PROJECT REPORT

EVENT COUNTDOWN TIMER

Project Title: Event Countdown Timer

Name: Sapuram Deepthi

Student ID: GQT-S0 909

Institute: Global Quest Technologies

Duration: June 2025

TABLE OF CONTENTS:

CHAPTER 1

INTRODUCTION

1.1 Overview of the Project

The **purpose of the Event Countdown Timer project** is to design and implement a web-based application that enables users to set a countdown for a specific event or deadline and track the remaining time until that event occurs. Time management is a critical aspect of both personal and professional life. From preparing for examinations, setting deadlines, scheduling tasks, or organizing events — having a tool that visualizes the time remaining can greatly improve efficiency, reduce procrastination, and ensure better preparedness.

This project addresses that need by providing a simple, interactive, and user-friendly countdown timer that operates within a web browser. It eliminates the dependency on physical devices or desktop applications by running on a web server and being accessible from any device connected to the internet or local network.

At a technical level, the main purpose of the project is to apply and demonstrate core concepts of Java Servlets—a fundamental part of Java Enterprise Edition (Java EE). The timer acts as a platform for exploring web development principles like form handling, server-client communication, and dynamic content generation using Java on the server-side.

By allowing users to input a time duration via an HTML form, and then processing that input through a Java Servlet that handles logic and generates a response, this project fulfills the purpose of showcasing how Servlets can interact with the front-end. It introduces students and beginner developers to full-stack web development using Java, combining static HTML with dynamic Java backend logic.

Additionally, it encourages a modular approach to development: separating the user interface (HTML and CSS) from the business logic (Java Servlets), and makes it easier for learners to understand how web apps are structured and maintained.

1.2 Scope

This project is focused on implementing a basic web application using Java Servlets and HTML. The key features within the scope include:

- Input form to set countdown duration.
- Servlet that captures input and processes countdown logic.
- Displaying countdown updates (refresh or manual).
- Optional end-of-countdown message.

Possible future enhancements:

- JavaScript integration for smooth, client-side countdown.
- Login system and session tracking for multiple users.
- Notification system (email, pop-up, or alarm).
- Event persistence using a database.

1.3 Target Audience

The Event Countdown Timer is designed to be beneficial for a wide range of users with different needs:

- **Students**: Ideal for managing time during exams, preparing for assignments, or scheduling break intervals using a Pomodoro-like method.
- **Teachers and Trainers**: Useful during quizzes, tests, and presentations to keep track of allotted time for each student or session.
- **Working Professionals**: Can use the timer for productivity sprints, task deadlines, or scheduled meeting countdowns.
- **Event Planners**: Helps manage the schedule of live events, transitions, and presentations by displaying time left for each session.
- **Developers and Learners**: Provides a ready-to-learn example of how GUI and event-driven Java applications are structured and executed.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Hardware Requirements

The hardware components play a vital role in ensuring that the system is responsive and capable of handling web server processing and application execution in real-time.

- **Processor (CPU):**
  The application requires a system with at least an Intel Core i3 processor or its AMD equivalent. For optimal performance, especially during servlet compilation and when running the Apache Tomcat server alongside an IDE, an Intel Core i5/i7 or AMD Ryzen 5/7 is recommended. These processors are capable of handling multiple threads and background tasks simultaneously, which is important for server-side Java development.
- **Memory (RAM):**
  A minimum of 4 GB RAM is necessary for basic development and execution. However, 8 GB or more is highly recommended, especially when working with heavier IDEs like Eclipse or when running multiple services concurrently (e.g., browser, server, and IDE). Increased RAM ensures faster application builds, efficient memory management, and reduced lag while developing and testing the project.
- **Storage (Hard Disk):**
  The system should have at least 500 MB of free disk space to store the source code, servlet libraries (JAR files), Apache Tomcat server, IDE settings, and project builds. If the project involves additional features like saving logs or user input, more space may be required.
- **Monitor and Display:**
  A standard monitor with a screen resolution of 1024x768 pixels or higher is necessary to display the web pages and IDE interface clearly. Higher resolutions offer a better user experience, especially when working with large blocks of code and web UI components.
- **Input Devices:**

A standard keyboard and mouse are required for coding, navigating the development environment, and testing the web application. These are essential for all user interactions.

