

CONCENTRIQ- A STUDENT FOCUS APP

An Internship Project Report Submitted in partial fulfillment of the requirements for the award of
the
degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND BUSINESS SYSTEMS

Submitted by

A.Bhavya Harshita(21071A3207)

B.V Vimal (21071A3209)

G.Tejaswi (21071A3219)

G.Pranav Kumar (21071A3224)

M.Sravanth (21071A3245)

P.LS Deepika (21071A3255)

Under the guidance of

Dr. G.S.Ramesh

(Assistant Professor, Department of CSE, VNRVJIET)



DEPARTMENT OF COMPUTER SCIENCE AND BUSINESS SYSTEMS

**VALLURUPALLI VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND
TECHNOLOGY**

VALLURUPALLI VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute ,NAACAccredited with 'A++' Grade NBA Accredited for CE,EEE,ME,ECE,CSE,EIE,IT B.Tech Courses Approved by AICTE ,New Delhi,Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified QS 1 GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet(S.O),Hyderabad -500090,TS,India

DEPARTMENT OF COMPUTER SCIENCE AND BUSINESS SYSTEMS



CERTIFICATE

This is to certify that **A.Bhavya Harshita(21071A3207),B.V Vimal(21071A3209),G.Tejaswi(21071A3219),G.PranavKumar(21071A3224),M.Sravanth(21071A3245),P.L.SDe epika(21071A3255)** have completed their internship project work at CSBS Department of VNR VJIET, Hyderabad entitled "**CONCENTRIQ-A STUDENT FOCUS APP**" in complete fulfilment of the requirements for the award of B.Tech degree during the academic year 2022-2023. This work is carried out under my supervision and has not been submitted to any other University/Institute for award of any degree/diploma.

Dr. G.S. Ramesh
Assistant Professor
CSE & CSBS
Department
VNR VJIET

Dr. S Nagini
Professor & HOD
CSE & CSBS
Department
VNR VJIET

VALLURUPALLI VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute ,NAAC Accredited with 'A++' Grade NBA Accredited for CE,EEE,ME,ECE,CSE,EIE,IT B.Tech Courses Approved by AICTE ,New Delhi,Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified QS 1 GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet(S.O),Hyderabad -500090,TS,India

DEPARTMENT OF COMPUTER SCIENCE AND BUSINESS SYSTEMS



DECLARATION

This is to certify that our project report titled "CONCENTRIQ-A Student Focus App" submitted to Vallurupalli Nageswara Rao Institute of Engineering and Technology in complete fulfilment of requirement for the award of Bachelor of Technology in Computer Science and Engineering is a bonafide report to the work carried out by us under the guidance and supervision of **Dr.G.S.Ramesh** , Assistant Professor, Department of Computer Science and Engineering, Vallurupalli Nageswara Rao Institute of Engineering and Technology. To the best of our knowledge, this has not been submitted in any form to other universities or institutions for the award of any degree or diploma.

A.Bhavya Harshitha	B.V Vimal	G.Tejaswi	G.Pranav	M.Sravanth	P.L.SDeepika
21071A3207	21071A3209	21071A3219	21071A3224	21071A3245	21071A3255

ACKNOWLEDGEMENT

Firstly, we would like to express our immense gratitude towards our institution VNR Vignana Jyothi Institute of Engineering and Technology, which created a great platform to attain profound technical skills in the field of Computer Science, thereby fulfilling our most cherished goal. We are very much thankful to our Principal, Dr. Challa Dhanunjaya Naidu, and our Head of Department, Dr. S.Nagini, for extending their cooperation in doing this project within the stipulated time. We extend our heartfelt thanks to our guide, Dr. G.S. Ramesh for his enthusiastic guidance throughout the course of our project.

Last but not least, our appreciable obligation also goes to all the staff members of the Computer Science & Engineering department and to our fellow classmates who directly or indirectly helped us.

A.Bhavya Harshita(21071A3207)

B.V Vimal (21071A3209)

G.Tejaswi (21071A3219)

G.Pranav Kumar (21071A3224)

M.Sravanth (21071A3245)

P.LS Deepika (21071A3255)

ABSTRACT

In the fast-paced digital landscape, "ConcentriQ" stands out as an innovative powerhouse, dedicated to elevating individuals' goal achievement. Seamlessly integrating a to-do list, a focus timer, a streak system, and a distinct friend collaboration feature, our platform transforms task management into a dynamic and rewarding experience. Users can effortlessly organize, edit, and track tasks, utilizing a pomodoro timer embedded with a music playlist and a smart website blocker. This unique timer activates only in full-screen mode, optimizing focused work sessions. The addition of a chat room fosters real-time communication among friends, enhancing connectivity and collaborative learning. With ConcentriQ, we redefine productivity, blending efficiency, motivation, and community engagement for a holistic goal-oriented experience.

INDEX

1. Introduction	8
2. Literature Survey/ Existing System	9
2.1 Literature Review	9
2.2. Existing System	10
2.3. Drawbacks Of the Existing System	10
3. System Requirement Analysis	12
3.1. Introduction	12
3.1.1 Document Purpose	12
3.1.2 Definitions	12
3.2. System Architecture	13
3.3. Functional Requirements	14
3.4. System Analysis	17
3.5. Non-Functional Requirements	17
3.6. Software Requirement Specification	18
3.7. Software Requirements	18
3.8. Hardware Requirements	18
4. Software Design	19
4.1. UML Diagrams	20
4.1.1 Use Case Diagram	19
4.1.2 Sequence Diagram	23
4.1.3 Activity Diagram	26
4.1.4 Class Diagram	29
4.1.5 Communication Diagram	31
4.1.6 State Chart Diagram	32

4.1.7 Component Diagram	34
4.1.8 Deployment Diagram	35
5. Proposed System	37
5.1. Methodology	38
5.2. Functionalities	40
5.3. Advantages Of Proposed System	42
6. Testing	44
6.1 Non-Functional Testing	44
6.1.1 Performance Testing	44
6.1.2 Security Testing	44
6.1.3 Usability Testing	44
6.2 Functional Testing	45
6.2.1 Integration Testing	45
6.2.2 Regression Testing	45
6.2.3 Unit Testing	45
7. Results	46
8. Conclusion And Future Scope	53
9. References	54

1. INTRODUCTION

ConcentriQ emerges as a revolutionary platform, transcending traditional task management with its innovative features and user-centric design. At its core, ConcentriQ provides users with a dynamic and user-friendly interface that grants them unparalleled control over their to-do lists. The platform goes beyond the basics, offering customizable options and deadlines that make organizing tasks a breeze. One of ConcentriQ's standout features is its built-in focus timer, a game-changer for those seeking structured work and study sessions. This timer is not your ordinary productivity tool – it comes embedded with a carefully curated playlist designed to enhance focus and productivity. Simultaneously, the website blocker feature ensures that access to distracting websites takes a backseat, creating an environment conducive to deep concentration. What sets ConcentriQ apart is its social productivity aspect. The platform transforms solo productivity into a collaborative and engaging experience. Users can compete with friends, injecting a sense of friendly rivalry into goal achievement. This gamified approach not only adds an element of fun but also fosters mutual accountability, encouraging users to push their limits and achieve more together. Addressing privacy concerns is a top priority for ConcentriQ. The platform ensures secure access through a user-friendly account creation process. Once logged in, users are greeted with a personalized task management dashboard. This tailored space not only enhances the user experience but also instills confidence in users, assuring them that their sensitive information is in safe hands. In essence, ConcentriQ is a comprehensive solution that seamlessly integrates task management, focused work sessions, social collaboration, and privacy features. It stands as a testament to the evolving landscape of productivity tools, catering to the diverse needs of individuals in the digital age. With ConcentriQ, productivity isn't just a task – it's a holistic and collaborative journey toward personal and collective success.

2. LITERATURE SURVEY

2.1 LITERATURE REVIEW

“Turning Time From Enemy into an Ally using the Pomodoro Technique” says

that Exploration of research on the challenges and opportunities associated with time management in the contemporary digital era. Analysis of strategies employed by existing tools to address time-related issues.[1]

An Analysis of Winning Streak's Effects in Language Course of "Duolingo" said

Examination of psychological literature on the role of streaks in habit formation and goal achievement.

Evaluation of existing platforms incorporating streak systems and their impact on user behavior.[2]

“Collaboration and community: Building strength in tertiary education” said

Study of collaborative tools and features in digital platforms that contribute to community building and mutual support.

Analysis of how friend collaboration features impact user motivation and accountability.[3]

User Experience and Interface Design:

[7] Charles et.al Exploration of research on user-friendly interface design and its influence on user engagement. Review of studies on the importance of secure login systems in digital platforms.[4]

Success Stories and User Testimonials:

[10] Bose et.al Inclusion of success stories and user testimonials related to the use of similar productivity tools. Analysis of user feedback and satisfaction in existing literature.[5]

Gaps in Current Literature and Future Directions:

[12] Wills et.al Identification of gaps in the current literature that ConcentriQ aims to fill suggestions for potential areas of improvement or future research related to digital productivity tools [6]

2.2 EXISTING SYSTEMS

In a landscape dominated by productivity tools such as HabitNow, Google Calendar, and Any.do, each tailored to enhance student focus and categorized as scheduler applications, it's essential to acknowledge their widespread usage while also recognizing certain limitations.

Any.do excels in providing users with the ability to create and schedule tasks, effectively functioning as a daily planner. However, it's crucial to note that its functionality may be constrained to task management.

Pomofocus, on the other hand, stands out as a customizable Pomodoro timer designed for desktop use, leveraging the Pomodoro Technique for effective time management. While beneficial for focused work sessions, its scope might be limited to time-based productivity.

Google Calendar offers a versatile platform for creating, editing, and managing events, including recurring ones. Nevertheless, its primary focus lies in event scheduling and may not cover all aspects of comprehensive task management.

Brain-FM offers AI-driven music streaming service designed to enhance focus, relaxation, and sleep. Offers a unique approach to using music as a tool for product

2.3 DRAWBACKS OF THE EXISTING SYSTEM

While popular productivity tools like HabitNow, Google Calendar, and Any.do offer valuable features, they also come with certain limitations. Here are some potential disadvantages of these existing systems:

Any.do:

Limited Functionality: Any.do primarily focuses on task creation and scheduling, potentially lacking in-depth features for comprehensive task management or collaborative efforts.

Scope: Its functionality may be confined to daily planning, potentially limiting its usefulness for long-term goal tracking.

Pomofocus:

Narrow Focus: Pomofocus excels as a Pomodoro timer but may lack the breadth of features required for comprehensive task organization and management.

Dependency on Technique: Users who do not resonate with or prefer alternative time management techniques might find Pomofocus less adaptable.

Google Calendar:

Task Management Limitations: While excellent for events and appointments, Google Calendar may not provide robust features for detailed task management, potentially leading to a fragmented productivity approach.

Interface Complexity: The interface, designed for various functionalities, might be overwhelming for users seeking a simpler task-oriented platform.

AI-driven Music Streaming Service:

Subjectivity: Music preferences vary greatly among individuals, and the effectiveness of the AI-driven music service may not be universal, potentially limiting its impact on focus and productivity.

Dependency on Music: Users who prefer a quiet work environment or have specific audio preferences may not find the music-focused approach suitable for their needs.

General Limitations:

Lack of Collaboration Features: Most of these systems may not have robust features for collaborative task management, limiting their suitability for group projects or team collaboration.

Learning Curve: Users may find a learning curve associated with the adoption of these tools, impacting their immediate usability and efficiency.

3. SOFTWARE REQUIREMENT ANALYSIS

3.1 INTRODUCTION

Software requirements for ConcentriQ encompass a set of specifications and tools that form the foundation for its development, functionality, and overall performance. Here's a description of the software requirements for ConcentriQ:

3.1.1 DOCUMENT PURPOSE

It outlines the primary goals and intentions behind creating the document. It serves as an introduction to provide readers with an understanding of why the document exists and what they can expect to find within its pages. The purpose is to set the context, define the scope, and clarify the objectives of the documentation.

3.1.2 DEFINITIONS

HTML, CSS, JavaScript:

These fundamental web development languages are employed to structure the content (HTML), style the user interface (CSS), and add dynamic behavior to the application (JavaScript). HTML provides the basic structure, CSS enhances the visual appeal, and JavaScript facilitates interactive and responsive features.

Node.js and Express:

Node.js, a server-side JavaScript runtime, is utilized along with the Express.js framework for building the backend of ConcentriQ. This combination enables the development of a robust and scalable server that handles various functionalities, such as task management, collaboration, and user authentication.

Socket.io:

Socket.io is integrated to facilitate real-time communication in the chat room feature. It enables seamless interaction between users, ensuring quick updates and responses in a collaborative environment.

Visual Studio Code (VSCode):

VSCode serves as the integrated development environment (IDE) for ConcentriQ's development team. It provides a feature-rich and extensible platform for coding, debugging, and version control, enhancing the efficiency of the development process.

Postman:

Postman is employed for API development and testing. It allows developers to design, test, and document APIs, ensuring that the application's backend functionalities are working correctly and efficiently.

Nodemon:

Nodemon is utilized as a development tool to automatically restart the server whenever changes are made to the code. This accelerates the development process by eliminating the need for manual server restarts after code modifications.

Chakra UI:

Chakra UI is implemented as a component library to enhance the user interface of ConcentriQ. It provides a set of accessible and customizable UI components, ensuring a modern and consistent design across the application.

