CODE SKETCH

# A PROJECT REPORT

**for**

**Mini Project-II (ID201B) Session (2024-25)**

**Submitted by**

**VICKY KUMAR(202410116100241) SHRISTI GOYAL(202410116100202) VIDHI SHARMA(2024101161000242) YASH CHAUHAN(202410116100252)**

**Submitted in partial fulfilment of the Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

**Under the Supervision of Ms. Shruti Aggarwal Associate Professor**



**Submitted to**

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad Uttar Pradesh-201206**

# (MAY- 2025)

**CERTIFICATE**

### Certified that VICKY KUMAR (202410116100241), SHRISTI GOYAL (202410116100202), VIDHI SHARMA (202410116100241), YASH CHAUHAN

**(202410116100252)** has/have carried out the project work having “**CODE SKETCH**” (**Mini Project-II, ID201B**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Ms. Shruti Aggarwal Dr. Akash Rajak**

**Associate Professor Dean**

**Department of Computer Applications Department of Computer Applications KIET Group of Institutions, Ghaziabad KIET Group of Institutions, Ghaziabad**

# ABSTRACT

*Code Sketch* is an innovative real-time collaborative coding platform designed to enhance remote programming, pair coding, and online teaching experiences. This web-based tool allows multiple users to simultaneously write, edit, and debug code in a shared environment — similar to a digital whiteboard for programmers. One of the key features of *Code Sketch* is its seamless session- sharing capability. A user can initiate a coding session and share it with collaborators, who can then immediately begin contributing to the same codebase from any location. This live interaction fosters better teamwork, faster learning, and efficient debugging. To further support communication, *Code Sketch* offers built-in integration with Zoom. Users can initiate or join a Zoom meeting directly from the coding interface, enabling face-to-face discussions, code walkthroughs, and mentoring sessions without switching between multiple apps. With real-time syntax highlighting, multi-language support, session saving, and version tracking, *Code Sketch* aims to be a powerful tool for developers, educators, and students alike — bridging the gap between distance and collaboration.

# ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, Dr. Vipin Kumar for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude to Dr. Akash Rajak, Professor and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions. Fortunately, I have many understanding friends, who have helped me a lot in many critical conditions. Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kinds of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**VICKY KUMAR SHRISHTI GOYAL VIDHI SHARMA YASH CHAUHAN**

# TABLE OF CONTENTS

|  |  |
| --- | --- |
| Certificate | ii |
| Abstract | iii |
| Acknowledgement | iv |
| Table of Contents | v |
| 1 Introduction | 6-7 |
| 1.1Overview |  |
| 1.2Project Description | 6 |
| 1.3 Project Scope | 6 |
| 1.4 Objectives | 7 |
| 1.5 Purpose |  |
| 2 Feasibility Study | 8-10 |
| 2.1 Technical feasibility | 8 |
| 2.2 Economic feasibility | 8 |
| 2.3 Operational feasibility | 10 |
| 2.4 Legal Feasibility |  |
| 2.5 Schedule Feasibility |  |
| 3 Project Objective | 11 |
| 4 Hardware and Software Requirements | 13 |
| 5 Project Flow | 14 |
| 6 Project Outcome | 24 |
| Conclusion |  |
| References | 28 |

**CHAPTER 1 INTRODUCTION**

In today’s fast-paced digital world, remote collaboration has become an essential part of software development and learning. Whether it's working with a team across different locations or conducting online programming classes, there is a strong need for tools that allow real-time interaction, smooth communication, and efficient collaboration.*Code Sketch* is a web-based platform designed to meet these needs by providing a shared coding environment where users can write, edit, and run code together in real time.

## OVERVIEW:

*Code Sketch* is a web-based, real-time collaborative coding platform specifically built for front-end development using HTML, CSS, and JavaScript. The platform is designed for developers, students, and educators who want to code together remotely in a live, interactive environment. Multiple users can join the same coding session, write and edit code simultaneously, and see changes reflected instantly. This real-time synchronization enhances communication, teamwork, and the learning experience.

One of the key features of *Code Sketch* is the live preview panel, which displays the output of HTML, CSS, and JavaScript code instantly as it is written. This allows users to immediately see the impact of their code without needing to refresh or open separate browser windows. For beginners, this helps in understanding how structure, design, and interactivity come together in web development.

To improve collaboration even further, *Code Sketch* includes Zoom integration. Users can start or join a Zoom meeting directly within the platform, enabling face-to-face communication alongside live coding. This is especially useful for online classes, remote pair programming, and virtual workshops where visual guidance and discussion are needed in real-time.

Other key features include:

* Syntax highlighting for HTML, CSS, and JavaScript
* Auto-indentation and basic error detection, especially for JavaScript
* Live session sharing via unique links
* Project saving and exporting options
* Lightweight and fast interface accessible via any modern web browser

It is important to note that *Code Sketch* is focused solely on front-end technologies. It does not support back-end languages or frameworks like PHP, Python, or Node.js. This makes the platform lightweight, easy to use, and perfect for learning, teaching, and collaborating on front-end web projects.

In summary, *Code Sketch* offers a simplified yet powerful environment that combines live coding, instant preview, and built-in video communication — creating a virtual space where users can build, learn, and collaborate on web projects from anywhere in the world.

## PROJECT DESCRIPTION:

*Code Sketch* is a real-time, web-based collaborative coding platform designed specifically for front-end development using HTML, CSS, and JavaScript. The project aims to create an interactive coding space where multiple users can work together on the same web development project remotely and in real time. It is ideal for online classes, coding interviews, pair programming sessions, and collaborative learning environments.

The core idea behind *Code Sketch* is to eliminate the limitations of screen-sharing or sending code back and forth. Instead, it allows all participants to join a shared coding session via a link, where they can type, edit, and view code together — with changes reflected instantly for everyone. This enables smoother collaboration and a better learning or teamwork experience.

A major highlight of the project is the integration of Zoom video conferencing directly into the platform. This allows users to communicate face-to-face while coding, without having to switch between multiple apps. Instructors can explain concepts in real-time, teammates can brainstorm ideas, and students can ask questions live while coding alongside their peers.

