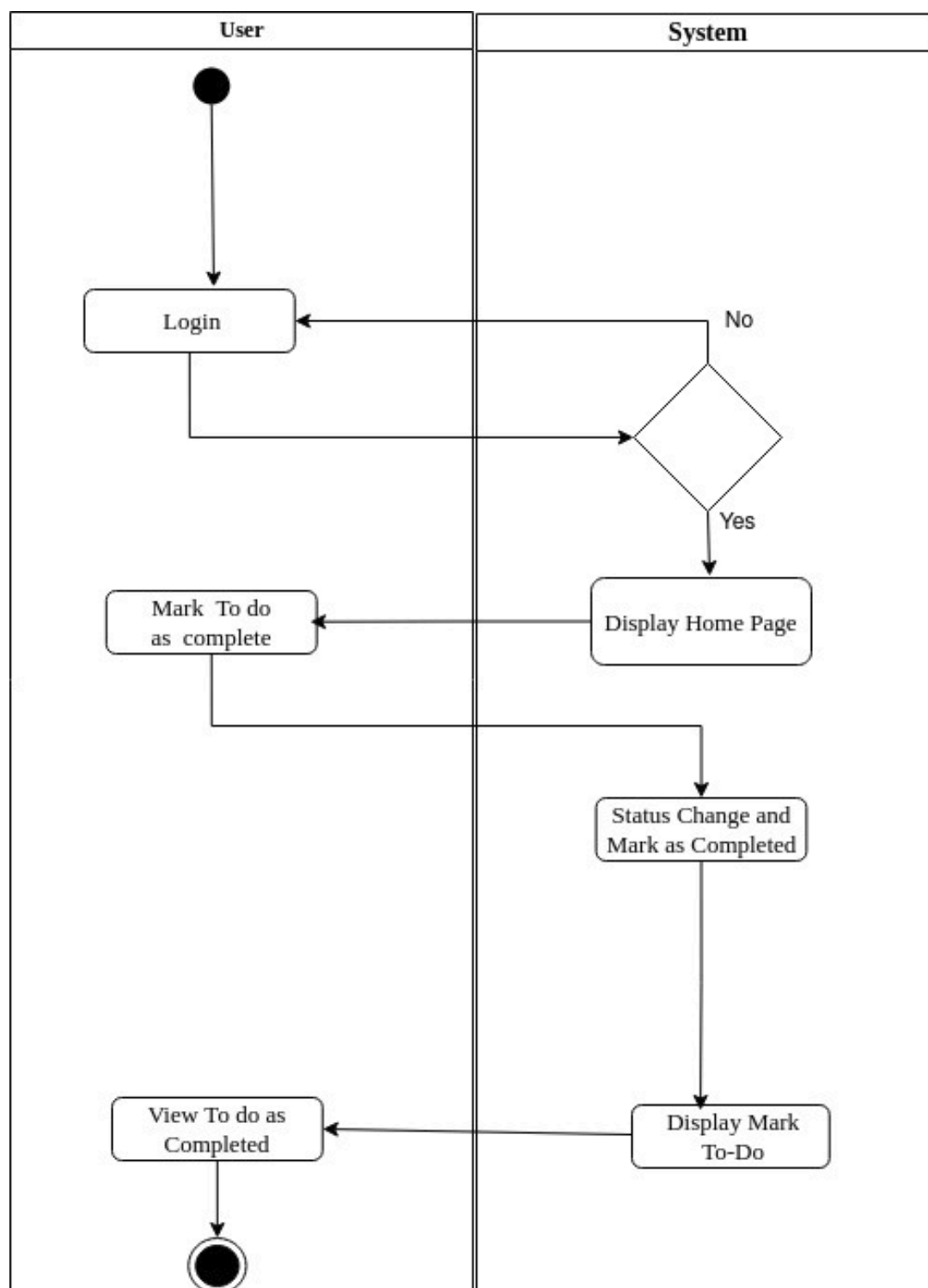
• ADD TO-DO ITEM



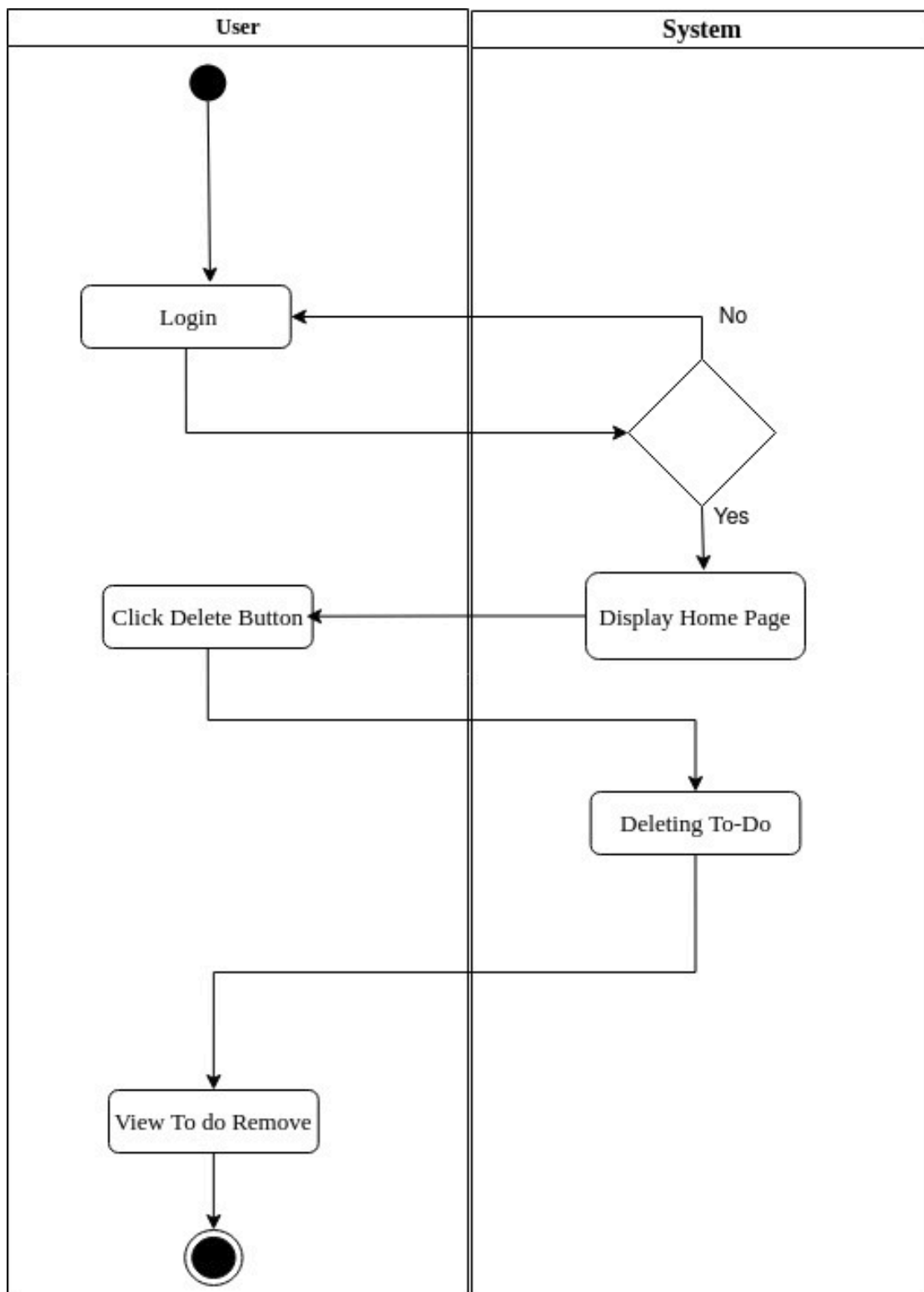
• EDIT TO-DO ITEM



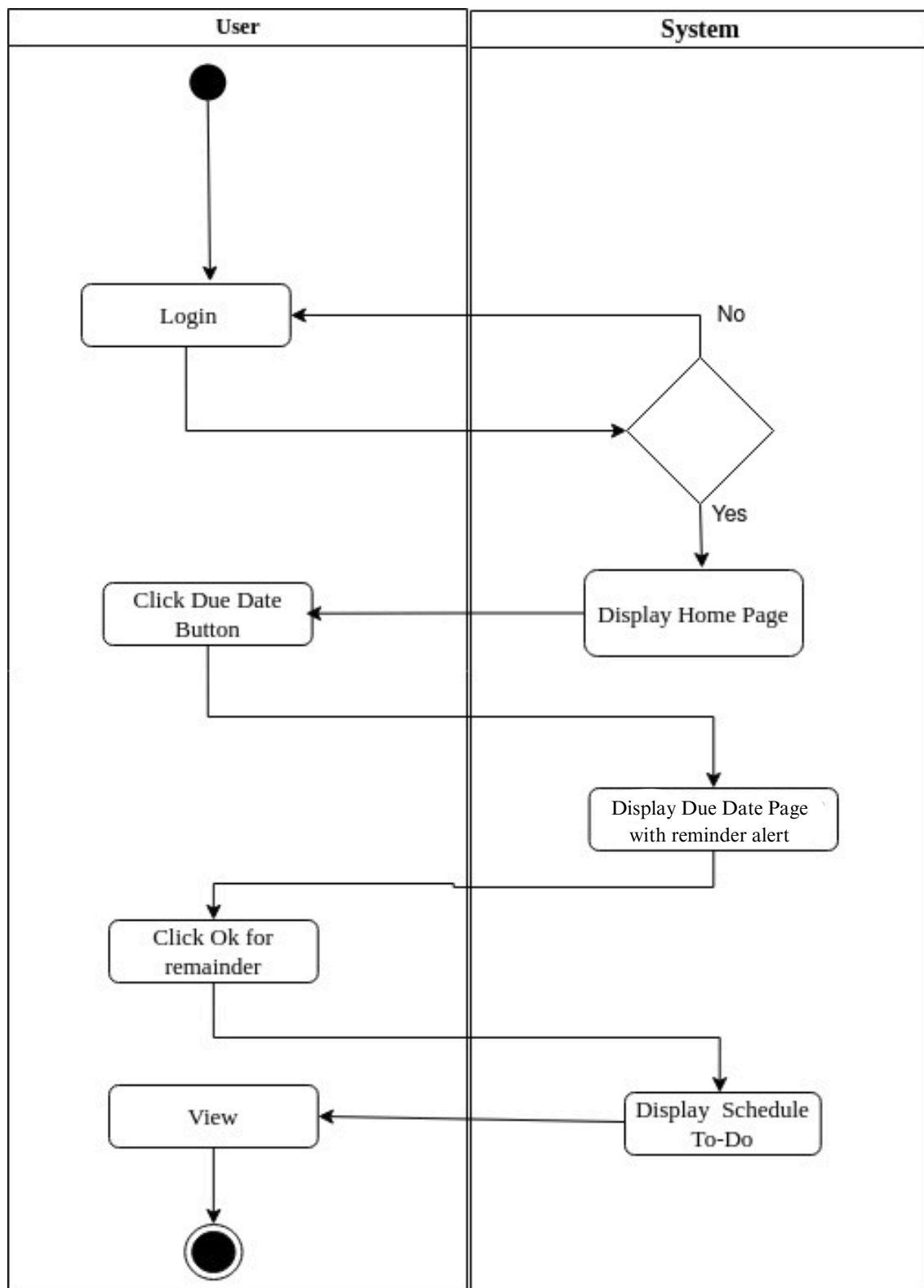
• MARK TO-DO ITEM



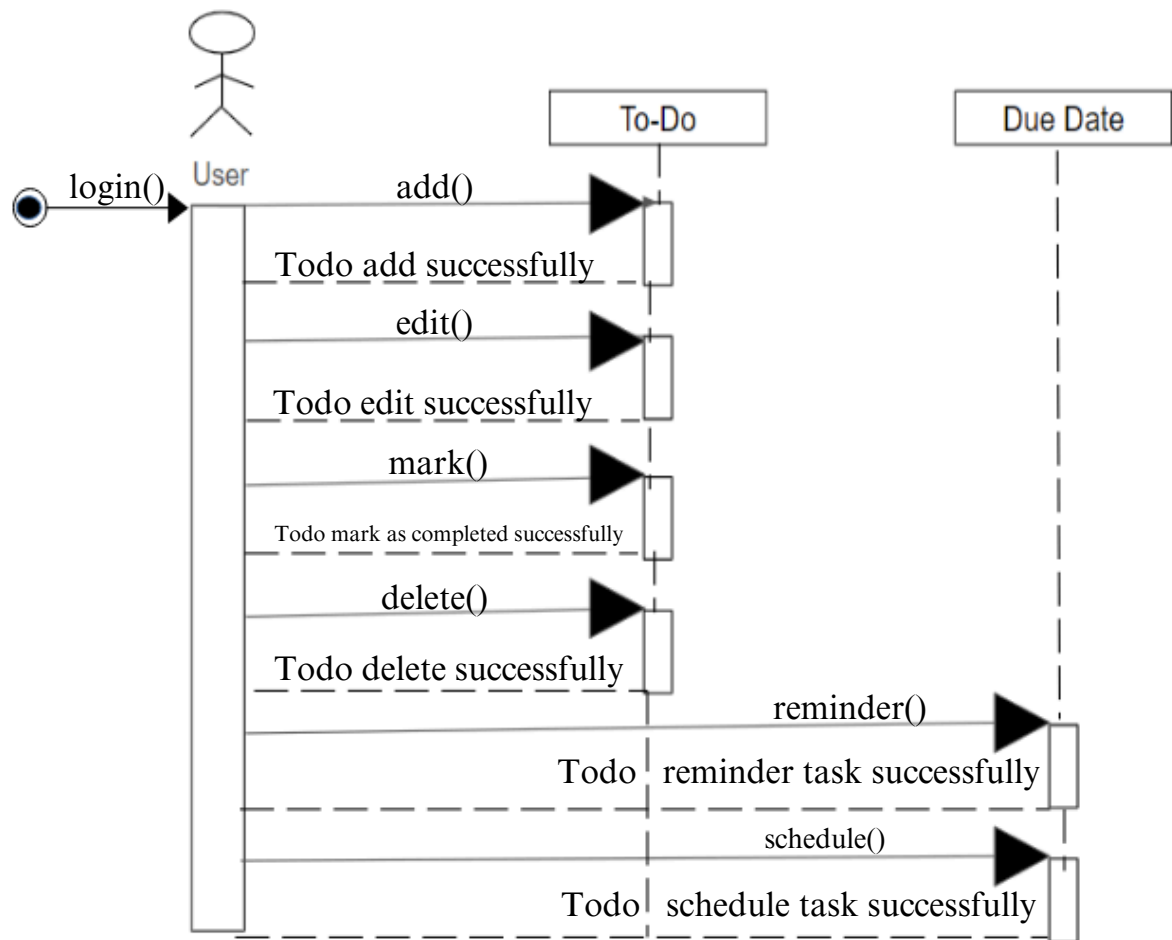
• DELETE TO-DO ITEM



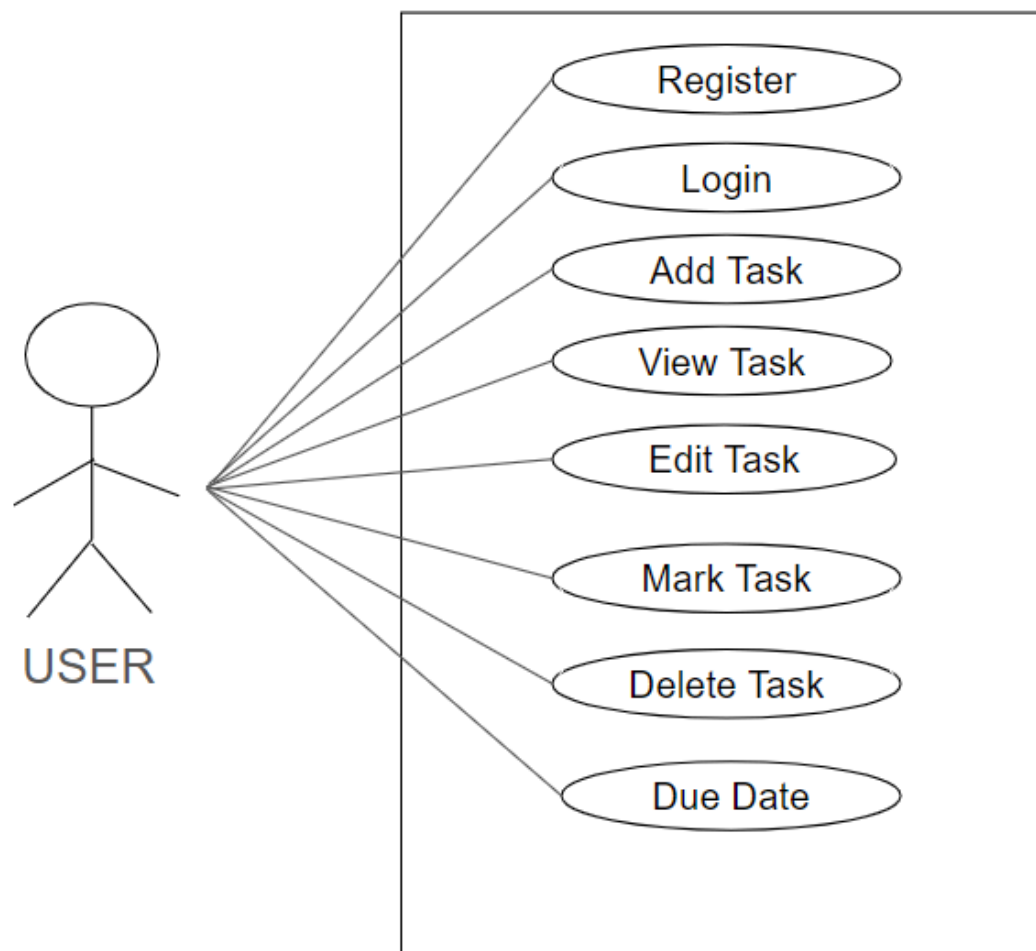
- **DUE DATE**



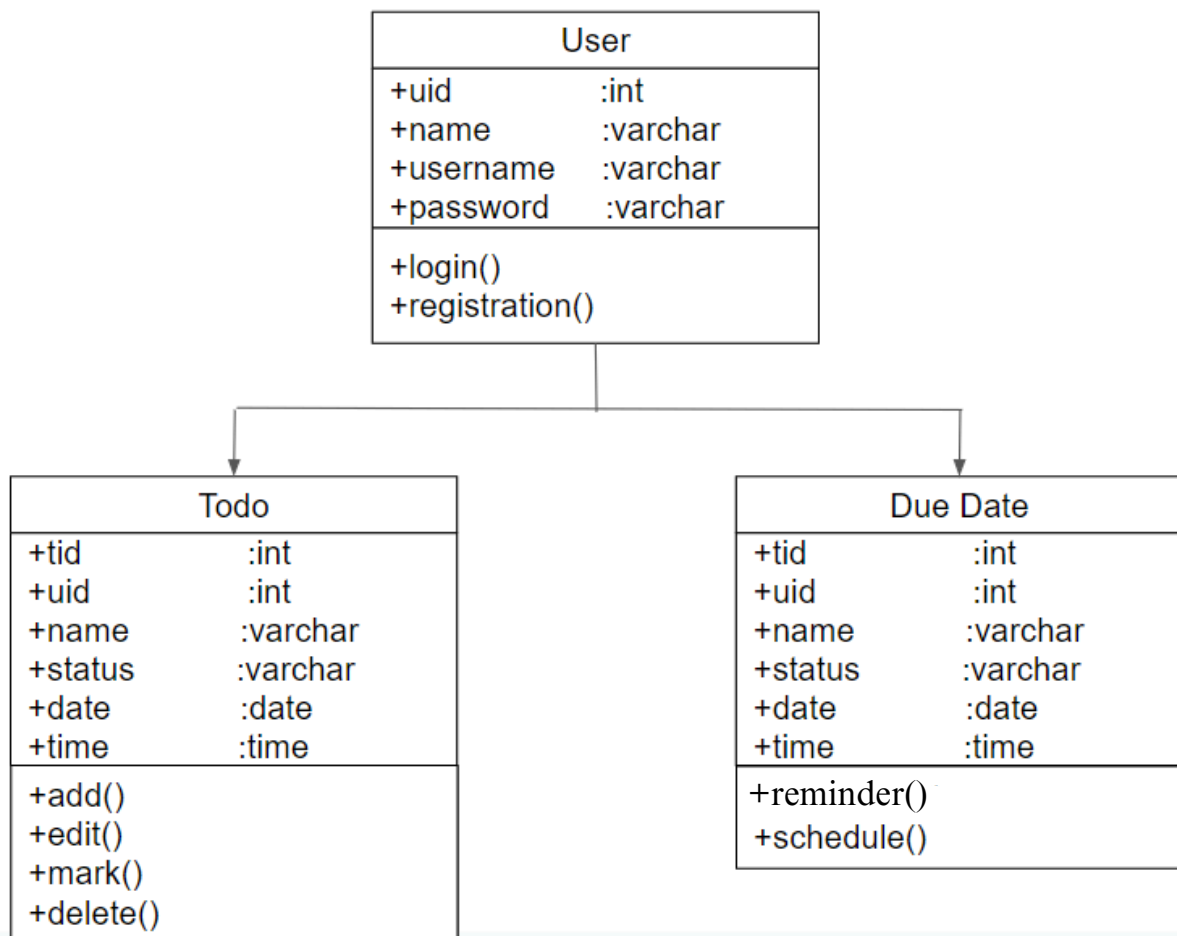
3.3.2 Sequence Diagram