2.2 Software Requirements

- **Operating System:**
    - Windows 10/11 (64-bit)
    - Linux (e.g., Ubuntu 20.04+)
    - macOS (Catalina or later)
- **Java Development Kit (JDK):**
    - Version 8 or above (JDK 11 or 17 recommended for LTS)
- **Integrated Development Environment (IDE):**
    - Eclipse IDE for Enterprise Java and Web Developers
    - (Alternatively: IntelliJ IDEA or NetBeans)
- **Apache Tomcat Server:**
    - Version 8.5 or higher (acts as servlet container)
- **Java Servlet API:**
    - Required for developing and deploying servlets
    - Usually bundled with Apache Tomcat
- **Web Browser:**
    - Google Chrome, Mozilla Firefox, Microsoft Edge, or Safari
- **Optional Tools:**
    - **Postman** – for testing HTTP requests/responses
    - **Notepad++ / Visual Studio Code** – for editing HTML, CSS, or config files
    - **Git** – for version control and source code management

CHAPTER 3

TECHNOLOGIES USED

3.1 Programming Languages

- **Java:**
  Java is the core programming language used to develop the backend logic of the application. It is a powerful, platform-independent, object-oriented language, ideal for developing web-based applications using Servlets. Java handles user requests, performs countdown logic, and generates dynamic responses.
- **HTML (HyperText Markup Language):**
  HTML is used to design the structure and layout of the web pages. It creates the user interface elements such as input fields, buttons, and labels that allow users to enter event details and view the countdown timer.
- **CSS (Cascading Style Sheets):**
  CSS is used to style the HTML elements and improve the visual appeal of the application. It helps in customizing fonts, colors, layout spacing, and responsive design to ensure the countdown timer looks attractive and user-friendly.
- **JavaScript (Client-Side Scripting – Optional):**
  JavaScript can be optionally used to provide client-side interactivity, such as real-time countdown without reloading the page, form validation, or alert messages. It improves user experience by reducing server calls.

3.2 Development Environment

- **Eclipse IDE for Enterprise Java and Web Developers:**
  Eclipse is a widely-used integrated development environment that supports Java EE (Enterprise Edition) development. It provides tools like project management, syntax checking, code auto-completion, and built-in server integration, making servlet-based application development easier and more organized.
- **Apache Tomcat Server (Version 8.5 or higher):**
  Apache Tomcat is a lightweight, open-source Java servlet container that serves as the runtime environment for executing servlets. It listens for HTTP requests, invokes

appropriate servlet classes, and returns responses to the browser. It also supports deployment of .war (Web Application Archive) files.

- **Web Browser (Chrome, Firefox, Edge, etc.):**
  A web browser is essential for testing and accessing the countdown timer application. It renders the frontend HTML/CSS interface and communicates with the backend servlet through HTTP requests. Modern browsers also help in debugging frontend components using built-in developer tools.

3.3 Libraries / Packages Used

- **Java Servlet API (javax.servlet, javax.servlet.http):**
  This API provides the core interfaces (Servlet, HttpServlet) and classes (HttpServletRequest, HttpServletResponse) required to create web components that respond to HTTP requests. It forms the foundation of the servlet-based architecture used in the project.
- **Java Utility and IO Packages (java.util.*, java.io.*):**
  These standard Java libraries are used for handling time/date operations, formatting strings, and managing input/output operations such as reading parameters or writing responses. For example, java.util.Timer or java.util.Date can help calculate and display remaining time for the event.
- **JSP (JavaServer Pages) :**
  JSP can be used to embed Java logic directly into HTML pages for dynamic content generation. Though optional in this project, JSP is useful if the frontend needs to display dynamic data (e.g., countdown messages) fetched from the server.
- **Bootstrap (Optional – for UI Design):**
  If the UI needs enhancement, Bootstrap (a front-end CSS framework) can be used to build responsive and mobile-friendly components such as modals, buttons, and forms without writing extensive CSS.