Key components of the project include:

* Live Code Editor with support for HTML, CSS, and JavaScript
* Real-Time Synchronization so that all users can see and edit the same code together
* Live Browser Preview panel that shows the output of code instantly as it's written
* Zoom Integration for embedded video/audio calls during coding sessions
* User Session Sharing via unique URLs, allowing easy access and collaboration
* Code Saving and Exporting, enabling users to store or share their final work

The platform is built using web technologies like JavaScript, WebSockets (for real-time sync), and cloud-based services for session handling and storage. The user interface is designed to be clean, responsive, and beginner-friendly, allowing both experienced developers and students to use it without any learning curve.

By focusing only on front-end technologies, *Code Sketch* provides a fast, lightweight, and distraction-free coding environment. It is not meant for backend development or complex full-stack projects but is optimized for HTML/CSS/JS collaboration — making it ideal for web design, interactive UI prototyping, and frontend-focused education.

In conclusion, *Code Sketch* provides a unique blend of real-time code collaboration and live communication, making it an excellent tool for remote web development and online learning in today’s digitally connected world.

## PROJECT SCOPE:

the *Code Sketch* project is to develop a real-time, browser-based collaborative coding platform specifically designed for front-end web development using **HTML, CSS, and JavaScript**. The platform will allow multiple users to write, edit, and preview code together within the same coding session, all from different locations. The main objective is to provide a virtual environment that enables seamless collaboration for remote teams, students, instructors, and web developers. Unlike traditional code editors, *Code Sketch* will offer live synchronization of code so that every change made by one participant is instantly visible to all other collaborators in real time.

A key feature within the project scope is the integration of **Zoom video conferencing**,

allowing users to engage in live audio and video communication directly within the platform. This built-in communication support removes the need to switch between coding and meeting applications, creating a more efficient and focused workflow for collaborative coding sessions, online classes, or coding interviews.

The project scope includes the development of an intuitive web-based user interface, a code editor that supports syntax highlighting and auto-indentation, and a **live output panel** to preview the result of HTML, CSS, and JavaScript code instantly. Users will be able to share their coding sessions through unique URLs, join existing sessions, and optionally save or export their work. The platform will be optimized for ease of use, requiring no installation — only a modern web browser and an internet connection.

However, the platform will remain focused strictly on front-end technologies. Features such as back-end language support (like PHP, Python, or Node.js), database integration, user authentication systems, and advanced version control are **not** included in the current scope. By keeping the platform lightweight and focused, *Code Sketch* aims to deliver a fast, responsive, and practical tool for real-time front-end collaboration.

In summary, the scope of *Code Sketch* covers all essential features required to build a live, collaborative front-end coding environment with real-time communication, while deliberately excluding full-stack development and advanced backend features to maintain simplicity and speed.

## OBJECTIVES:

The primary objective of the *Code Sketch* project is to create an interactive, real-time collaborative platform for front-end web development using **HTML, CSS, and JavaScript**. This platform aims to provide a seamless and efficient coding environment for developers, students, and instructors. The specific objectives of the project are:

1. Enable Real-Time Collaboration:

Allow multiple users to work on the same code simultaneously, with live updates and

changes reflected instantly across all participants' screens. This feature fosters effective collaboration among remote teams, online coding classrooms, and peer programming sessions.

1. Support Front-End Web Development:

Focus on HTML, CSS, and JavaScript to provide a streamlined platform for creating and editing front-end web pages. The goal is to offer users a practical tool for designing and testing web pages and UI components in real time.

1. Provide Live Output Preview:

Integrate a live preview window where users can instantly see the output of their HTML, CSS, and JavaScript code. This feature allows developers to quickly visualize the results of their work, improving the learning process and the efficiency of collaborative projects.

1. Enhance Communication with Integrated Video Conferencing:

Integrate Zoom video conferencing within the platform to allow users to communicate and collaborate face-to-face while working on code. This feature is designed to simulate in-person collaboration and is especially useful for online coding classes, tutorials, and team meetings.

1. Simplify Session Management and Sharing:

Enable easy session sharing via unique links, allowing users to join ongoing sessions with minimal setup. This objective ensures that the platform is user-friendly and accessible, with an intuitive process for creating, joining, and saving sessions.

1. Provide a Lightweight, Easy-to-Use Platform:

Design a clean, responsive interface that requires no installation or complex setup, making it easy for both beginners and experienced developers to start coding immediately. The platform will be fully web-based, ensuring accessibility from any modern browser.

1. Support Code Export and Session Saving:

Allow users to save their ongoing projects, resume coding sessions, and export their

work for later use or sharing. This functionality supports long-term projects and provides a convenient way for users to keep track of their progress.

## PURPOSE:

The purpose of the *Code Sketch* project is to provide a platform that enhances the collaborative experience for front-end developers, students, and instructors working with **HTML, CSS, and JavaScript**. The platform aims to address the growing need for real-time coding environments where users can collaborate seamlessly, especially in remote or distributed settings.

By offering a space for multiple users to write, edit, and test code together in real time, *Code Sketch* aims to improve the efficiency of team-based projects, pair programming, and interactive learning. This tool is designed to make coding more interactive and accessible, allowing participants to collaborate regardless of location while receiving immediate feedback through live code previews.

Additionally, the integration of **Zoom video conferencing** directly into the platform aims to create a more cohesive environment where users can communicate and solve problems together face-to-face, without the need to switch between separate applications. This makes it particularly useful for online coding workshops, virtual classrooms, and remote development teams.

Ultimately, the purpose of *Code Sketch* is to streamline the coding process for front-end development, fostering a more engaging and productive environment for both individual learners and teams. It seeks to make coding accessible, efficient, and more interactive, ensuring that users can focus on learning and creating without the technical distractions often associated with managing multiple tools.

# CHAPTER 2 FEASIBILITY STUDY

The feasibility study for *Code Sketch* analyzes the practicality and viability of developing and implementing a real-time collaborative coding platform specifically for **HTML, CSS, and JavaScript**. This study considers technical, operational, and financial factors to evaluate whether the project can be successfully executed and sustained.