Bcrypt:

Bcrypt is employed for secure password hashing. It adds a layer of security by encrypting and protecting user passwords, reducing the risk of unauthorized access in the case of a data breach. These software requirements collectively contribute to the development, functionality, and security of ConcentriQ, providing a comprehensive technological framework for a seamless and feature-rich user experience.

3.1.3 SYSTEM ARCHITECTURE

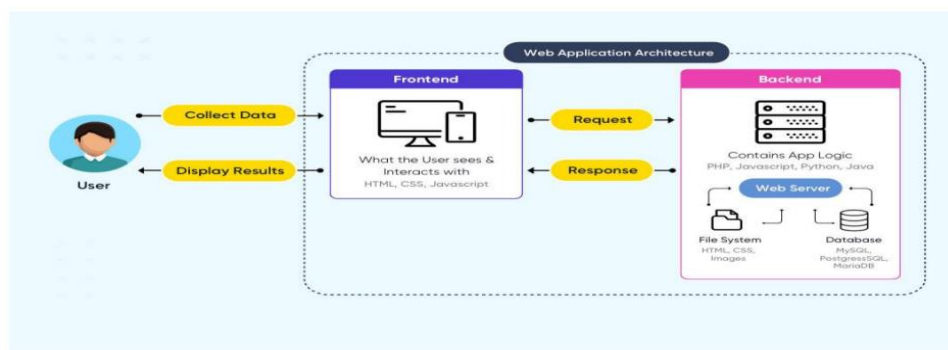


Fig 3.1: System Architecture

ConcentriQ adopts a three-tier architecture that strategically separates its components, fostering a scalable and maintainable framework. At the presentation layer, the frontend, ConcentriQ employs React.js to create a responsive and dynamic user interface. This Single Page Application (SPA) model enhances user interaction, with components structured for modularity and reusability. Chakra UI is integrated to maintain visual consistency and provide customizable UI elements. In the business logic layer, constituted by the backend, Node.js and Express.js handle core functionalities like task management, focus timer, social collaboration, and user authentication. This tier ensures efficient processing of requests, and Socket.io is integrated for real-time communication, particularly in the chat room feature. The data storage layer, the third tier, relies on MongoDB as the NoSQL database. MongoDB's flexibility in handling JSON-like documents aligns well with the dynamic nature of ConcentriQ's data, offering secure and efficient storage for user details, tasks, and collaborative information. Communication between tiers is facilitated by a RESTful API, enabling seamless data exchange between the frontend and backend. This API ensures a smooth flow of information and supports ConcentriQ's robust functionalities. Real-time communication is achieved through WebSocket connections, enhancing collaborative features like the chat room. In essence, ConcentriQ's architecture integrates frontend, backend, and database layers cohesively, delivering a seamless and feature-rich user experience. This three-tier design positions ConcentriQ for scalability and adaptability, crucial for its evolution and enhancement over time.

3.3 FUNCTIONAL REQUIREMENTS

Functional requirements for ConcentriQ outline the specific features and capabilities that the platform must possess to meet user needs and expectations. These requirements focus on the functionalities that users will directly interact with. Here's a description of the functional requirements for ConcentriQ:

Task Management: ConcentriQ's task management feature allows users to create, edit, prioritize, and delete tasks effortlessly. Users can customize their to-do lists, set deadlines, and mark tasks as completed. The system ensures a streamlined and intuitive interface for efficient organization and management of tasks.

Focus Timer: The built-in focus timer is a key functional component of ConcentriQ. Users can set timers for work and study sessions, leveraging the Pomodoro Technique. The timer is embedded with a curated music playlist designed to enhance focus. Additionally, a website blocker feature ensures that users stay focused by restricting access to distracting websites during work sessions.

Social Collaboration: ConcentriQ introduces a social dimension to productivity. Users can connect with friends, engage in friendly competition, and build mutual accountability. The platform allows users to set challenges, share progress, and collaborate with like-minded individuals striving to achieve their goals.

Chat Room:The chat room feature fosters communication and collaboration among users. It provides a space for users to interact, share insights, and motivate each other. This functionality enhances the sense of community within ConcentriQ, promoting a supportive environment for personal and collective growth.

User Authentication and Personalized Dashboards:

To ensure privacy and security, ConcentriQ incorporates user authentication. Users can create accounts, log in securely, and access personalized task management dashboards. This personalized space caters to individual preferences and ensures that user information is kept confidential. These functional requirements collectively contribute to ConcentriQ's goal of providing a comprehensive and user-centric task management platform. By focusing on features like task organization, focus enhancement, social collaboration, and user security, ConcentriQ aims to deliver a well-rounded and effective solution for individuals striving for productivity and goal achievement.

3.4 SYSTEM ANALYSIS

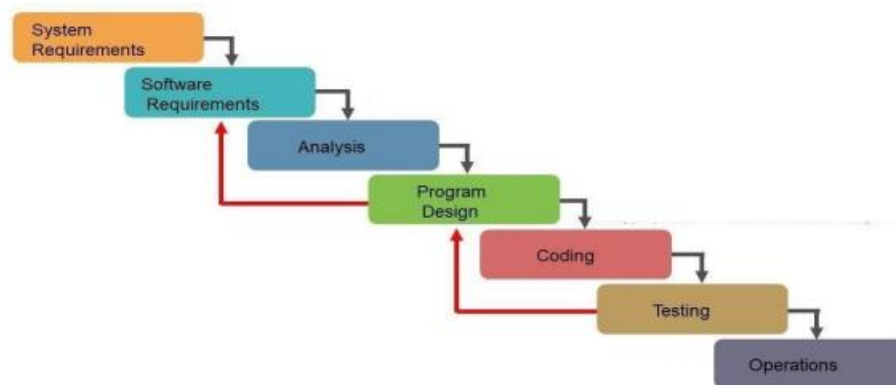


Fig 3.2: System Analysis

Requirements Gathering:

The system analysis begins with an extensive requirements gathering phase. ConcentriQ's development team collaborates with stakeholders to understand and document user needs. This involves identifying features like task management, focus timer, social collaboration, and privacy concerns. Clear and detailed requirements are crucial to guide the subsequent phases of development.

Feasibility Study:

Following requirements gathering, a feasibility study is conducted to assess the practicality and viability of implementing ConcentriQ. This includes evaluating technical, operational, and economic aspects. The feasibility study informs decision-making and ensures that ConcentriQ aligns with both user needs and development capabilities.

System Design:

Once feasibility is established, the system design phase begins. This involves creating detailed specifications for the system's architecture, components, modules, and data structures. For ConcentriQ, this phase outlines the three-tier architecture, the functionalities of the frontend and backend components, and the data storage structure using MongoDB.

Implementation:

In the implementation phase, the ConcentriQ system is built according to the specifications outlined in the system design. Frontend and backend components are developed using technologies like React.js, Node.js, and Express.js. The integration of Socket.io for real-time communication and MongoDB for data storage is executed during this phase.

Testing:

The testing phase involves systematically validating the functionalities and components of ConcentriQ. Unit testing ensures individual modules work as intended, integration testing verifies that components work together seamlessly, and system testing evaluates the overall functionality of the system. This phase ensures that ConcentriQ meets the specified requirements without errors.

Deployment:

After successful testing, ConcentriQ is deployed for public use. This phase involves making the system accessible to users, setting up servers, and ensuring that the platform is stable and performant. Deployment marks the transition from development to user accessibility.

Maintenance:

The final phase of the waterfall model is maintenance. Post-deployment, ConcentriQ undergoes ongoing maintenance to address any issues, implement updates, and enhance features based on user feedback. This ensures the longevity and continuous improvement of the system.

In summary, the system analysis of ConcentriQ using the waterfall model involves a structured progression from requirements gathering to deployment and maintenance. Each phase contributes to the overall development process, ensuring that ConcentriQ aligns with user needs, technical feasibility, and quality standards.

3.5 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements for ConcentriQ outline the qualities and characteristics that are not directly related to specific functionalities but are essential for ensuring the overall performance, usability, and security of the system. Here's a description of the non-functional requirements for ConcentriQ:

Performance: ConcentriQ is expected to demonstrate high performance, ensuring quick response times for user interactions such as task creation, timer setting, and chat room participation. The system should efficiently handle simultaneous user activities without experiencing significant latency.

Scalability: As user numbers grow, ConcentriQ should exhibit scalability, accommodating an increasing user base without compromising performance. This includes the ability to handle a growing database of tasks, user accounts, and collaborative interactions.

Reliability: ConcentriQ should be highly reliable, minimizing system downtimes and ensuring consistent availability for users. Reliability is crucial for users who rely on the platform for task management and time-sensitive activities.

Security: Security is a paramount concern for ConcentriQ. User data, including account information and task details, must be securely stored and transmitted. User authentication and authorization mechanisms should be robust to prevent unauthorized access.

Usability: ConcentriQ aims for a high level of usability, ensuring that users can easily navigate the interface, create tasks, set timers, and engage in social collaboration. The platform should be intuitive, requiring minimal learning curve for users to maximize its effectiveness.

Compatibility: ConcentriQ should be compatible with various web browsers and devices to ensure a seamless user experience for a diverse user base. Compatibility considerations extend to different operating systems and screen sizes.

Maintainability: The system should be designed and implemented in a way that facilitates ease of maintenance. Updates, bug fixes, and feature enhancements should be manageable without causing disruptions to the overall functionality of the platform.

Privacy: Privacy is a critical non-functional requirement. ConcentriQ must adhere to data protection regulations and ensure that user information is handled with the utmost confidentiality. Secure login mechanisms and encrypted communication channels contribute to maintaining user privacy.

Accessibility: ConcentriQ should be designed with accessibility in mind, catering to users with diverse needs. This includes considerations for users with disabilities, ensuring that the platform is navigable and functional for all.

3.6 SOFTWARE REQUIREMENT SPECIFICATION

ConcentriQ, a dynamic task management platform, features a three-tier architecture with a React.js frontend, Node.js and Express.js backend, and MongoDB for data storage. Functionalities include task management, a focus timer, social collaboration, and chat rooms. Non-functional requirements emphasize performance, scalability, reliability, security, usability, and privacy. Software requirements involve React.js, Node.js, Express.js, MongoDB, and Socket.io. Development tools include VSCode and Git. Server management utilizes Nodemon, while bcrypt ensures secure user authentication. ConcentriQ's hardware requirements include a robust server infrastructure, stable network, and compatibility with modern client devices.

3.7 SOFTWARE REQUIREMENTS

Software:

1)Frontend:React.js, Chakra UI

2)Backend:Node.js, Express.js

3)Database:MongoDB

4)Real-Time Communication:Socket.io

5)Development Tools:Visual Studio Code (VSCode),Git

6)Server Management:Nodemon

7)User Authentication:Bcrypt

Operating System: Linux distributions (e.g., Ubuntu, CentOS) for server deployment.

Technology Stack: Three-tier architecture with React.js for the frontend, Node.js and Express.js for the backend, and MongoDB for data storage. Socket.io facilitates real-time communication. The technology stack ensures a scalable, secure, and responsive task management platform.

3.8 HARDWARE SPECIFICATION

Minimum 8GB Ram Laptop, Internet Connection

4. SOFTWARE DESIGN

4.1 UML DIAGRAMS

Application requirements, operating state, application and subsystem functionality, document and repository configuration, input locations, yield types, human-machine interfaces, management justification, and external interfaces are all covered in the Device Architecture Manual. Software developers can express an analytical model through documents that have a large number of syntactic and semantic instructions with the help of the Unified Modelling Language (UML). A UML context is described as five diverse points of view that each show the system in a distinctively different way. The parts resemble modules that can be put together in different ways to form a complete UML diagram. Therefore, understanding the various diagrams is crucial to applying the knowledge in actual systems. Drawing diagrams or pictures of any complex system is the greatest way to comprehend it. These patterns have a greater impact on our comprehension. We can tell from looking around that infographics are not a new idea, but they are commonly used in many different types of organizations in different ways.

USER MODEL VIEW

In the context of system design, the client perspective refers to how the system appears to the end- users, while the exam's illustration provides an example of how the system would be used by actual users. The user view of the system defines how it operates and meets the expectations of the user, offering a glimpse of the system from the user's standpoint.

STRUCTURAL MODEL VIEW

This perspective refers to the software design that represents the features and details of the device. It outlines the static structures and includes activity diagrams, sequence diagrams, and state machine diagrams.

BEHAVIORAL MODEL

View The client model and basic model view of UML focuses on the social dynamics and collaboration between different components of the system. On the other hand, UML Behavioral Diagrams are used to illustrate the dynamic aspects of the system and how they interact over time. This includes diagrams such as interaction diagrams, use case diagrams, activity diagrams, and state-chart diagrams, which provide a visual representation of the system's behavior and functionality.

IMPLEMENTATION MODEL VIEW

In this, the necessary procedures for producing the frame pieces are detailed. Also known as the implementation view, this is. A UML Component diagram is used to describe system Components. One UML diagram used to display the development view is the package diagram.

ENVIRONMENTAL MODEL VIEW

This was how the world in which the program is to be launched was expressed on a systemic and functional level. The environmental view's graphic outlines the software model's post-deployment behavior. Typically, this diagram depicts how users interact with the system and how software impacts it. The environmental model includes the following diagrams: Illustration of deployment. The UML model is made up of two separate domains: 1. An overview of UML Analysis, focusing on the client model and auxiliary model perspectives. 2. Demonstrations use cases, and natural model perspectives are emphasized in UML configuration presentations.