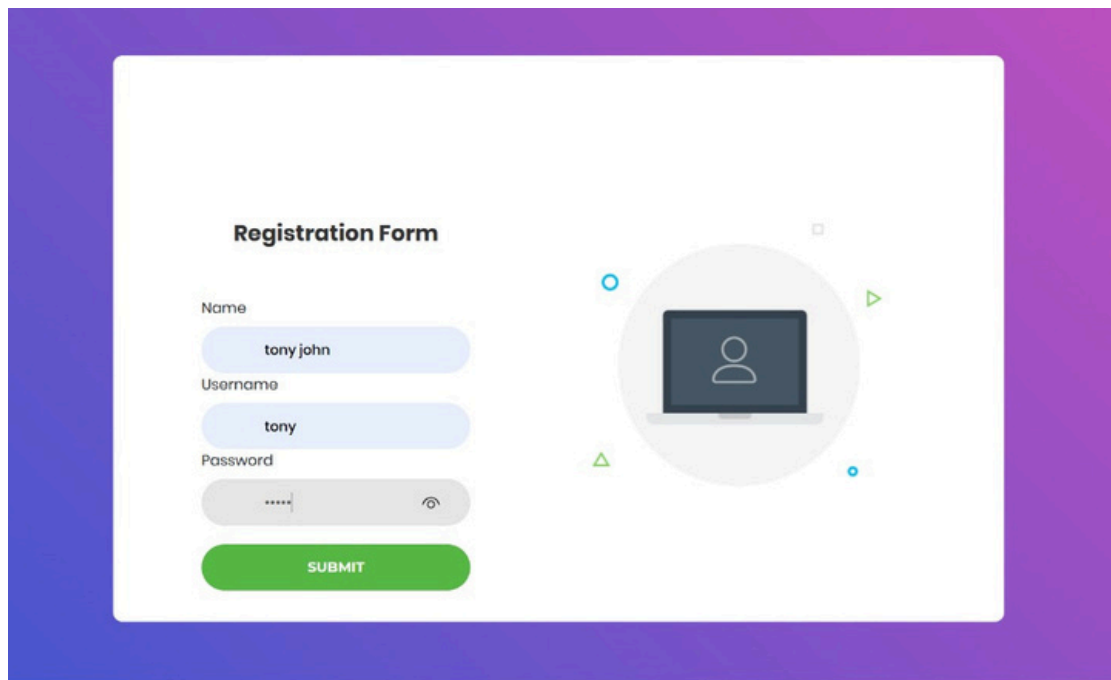
3.3.3 Use Case Diagram



3.3.4 Class Diagram



3.4 INPUT DESIGN



The registration form is titled "Registration Form" and is set against a white background with a purple gradient border. It features three input fields: "Name" with the text "tony john", "Username" with the text "tony", and "Password" with masked characters "*****" and a toggle