* 1. **TECHNICAL FEASIBILITY:**

### Technologies Overview

* + - * **Platform Type**: Browser-based, real-time collaborative coding platform for **HTML, CSS, and JavaScript**.

### Frontend Technologies:

* + - * + **HTML5**, **CSS3**, **JavaScript** for user interface and interactivity.
        + **Monaco Editor** for code editing with syntax highlighting, auto-completion, and error hints.

### Real-Time Collaboration:

* + - * + **WebSockets** for live synchronization of code changes across all users.

### Backend Technologies:

* + - * + **Node.js** for server-side logic and handling WebSocket connections.
        + **Express.js** for API endpoints.

### Video Integration:

* + - * + **Zoom API** for embedded video conferencing to facilitate communication during sessions.

### Cloud Infrastructure:

* + - * + Hosted on **AWS** or **Google Cloud** for scalability and availability.
        + **Cloud storage (AWS S3 / Google Cloud Storage)** for session saving and file

storage.

### System Architecture

1. Client-Side (Frontend)
   * Technologies: HTML5, CSS3, JavaScript.
   * Code Editor: Monaco Editor (syntax highlighting, auto-completion).
   * Real-Time Preview: Live code output displayed as users edit.
   * Video Conferencing: Zoom API integration for embedded video calls.
   * Real-Time Sync: WebSockets for live collaboration between users.
2. Server-Side (Backend)
   * Server Framework: Node.js with Express.js.
   * Session Management: Unique session IDs for user sessions.
   * WebSockets: Handles real-time communication and synchronization of code changes.
   * Zoom Integration: Backend manages Zoom meetings via Zoom API.
3. Cloud Infrastructure
   * Hosting: Cloud services like AWS or Google Cloud for scalability.
   * Database: NoSQL database (MongoDB) for session and project data.
   * Storage: Cloud storage (AWS S3/Google Cloud Storage) for saving user projects.
4. Security and Authentication
   * Data Encryption: SSL/TLS encryption for secure data transfer.
   * Session Security: Session IDs validated for access control.
   * Future Authentication: JWT token-based user authentication.
5. Real-Time Code Synchronization
   * WebSockets: Synchronizes code in real time across all users.
   * State Management: Backend maintains the latest code state for each session.
6. User Flow
7. Session Creation: User creates a session, server generates a unique ID.
8. Join Session: Other users join using the session ID/link.
9. Collaboration: Real-time code editing and syncing via WebSockets.
10. Live Preview: Instant output preview based on code changes.
11. Zoom Call: Embedded Zoom for video communication.

### Security Measures

Security is paramount given the sensitive nature of the communication. Key measures include:

* + - * **Data Encryption:** All data transmitted between users and the server is encrypted using **SSL/TLS** to protect sensitive information (e.g., code, session details).
      * **Session Expiration**: Sessions can have expiration times or manual termination to prevent unauthorized access after a session is no longer active.
      * **Role-Based Access Control (RBAC)**: Implement role-based permissions, such as Admin (create sessions) or Viewer (view-only access), to control access to features.
      * **Secure WebSockets (WSS)**: Communication over WebSockets will be encrypted using **WSS** (WebSocket Secure) to ensure secure, real-time collaboration.

### Development Considerations Development Tools:

* + - * **VS Code** or **Sublime Text** (Code editors).
      * **Git** for version control, **GitHub**/**GitLab** for hosting repositories.
      * **Sass/LESS** for CSS preprocessing.
      * **React.js** or **Vue.js** (optional for UI components).
      * **Node.js** (for backend development).
      * **Express.js** (for API handling).
      * **Socket.io** for WebSocket communication (real-time collaboration).
      * **AWS** or **Google Cloud** for hosting and scalability.
      * **Jest**/**Mocha** for unit testing.
      * **Cypress** for end-to-end testing.
      * **ESLint** for JavaScript code linting.

## ECONOMIC FEASIBILITY

### Cost Analysis

A thorough economic analysis identifies various cost components relevant to the development and maintenance of the API:

* + - * **Development Costs:** Personnel (developers, project managers): Estimated at

$100,000 for the team over six months. Infrastructure (servers, databases): Initial setup costs around $10,000.

* + - * **Operational Costs:** Hosting services (AWS, Google Cloud): Approximately

$100/month initially.

### Return on Investment

* + - * **Projected Revenue:** Based on usage, potential revenue streams such as subscription models or pay-per-use services could generate estimated revenues of $15,000 to

$50,000 monthly after launch.

* + - * **Break-Even Point:** Expected within the first year, assuming moderate adoption.

### Market Potential

* + - * **Target Audience:** Individuals and organizations needing secure communication.
      * **Trends:** Increasing demand for privacy-focused solutions, especially in compliance- heavy industries.
      * **Competitive Analysis:** Review existing tools and identify unique value propositions (e.g., ephemeral message storage).

### Pricing Strategy

* + - * **Freemium Model:** Basic service free, with advanced features available through subscription plans.
      * **Pricing Tiers:** Individual users: $5/month, Business users: $20/month for enhanced features like larger message limits and additional security measures.

## OPERATIONAL FEASIBILITY

### User Experience

* + - * **User Interface (UI):** Simple, intuitive design that requires minimal training.
      * **User Journey:** Streamlined process for creating and retrieving messages.

### Implementation Strategy

* + - * **Phased Rollout:** Begin with a pilot program among select users to gather feedback and make improvements.
      * **Marketing Plan:** Utilize content marketing, social media, and partnerships to promote the API.

### Support and Maintenance

* + - * **Technical Support:** Establish a support team for user inquiries and issues.
      * **Continuous Updates:** Regularly introduce new features based on user feedback and technology advancements.

### Adoption Factors

* + - * **Learning Curve:** Ensure onboarding materials (tutorials, FAQs) are easily accessible.
      * **Feedback Mechanism:** Encourage user feedback for continuous improvement, fostering a community around the API.

## LEGAL FEASIBILITY

### Compliance with Data Protection Laws

* + - * **User Consent**: Obtain clear consent before collecting personal data.
      * **Right to Access & Delete**: Users can request to view or delete their data.
      * **Data Minimization**: Collect only essential information (e.g., session ID, email).
      * **Data Encryption**: Encrypt all stored and transmitted data.
      * **Privacy Policy**: Clear and accessible privacy policy outlining data use.