The UML model is made up of two separate domains:

1. An overview of UML Analysis, focusing on the client model and auxiliary model perspectives.
2. Demonstrations use cases, and natural model perspectives are emphasized in UML configuration presentations.

4.1.1 USE CASE DIAGRAM

A use case diagram's goal is to demonstrate a system's dynamic nature. This description is too general to adequately capture the intention, though, as the goal of the other four images is the same. We'll examine what makes it unique compared to the other four diagrams. Use case diagrams are used to compile the requirements for a system, taking into account numerous variables. Most of these requirements are design requirements. As a result, when studying a system to gather its functions, use cases are built and actors are identified.

Use case diagrams are made to represent the outside view once the primary task is completed. In conclusion, use case diagrams are useful for the following purposes:

1. System requirements are collected using this form.
2. Used to get a bird's-eye view of a system.
3. Determine various factors that are influencing the system.
4. Display the interaction of the requirements as actors.

The analysis of a system's high-level requirements is done using use case diagrams. When the requirements are examined, use cases are created to document a system's functionality. The term "use cases" refers to "system functionalities written in a logical order." The second crucial pillar of use cases is the actors. Actors are any entities that communicate with the system. Internal applications, human users and external applications can all be actors. The following factors should be kept in mind when constructing a use case diagram.

1. As a use case, functionalities will be represented.
2. Actors.
3. Relationships between use cases and actors.

USE CASES

A use case is a written explanation of how visitors will complete tasks on your website. It defines how a system responds to a request from the user's point of view. Each use case is defined by a sequence of essential activities that begin with the user's objective and finish when that goal is met.

Construction of Use-case:

Outlines provide a graphic illustration of the framework's behavior. These graphs demonstrate how the framework is used effectively when viewed from the viewpoint of an untouchable (actor). A utilization case graph may show all or only some of the work instances for a framework.

A use-case diagram may include the following elements:

- i. Actors.
- ii. Use cases.

RELATIONSHIPS IN USE CASES

Use case diagrams commonly depict active relationships, also referred to as behavioral relationships, as a sort of interaction. Include, Communicate, Generalize, and Extend are the four main behavioral connection kinds.

COMMUNICATES

An actor and a use case are connected by a behavioral relationship. Keep in mind that the usecase's goal is to assist the system's actor in some way. It is crucial to record these interactions between actors and use cases as a result. An actor and a use case are linked by a line without any arrowheads.

INCLUDES

The circumstance in which a use case contains behavior that is shared by several use cases is referred to as the includes relationship (also known as the uses connection). In other words, the additional use cases include the common use case. The typical use case is pointed to by a dotted arrow, signifying the included relationship.

EXTENDS

When one use case has behavior that enables another use case to handle a variation or exception from the core use case, this is known as an extended link. Exceptions to the fundamental use case

are handled by a different use case. The basic and extended use cases are connected by the arrow.

GENERALIZES

One thing is more common than another, according to the generalized connection. This connection may exist between two actors or two use cases. In the UML, the arrow directs attention to a "thing" that is more general than another "thing".

In this system Use Case diagram:

Use-case diagrams serve as a visual representation of the high-level functionalities and boundaries of a system. These diagrams are instrumental in illustrating the interactions between the system and its actors. The use cases and actors depicted in these diagrams provide a comprehensive overview of the system's operations and how various actors utilize its features.

In the provided use-case diagram, users have the capability to perform a range of actions, each represented as a use case. These actions include adding a task, deleting a task, playing music, and entering personal details. Each use case outlines a specific functionality or task that the system supports. Actors, which can be individuals, other systems, or external entities, are also identified to showcase the external elements interacting with the system.

For example, the "Add Task" use case signifies that users can input new tasks into the system. The "Delete Task" use case indicates the ability to remove tasks from the system. "Play Music" suggests a feature allowing users to play music within the system. Additionally, the "Add Personal Details" use case implies that users can input and update their personal information.

In summary, use-case diagrams provide a bird's-eye view of the system's capabilities, interactions, and the ways in which various actors engage with the system's features.



Fig - 4.1 Use-Case Diagram

4.1.2 SEQUENCE DIAGRAM

A sequence diagram is a type of UML diagram that illustrates the interactions between objects in a system over time. It depicts the flow of messages between objects and the order in which those messages are sent. The diagram uses vertical lifelines to represent objects and horizontal arrows to represent messages. Sequence diagrams are useful for understanding the behavior of a system and can be used for design, testing, and documentation purposes.

Make the diagram to show:

- i. Explains the details of a UML use case.
- ii. Make a logical model of a complex method, function, or operation.
- iii. Examine how objects and components interact with one another in order to complete a process.

Plan and comprehend the specific functionality of a current or future scenario. The following scenarios lend themselves well to the use of a sequence diagram: A utilization scenario is a diagram that illustrates potential uses for your technology in the future.

Making sure you've considered every conceivable system usage scenario is a great strategy.

Method logic:

A UML sequence diagram, like a use case diagram, may be used to examine the logic of any function, technique, or complex process.

If you conceive of a service as a high-level approach utilized by many customers, a sequence diagram is an excellent way to write out service logic.

Object: An object has a lead, a personality, and a state. In their fundamental class, items that are practically identical are represented in terms of their structure and orientation. A specific instance of a class is represented by each object in a diagram. An object is an order case.

Message: A message is when two articles communicate information to bring about an occurrence. From the source point of control convergence to the objective point of control convergence, information is transmitted via a message.





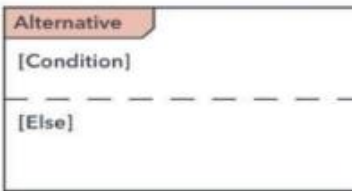

Link: A connection between two items, including their class, suggests a connection between the classes that are in opposition to one another. Use the image's hover adjustment if an object starts to associate with itself. **Lifeline:** As it descends, it reflects the passing of time. This vertical dashed line represents the events that happen in order to an item during the monitoring process. A lifeline could begin with a predetermined rectangle shape or an actor symbol.

Actor: Entities which communicate with the system or are external to the system are displayed. **Message Symbols** **Synchronous message:** This is represented by an arrowhead and a solid line. This symbol is used when a sender must wait for a response to a query before proceeding. The graphic should depict both the request and the answer.

Asynchronous message: This is shown as a solid line with a lined arrowhead. A response is not required for asynchronous communications in order for the sender to continue. Only the call should be shown in the diagram.

Delete message: An X follows a solid line with a solid arrowhead. This message has the effect of causing an object to be destroyed.

Sequence diagram components:

Name	Description	Symbol
Object symbol	In UML, it represents a class or object. The object symbol depicts an item's behavior within the framework of the system. It is inappropriate to list class attributes in this format.	
The activation box	Depicts the amount of time required for an object to finish a task. The activation box gets longer the longer it will take to complete the assignment.	
Actor symbol	Displays entities that are external to the system or have interactions with it.	
Lifeline symbol	Represents time passing by extending downward. The consecutive events that affect an object during the charting process are depicted by this vertical dashed line. Lifelines may start with an actor symbol or a designated rectangle form.	
Alternative symbol	Symbolizes a decision between two or more message sequences (which are typically mutually exclusive). Use the marked rectangle shape with a dashed line within to symbolize options.	
Message symbol	When a sender needs to transmit a message, they <u>utilise</u> this symbol.	

In our project the Sequence diagram is:

A sequence diagram is a visual representation used in system modelling to depict the chronological order of interactions between objects or components within a system. It shows how messages and calls are exchanged between these elements over time. The diagram shows sequential order from user logging in and then authenticate from database, after logging in the user interact with website to select a desired function, diagram shows selection of pomodoro method and further option of website blocker is turned on.

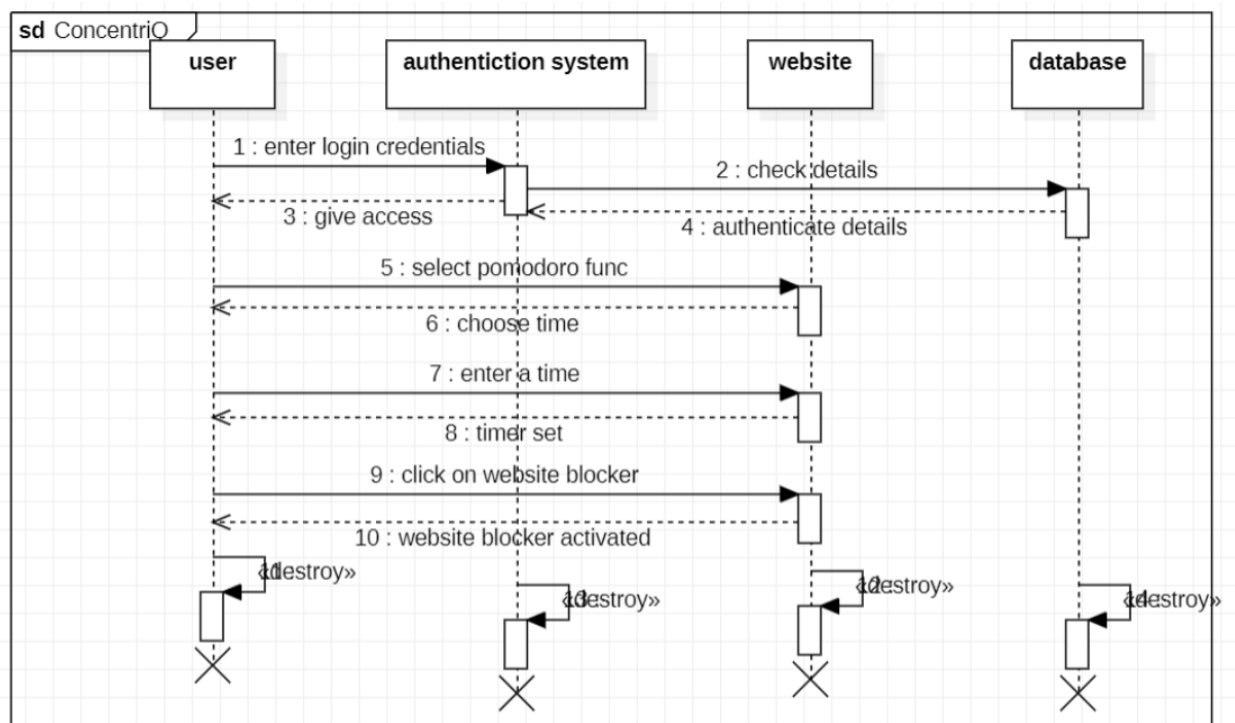


Fig-4.2 Sequence Diagram

4.1.3 ACTIVITY DIAGRAM

An activity diagram is a flowchart that displays the movement of information from one action to the next. A system operation can be used to describe the activity. The control flow is led from

one operation to the next. This flow in nature could be sequential, branched, or concurrent. Activity diagrams handle various types of flow control by utilizing numerous sections, such as join, fork, and so forth. The basic functions of the other four diagrams are also provided by activity diagrams. It captures the system's dynamic behavior. Message flow from one item to the next is shown in the other four diagrams, but not in the activity diagram.

An activity is a particular type of system operation. There is no evidence of communication between one activity and the next. Flowcharts and activity diagrams are frequently used in the same sentence. The diagrams are not flowcharts, despite their appearance.

Notations

Initial point or start point

Any activity diagram's beginning action state or starting point is represented by a small, filled circle and an arrow. Make sure the top left corner of the first column is where an activity diagram with swim lanes should begin.

Activity or Action state

An action state is a representation of an object's non-interruptible action.

In StarUml, you can create an action state by drawing a rectangle with rounded corners.

Action flow

Transitions from one action state to another are depicted by action flows, also known as edges and routes. An arrowed line is commonly used to depict them.

Decisions and branching

A diamond denotes a pick from several options. When one activity requires a choice before continuing, place a diamond between them. The outgoing alternates should be labeled with a condition or guard expression. Another way to name one of the pathways is "else."

In our project, the activity diagram is:

Activity diagrams depict the flow of actions within a system, illustrating the sequence of activities and their dependencies. They are part of the (UML) and are particularly useful for visualizing business processes, workflow, or system behavior. The activity diagram that has the following states as authentication then functions like to do list,Pomodoro,add,delete,and edit a task,completion of the task,set timer duration,playmusic,turn on website blockers.Fork that is used to divide a single task into multiple tasks and join that is used for combining multiple sates into a single state.It has three swim lanes namely user,database and website.