icon. A green "SUBMIT" button is at the bottom. To the right is a circular graphic with a laptop icon and a person silhouette, surrounded by small blue and green geometric shapes.

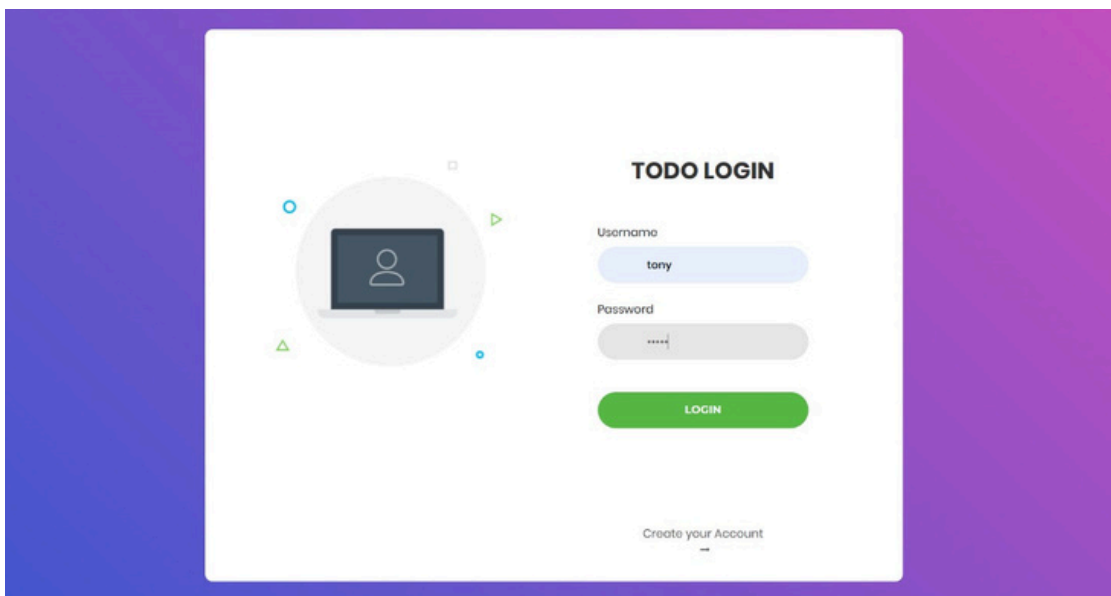
Registration Form

Name
tony john

Username
tony

Password

SUBMIT



The login form is titled "TODO LOGIN" and is set against a white background with a purple gradient border. It features two input fields: "Username" with the text "tony" and "Password" with masked characters "*****". A green "LOGIN" button is at the bottom. Below the button is a link "Create your Account" with a right-pointing arrow. To the left is a circular graphic with a laptop icon and a person silhouette, surrounded by small blue and green geometric shapes.