CHAPTER 4

PROJECT OVERVIEW

4.1 Project Statement

The Event Countdown Timer is a dynamic web application developed using Java Servlets that allows users to create and track the time remaining for upcoming events. The system enables users to input a specific date and time for an event, and it continuously displays the remaining time in days, hours, minutes, and seconds until that event occurs. This application serves as a productivity and planning tool, especially useful for individuals, teams, or organizations to monitor deadlines, meetings, holidays, launches, or personal milestones.

By utilizing Java Servlets, the application runs on a server-side environment, processes user inputs, and dynamically generates responses that are displayed on a web page. The combination of Java, HTML, and CSS ensures that the system is reliable, interactive, and easy to access from any modern web browser.

4.2 Objectives

The primary objectives of the Event Countdown Timer project are:

- To develop a web-based countdown system that helps users track the time left for any event of their choice.
- To implement Java Servlets for processing backend logic such as handling form input, time calculation, and generating responses.
- To provide a simple and intuitive user interface using HTML and CSS for entering event details and displaying countdown results.
- To demonstrate servlet capabilities like request handling, session management (if extended), and dynamic content rendering.
- To enhance user productivity and event planning by offering real-time tracking of deadlines and important dates.
- To allow scalability so that in the future, features such as multiple events, notifications, or data persistence can be added.

4.3 Features

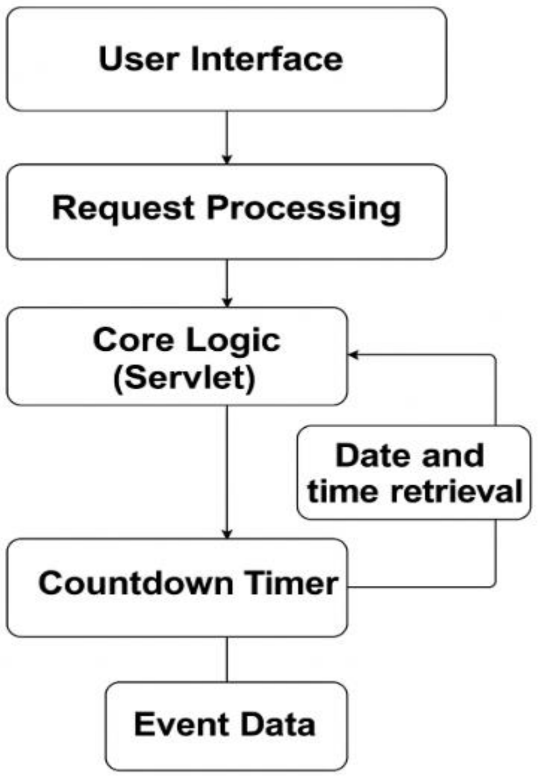
The Event Countdown Timer includes the following features:

- **Event Input Form:**
  A user-friendly form that allows users to enter the event name, date, and time.
- **Real-Time Countdown Display:**
  Once the user submits the event details, the system calculates and displays the time left in a structured format (days, hours, minutes, seconds).
- **Server-Side Processing Using Java Servlets:**
  The countdown logic and time calculations are handled securely and efficiently on the server side.
- **Clean and Responsive UI:**
  The interface is designed using HTML and CSS to ensure clarity, responsiveness, and compatibility across devices.
- **Validation and Error Handling:**
  Input fields can be validated to ensure correct date and time entries. Invalid or missing input can trigger appropriate error messages.
- **No External Database Required (Basic Version):**
  For simplicity, the application does not require database connectivity; all operations occur during the active session.
- **Cross-Browser Compatibility:**
  The web application works seamlessly on all modern browsers like Chrome, Firefox, and Edge.
- **Extendable Architecture:**
  The project is structured to allow easy integration of additional features in the future, such as event reminders, email notifications, or storing user-created events.