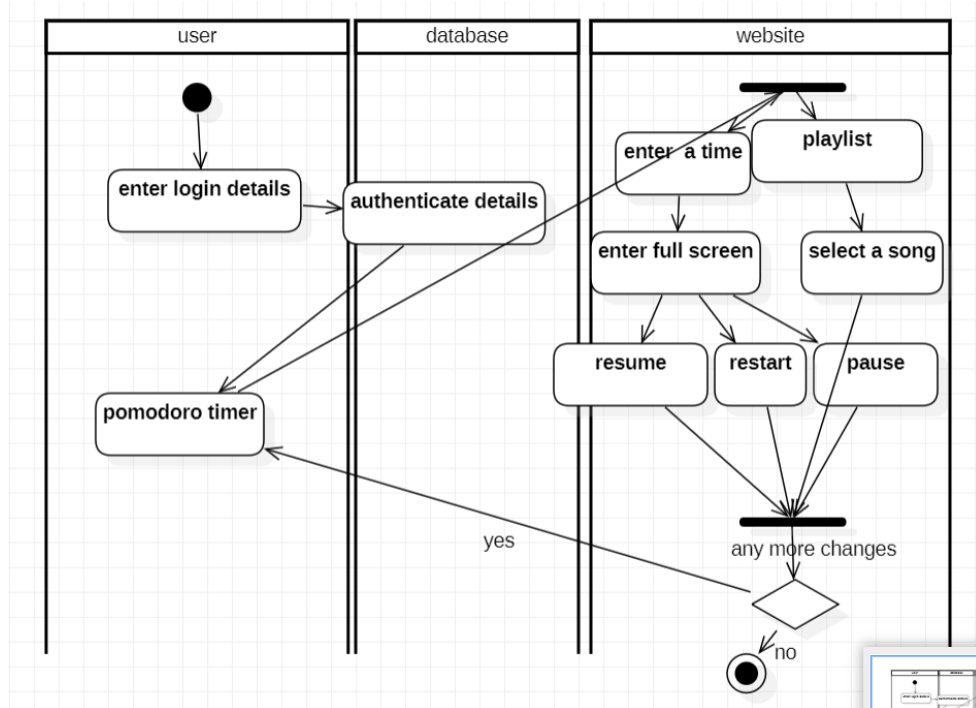


Fig-4.3 Activity Diagram for Pomodoro Timer

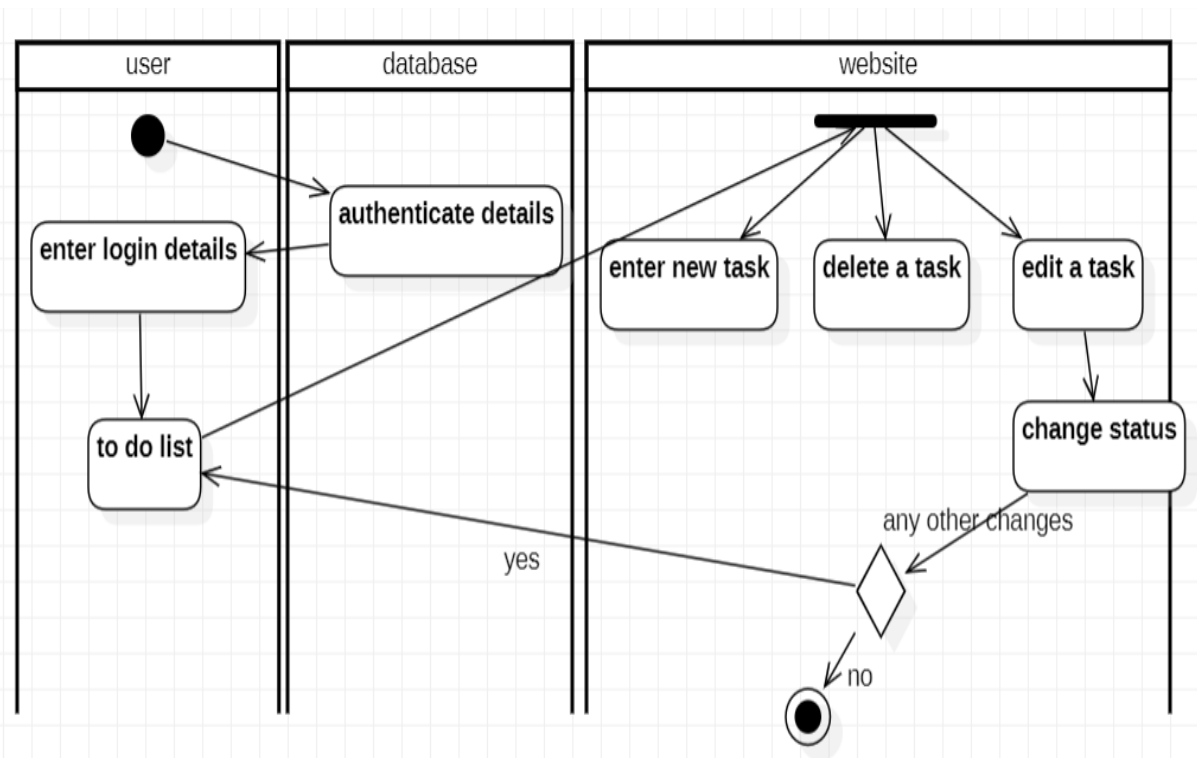


Fig-4.4 Activity Diagram for Task Manager

4.1.4 CLASS DIAGRAM

A class diagram is a visual representation of the structure and relationships among classes in a system.

CLASS:

Definition: A blueprint or template for creating objects. It defines the attributes (data members) and behaviors (methods) common to all objects of a certain type.

Attributes: Definition: Properties or characteristics of a class that describe the state of objects belonging to that class.

METHODS:

Definition: Functions or operations associated with a class that define its behavior. Methods manipulate the state of the object or provide some functionality.

ASSOCIATION:

Definition: Represents a relationship between two classes, indicating that objects of one class are connected with objects of another class. It can be one-to-one, one-to-many, or many-to-many.

AGGREGATION:

Definition: A special form of association that represents a "whole part" relationship, where one class is a container for another class. The aggregated class can exist independently.

COMPOSITION:

Definition: Similar to aggregation but with a stronger association, indicating that the composed class is part of the composite class. The composed class cannot exist independently of the composite class.

INHERITANCE:

Definition: Represents an "is-a" relationship between classes. It allows a subclass to inherit attributes and behaviors from a superclass, promoting code reuse and establishing a hierarchy.

DEPENDENCY:

Definition: Denotes a relationship where one class depends on another class, usually in terms of method calls. Changes in one class may impact on the other class.

INTERFACE:

Definition: A contract that defines a set of methods that a class must implement. It abstracts the behavior of a class without specifying the implementation details.

MULTIPLICITY:

Definition: Indicates the number of instances of one class related to a single instance of the other class in an association

In our project the Class diagram is:

A class diagram that defines and provides the overview and structure of a system in terms of classes, attributes and methods, and the relationships between different classes. The class diagram for the website concentriq comprises the main classes as 'user','pomodoro','to do list','friends'.Task is taken as the child of 'to do list'.All the classes are associated with each therefore association relationship is defined.As task is the child there is generalization relation between 'to do list' and the 'task' that defines the inheritance concept.Both 'pomodoro' and 'friends' are dependent on the task hence dependency relationship is defined.

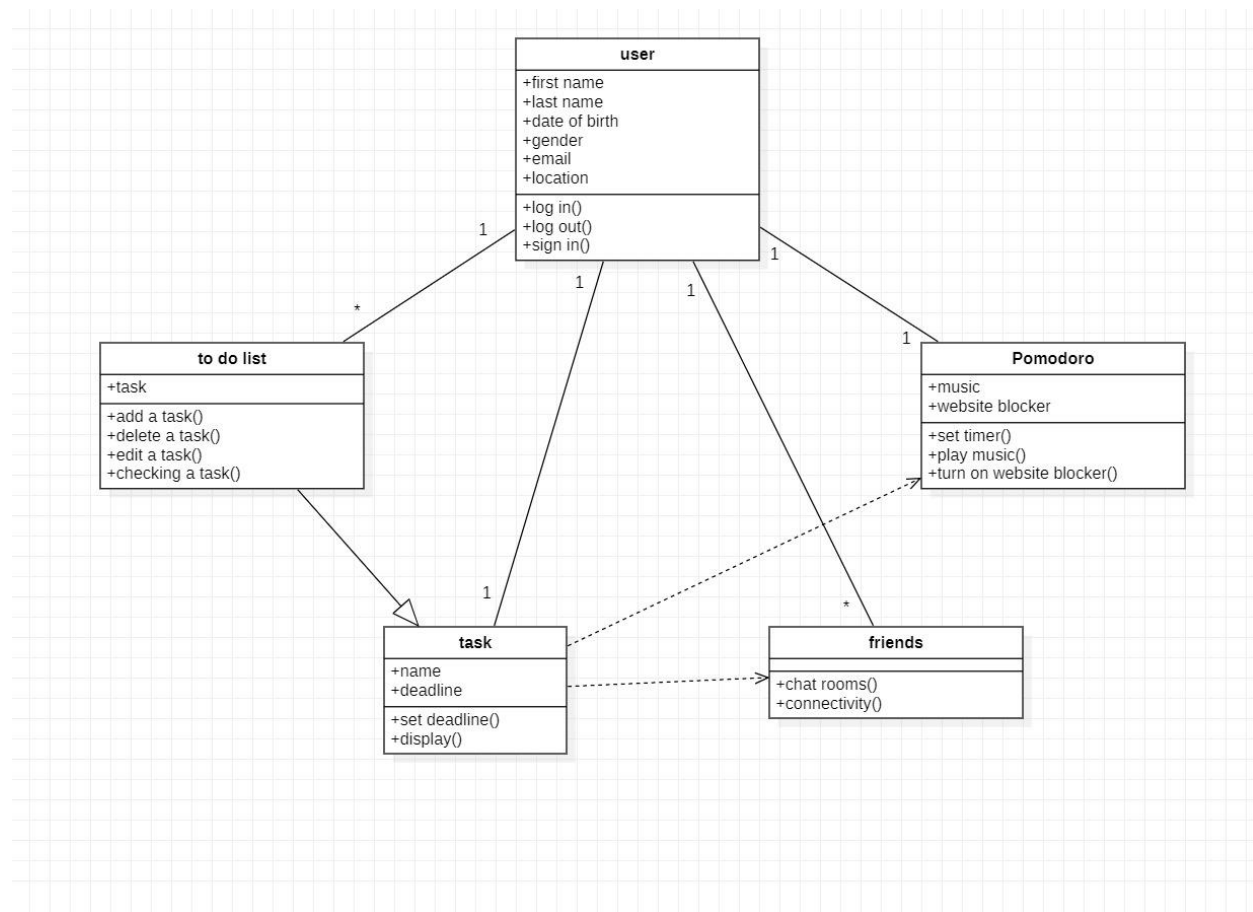


Fig- 4.5 Class Diagram

4.1.5 STATECHART DIAGRAM

A state chart diagram, a type of behavioral diagram in UML (Unified Modeling Language), depicts the various states an object can be in and how it transitions between these states in response to events. State chart diagrams are particularly useful for modeling the dynamic behavior of systems over time.

Notations

State Represented by rounded rectangles, each state represents a condition or situation of an object.

Transition

Represented by arrows between states, transitions depict the movement from one state to another. They are triggered by events.

Initial State

Denoted by a filled circle, it represents the starting point of the state chart.

Final State

Denoted by a filled circle with a dot inside, it represents the end point of the state chart.

Event

Trigger for a state transition, typically shown as a labeled arrow between states.

Guard Condition

A condition that must be true for a transition to occur. It is often written next to the transition arrow within square brackets.

Composite State

A state that contains nested states within it. Represented by a small black rectangle drawn on the side of the state.

In our project ,theStatechart diagram is:

Statechart diagram describes the flow of control from one state to another state. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.

The state chart diagram that has the following states as login,to do list,Pomodoro,add,delete,and edit a task,completion of the task,set timer duration,playmusic,turn on website blockers.Fork that is used to divide a single task into multiple tasks and join that is used for combining multiple sates into a single state.

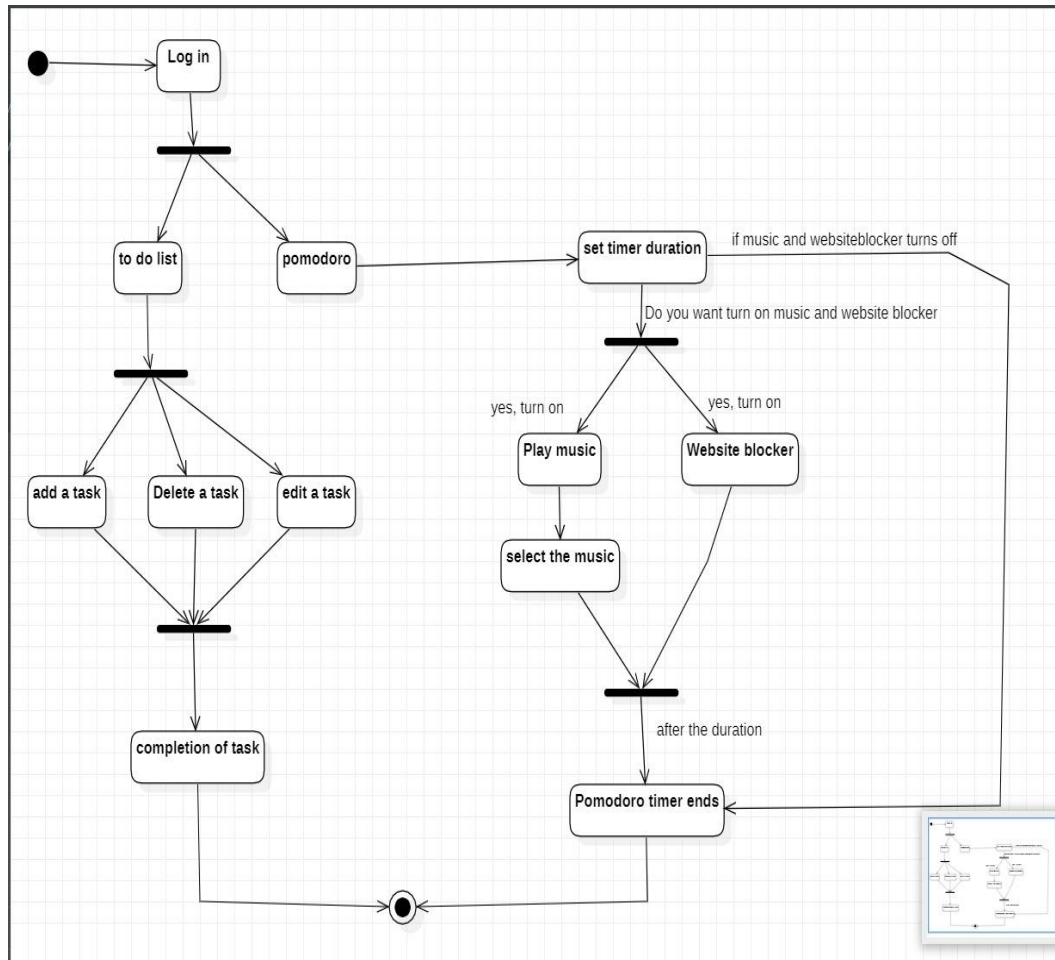


Fig-4.6 Sequence Diagram

4.1.6 COMMUNICATION DIAGRAM

A communication diagram, also known as a collaboration diagram, is a type of interaction diagram that visualizes the interactions between objects or components within a system. It illustrates how various elements communicate with each other to accomplish a specific task or process. The Unified Modeling Language (UML) provides standardized notations for creating communication diagrams.