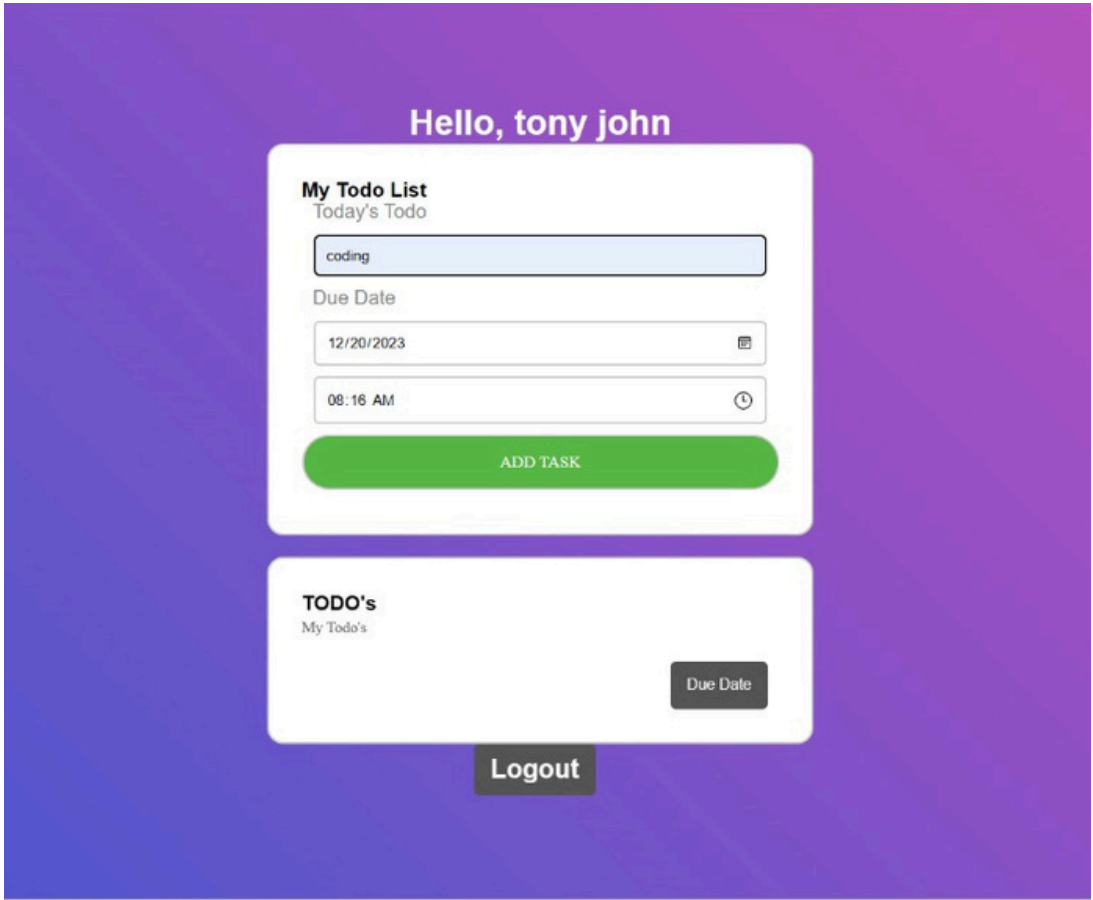
TODO LOGIN

Username
tony

Password

LOGIN

Create your Account →



3.5 OUTPUT DESIGN

Hello, tony john

My Todo List

Today's Todo

Due Date

12/20/2023

08:20 AM

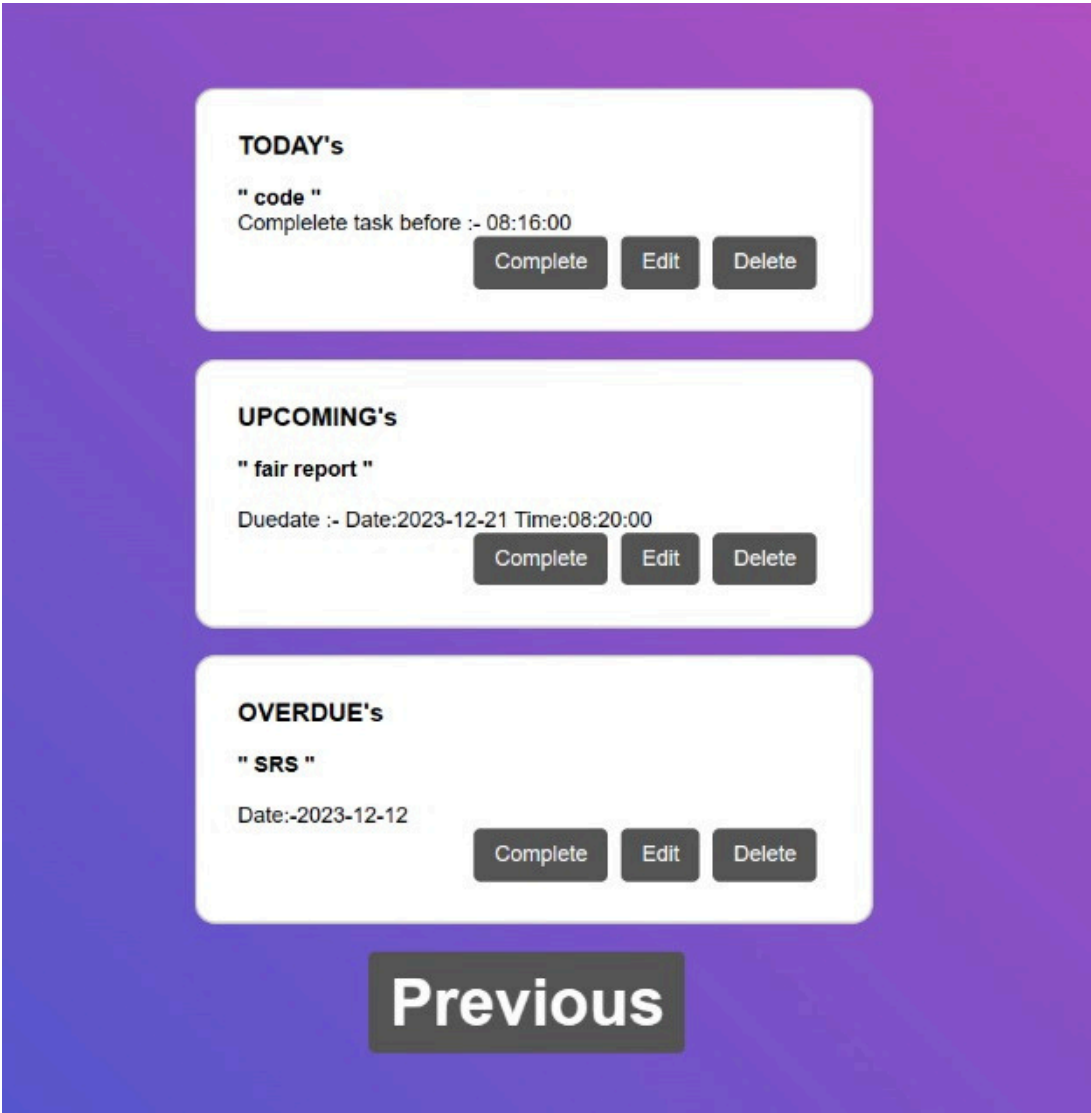
ADD TASK

Today's todos created.

TODO's

My Todo's

SRS	Complete	Edit	Delete
code	Complete	Edit	Delete
fair report	Complete	Edit	Delete
			Due Date



4 SYSTEM ENVIRONMENT

4.1 INTRODUCTION

A web-based todo application requires a variety of software and hardware components to function properly. These components include a web server, a programming language, a database, and an operating system. The specific requirements of each component will vary depending on the specific application, but in general, a web-based todo application will require a web server such as Apache or Nginx, a programming language such as PHP, a database such as MySQL, and an operating system such as Linux, Windows, or macOS. In addition to these core components, a web-based todo application may also require additional tools and platforms, such as an integrated development environment (IDE) and a version control system. By ensuring that all of the required components are properly installed and configured, developers can ensure that their web-based todo application will run smoothly and efficiently.

4.2 Software Requirement Specification

IDE : Visual studio code

Front end technologies : HTML, CSS

Back end technology : PHP, MySQL

4.3 Hardware Requirement Specification

Hardware requirements needs for the system

Processor : Pentium 1.80 GHz or more

RAM : 4GB Hard disk : 90 GB

Monitor : SVGA Color

Keyboard : 104 Keys

Mouse : Optical

4.4 Tools, Platforms

4.4.1 Frontend Tool :

Front end : HTML, CSS

- HTML: Hyper Text Markup Language is the standard markup language for creating web pages. It is used to define the structure and content of a web page.
- CSS: Cascading Style Sheets are used to style the appearance of a web page. CSS can be used to control the layout, colors, fonts, and other aspects of a web page's appearance.

In a frontend tool refers to a software application or library that is used to create the user interface (UI) of a web application. Frontend tools typically include HTML, CSS. These tools are used to create the layout, styling, and interactivity of a web page, and they are essential for creating a user-friendly and visually appealing experience.

For a web-based todo application, frontend tools would be used to create the interface for adding, editing, and deleting tasks, as well as for managing other aspects of the application, such as user accounts and settings. The specific frontend tools that are used will depend on the specific requirements of the application, but in general, any tool that can be used to create HTML, CSS, and JavaScript code can be used to develop the frontend of a web-based todo application.

4.4.2 Backend Tool :

Backend: PHP, MYSQL

- **PHP:** PHP is a scripting language that is commonly used for web development. It is a versatile language that can be used to develop a wide variety of web applications.
- **MYSQL:** MySQL is a relational database management system (RDBMS) that is commonly used to store the data for web-based applications. It is a popular choice for web applications because it is easy to use, scalable, and reliable.