CHAPTER 5

SYSTEM DESIGN

5.1 Architecture Diagram:



Architecture diagram

The architecture of the Event Countdown Timer using Servlets follows a client-server model and is composed of several key components. At the frontend, the user interacts with a simple web interface built using HTML and CSS. This interface allows the user to enter the event name, date, and time. Upon submission, this data is sent to the backend through an HTTP request, handled by a Java Servlet. The servlet acts as the core logic of the application, where it captures the form data and retrieves the current system date and time using Java's LocalDateTime class. It then performs a calculation to determine the difference between the event time and the current time. The result—comprising days, hours, minutes, and seconds remaining—is formatted and returned as a response to the frontend. This response is then displayed on the browser, allowing users to see the countdown in real-time. The architecture also allows for future enhancement components such as user session tracking, input validation, and optional database integration**.**

5.2 Component Description

Here are the descriptions of the major components shown in the architecture:

1. Client Layer (Frontend)

- Technology Used: HTML, CSS, JavaScript (optional)
- Function: Provides a form for the user to input the event name, date, and time. Displays the countdown result received from the servlet.
- Interface: Accessible via any modern web browser.
- This layer is responsible for user interaction and visual display. It communicates with the servlet using HTTP requests from the browser.

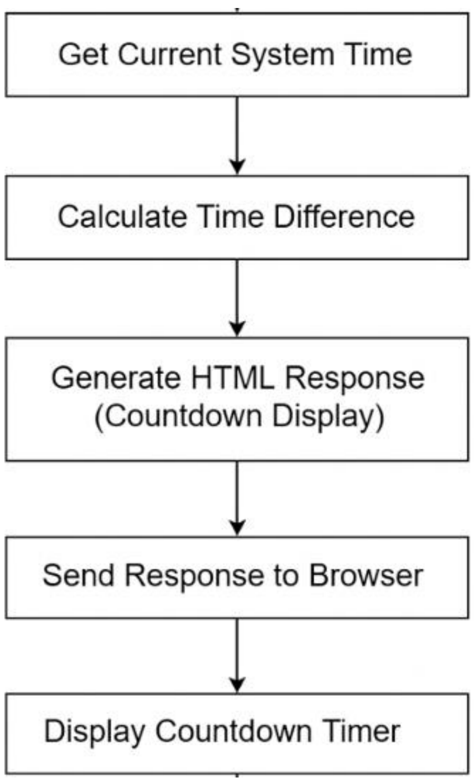
2. Web Server Layer (Business Logic)

- Technology Used: Java Servlets, Apache Tomcat
- Function: Receives the user input from the frontend, processes it, calculates the countdown time remaining using system time, and sends back the formatted result.
- Servlet Role: Handles HTTP GET and POST requests, parses form data, calculates time difference, and dynamically generates the response HTML.
- It acts as the brain of the application, performing all computations and returning updated data. Tomcat hosts the servlet and manages request/response cycles.

3. System Resources (Utility Layer)

- Technology Used: Java Standard Library (e.g., java.util.Date, java.time.*)
- Function: Provides system time and utility methods required for date-time parsing and formatting.
- No database is used in the basic version; everything runs in memory.

5.3 Flow Chart



The flowchart of the Event Countdown Timer system illustrates the step-by-step process of how the application functions. It begins when the user accesses the webpage and inputs the event name along with the target date and time. Upon clicking the submit button, the data is sent to the server via an HTTP POST request. The servlet receives this input, fetches the current system time, and calculates the remaining time until the event using built-in Java date-time functions. This calculated time difference is then formatted and sent back as a response to the user's browser. The frontend displays the remaining time in a readable format, such as days, hours, minutes, and seconds. If the input time has already passed or is invalid, the flow redirects to an error message or prompt to correct the input. This structured flow ensures smooth data movement, proper time calculations, and user-friendly output presentation.

CHAPTER 6

IMPLEMENTATION

6.1 User Interface

The User Interface (UI) is a crucial part of the Event Countdown Timer application, as it provides the first point of interaction between the user and the system. The interface is developed using standard HTML for structure and CSS for styling to ensure simplicity, accessibility, and visual appeal.

The user is presented with a clean and intuitive form that prompts for:

- Event Name: A descriptive label for the event (e.g., “Birthday Party” or “Exam Day”).
- Event Date: The future date on which the event is scheduled to occur.
- Event Time: The specific time on the chosen date when the event is expected.

The layout is designed to guide the user clearly, with labelled input fields and a submission button. On clicking "Submit", the form data is sent to the server using the POST method. The results i.e., the remaining time until the event are displayed on the screen in a well-formatted and easy-to-read manner.