Notations

Objects

Represented by rectangles with the object's name at the top. Objects are instances of classes or components involved in communication.

Lifelines

Vertical dashed lines extending from an object, representing the object's lifespan during the interaction.

Messages

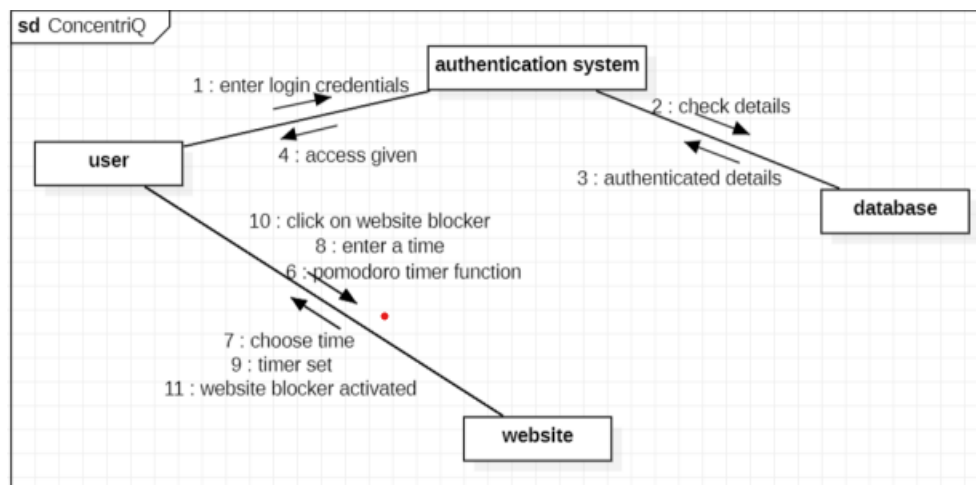
Arrows connect lifelines, indicating communication or interaction between objects. Messages can be synchronous or asynchronous.

Association Relationships

Dashed lines connecting objects, representing relationships between objects. These lines can have labels to describe the association.

In our Project ,the Communication Diagram is:

A communication diagram is a visual representation used in system modeling to illustrate interactions between components or objects within a system. It employs arrows and labels to depict the flow of messages or information, facilitating the understanding of how elements collaborate to achieve specific tasks. There are four life lines: user, website, authentication system, and database. The diagram shows how each lifeline communicates to activate the pomodoro timer and website blocker.



4.7 Communication Diagram

4.1.7 COMPONENT DIAGRAM

A component diagram in UML (Unified Modeling Language) provides a visual representation of the high-level components within a system and their interactions. It illustrates how various software components, such as classes, interfaces, and packages, are organized and connected in a system.

Notations

Component Represented by rectangles with two tabs at the top, a component represents a modular, deployable, and replaceable part of a system. It can encapsulate classes, interfaces, or other components.

Interface

Represented by a small circle attached to a component, an interface specifies a contract that a component or a class must adhere to.

Dependency Relationship

Represented by a dashed arrow with an open arrowhead, a dependency relationship indicates that one component depends on another. It may be a runtime dependency or a design-time dependency.

Association Relationship

Represented by a solid line connecting two components, an association relationship signifies a connection between components. It may represent a more structural relationship compared to dependency.

In our project the Component Diagram is:

A component diagram, also known as a UML component diagram, describes the organization and wiring of the components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development. Here, in the given component diagram, there are various components that typically describe the functions software units in our project such as dialog.dll, toDo.html, pomodoro.html, etc.

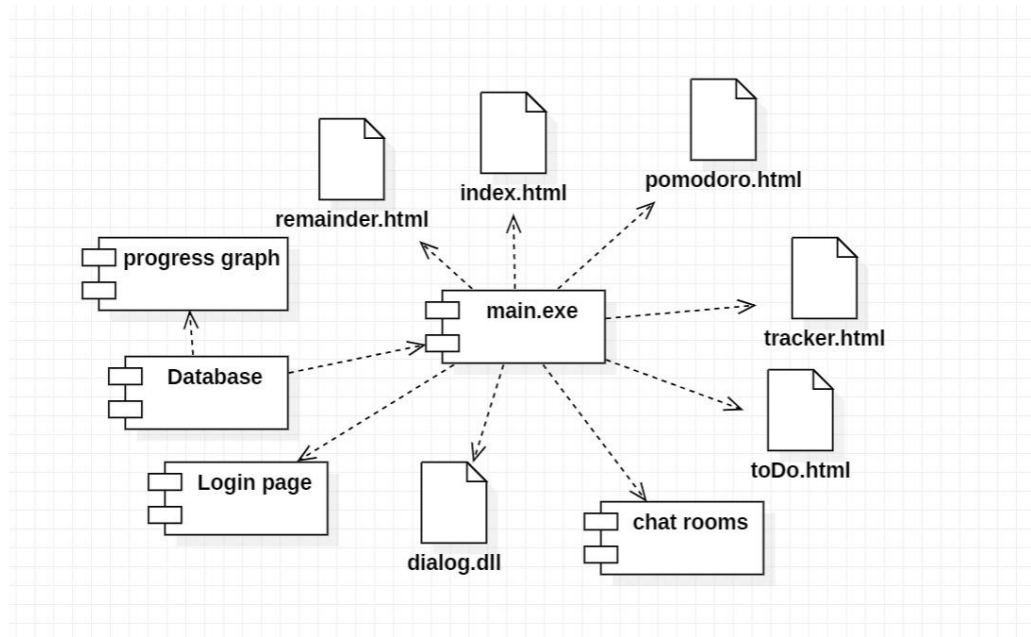


Fig-4.8 Component Diagram

4.1.8 DEPLOYMENT DIAGRAM

A deployment diagram in UML (Unified Modeling Language) provides a visual representation of the physical deployment of software components in a system, illustrating how software and hardware elements interact in a distributed environment.

Notations

Node

Represented by a box, a node can be a hardware device, such as a server or a PC, or a software execution environment, such as a database server or application server.

Component

Represented by rectangles with two tabs at the top, a component represents a modular, deployable, and replaceable part of a system. It can encapsulate classes, interfaces, or other components.

Artifact

Represented by rectangles with a folded corner, artifacts represent physical files or libraries associated with components. They can also be placed inside nodes to show their deployment.

Dependency Relationship

Represented by a dashed arrow with an open arrowhead, a dependency relationship indicates that one component depends on another. It may be a runtime dependency or a design-time dependency.

Association Relationship

Represented by a solid line connecting two components, an association relationship signifies a connection between components. It may represent a more structural relationship compared to dependency.

In our project, the Deployment diagram is:

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths. Here , mainly the nodes used are the User PC , Web Server and the DB Server. They also contain various artifacts such as the HTML% and the php files.

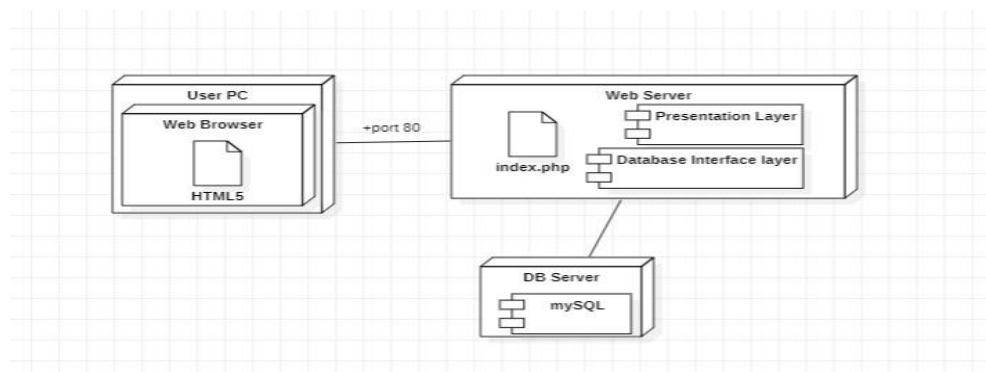


Fig-4.9 Deployment Diagram

5. PROPOSED SYSTEM

5.1 METHODOLOGY

1. Requirement Analysis:

User Stories and Use Cases: Gather user stories and define use cases to understand the functionalities and features required.

Stakeholder Interviews: Conduct interviews with potential users to gather insights and expectations.

Functional and Non-functional Requirements: Clearly document functional and non-functional requirements, considering features like task management, focus timer, collaboration, and security.

2. Technology Selection:

Frontend:

HTML, CSS, JavaScript for the core structure and styling.

React for building a responsive and dynamic user interface.

Chakra UI for a modern and accessible UI component library.

Backend:

Node.js for server-side development.

Express.js for building the RESTful API.

Socket.io for real-time communication in the chat room.

Bcrypt for password hashing.

Development Tools:

VSCode as the integrated development environment.

Postman for API development and testing.

Nodemon for automatic server restart during development.

3. User Interface Design:

Wireframing and Mockups: Create wireframes and interactive mockups to visualize the user interface and user experience.

Responsive Design: Ensure the design is responsive, providing a seamless experience across devices.

Accessibility: Implement accessible design principles using Chakra UI.

4. Development:

Frontend Development:

Implement the user interface using HTML, CSS, and React.

Integrate Chakra UI components for a consistent and visually appealing design.

Backend Development:

Develop server-side logic using Node.js and Express.js.

Implement RESTful API endpoints for tasks, timers, and user interactions.

Integrate Socket.io for real-time communication in the chat room.

Database Integration:

Utilize MongoDB for efficient data storage and retrieval.

5. Search Engine Visibility:

SEO-Friendly Content: Ensure content is structured with relevant keywords, titles, and descriptions for better search engine visibility.

Sitemap Submission: Submit a sitemap to search engines to help them index and understand the structure of your website.

Responsive Design: Optimize the website for mobile devices, as search engines prioritize mobile-friendly websites.

Content Quality: Focus on creating high-quality, valuable content that aligns with user intent and search queries.

6. Communication and Security:

Real-time Communication:

Implement Socket.io for real-time updates and messaging in the chat room.

Security Measures:

Use Bcrypt for password hashing and storage.

Implement secure data transmission using HTTPS.

Apply user authentication and authorization.

7. Testing:

Unit Testing: Conduct unit testing for individual components and functions.

Integration Testing: Test the integration of frontend and backend components.

User Acceptance Testing (UAT): Engage users to perform UAT to ensure the system meets their expectations.

8. Deployment:

Server Deployment: Host your Node.js server on the chosen hosting platform. Options include services like DigitalOcean, Linode, or your preferred server hosting provider.

Database Hosting: For MongoDB, consider using cloud-based solutions like MongoDB Atlas or any other provider that aligns with your requirements.

Frontend Deployment: Choose a hosting service that suits your needs. You may consider platforms like GitHub Pages or GitLab Pages for simplicity.

Continuous Deployment: Implement continuous deployment through your version control system (e.g., Git) and a tool like Nodemon for automatic server restarts during development.

9. Monitoring and Maintenance:

Implement monitoring tools to track system performance and identify potential issues. Schedule regular updates for feature enhancements and security patches.

5.2 FUNCTIONALITIES

1. To-Do List Management (Task Manager):

Create, Prioritize, and Manage Tasks: Enjoy a user-friendly interface that empowers you to create, prioritize, edit, and delete tasks effortlessly.

Customizable Options and Deadlines: Tailor your task list with customizable options and deadlines, ensuring a personalized and organized approach to your goals.

2. Built-in Focus Timer:

Structured Work and Study Sessions: Boost your productivity with ConcentriQ's built-in focus timer. Structure your work or study sessions effectively to enhance concentration and efficiency.

Website Blocker: Say goodbye to distractions! The focus timer comes with a website blocker, allowing you to enter full-screen mode and block access to distracting websites during your focused sessions.

3. Compete with Friends and Mutual Accountability:

Friendly Competition: Turn productivity into a game! Compete with friends on ConcentriQ, setting challenges and spurring each other to achieve more.

Mutual Accountability: Build mutual accountability by sharing your goals and progress with friends. Encourage each other to stay on track and celebrate collective successes.

4. Privacy and Personalized Dashboards:

Secure Account Creation: ConcentriQ prioritizes your privacy. Create your account with confidence, knowing that your information is handled securely.

Personalized Task Management Dashboard: Access a personalized dashboard for managing your tasks. ConcentriQ ensures that your data is in safe hands, providing a secure and private task management experience.