In A web development, a backend tool refers to a software application or library that is used to develop the server-side logic of a web application. Backend tools typically include programming languages such as PHP. These tools are used to handle user requests, process data, and communicate with MYSQL databases.

4.4.3 Operating System

Since it is a web application it does not require any specific Operating System. It supports all OS that has a browser

5. SYSTEM IMPLEMENTATION

5.1 INTRODUCTION

Implementing a web-based to-do application involves several steps, including front-end and back-end development, database integration, and deployment.

Define requirements such as define the features and functionality you want in your to-do application. Consider features such as user authentication, task creation, task modification, task deletion, and task categorization. Select the technologies you'll use for front-end, back-end, and database development. Organize your project into a clear structure, separating front-end and back-end components. Then Coding to create the user interface using the chosen front-end framework. Implement components for task listing, task creation, task modification, and authentication if necessary. Build the server-side logic to handle HTTP requests and interact with the database. Implement routes for CRUD operations (Create, Read, Update, Delete). Set up user authentication if needed. Connect todo application to the chosen database. Design tables to store user information, tasks, and any other necessary data. Implement database queries for CRUD operations and remiander. Test the application and deployment the model.

5.2 CODING

5.2.1 SAMPLE CODES

Home.php

```
<?php
session_start();
if (isset($_SESSION['uid']) && isset($_SESSION['user_name']))
{
    $ui=$_SESSION['uid'];
    $dbc = mysqli_connect('localhost', 'root', '', 'to');
    if(isset($_POST['submit']))
    {
        $todo = $_POST['todo'];
        $date = $_POST['dueDate'];
        $time = $_POST['duetime'];
        //insert data
        $sql = "insert into todos (todo,uid,date,time) values ('$todo','$ui','$date','$time')";
        mysqli_query($dbc, $sql);
        $msg = "Today's todos created.";
    }

?>
```



```

<!DOCTYPE html>
<html>
<head>
  <title>HOME</title>
  <link rel="stylesheet" type="text/css" href="s.css">

</head>
<body>
<h1>Hello, <?php echo $_SESSION['name']; ?></h1>

  <form action="home.php" method="post">
    <h3>My Todo List</h3>

    <label>Today's Todo</label>
    <center>
      <input type="text" name="todo" >
    </center>
    <label>Due Date</label>
    <center>
      <input type="date" name="dueDate" value="<?php echo date('Y-m-d'); ?>">
      <?php $currentTime = new DateTime(date('H:i'));
      $timeInterval = new DateInterval('PT5H30M'); // 5 hours and 30 minutes
      $targetTime = $currentTime->add($timeInterval); ?>
      <input type='time' name='duetime' value="<?php echo $targetTime->format('H:i');?
      >">
    </center>
    <input type="submit" name="submit" value="Done">
    <?php if (isset($_POST['submit'])) {?>
      <p><?=$msg;?></p>
    <?php } ?>
  </form>

  <form>

    <h3>TODO's</h3>
    <p>My Todo's</p>
    <p>
    <ol>
    <?php
    //fetch data from table;
    $sql = "SELECT * from todos where `uid`='Sui' order by date,time";
    $result = mysqli_query($dbc, $sql);

```

```

while ($row=mysqli_fetch_array($result)) {
    $todo = $row['todo'];
    $todo_id = $row['tid'];
    $status = $row['status'];
    ?>
    <br>
    <li>
    <?=$todo?>
    <?php if ($status==0) { ?>
    &nbsp;
    <a href="del.php?tid=<?=$todo_id?>">Delete
    <i aria-hidden="true"></i>
    </a>
    <a href="edit.php?tid=<?=$todo_id?>">Edit
    <i aria-hidden="true"></i>
    </a> &nbsp;
    <a href="comp.php?tid=<?=$todo_id?>">Complete
    <i aria-hidden="true"></i>
    </a> <?php } else{ ?> &nbsp;
    <a href="del.php?tid=<?=$todo_id?>">Delete
    <i aria-hidden="true"></i>
    </a> &nbsp;
    <a href="comp.php?tid=<?=$todo_id?>">Completed
    <i aria-hidden="true"></i>
    </a> &nbsp;
    <?php } ?>
    </li><br>
    <?php } ?>
</ol></p><pre>
<p> <a href="duedate.php">Due Date</a></p></pre>
</form>
<h1> <a href="logout.php">Logout</a></h1>
</body>
</html>

<?php
} else{
    header("Location: index.php");
    exit();
?>

```

duedate.php

```

<form>
  <h3 >TODAY's</h3> <p><ol >
  <?php
    //fetch data from table;
    $date=date('Y-m-d');
    $sql = "SELECT * from todos where `uid`='$ui' and `date`='$date' and `status`=0 ";
    $result = mysqli_query($dbc, $sql);
    while ($row=mysqli_fetch_array($result))
    {
      $todo = $row['todo'];
      $todo_id = $row['tid'];
      $status = $row['status'];
      $date= $row['date'];
      $time=$row['time'];
      $current_time = time();
      // Check if it's time for the reminder
      if ($current_time >= $time) {
        // Time to show the reminder
        echo "<script>alert('Reminder: Time to complete the task - $todo');</script>";
      }
    }
  ?>  <br><li >
  <?php
    echo '<h4>' . $todo;
    echo $todo;
    echo ' "</h4>';
    echo " Complete task before :- ";
    echo $time;
  ?>
  <?php if ($status==0) { ?>
    <a href="del.php?tid=?=$todo_id?">Delete
    <i aria-hidden="true"></i> </a>
    <a href="edit.php?tid=?=$todo_id?">Edit
    <i aria-hidden="true"></i> </a> &nbsp;
    <a href="comp.php?tid=?=$todo_id?">Complete
    <i aria-hidden="true"></i> </a> <?php } else{ ?> &nbsp;
    <a href="del.php?tid=?=$todo_id?">Delete
    <i aria-hidden="true"></i></a> &nbsp;
    <a href="comp.php?tid=?=$todo_id?">Completed
    <i aria-hidden="true" ></i> </a> &nbsp; <?php } ?>
  </li><br> <?php } ?>
</ol>
</form>

```

```

<form>
  <h3 >UPCOMING's</h3>
  <p>
  <ol >
    <?php
      //fetch data from table;
      $date=date('Y-m-d');
      $sql = "SELECT * from todos where `uid`='Sui' and `date`>'$date' and `status`=0 ";
      $result = mysqli_query($dbc, $sql);
      while ($row=mysqli_fetch_array($result)) {
    $todo = $row['todo'];
      $todo_id = $row['tid'];
      $status = $row['status'];
      $date= $row['date'];
      $time=$row['time']; ?> <br> <li >
    <?php
      echo '<h4>" ';
      echo $todo;
      echo ' "</h4>';
      echo " <br> ";
      echo "Due date :- Date:$date Time:$time ";
      ?><br>
      <?php if ($status==0) { ?>
        <a href="del.php?tid=?=$todo_id?>">Delete
        <i aria-hidden="true"></i>
        </a>
        <a href="edit.php?tid=?=$todo_id?>" >Edit
        <i aria-hidden="true"></i>
        </a> &nbsp;
        <a href="comp.php?tid=?=$todo_id?>" >Complete
        <i aria-hidden="true"></i>
        </a> <?php } else{ ?> &nbsp;
        <a href="del.php?tid=?=$todo_id?>">Delete
        <i aria-hidden="true"></i>
        </a> &nbsp;
        <a href="comp.php?tid=?=$todo_id?>" >Completed
        <i aria-hidden="true" ></i>
        </a> &nbsp;
      <?php } ?>
    </li><br>
    <?php } ?>
  </ol>
</form>