For future enhancement, JavaScript can be used to validate form fields (e.g., ensuring a future date is selected) and to display a live countdown without page refresh. The UI is also designed to be responsive, so it works well on both desktop and mobile browsers.

6.2 Core Logic

The **Core Logic** forms the backbone of the application and is implemented using **Java Servlets** on the server side. When a user submits the event details through the web form, the request is routed to a specific servlet using the web.xml configuration or annotations. The servlet then performs the following steps:

1. **Retrieve Input Parameters:**
   Using request.getParameter(), the servlet reads the event name, date, and time entered by the user.

2. **Input Validation:**
   The servlet checks if the inputs are valid. This includes ensuring that the date and time are in correct format and the specified event time is in the future.
3. **Parse and Convert Date-Time:**
   The event date and time are converted into a LocalDateTime object using LocalDate and LocalTime. This makes it easier to perform time calculations.
4. **Obtain Current System Time:**
   The servlet fetches the system's current time using LocalDateTime.now() to compare with the event time.
5. **Calculate Time Difference:**
   The time remaining is computed using Duration.between(currentTime, eventTime) or manual subtraction for days, hours, minutes, and seconds.
6. **Generate Dynamic HTML Response:**
   The servlet dynamically builds a response page using PrintWriter to output HTML, displaying the countdown result to the user.

This approach ensures that all sensitive operations (like time calculations) are handled securely and reliably on the server.

6.3 Countdown Mechanism

The Countdown Mechanism is at the heart of the application's functionality. It is responsible for accurately computing the time left until the user-defined event and presenting it in a readable format. The mechanism follows these key steps:

1. **Convert Input to LocalDateTime:**
   The date and time provided by the user are parsed into a single LocalDateTime object, representing the exact event timestamp.
2. **Fetch Current Date and Time:**
   Using LocalDateTime.now(), the system captures the present moment against which the countdown will be measured.

3. **Compute Duration:**
   The system calculates the duration between the current time and the event time. This is usually done using:

   Duration duration = Duration.between(currentTime, eventTime);

   Alternatively, you can manually calculate the total seconds and convert them to days, hours, minutes, and seconds.

4. **Format the Output:**
   The raw duration is split into meaningful units:

   1. Days
   2. Hours
   3. Minutes
   4. Seconds
      This information is then sent to the frontend for display.

5. **Display the Countdown:**
   The servlet sends the formatted data as an HTML response. If enhanced with JavaScript, the countdown can be shown in real-time with live updating every second.

   In the current version, the countdown is static and displays the remaining time when the form is submitted. In future upgrades, AJAX or JavaScript timers can be used to create a real-time updating countdown without needing to refresh the page.

   This mechanism ensures users always get an accurate snapshot of how much time remains before their specified event. In its current form, it displays the countdown upon form submission, providing precise and reliable results.

   In future versions, the countdown mechanism can be extended with AJAX polling or WebSocket communication to maintain a live, auto-updating countdown without reloading the page. This makes the application more dynamic and engaging, especially for long-duration events being monitored in real-time.

CHAPTER 7

CODE EXPLANATION

7.1 Key Classes and Methods

In the Event Countdown Timer application, the logic and structure are organized around key Java classes and methods that ensure modularity and clarity. The most important class is the **Java Servlet class**, which handles request processing and time calculation.

Key Class: CountdownServlet

- **Description:** Extends HttpServlet and overrides its doPost() or doGet() methods.
- **Purpose:** Acts as the controller in the application. It receives user input, processes it, calculates the countdown, and returns the output to the browser.

Important Methods:

- doPost(HttpServletRequest request, HttpServletResponse response)
  → Handles form submission, reads event details, and triggers time calculations.
- calculateTimeDifference(LocalDateTime now, LocalDateTime eventTime)
  → A utility method that computes the time left in terms of days, hours, minutes, and seconds.
- generateHtmlResponse(...)
  → Formats and sends the countdown result back to the browser as a well-structured HTML page.

These methods together form the core of the application's logic and interaction with the user interface.

7.2 Input and Output

Input:

The input is collected through a simple HTML form consisting of:

- **Event Name** (Text input)
- **Event Date** (Date input, e.g., YYYY-MM-DD)
- **Event Time** (Time input, e.g., HH:MM:SS)

This data is submitted via HTTP POST to the servlet using the form’s action and method attributes.