5. Pomodoro Timer with Music Playlist:

Structured Time Management: Use the Pomodoro timer to break your work into focused intervals, boosting productivity and preventing burnout.

Embedded Music Playlist: Enhance your concentration with a curated music playlist embedded within ConcentriQ. Enjoy the synergy of focused work and immersive music.

6. Website Blocker in Full-Screen Mode:

Distraction-Free Focused Sessions: Activate the focus timer in full-screen mode to ensure a distraction-free environment. ConcentriQ's website blocker restricts access to distracting websites, allowing you to stay fully engaged in your tasks.

7. Chat Room for Enhanced Connectivity:

Connectivity Among Learners: ConcentriQ goes beyond individual productivity. Engage with a chat room to foster connectivity among learners and friends. Collaborate, share insights, and motivate each other towards collective success.

ConcentriQ is your all-in-one solution for effective task management, productivity enhancement, and building a supportive community of learners and achievers. Experience a seamless blend of features that cater to your unique needs and preferences.

5.3 ADVANTAGES OF OUR PROPOSED SYSTEM:

ConcentriQ presents several advantages over existing systems such as Google Calendar, Any.do, and Brain.fm, offering a comprehensive and integrated solution for users. Here's a breakdown of the advantages:

1. All-in-One Task Management:

Advantage (ConcentriQ): ConcentriQ stands out as an all-in-one platform, combining task management, focus timers, and collaboration features in a single interface.

Comparison: While other systems like Google Calendar and Any.do primarily focus on task scheduling, ConcentriQ provides a more holistic approach to productivity.

2. Integrated Focus Timer with Website Blocker:

Advantage (ConcentriQ): The built-in focus timer, combined with the website blocker in full-screen mode, offers a unique advantage. Users can enhance concentration and productivity by eliminating distractions during focused work sessions.

Comparison: This integrated feature sets ConcentriQ apart from systems like Google Calendar and Any.do, which lack dedicated focus timer functionalities.

3. Friend Collaboration and Accountability:

Advantage (ConcentriQ): The friend collaboration feature fosters friendly competition and mutual accountability, creating a supportive community to achieve shared goals.

Comparison: Unlike Google Calendar and Any.do, which are primarily individual-focused, ConcentriQ emphasizes social connectivity, making productivity a collaborative and enjoyable experience.

4. Pomodoro Timer with Embedded Music Playlist:

Advantage (ConcentriQ): The Pomodoro timer, coupled with an embedded music playlist, offers a unique approach to time management, providing users with a structured and immersive work environment.

Comparison: This feature distinguishes ConcentriQ from existing systems like Google Calendar and Any.do, which lack integrated Pomodoro timers and music functionalities.

5. Secure and Personalized Dashboard:

Advantage (ConcentriQ): ConcentriQ prioritizes user privacy with secure account creation and offers a personalized task management dashboard.

Comparison: While Google Calendar and Any.do also provide secure access, ConcentriQ emphasizes a more personalized and user-centric experience.

6. Chat Room for Community Building:

Advantage (ConcentriQ): The chat room feature promotes connectivity among learners and users, fostering a sense of community and collaboration.

Comparison: Google Calendar and Any.do lack dedicated social features, focusing primarily on individual task management rather than community-building aspects.

7. Enhanced User Control and Customization:

Advantage (ConcentriQ): ConcentriQ provides users with greater control over tasks, timers, and collaboration settings, allowing for a highly customizable experience.

Comparison: While Google Calendar and Any.do offer customization options, ConcentriQ's holistic approach and versatility set it apart.

In summary, ConcentriQ's advantages lie in its holistic integration of task management, focus timers, collaboration features, and community-building aspects, making it a standout choice for users seeking a comprehensive productivity solution.

6. TESTING

Testing is mostly used in web development to assess the effectiveness.

Primary goals are to ensure:

1. Quality Assurance
2. Detect bugs and flaws

6.1 NON-FUNCTIONAL TESTING

Non-functional testing for ConcentriQ involves evaluating aspects of the system that are not related to specific behaviors or functions but rather focus on performance, usability, security, and other quality attributes. Here's a breakdown of non-functional testing considerations for ConcentriQ:

6.1.1 PERFORMANCE TESTING:

Performance testing is essential to ensure ConcentriQ functions optimally under various scenarios. Load testing evaluates the system's responsiveness under normal and peak loads, while stress testing assesses stability under extreme conditions. Scalability testing measures the system's ability to handle increasing loads as user numbers grow.

6.1.2 USABILITY TESTING:

Usability testing focuses on the user experience. It involves evaluating the intuitiveness and responsiveness of the user interface (UI) across different devices and screen sizes. Accessibility testing ensures compliance with accessibility standards, making the application usable for individuals with disabilities. Navigation testing evaluates the ease of use and overall user experience.

6.1.3 SECURITY TESTING:

Security testing is crucial for safeguarding user data. Authentication and authorization mechanisms are tested for their effectiveness. Data encryption protocols are scrutinized to ensure sensitive information is secure during transmission and storage. Vulnerability scanning is performed to identify and address potential security risks.

6.2 FUNCTIONAL TESTING

Functional testing is a software testing technique that checks if a software system fulfils its functional requirements and specifications. It concentrates on testing the system's features and functionalities to ensure that they operate as intended. Functional testing can be done manually or using automated tools, and can include black-box, white-box, and gray-box testing. The objective of functional testing is to ensure that the software system meets the end-user's requirements and expectations.

6.2.1 UNIT TESTING:

Unit testing involves evaluating individual components or units of code in isolation to confirm their correctness. For ConcentriQ, this means assessing each function, module, or feature independently to ensure they produce the expected output. Unit tests are crucial for identifying and fixing errors at an early stage of development.

6.2.2 INTEGRATION TESTING:

Integration testing assesses the interaction and cooperation between different components of ConcentriQ. It verifies that various modules and features work seamlessly together as an integrated system. Integration testing ensures that data flows correctly between components and that the application as a whole functions cohesively.

6.2.3 REGRESSION TESTING:

Regression testing is essential for ensuring that new code changes or feature additions do not adversely affect existing functionalities. As ConcentriQ evolves, regression testing helps identify any unintended side effects on previously tested features. It ensures that the application maintains overall stability and functionality across updates.

7.RESULTS

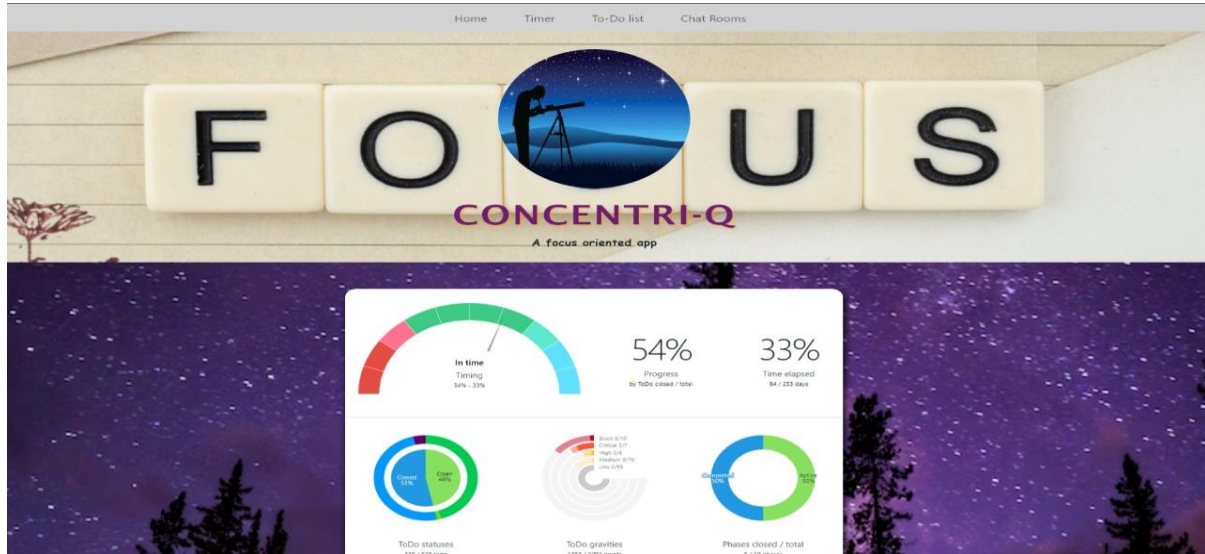


Fig-7.1 Home Page

The screenshot shows the app interface with a dark, starry night sky background. The top section displays the same dashboard as Fig-7.1. Below the dashboard, there are two main sections: "Pomodoro Timer" and "Task Manager".

Pomodoro Timer

The Pomodoro Technique is a time management method that uses a timer to break down work into intervals, traditionally 25 minutes in length, separated by short breaks. The method is named after the Italian word for "tomato" because of the tomato-shaped kitchen timer that the inventor used.

[Visit Pomodoro Timer >>](#)

Task Manager

A task manager is a tool that helps you organize and keep track of your tasks and activities. It allows you to create, update, and prioritize tasks, making it easier to manage your workload and stay productive.

[Visit Task Manager >>](#)

7.2 View options

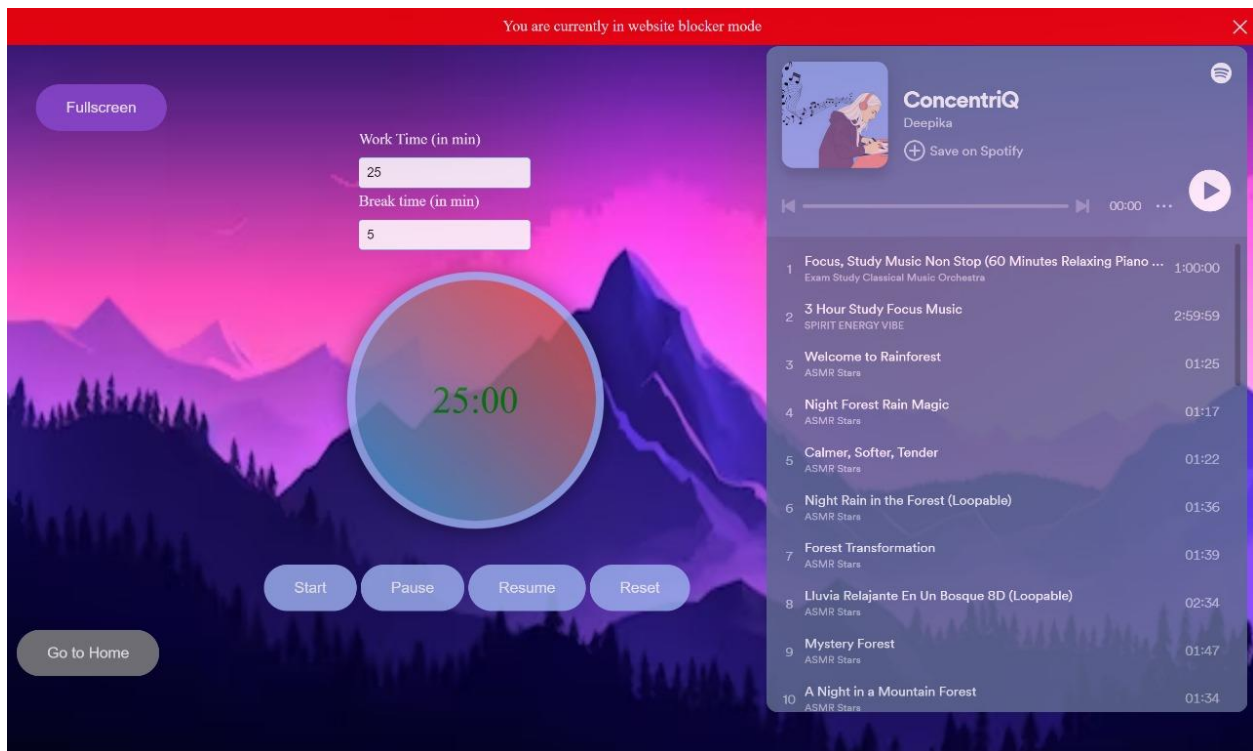


Fig 7.3 Open Pomodoro timer (can only be enabled in full screen mode)