```

```

<form><h3 >OVERDUE's</h3>
  <p><ol >
    <?php
      //fetch data from table;
      $date=date('Y-m-d');
      $sql = "SELECT * from todos where `uid`='$ui' and `date`<$date' and `status`=0 ";
      $result = mysqli_query($dbc, $sql);
      while ($row=mysqli_fetch_array($result)) {
        $todo = $row['todo'];
        $todo_id = $row['tid'];
        $status = $row['status'];
        $date= $row['date'];
        ?>    <li >
          <?php
            echo '<h4>" ' ;
            echo $todo;
            echo ' "</h4>';
            echo " <br> ";
            echo " Date:-$date "; ?> <br>
            <?php if ($status==0) { ?>
              &nbsp;
              <a href="del.php?tid=<?=$todo_id?>">Delete<i aria-hidden="true"></i>
              </a><a href="edit.php?tid=<?=$todo_id?>" >Edit
                <i aria-hidden="true"></i>          </a> &nbsp;
              <a href="comp.php?tid=<?=$todo_id?>" >Complete
                <i aria-hidden="true"></i>          </a> <?php } else{ ?> &nbsp;
              <a href="del.php?tid=<?=$todo_id?>">Delete
                <i aria-hidden="true"></i></a> &nbsp;
              <a href="comp.php?tid=<?=$todo_id?>" >Completed
                <i aria-hidden="true" ></i> </a> &nbsp;
            <?php } ?> </li><br>
          <?php } ?>
        </ol>
      </form>
    <h1> <a href="home.php">Previous</a></h1>
  </body>
</html>
<?php
}else{
  header("Location: index.php");
  exit();
}
?>

```

5.2.2 Code Validation and Optimization

Code Validation

Syntax Checking: The first step in code validation is to ensure that the syntax of the code is correct. This can be done using a syntax checker or a code editor that has built-in syntax checking capabilities. The syntax checker will highlight any errors or warnings in the code, which can then be corrected.

Input validation: Ensure all user inputs are validated before processing. Use libraries or custom functions to check for valid data types, lengths, and formats .

Database validation: Sanitize all user inputs before inserting them into the database to prevent SQL injection attacks. Use prepared statements and escape special characters.

Data integrity: Validate data retrieved from the database to ensure it's consistent and accurate. Check for null values, empty strings, and unexpected formats.

Form validation: Implement server-side form validation before relying on client-side validation . This ensures invalid forms don't reach the server and trigger errors.

Optimization

Database queries: Optimize your SQL queries to minimize database calls and data transfer. Use indexes for frequently used fields, avoid unnecessary joins, and write efficient WHERE clauses.

Caching: Implement caching mechanisms (e.g., Redis, Memcached) to store frequently accessed data in memory, reducing database load and improving response times.

Sessions: Avoid unnecessary session regenerations and use lightweight session storage methods (e.g., database sessions with optimized tables) to improve app performance.

Error handling: Implement proper error handling to avoid cascading errors and unexpected behavior. Log errors and display user-friendly messages instead of technical details.

Code structure: Write clean, well-organized code with appropriate functions, classes, and comments. This makes the code easier to maintain, debug, and optimize.

Libraries and frameworks: Consider using established PHP libraries and frameworks for common tasks like authentication, authorization, and routing. This can save time and improve code quality.

Testing: Implement unit and integration tests to ensure your code functions as expected. This helps identify and fix bugs early in the development process.

6. SYSTEM TESTING

6.1 Introduction

System testing is the process of testing the entire web-based todo application as a whole. It ensures that all components work together as expected, and the application meets the specified requirements. System testing is a level of software testing where the complete and integrated software is tested as a whole to ensure that it meets the specified requirements. The primary objective is to validate the system's functionality, performance, reliability, and other non-functional aspects.

6.2 Unit Testing

To implement unit testing for a web-based to-do application built with PHP and MySQL, you can use a testing framework like PHP Unit. Start by creating test cases that cover different aspects of your application, such as task creation, completion status updates, and database interactions. For example, write a test to ensure that adding a task results in a valid database entry and another test to verify the correct toggling of task completion status. Mocking the database connections or using an in-memory database during tests can isolate the tests and improve their reliability. Integrate these tests into your development workflow, running them automatically on code changes to catch potential issues early in the development process. Unit testing helps ensure that individual components of your application function as expected, making it easier to maintain and enhance the application over time.

6.3 Integration Testing

Integration testing involves testing the interaction between different components. Ensure that adding a task, updating task status, and displaying tasks work seamlessly together. Integration testing for a web-based to-do application with PHP and MySQL involves testing the interaction and collaboration between different components of the system, such as the database, server-side scripts, and client-side functionalities. The testing process should verify that these components work seamlessly together to achieve the application's intended behavior. In the context of a to-do application, integration testing may include ensuring that tasks added through the user interface are correctly stored in the database, updating the completion status of tasks reflects accurately, and that any changes made in the database are correctly reflected in the user interface. Test cases should cover typical user interactions and edge cases, checking the flow of data between the PHP server scripts and the MySQL database. Automated testing frameworks, such as PHPUnit for PHP, can be employed to facilitate integration testing, allowing for the creation of repeatable and comprehensive test suites that simulate real-world usage scenarios and help identify and address potential issues in the integration of various application components..

6.4 SYSTEM TESTING

For system testing of a web-based to-do application developed with PHP and MySQL, a comprehensive test plan and test cases need to be established. The test plan should outline the scope, objectives, resources, and schedule for the testing process. It should identify the different modules or components of the application to be tested, including user interfaces, database interactions, and task management functionalities. Test cases should be designed to cover various scenarios such as adding tasks, updating task statuses, and handling user interactions. They should encompass positive test cases to validate expected behavior and negative test cases to identify potential vulnerabilities or edge cases. Additionally, test cases should consider concurrent user interactions, data integrity, and error handling. Automated testing tools can be employed to enhance efficiency, especially for repetitive or regression testing. The overall goal is to ensure the robustness, reliability, and security of the web-based to-do application, addressing both functional and non-functional requirements through a systematic and thorough testing process. Regular reviews and updates to the test plan and test cases should be conducted to accommodate changes in the application's features or architecture.

6.4.1 Test plan

The system testing phase for the web-based to-do application involves creating a comprehensive test plan to ensure the application functions correctly and meets user requirements. The test plan outlines the testing approach, resources, schedule, and deliverables. It defines the testing scope, including the features and functionalities to be tested, as well as any specific testing environments or configurations required. Additionally, the plan addresses the testing methods to be employed, such as manual testing and automated testing tools, and specifies the criteria for test execution and completion.

6.4.2 Test cases

Within the test plan, detailed test cases are developed to systematically validate each aspect of the to-do application. Test cases cover various scenarios, including positive and negative test scenarios, input validation, boundary testing, and integration testing. For the PHP and MySQL-based to-do application, test cases might include verifying the correct insertion of tasks into the database, validating user input in the HTML forms, ensuring proper task display, and testing the functionality of task completion checkboxes. Additionally, error-handling scenarios, such as testing the application's response to incorrect database connections, should be included in the test cases. The test cases aim to ensure the application's reliability, performance, and user-friendliness under diverse conditions.