### Intellectual Property Considerations

1. Ownership of Code & Content
   * User-Generated Content:
     + All code written by users remains their own property.
     + The platform does **not claim ownership** of user-created projects or code.

### Platform Source Code

* + The platform's backend, frontend, and core features (excluding user content) are intellectual property of the developers/company behind *Code Sketch*.
  + Protect through copyright and/or software licenses.

### User Agreement and Terms of Service

1. **Acceptance of Terms**
   * By using *Code Sketch*, users agree to these terms and conditions.
   * Continued use signifies consent to any updates made to the agreement.

### User Responsibilities

* + Users are responsible for any content/code they create or share.
  + Users must not upload malicious code, spam, or illegal content.
  + Users must respect others during collaborative sessions (no abuse, threats, etc.).

## SCHEDULE FEASIBILITY

### Development Timeline

The estimated timeline for the development of the One-Time Secret Message API is broken into several phases:

* + - * Phase 1: Planning & Research (Week 1–2)
      * Phase 2: Design (Week 3–4)
      * Phase 3: Frontend Development (Week 5–7)
      * Phase 4: Backend Development (Week 8–10)
      * Phase 5: Online Integration (Week 11)
      * Phase 6: Testing & Debugging (Week 12–13)
      * Phase 7: Deployment & Hosting (Week 14)
      * Phase 8: Documentation & Launch (Week 15)

# CHAPTER 3 PROJECT/RESEARCH OBJECTIVES

## 3.1 INTRODUCTION

The primary objective of this research is to explore the development of a real-time collaborative coding platform that supports HTML, CSS, and JavaScript, with integrated video conferencing capabilities. The research focuses on understanding how existing collaborative tools, such as Google Docs and online code editors like Replit and JSFiddle, manage multi-user editing and live synchronization. A significant part of the study involves evaluating WebSocket technologies, particularly Socket.io, to enable smooth, real-time code sharing between users. Additionally, the project investigates the Zoom API to enable seamless video communication within the platform, ensuring secure and efficient integration.

### User Authentication and Security

1. **User Authentication**
   * Login/Sign-Up System for registered users
   * JWT (JSON Web Tokens) for secure session handling
   * Guest Access with limited permissions (no saving sessions)
   * OAuth Integration (future scope for Google, GitHub, Zoom logins)

### Session Security

* + Unique session ID for each collaboration room
  + Access roles: Host, Collaborator, Viewer
  + Session links can expire or be revoked manually
  + Only authorized users can join a session

### Create unique, encrypted links

**Objective:** To generate secure, unique, and encrypted session links that prevent unauthorized access to collaborative coding sessions

### Details:

The goal is to ensure that every collaborative session created on *Code Sketch* has

a **unique and private link** that cannot be guessed, reused, or accessed by unauthorized users. This is achieved by generating a **random session ID** using cryptographic functions and optionally **encrypting** it before adding it to the session URL.

This process ensures that:

* Each session has a **one-of-a-kind identifier**.
* The session link is **obfuscated and secure**, even if shared publicly.
* **Only users with the encrypted link** can join or collaborate.
* The session ID is not exposed in raw form, preventing misuse or guessing.
* You can implement **link expiration**, adding an extra layer of control by deactivating links after a set time or once the session ends.

1. **Minimize the Attack Surface**

**Objective: To minimize the platform's attack surface by reducing unnecessary access points, services, and vulnerabilities**

**Details:**

Minimizing the attack surface means **limiting the number of ways an attacker can interact with or exploit the system**. In *Code Sketch*, this involves both frontend and backend security practices.

Key strategies include:

### Limit API Exposure

* Only expose **essential API endpoints**.
* Use proper **authentication** and **authorization** for every endpoint.
* Disable or remove unused routes and test APIs from production.

### Input Validation & Sanitization

* Validate all user input to prevent attacks like **XSS (Cross-Site Scripting)** and **SQL/NoSQL Injection**.
* Sanitize input fields in the code editor, chat box (if added), and login forms.

### Role-Based Access Control (RBAC)

* Assign roles like **Admin**, **Editor**, and **Viewer**.
* Limit access based on user role — for example, only hosts can end a session or invite new users.

### Disable Unused Features/Ports

* Turn off unused backend services, WebSocket ports, and third-party integrations.
* Restrict open ports and unnecessary system services on the server.

### Secure File Access and Storage

* Prevent public access to backend folders or configuration files.
* Use signed URLs or token-based access for any downloadable files.

### Regular Code Reviews and Patching

* Review code for security issues.
* Apply security updates to all dependencies and libraries (e.g., Express, Socket.io).

### Environment Separation

* Separate **development**, **testing**, and **production** environments.
* Never expose test data or debug tools in the live app.

### Secure Access Links

**Objective:** To implement secure access links that ensure only authorized users can join and participate in a coding session.:

### Details:

Secure access links are special URLs that act as private keys to enter a session. The main goal is to prevent unauthorized users from joining or hijacking a session, while keeping the process simple and user-friendly.

# CHAPTER 4 HARDWARE/SOFTWARE REQUIREMENTS

The hardware requirements depend on the scale and the environment in which the API is deployed (development, testing, production). For a basic setup or a development environment:

### Hardware Requirements

* + 1. **Development and Testing:**
       - Processor: Intel i5 or higher (quad-core or above recommended)
       - RAM: 8GB or more
       - Storage: 100GB of available space (depends on the number of messages stored, but since messages are encrypted and removed quickly, storage needs won't be massive unless the system is scaling up)
       - Internet Connection: Stable high-speed internet for accessing cloud services, dependencies, and updates.

### Production/Enterprise:

* + - * Processor: Multi-core processors (Intel Xeon, AMD EPYC for scalability in enterprise environments)
      * RAM: 16GB or more, depending on expected API traffic and number of concurrent requests.
      * Storage: SSD storage for fast data retrieval, with scalability (cloud storage can be used to

scale storage needs). About 100GB or more for storing encrypted messages (depends on usage).