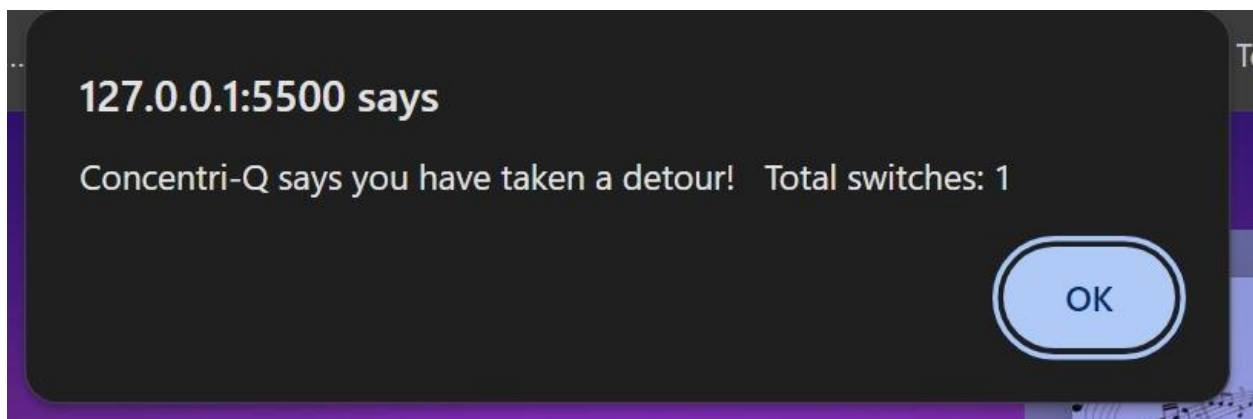


Fig 7.4 Updating the tab switch count

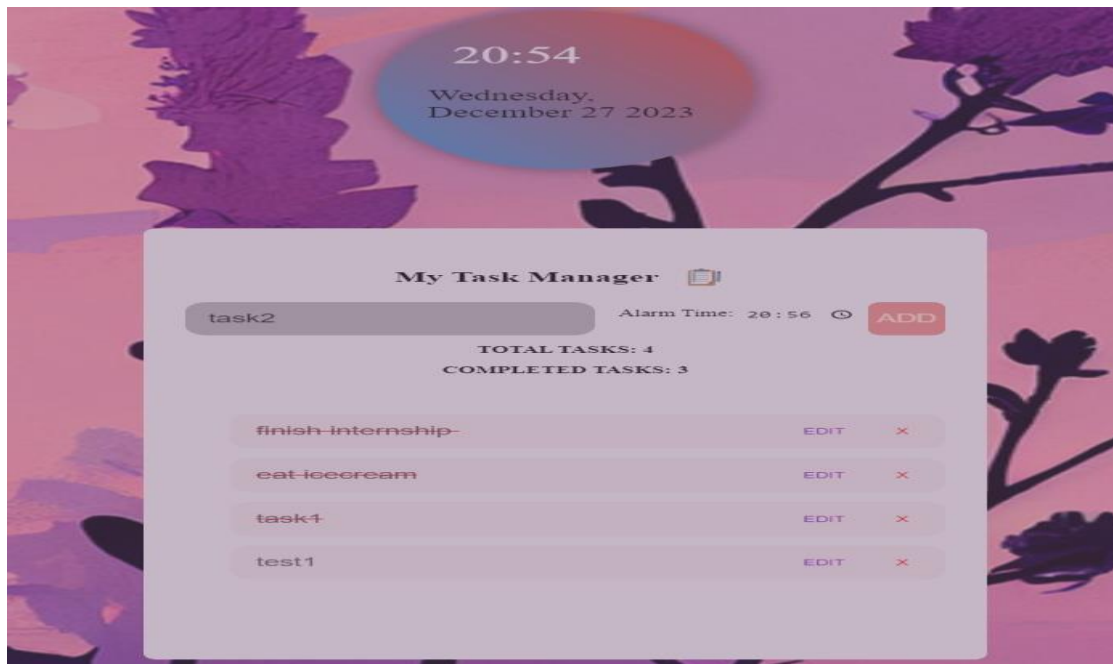


Fig 7.5 Task Manager

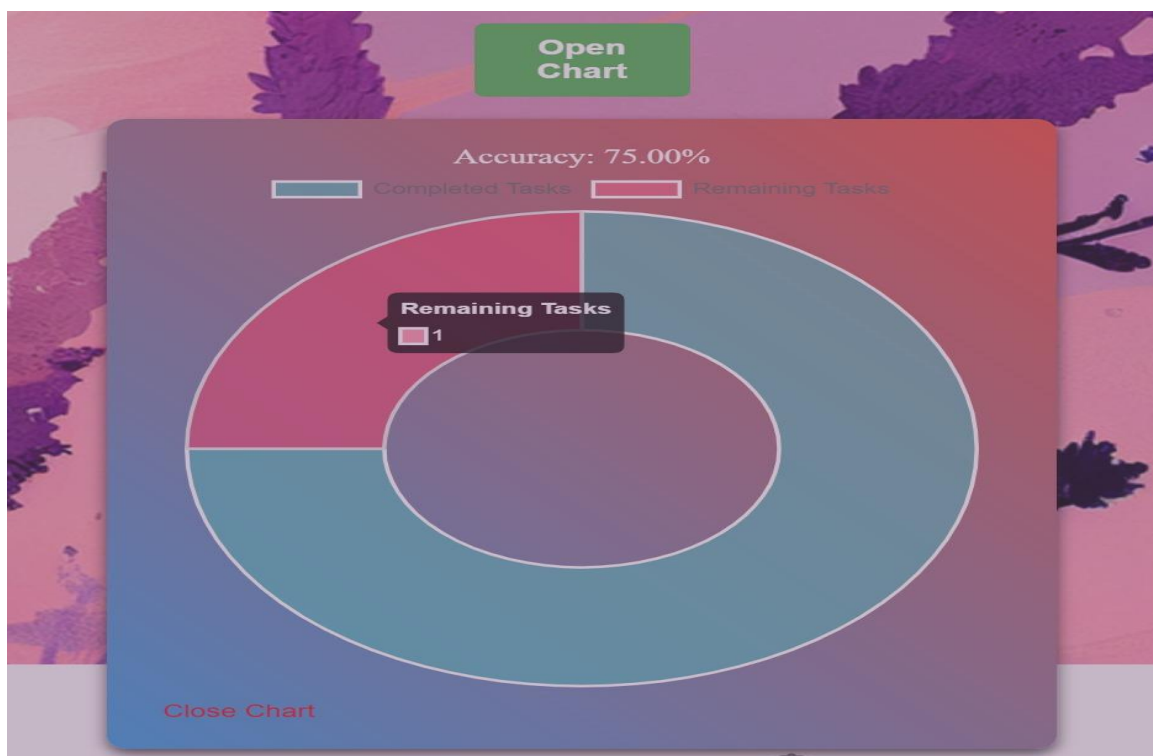


Fig 7.6 Donut Chart

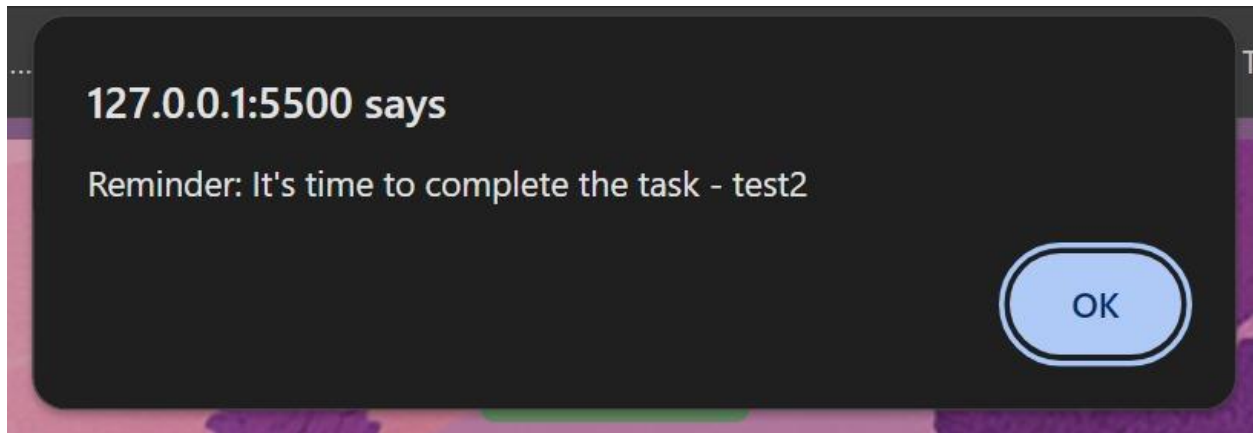


Fig 7.7 Reminder to complete the task

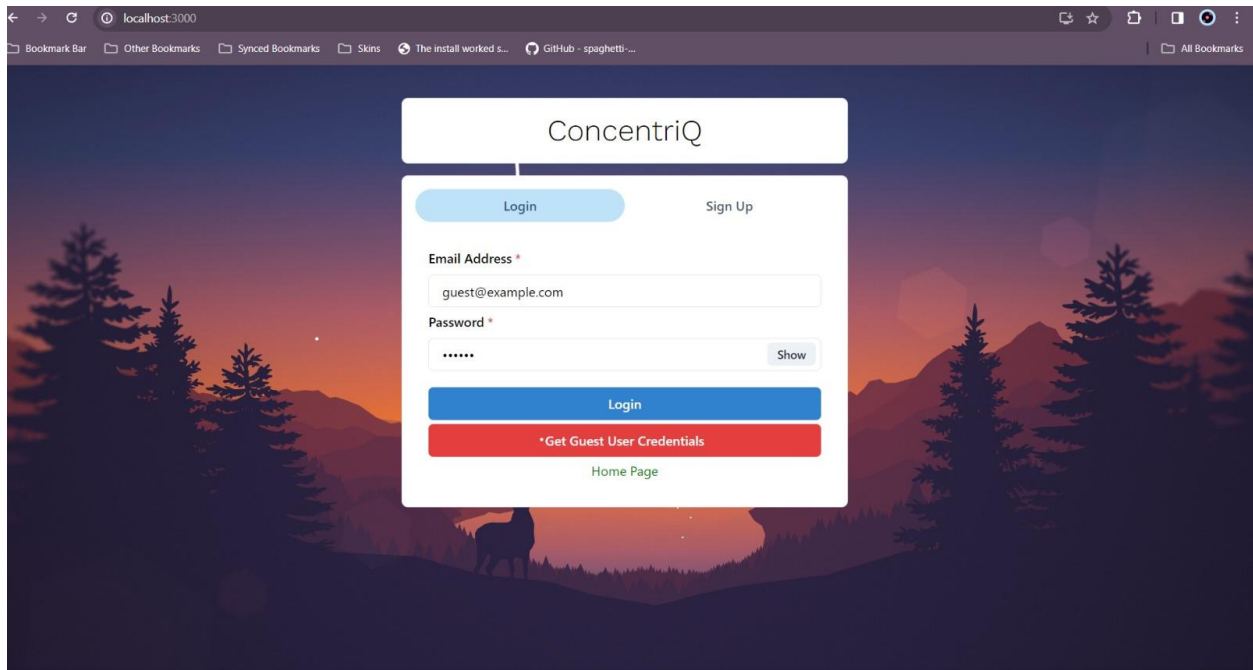
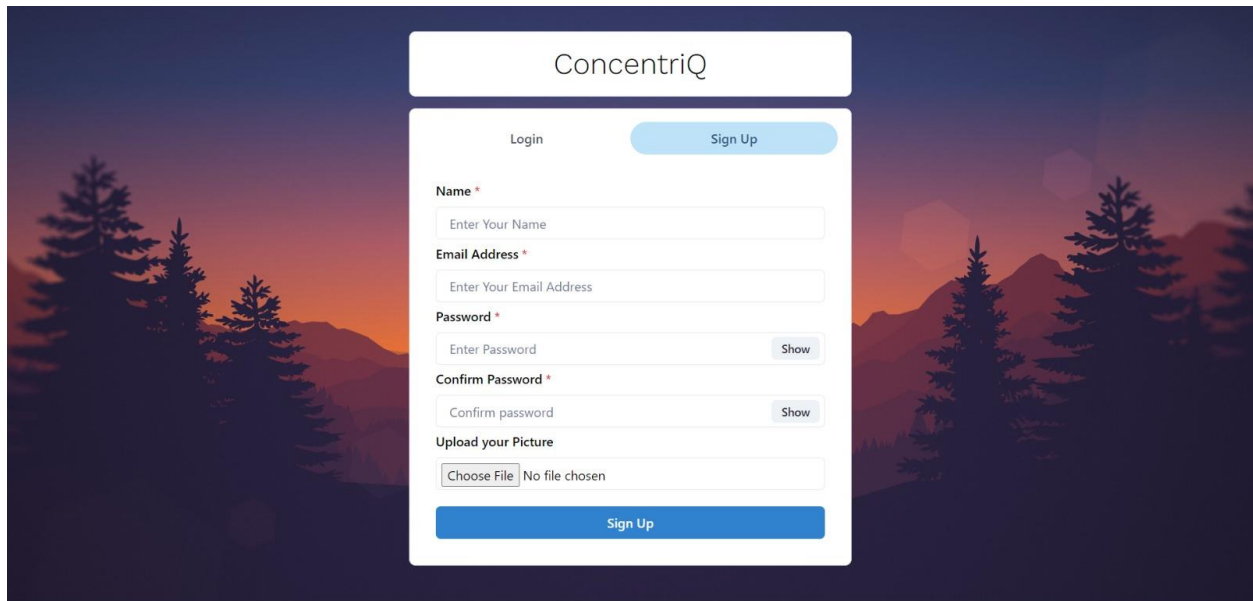


Fig 7.8 Login Page For Chat Room



The image shows a web application interface for ConcentriQ. At the top, the name "ConcentriQ" is displayed in a white box. Below it, there are two tabs: "Login" and "Sign Up", with "Sign Up" being the active tab. The form contains several input fields: "Name *" with a placeholder "Enter Your Name", "Email Address *" with a placeholder "Enter Your Email Address", "Password *" with a placeholder "Enter Password" and a "Show" button, and "Confirm Password *" with a placeholder "Confirm password" and a "Show" button. There is also an "Upload your Picture" section with a "Choose File" button and the text "No file chosen". At the bottom of the form is a large blue "Sign Up" button. The background of the page features a dark, scenic image of mountains and trees at sunset.

Fig 7.9 If not Registered,Sign Up.