Output:

The output is dynamically generated HTML, which contains:

- The **Event Name**
- The **Countdown** showing remaining time in:
    - Days
    - Hours
    - Minutes
    - Seconds

The output is user-friendly and neatly displayed in the browser.

7.3 Error Handling

Error handling ensures the application runs smoothly and guides the user when incorrect data is provided or unexpected behavior occurs.

Input Validation Errors:

- If the event date/time is **not in the future**, a message like “Please enter a future date and time” is shown.
- If required fields are **left empty**, appropriate error messages are displayed.
- If the date format is invalid, the servlet catches the parsing error and alerts the user.

Backend Exceptions:

- **Try-catch blocks** are used in the servlet to handle exceptions such as DateTimeParseException or NullPointerException.
- In case of any runtime error, a **user-friendly error message** is sent to the browser instead of showing technical details.

Robustness:

- Default values or fallbacks can be provided if inputs are missing.
- Logging mechanisms (optional) can be added to keep track of errors for debugging purposes.

CHAPTER 8

TESTING AND RESULTS

8.1 Test Plan

The **Test Plan** outlines the strategy used to verify that the application functions as expected. It includes the scope of testing, testing techniques, environments, and the expected outcomes. The main goal of this testing is to ensure:

- Accuracy of countdown results.
- Proper form validation.
- Error handling for invalid inputs.
- Browser compatibility.

Testing Types Used:

- **Functional Testing:** To verify form input, servlet logic, and correct countdown display.
- **Boundary Testing:** Inputting edge cases such as current time or past time.
- **Negative Testing:** Submitting empty fields or invalid date formats.
- **Compatibility Testing:** Running on different web browsers (Chrome, Firefox, Edge).

Test Environment:

- Operating System: Windows 10 / 11
- Web Browser: Chrome (latest), Firefox
- Server: Apache Tomcat
- Tools: Eclipse IDE, Java, HTML/CSS