* + - * Network: High-speed internet connection with redundancy for reliable cloud hosting or on- premise deployment.

### Optional:

* + - * GPU (for heavy encryption operations or AI integration): A high-performance GPU if your application has intensive encryption or AI components.
      * Backup/Redundancy: For cloud deployments, ensure multiple availability zones for redundancy and uptime.

**4.2 Software Requirements Frontend Requirements**

## HTML5 & CSS3

* + For basic page structure and styling.
  + Compatible with modern browsers.

### JavaScript (ES6+)

* + Used for client-side interactivity.
  + Supports real-time events and DOM manipulation.

### React.js

* + Framework for building dynamic UI components.
  + Manages component state, rendering, and routing.
  + Allows reusable component-based structure.

### Tailwind CSS

* + Utility-first CSS framework for rapid UI development.
  + Enables clean, responsive design with minimal custom CSS.

### Backend Requirements

1. **Node.js**
   * JavaScript runtime environment for executing backend logic.
   * Efficient and scalable for real-time applications.

### Express.js

* + Lightweight web framework for Node.js.
  + Used to build RESTful APIs and route requests efficiently.

### WebSockets (via Socket.io)

* + Enables real-time, bidirectional communication between users.
  + Essential for collaborative code editing and live sessions.

### Database Requirements

1. **MongoDB**
   * NoSQL database for storing session data, user info, and code snippets.
   * Scalable, fast, and well-suited for JSON-like document structures.
   * **Mongoose.js** can be used as an ORM for schema management.

### Other Supporting Tools

* + **npm/yarn** – Package managers for managing project dependencies.
  + **Webpack/Vite** – Module bundlers for frontend optimization.
  + **Postman** – For testing API endpoints.
  + **Git & GitHub** – Version control and collaboration.

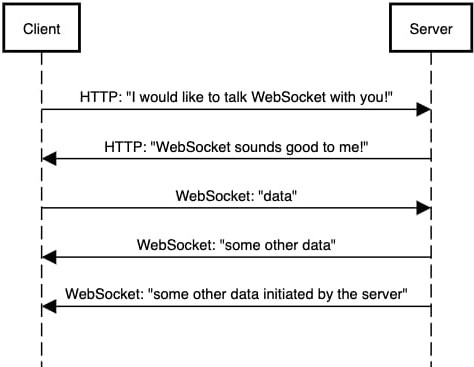
### System Requirements (Minimum)

* + **Browser**: Latest Chrome, Firefox, Edge
  + **Node.js Version**: v16 or above
  + **MongoDB**: v5 or above
  + **RAM**: 4GB (Development), 8GB+ (Production)
  + **OS**: Windows, macOS, or Linux

# CHAPTER 5 PROJECT FLOW

* 1. **Flowchart**

Flowchart is a diagrammatic representation of sequence of logical steps of a program.Flowcharts use simple geometric shapes to depict processes and arrows to show relationships and process/data flow.



### Fig No. 5.1

* + 1. **Steps in the Flowchart**
       1. Session Creation

One user starts a coding session.

### Join Session

Other collaborators join using a unique ID or link.

* + - 1. Collaboration (Real-time code editing and syncing) This is where WebSockets come into play!

### Live Preview

Changes in the code are instantly reflected. WebSockets facilitate this.

### Embedded

For direct communication alongside coding.

### Connecting the Dots with the WebSocket

* + - 1. Initial Handshake
         * When a user joins a CodeSketch session, their web browser (the **Client**) needs a persistent connection to the CodeSketch server (**Server**) to enable real-time updates.
         * The process starts with an HTTP "upgrade" request, just like in the first line of the diagram: "HTTP: I would like to talk WebSocket with you!"
         * The CodeSketch server responds positively, as shown in the second line: "HTTP: WebSocket sounds good to me!" This establishes the WebSocket connection.
      2. Real-Time Code Editing and Syncing
         * Once the WebSocket connection is established, whenever a user types or edits code in their CodeSketch window, these changes are sent immediately to the server via the open WebSocket connection (similar to "WebSocket: data" and "WebSocket: some other data" in the diagram).
         * The CodeSketch server then instantly broadcasts these changes to all other connected users in the same session through their respective WebSocket connections. This is how everyone sees the code being updated in real-time.
      3. Live Preview:

Similarly, when the code changes, the server can process these changes and push the updated preview information back to all connected clients in real-time using the same persistent WebSocket connection. This eliminates the need for users to manually refresh their browsers to see the latest output.

* 1. **Data Flow Diagram (DFD) Levels**

### Level 0: Context Diagram

This level provides a high-level view of the system.

### External Entities:

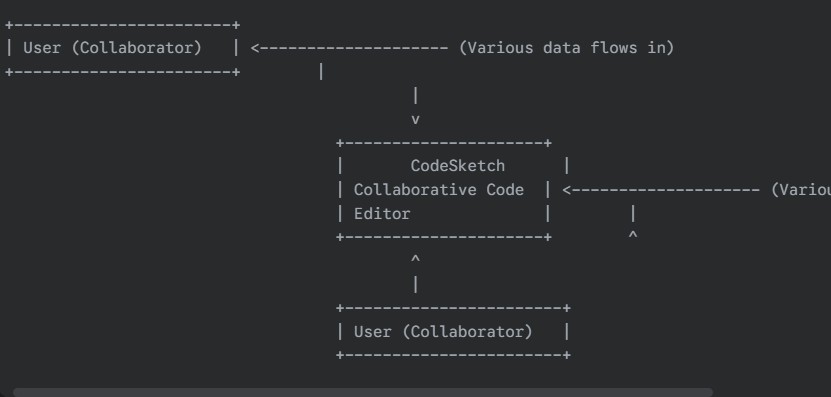
* **User**: Individuals who use CodeSketch to write and collaborate on code.

### Processes::

* + - * + Online Collaborative Code Editor with Live Preview and Communication

### Data Stores:

* **Database**: Code Input, Collaboration Data, Live Preview Request, Live Preview Output, Communication Data, etc.
  + - 1. **Diagram Representation:**

****

**Fig No.5.2.1**

* + 1. **Level 1: Detailed DFD**

This level breaks down the processes into more detailed steps.