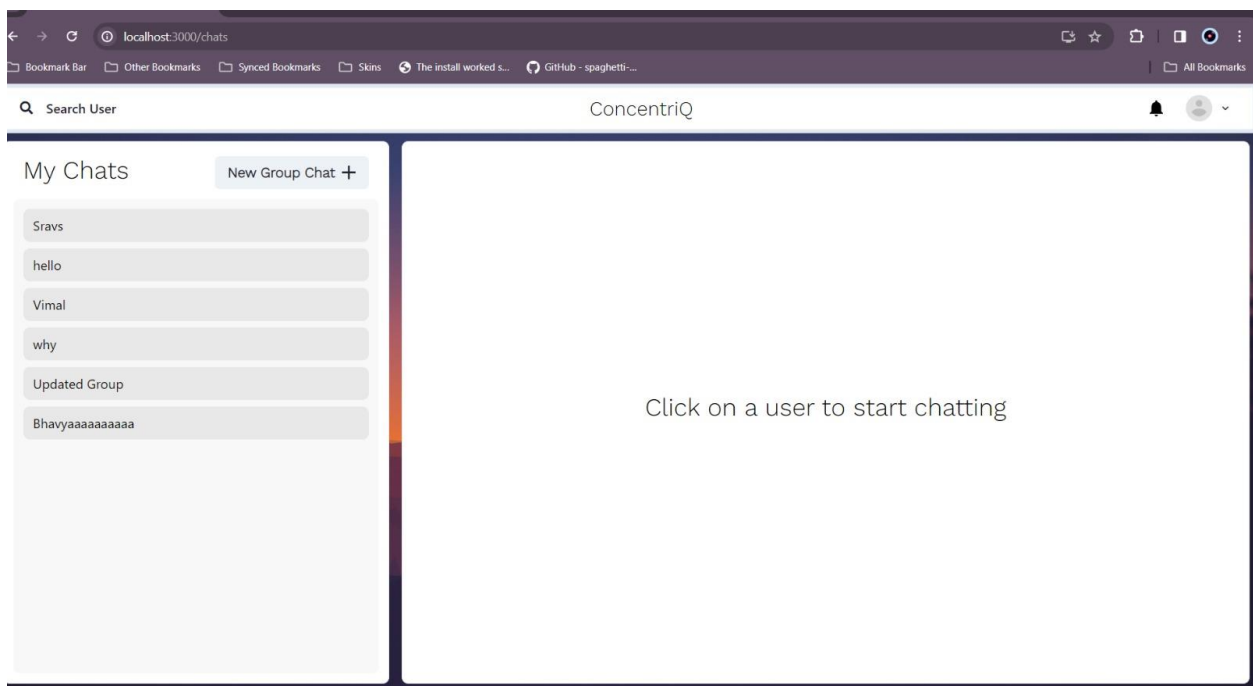
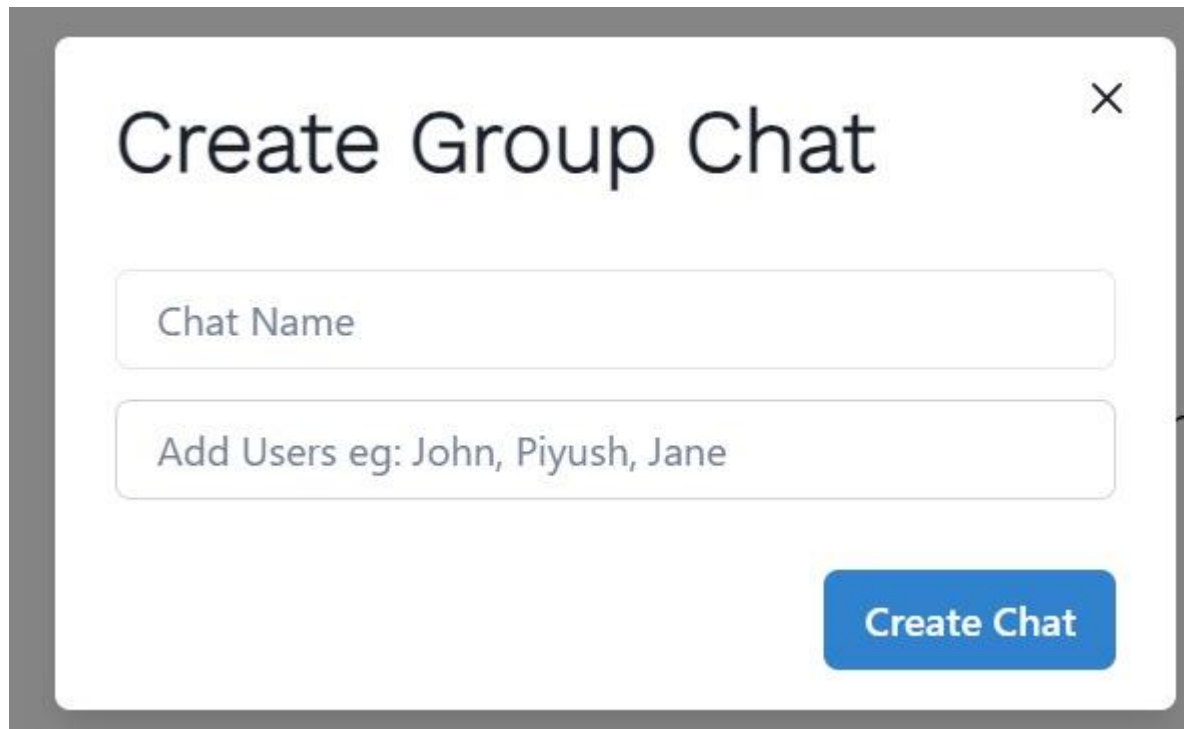
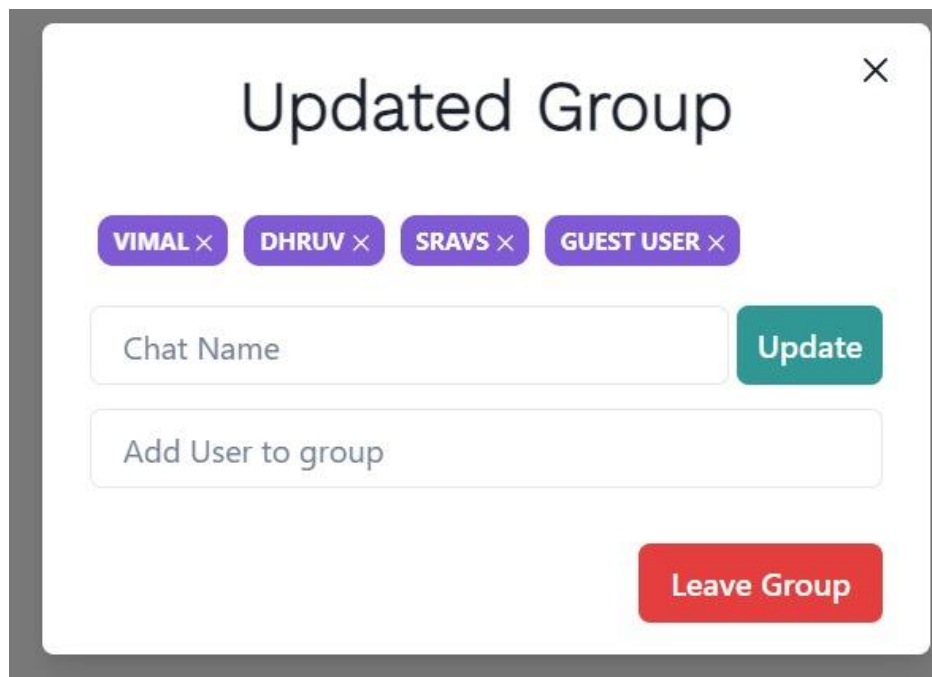


Fig 7.10 Chat page



A dialog box titled "Create Group Chat" with a close button (X) in the top right corner. It contains two input fields: "Chat Name" and "Add Users eg: John, Piyush, Jane". A blue "Create Chat" button is located at the bottom right.

Fig 7.11 You can Create group Chat if you want



A dialog box titled "Updated Group" with a close button (X) in the top right corner. It displays four user avatars: VIMAL, DHRUV, SRAVS, and GUEST USER, each with a close button (X). Below the avatars is an input field for "Chat Name" and a green "Update" button. At the bottom, there is an input field for "Add User to group" and a red "Leave Group" button.

Fig 7.12 Viewing the group members and option to add members

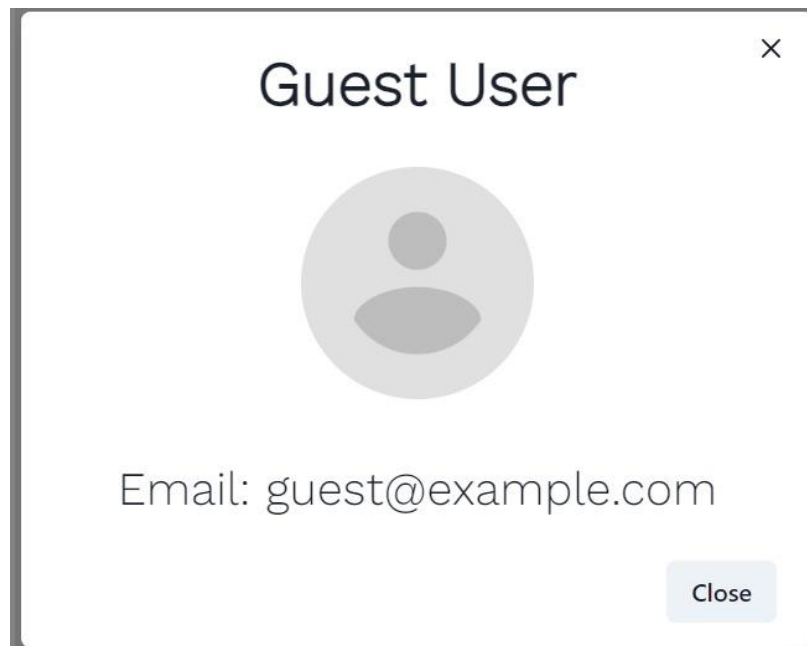


Fig 7.13 Viewing the details of a member

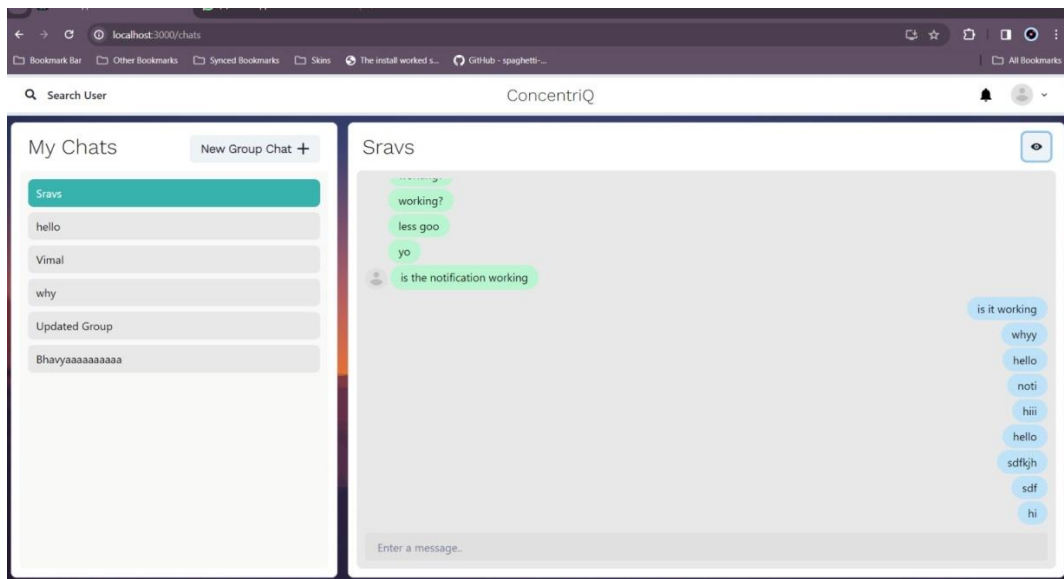


Fig-7.14 Chatting with a friend

8. CONCLUSION AND FUTURE SCOPE

ConcentriQ stands as a robust task management platform, offering users a comprehensive solution for enhanced productivity and goal achievement. The innovative integration of a dynamic to-do list, focus timer, streak system, and friend collaboration features provides a unique and efficient approach to task organization. The user-friendly interface, coupled with privacy measures, ensures a secure and intuitive experience. ConcentriQ's emphasis on consistency and accountability fosters positive habits and a supportive community. In conclusion, ConcentriQ emerges as a valuable tool in the digital landscape, addressing the contemporary need for effective time management and productivity.

In the realm of future developments, ConcentriQ envisions an exciting trajectory. The platform's potential for growth lies in continuous refinement and innovation. Anticipated advancements include the integration of artificial intelligence for personalized task recommendations, the introduction of a dedicated mobile application to enhance accessibility, and collaborations with external tools to offer users a seamlessly integrated experience. Furthermore, ConcentriQ envisions expanding its global presence through language localization and cultural customization. Advanced security measures, user analytics, and ongoing feature enhancements stand as key pillars in its journey, ensuring that ConcentriQ remains at the forefront of empowering individuals in their pursuit of enhanced productivity and goal attainment.

9.REFERENCES

- [1] Collaboration and community: Building strength in tertiary education - Shelda-Debowski**
- [2] Turning Time From Enemy into an Ally using the Pomodoro Technique- Xiaofeng Wang**
- [3] An Analysis of Winning Streak's Effects in Language Course of "Duolingo" – Duy Hyuynh**