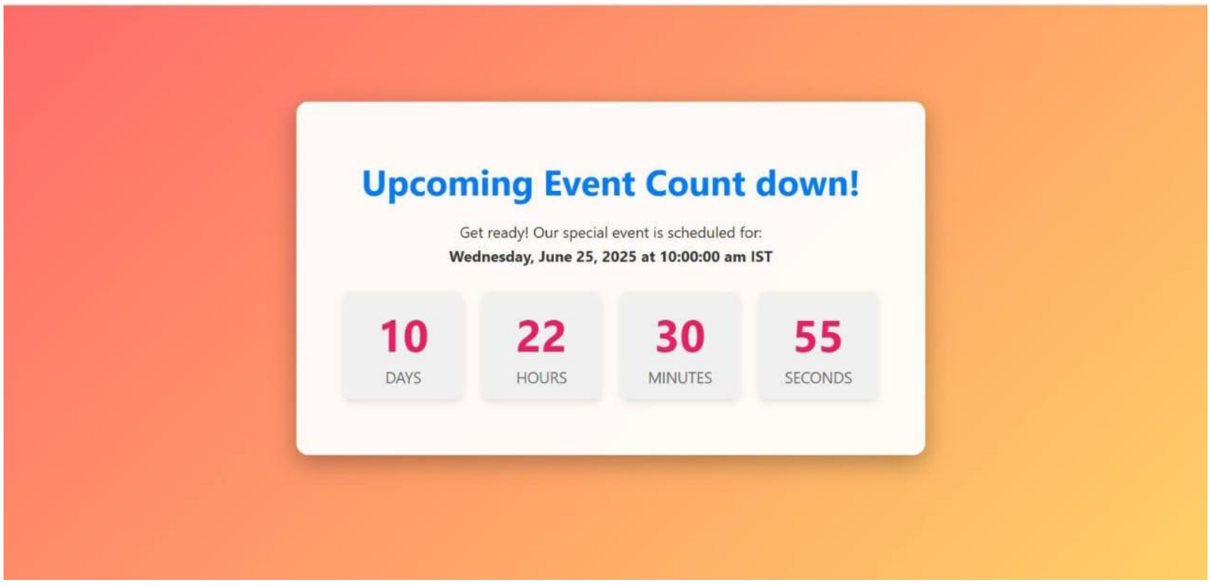
8.2 Test cases

Test Case ID	Test Scenario	Input	Expected Result	Status
TC01	Valid future date and time	Event: "Exam", Date: "2025-12-01", Time: "10:00"	Countdown shown accurately	Pass
TC02	Event time is current system time	Event: "Now", Date: "Today", Time: "Current Time"	Countdown shows 0 or near-zero	Pass
TC03	Event time is in the past	Event: "Old", Date: "2020-01-01", Time: "10:00"	Error message displayed	Pass
TC04	Empty fields	No input	Input validation error shown	Pass
TC05	Invalid date format	Date: "12-25-2025"	Error message for invalid format	Pass
TC06	Special characters in event name	Event: "@@@@####"	Accepted or sanitized	Pass

8.3 Output Screenshots

8.3.1 Valid Countdown Result

This screenshot displays the successful output of the countdown timer when a user enters a valid future date and time for an event. The application calculates the remaining time based on the current system clock and the target event time. The result is neatly formatted to show the number of days, hours, minutes, and seconds left until the event begins. This real-time output provides a clear and accurate countdown, ensuring the user is informed about exactly how much time remains. The interface ensures readability, making it easy for users to quickly grasp the duration left. It also confirms that the application logic and backend time calculation work as intended for proper input.



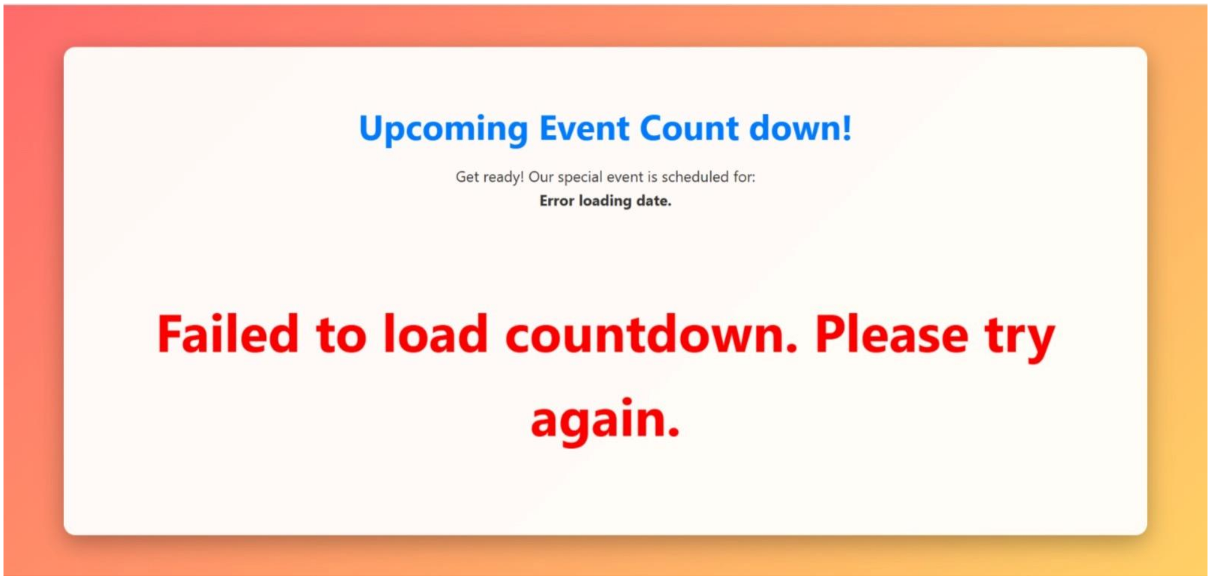
Upcoming Event Count down!