### Manage Session:

* Receives "Session Creation Request" and "Join Request" from the User.
* Sends "Session ID/Link" to the User.

### Collaborative Editing:

* Receives "Code Input" and "Code Changes" from the User.
* Exchanges "Code Changes" with other instances of the "Collaborative Editing" process for other connected users (this flow would be internal to the system but is crucial for the collaborative aspect).
* Sends "Code for Preview" to the "Process Code & Generate Preview" process.
* Sends "Code Changes" back to the User (reflecting changes from other collaborators).

### Process Code & Generate Preview:

* + - * + Receives "Code for Preview" (HTML, CSS, JavaScript).
        + Processes the code to generate the output.
        + Sends "Live Preview Output" to the User.

### Facilitate Communication:

* + - * + Receives "Communication Data (Audio/Video)" from the User.
        + Routes "Communication Data (Audio/Video)" to other connected Users.

### Manage Saving:

* Receives "Save Request" from the User.
* Manages the storage of the code.
* Provides "Saved Code" (potentially to the User or for loading later).

**1. Diagram Represenatatio:**



**Fig No.5.2.2**

* + 1. **Level 2: Sub-Process Detail**

**Purpose**: Further breaks down key processes for additional clarity, especially important processes like "Save to Database" and "Retrieve Message".

### Detailed Breakdown

* + - 1. **Manage Session:**
* **1.1 Create Session:** Handles the user request to start a new session and generates a unique Session ID.
  + *Input:* Session Creation Request (from User)
  + *Output:* Session ID (to User), Session Details (internal)
* **1.2 Join Session:** Allows users to join an existing session using the Session ID/Link.
  + *Input:* Join Request, Session ID/Link (from User)
  + *Output:* Session Confirmation (to User), User Added to Session (internal)

### Collaborative Editing:

* **2.1 Receive Code Input:** Accepts code entered or modified by a user.
  + *Input:* Code Input (from User)
  + *Output:* Code Changes (internal)
* **2.2 Synchronize Changes:** Broadcasts code changes to all other connected users in the session.
  + *Input:* Code Changes (internal)
  + *Output:* Code Updates (to other Collaborative Editing processes)
* **2.3 Apply Remote Changes:** Receives and applies code changes from other collaborators to the local editor.
  + *Input:* Code Updates (from other Collaborative Editing processes)
  + *Output:* Updated Code (internal), Updated Code (to User interface)

### Process Code & Generate Preview:

* **3.1 Receive Code for Preview:** Accepts the HTML, CSS, and JavaScript code to be rendered.
  + *Input:* Code for Preview (from Collaborative Editing)
  + *Output:* Processed Code (internal)
* **3.2 Render Preview:** Interprets and renders the code to generate the visual output.
  + *Input:* Processed Code (internal)
  + *Output:* Preview Data (internal)
* **3.3 Send Preview Output:** Transmits the rendered preview to the requesting user(s).
  + *Input:* Preview Data (internal)
  + *Output:* Live Preview Output (to User)

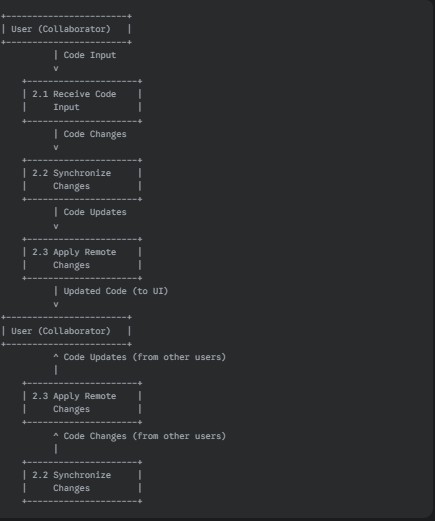
### Facilitate Communication:

* **4.1 Receive Communication Data:** Accepts audio and/or video data from a user.
  + *Input:* Communication Data (from User)
  + *Output:* Communication Stream (internal)
* **4.2 Route Communication Data:** Forwards the audio/video stream to other connected users in the session.
  + *Input:* Communication Stream (internal)
  + *Output:* Communication Stream (to other Facilitate Communication processes)
* **4.3 Send Communication Data:** Transmits the audio/video stream to the user's interface.
  + *Input:* Communication Stream (from other Facilitate Communication processes)
  + *Output:* Communication Data (to User)

### Manage Saving:

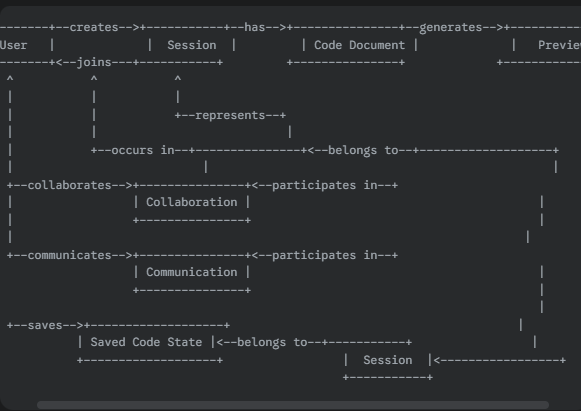
* **5.1 Receive Save Request:** Accepts the user's request to save the current code.
  + *Input:* Save Request (from User)
  + *Output:* Code to Save (internal)
* **5.2 Store Code:** Persists the code in a storage mechanism (e.g., database, local storage).
  + *Input:* Code to Save (internal)
  + *Output:* Save Confirmation (internal)
* **5.3 Provide Saved Code (Loading):** Retrieves saved code when a user requests it (this could be another sub-process or part of session loading).
  + *Input:* Load Request, Session ID/User ID (from User)
  + *Output:* Saved Code (internal), Loaded Code (to Collaborative Editing)