Test Case ID	Test Objective	Precondition	Test Steps	Test Data	Expected Result	Post-condition
TC_UI_1	Verify Login Functionality	User should have a valid account	1. Open the w login page. 2. Enter valid username and password. 3. Click the "Login" button.	Username: tony Password: 123	User is redirected to the home page.	User is logged in, and the system is in the home state.
TC_UI_2	Verify Adding a Task	User should be logged in	1. Enter To-do detail on box. 2. Click on the "Add Task" button.	To-do: Task 1 Due Date: 2023-12-12 1:00 AM	Task is added to the list,	Task is added to the database and displayed,
TC_UI_3	Verify Editing a Task	User should have at least one task	1. Click on the "Edit" correspond a task. 2. Modify task details and click "Save Changes."	To-do: Task 1 Due Date: Updated	Task details are updated,	Task details are updated in the database and displayed on the dashboard.
TC_UI_4	Verify Task as Completed	User should have at least one task	1. Click on the "Mark Complete" button next to a task.	-	Task is marked as completed	Task status is updated in the database and displayed,
TC_UI_5	Verify Deleting a Task	User should have at least one task	1. Click on the "Delete" icon next to a task. 2. Confirm the deletion.	-	Task is removed,	Task is deleted from the database, and it is no longer displayed,
TC_UI_1	Verify Due date	User should have at least one or more task	1. Click on the "Due Date" icon	-	User is see the due date of to-do with reminder	Task with reminders and schedules

7. SYSTEM MAINTENANCE

7.1 INTRODUCTION

In the realm of web-based to-do applications developed with PHP and MySQL, the concept of system maintenance plays a pivotal role in ensuring the continued reliability, security, and optimal performance of the application. Maintenance activities encompass a range of tasks that collectively contribute to the long-term viability of the system. One crucial aspect is database maintenance, involving regular checks, optimization of queries, and ensuring data integrity. This is achieved through the periodic examination and potential repair of tables, addressing issues such as index fragmentation and updating statistics to enhance query efficiency. Additionally, PHP code maintenance involves ongoing reviews and updates to ensure compatibility with the latest language versions, addressing any deprecated functions, and adhering to coding best practices. Regular backups of both the application codebase and the database are imperative, safeguarding against unforeseen issues and allowing for swift recovery in the event of data loss or system failures.

Security is a paramount consideration in system maintenance. Regular audits and updates are necessary to patch vulnerabilities, ensuring the application remains resilient against potential cyber threats. This involves staying informed about the latest security patches for PHP, MySQL, and any third-party libraries or frameworks used in the application. Continuous monitoring of user interactions and logs is essential to detect and respond promptly to any anomalous activities that may signal a security breach. Lastly, performance optimization is an ongoing concern, involving the identification and resolution of bottlenecks in both PHP code and database queries. Through systematic system maintenance, web-based to-do applications can uphold a high standard of reliability, security, and performance, providing users with a seamless and trustworthy experience. System maintenance is a critical aspect of ensuring the continuous and efficient operation of a web-based to-do application developed using PHP and MySQL. Maintenance involves a series of planned activities aimed at preserving and enhancing the system's performance, security, and usability over time. It is a proactive approach that addresses potential issues, improves functionality, and adapts the application to evolving requirements.

7.2 MAINTENANCE

Maintenance tasks for a web-based to-do application typically include regular database backups to safeguard against data loss, updating PHP and MySQL versions to benefit from performance enhancements and security patches, and reviewing and optimizing the codebase for improved efficiency. Additionally, monitoring server resources, such as CPU and memory usage, ensures optimal performance. User feedback and usage patterns should be analyzed to identify areas for enhancement or bug fixes. Regular testing, both automated and manual, helps identify and rectify issues promptly, ensuring a smooth and reliable user experience. Embracing best practices and staying current with technology updates ensures the long-term viability of the application. Regular maintenance is key to fostering a robust and secure environment for users, providing a seamless and reliable to-do management experience.

- Regular Database Maintenance:
 - To ensure the optimal performance and integrity of the web-based to-do application, regular database maintenance is crucial. This involves tasks such as database backups, indexing, and purging of outdated or completed tasks. Scheduled backups, preferably automated, should be implemented to safeguard against data loss or corruption. Indexing can enhance query performance, especially as the dataset grows. Additionally, periodic purging of completed tasks can help manage database size and keep the application responsive. Monitoring tools can be employed to track database metrics, identifying potential issues before they impact the user experience. Addressing these aspects in the maintenance routine ensures the long-term stability and reliability of the application's data storage.
- Codebase and Security Updates:
 - The maintenance of the web-based to-do application also involves keeping the codebase up-to-date and addressing security vulnerabilities promptly. Regularly checking for updates to PHP and MySQL is essential, as updates often include performance improvements, bug fixes, and security patches. Furthermore, adopting version control systems like Git enables efficient code management and version tracking. Performing regular security audits helps identify and fix potential vulnerabilities in the application code. It is advisable to follow coding best practices, such as input validation and output sanitization, to prevent common security threats like SQL injection and cross-site scripting. By incorporating these practices into the maintenance routine, the application remains robust, secure, and aligned with the latest industry standards.

8. FUTURE ENHANCEMENT AND SCOPE OF FURTHER DEVELOPMENT

8.1 INTRODUCTION

The future enhancement and development of the web-based to-do application built with PHP and MySQL present a promising opportunity to expand its functionality and user experience. One potential avenue for improvement is the introduction of user authentication and authorization, allowing each user to have a personalized to-do list. This would require user account management, secure login mechanisms, and the association of tasks with specific users. Additionally, the application could benefit from the incorporation of task prioritization, due dates, and categories to help users organize and manage their tasks more effectively.

8.2 MERITS OF THE SYSTEM

The current system provides a solid foundation for a to-do application, offering basic task management functionalities. The use of PHP and MySQL ensures a robust and scalable backend, while the simple HTML and JavaScript front end ensures broad accessibility. Future enhancements can build upon this foundation, improving the user experience, increasing security, and enabling collaborative features. The modular structure of the code allows for easy integration of new features, and the separation of concerns facilitates maintainability. With the potential introduction of mobile responsiveness and offline capabilities, the system could become even more versatile, catering to a diverse range of users and their evolving needs.

8.3 LIMITATIONS OF THE SYSTEM

Despite its merits, the current system has limitations that could be addressed in future development. Notably, the absence of user authentication exposes the application to potential misuse or unauthorized access. The lack of data validation and sanitization could leave the system vulnerable to security threats such as SQL injection. Additionally, the application's current simplicity may limit its appeal to users seeking more advanced task management features. These limitations highlight the importance of incorporating robust security measures and expanding the feature set to ensure the application remains competitive and meets the expectations of a broader user base.

8.4 CONCLUSION AND FURTHER ENHANCEMENT OF THE SYSTEM

In conclusion, the web-based to-do application demonstrates promise as a foundation for further development. By addressing the limitations and incorporating user feedback, the system can evolve into a more comprehensive and secure task management tool. Future enhancements should focus on user authentication, security measures, task prioritization, and collaborative features to elevate the application's utility and appeal. Regular updates and maintenance will be essential to keep the system aligned with evolving web technologies and user expectations.

8.4 CONCLUSION AND FURTHER ENHANCEMENT OF THE SYSTEM

In conclusion, the web-based to-do application demonstrates promise as a foundation for further development. By addressing the limitations and incorporating user feedback, the system can evolve into a more comprehensive and secure task management tool. Future enhancements should focus on user authentication, security measures, task prioritization, and collaborative features to elevate the application's utility and appeal. Regular updates and maintenance will be essential to keep the system aligned with evolving web technologies and user expectations.