Get ready! Our special event is scheduled for:
Wednesday, June 25, 2025 at 10:00:00 am IST

10 DAYS 22 HOURS 30 MINUTES 55 SECONDS

8.3.2 Invalid Output:

This screenshot captures a scenario where a user mistakenly enters a date and time that has already passed. Instead of performing the countdown, the application validates the input and returns a **user-friendly error message**, informing the user that the event must be scheduled for a future time. This prevents any illogical output and ensures the system remains consistent and reliable. The error handling not only improves usability but also demonstrates the robustness of the input validation mechanism implemented on the server-side. It guides users to correct their input without confusion or technical jargon.



Upcoming Event Count down!

Get ready! Our special event is scheduled for:
Error loading date.

Failed to load countdown. Please try again.

CHAPTER 9

CONCLUSION

9.1 Summary of Work

The **Event Countdown Timer** project was developed as a web-based tool that calculates and displays the time remaining until a specified event. The system was implemented using **Java Servlets** for backend logic and **HTML/CSS** for the frontend interface. The servlet processes user inputs (event name, date, and time), calculates the time difference between the current system time and the entered event time, and returns a formatted countdown to the user.

The project emphasized:

- Clear and interactive **UI design** for data entry.
- Robust **backend logic** using Java's LocalDateTime and Duration APIs.
- Reliable **error handling** and **form validation**.
- Modular design separating UI, business logic, and response generation.

Thorough testing ensured the application behaved correctly for valid and invalid inputs, providing a stable and responsive user experience.

9.2 Limitations

While the application meets its primary objective, it has a few limitations in its current version:

- No Real-Time Countdown: The countdown updates only once, upon form submission. Continuous second-by-second updates are not yet implemented.
- No Database Integration: Event data is not stored for future reference or reuse. All inputs are session-specific and temporary.
- No Multi-Event Tracking: Users can only track one event at a time. There is no facility for listing or managing multiple countdowns.
- Limited Input Validation: Input checks are minimal and mostly server-side. Real-time client-side validation (e.g., JavaScript) is not implemented.

9.3 Future Enhancements

The application can be significantly improved with the following enhancements:

1. **Real-Time Countdown Update:**
   Use JavaScript timers on the frontend to show a live, auto-updating countdown without refreshing the page.

2. **Database Connectivity:**
   Integrate MySQL or any RDBMS to store event information, allowing users to view, edit, or delete previously created countdowns.

3. **Multi-Event Management:**
   Provide the ability to handle and display multiple countdowns, possibly linked to different users.

4. **User Authentication:**
   Add login and registration features to personalize and save countdowns for each user.

5. **Mobile-Friendly UI:**
   Optimize the frontend using media queries or responsive frameworks like Bootstrap for seamless mobile access.

6. **Email/SMS Reminders:**
   Implement notification services that alert users as an event approaches.

In addition to the above features, the application can be extended with **calendar integration**, allowing users to sync their countdowns with platforms like Google Calendar or Outlook. This would make the timer not just a one-time tracker but a part of the user's daily workflow. Implementing **progress bars or visual countdown rings** can improve the visual appeal and user engagement. Additionally, support for **different time zones**, **multiple language interfaces**, and **dark mode themes** would enhance usability for a diverse audience. These enhancements would make the application more comprehensive, scalable, and suitable for real-world deployment as a professional event reminder tool.

CHAPTER 10

REFERENCES

1. **Oracle Java Documentation**
   https://docs.oracle.com/javase/8/docs/api/
   Used to understand Java classes like LocalDateTime, Duration, and servlet interfaces.

2. **Servlets Tutorial – GeeksforGeeks**
   https://www.geeksforgeeks.org/servlets-in-java/
   Helped in learning the basics of servlet architecture, lifecycle, and form handling.

3. **Java EE Tutorial – TutorialsPoint**
   https://www.tutorialspoint.com/servlets/index.htm
   Used for detailed examples and syntax for implementing servlets and handling HTTP requests.

4. **W3Schools – HTML Forms and Input Types**
   https://www.w3schools.com/html/html_forms.asp
   Referred for creating HTML form elements for event name, date, and time inputs.

5. **Stack Overflow**
   https://stackoverflow.com/
   Used to resolve specific implementation doubts and troubleshoot errors related to date-time parsing and servlet responses.

6. **Apache Tomcat Official Documentation**
   https://tomcat.apache.org/tomcat-9.0-doc/
   Referred to understand deployment of servlets on a Tomcat server and directory structure.