**1. Diagram Representation**

****

**Fig No.5.2.3**

* 1. **Entity Relationship Diagram**

****

**Fig No.5.3**

* **Entitties:**
  + User: Represents an individual who uses the CodeSketch platform.
  + Session: Represents a collaborative coding environment created by a user.
  + Code Document: Represents the HTML, CSS, and JavaScript code written within a session.
  + Collaboration: Represents the real-time interaction between users within a session.
  + Preview: Represents the rendered output of the code.
  + Communication: Represents the audio/video interaction between users within a session.
  + Saved Code State: Represents a saved version of the code document**.**

### Relationships:

* User Creates Session: A User can create one or more Sessions. A Session is created by one User (the owner). (One-to-Many relationship from User to Session)
* User Joins Session: A User can join one or more Sessions (as a collaborator). A Session can have many Users (collaborators). (Many-to-Many relationship between User and Session)
* Session Has Code Document: A Session has one Code Document (containing HTML, CSS, and JavaScript). A Code Document belongs to one Session. (One-to-One relationship between Session and Code Document)
* Session Involves Collaboration: A Session involves Collaboration between multiple users. Collaboration occurs within a Session. (One-to-Many relationship from Session to Collaboration, or a relationship with multiple participating Users)
* Code Document Generates Preview: A Code Document generates one Preview. A Preview is generated from one Code Document. (One-to-One relationship between Code Document and Preview)
* Session Supports Communication: A Session supports Communication between its participants. Communication occurs within a Session. (One-to-Many relationship from Session to Communication, involving multiple Users)
* User Saves Code Document: A User can save a Code Document at a specific point, resulting in a Saved Code State. A Code Document can have many Saved Code States (per user or session). (One-to-Many relationship from Code Document to Saved Code State, potentially with User as a factor)
* Session Has Saved Code States: A Session can have many Saved Code States. A Saved Code State belongs to a Session. (One-to-Many relationship from Session to Saved Code State)

### Use Case Diagram

A Use Case Diagram visually represents the interactions between users (actors) and the system. It outlines the functionalities (use cases) that the system offers to the users and the relationships between those functionalities.

### Key Components of a Use Case Diagram Actors:

* User (Collaborator): Any individual who interacts with the CodeSketch system.

### Use Cases:

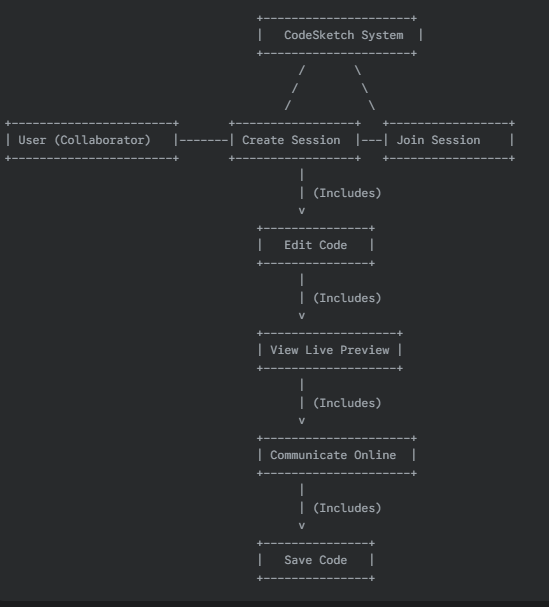
* Create Session: A user starts a new collaborative coding session.
* Join Session: A user joins an existing coding session using a session ID or link.
* Edit Code: Users collaboratively write and modify HTML, CSS, and JavaScript code in real-time.
* View Live Preview: Users instantly see the rendered output of their code changes.
* Communicate Online: Users engage in audio/video communication within the coding session.
* Save Code: Users save the current state of their code.

### System Boundary:

This is the boundary that defines what is inside the system and what is external. Use cases that fall within the system indicate functionalities provided.

### Relationships:

* Include: Represents functionality that is included within another use case.
* Extend: Represents optional functionality that can extend another use case under certain conditions.
* Association: Represents the interaction between an actor and a use case.



### Use Case Diagram Description

* Actor: The "User (Collaborator)" is represented by the stick figure on the left.
* System Boundary: The rectangle labeled "CodeSketch System" encloses all the use cases.
* Use Cases: Each oval represents a specific use case (functionality) that the user can perform.
* Associations: Lines connect the "User (Collaborator)" actor to the use cases they interact with. For example, the user can "Create Session," "Join Session," "Edit Code," "View Live Preview," "Communicate Online," and "Save Code."
* Include Relationships: Dashed arrows labeled "&lt;&lt;include>>" point from a base use case to an included use case, indicating that the included use case is always performed as part of the base use case. In this diagram, "Edit Code" includes "View Live Preview" and "Communicate Online," and "Edit Code" also includes "Save Code" (as saving is a part of the editing workflow). "Create Session" also implicitly leads to "Edit Code."

### Benefit of Use Case Diagram

* Understanding user needs.
* Defining system scope and functionality.
* Communicating effectively with stakeholders.
* Guiding the development process.
* Ensuring the system meets user requirements.

# CHAPTER 6 PROJECT OUTCOME

### Functional Outcomes

* + 1. **Create Session**
       - Observable Behavior: The system generates and displays a unique session ID or link.
       - Result/Output: A new, empty coding session is created and accessible via the ID/link.
       - User Value: The user can initiate a collaborative coding environment.

### Join Session

* + - * Observable Behavior: The user enters a session ID/link, and the system navigates them to the shared coding interface.
      * Result/Output: The user is now connected to the existing coding session and can see the current code.
      * User Value: The user can participate in collaborative coding with others..

### Edit Code

* + - * **Observable Behavior:** Keystrokes and code modifications are reflected in the editor.
      * **Result/Output:** The shared code document is updated in real-time for all participants.
      * **User Value:** Users can collaboratively build and modify code.

### View Live Preview

* + - * **Observable Behavior:** The system displays a rendered output of the HTML, CSS, and JavaScript code.
      * **Result/Output:** A visual representation of the code's effect is shown.
      * **User Value:** Users can instantly see the results of their coding changes.

### Communicate Online

* + - * Observable Behavior: Audio and video streams are transmitted and displayed between users.
      * Result/Output: Users can have real-time voice and/or video conversations.
      * User Value: Users can discuss and coordinate their coding efforts directly within the platform.
    1. Save Code
       - **Observable Behavior:** The system indicates that the code has been saved.
       - **Result/Output:** The current state of the code is stored for later retrieval.
       - **User Value:** Users can preserve their work and return to it later.

### Technical Outcomes

* + 1. **System Performance**
       - **Speed**: The time it takes for the system to react to a user's input or a request. Lower latency generally indicates better performance. For users in Muradnagar, this could mean how quickly a webpage loads or how fast their code changes are reflected in CodeSketch.
       - **Scalability**: The system's ability to operate without failures or errors over a sustained period. Performance issues can sometimes be linked to instability.

### Security

* + - * **Data Protection**: Protecting data at rest, in transit, and in use. This encompasses encryption, data loss prevention (DLP) tools, access controls, and data classification.
      * **Data Integrity**: Securing individual devices like computers, laptops, and mobile phones that connect to the network. This involves antivirus software, anti-malware tools, personal firewalls, and endpoint detection and response (EDR) solutions..

### Usability and Accessibility

* + - * **User-Friendly Interface**: New users can quickly figure out how to perform basic tasks.
      * **Error-tolerant:** The design minimizes errors and provides easy ways to recover from them.

### User Outcomes

* + 1. **User Satisfaction**
       - **Functionality**: The extent to which the system provides the features and capabilities that users need and expect. If CodeSketch lacks essential collaboration or coding features, users will be dissatisfied.
       - **Reliability:** The system's consistency and dependability. Frequent crashes, errors, or data loss will severely damage user satisfaction.

### Enhanced Communication

* + - * **Simultaneous Editing with Clear Presence Indicators:** Showing who is editing which part of the code in real-time with distinct cursors and names.
      * **Interactive Whiteboarding:** A shared digital whiteboard within the session where users can sketch diagrams, flowcharts, or jot down ideas collaboratively.

### Business Outcomes

* + 1. **Increased User Base**
       - **Higher Number of Active Users:** More individuals and teams are logging in, creating sessions, and actively coding on the platform.
       - **Broader Geographic Distribution:** Users might be joining from diverse locations, both domestically and internationally.

### Monetization Opportunities

* + - * **Advanced Collaboration Features:** Enhanced access controls, detailed version history, advanced merging tools.
      * **Priority Support:** Faster response times and dedicated support channels.

### Brand Building

* + - * **Blog:** Share valuable content related to web development, collaborative coding best practices, tips for using CodeSketch, and industry insights.
      * **Tutorials and Documentation:** Provide comprehensive and easy-to-understand guides and documentation for using all of CodeSketch's features. Consider creating content in multiple languages to cater to a global audience.

### Project Management Outcomes

* + 1. **Timely Delivery**
       - **Competitive Advantage:** In the market of online collaboration tools, a reputation for reliability and timely service can be a key differentiator.
       - **Global User Base:** Ensuring timely and consistent performance across different geographical locations, considering varying internet infrastructure, is vital for a global platform.

### Budget Management

* + - * **Sustainability:** Ensuring that the platform generates enough revenue to cover its operating costs and continue to function in the long term, providing reliable service to its global user base.
      * **Growth:** Providing the financial resources necessary to invest in development, marketing, and infrastructure to attract more users and enhance the platform.

### Learning Outcomes

* + 1. **Technical Learning**
       - **Effective Use of the Platform:** Understanding the features allows users to leverage CodeSketch to its full potential, maximizing their productivity and collaboration efficiency.
       - **Improved Coding Skills:** The platform provides a practical environment for honing web development skills in a collaborative setting.

### Feedback for Future Projects

* + - * Focus on Improvement & Innovation: Suggest ways to make CodeSketch better and introduce new valuable features.
      * Be Specific & Contextual: Clearly explain what you're referring to and why it's important in your workflow.
      * Prioritize Suggestions: Indicate what issues are most critical or what features would be most impactful.

# CONCLUSION

Code Sketch, as a collaborative online code editor, holds significant potential for users globally, including those in regions like Muradnagar, Uttar Pradesh, India, by facilitating real-time teamwork on web development projects. Its success hinges on a delicate balance of factors including robust system performance that accommodates varying internet conditions, stringent security measures to protect user data and creations, intuitive usability and broad accessibility for a diverse user base, and ultimately, high user satisfaction driven by a positive and efficient collaborative experience. Continuous brand building, strategic monetization opportunities that respect different economic contexts, and a commitment to timely delivery of features and support are crucial for sustained growth. Furthermore, fostering technical learning among its users and actively soliciting and incorporating specific, prioritized, and constructive feedback will be vital in guiding future development and ensuring Code Sketch evolves to meet the ever-changing needs of the collaborative coding community worldwide.

# REFERNCES

* + - * **Shaw, M., & Garlan, D. (1996).** *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall. (A foundational text on software architecture principles.)
      * **Fowler, M. (2002).** *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Professional. (Provides context for understanding use case diagrams and other UML models.)
      * **Jacobson, I., Booch, G., & Rumbaugh, J. (1999).** *The Unified Software Development Process*. Addison-Wesley Professional. (Discusses iterative development and the role of use cases in the development lifecycle.)
      * **Norman, D. A. (2013).** *The Design of Everyday Things*. Basic Books. (While focused on product design, it offers crucial insights into usability and user-centered design principles relevant to software interfaces.)
      * **Nielsen, J. (1994).** *Usability Engineering*. Morgan Kaufmann. (A classic text on the principles and methods of usability in software development.)
      * **Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994).** *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional. (While focused on object-oriented design, understanding design patterns informs architectural decisions.)
      * **Brooks Jr., F. P. (1995).** *The Mythical Man-Month: Essays on Software Engineering*. Addison- Wesley. (A seminal work offering insights into the complexities of software project management and decision-making.)
      * **McConnell, S. (2004).** *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press. (Provides practical guidance on software development practices, including aspects relevant to collaborative coding.)
      * **Kruchten, P. (2003).** *The Rational Unified Process: An Introduction*. Addison-Wesley. (Another perspective on a structured software development process that incorporates use cases.)