Algosikhi

An Online learning platform for Data Structures and Algorithms

Submitted By

Jamil Hasan Fahim

211-35-688

Supervised By

Musabbir Hasan Sammak

Senior Lecturer
Department of Software Engineering,
Daffodil International University

This project report has been submitted in fulfilment of the requirements for the degree of **Bachelor of Science in Software Engineering**

APPROVAL

This thesis titled on "**Algosikhi-A learning platform for Data Structure and Algorithm Visualization**", submitted by **Jamil Hasan Fahim (ID: 211-35-688)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Chairman

Dr. S M Hasan Mahmud
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1

Tapushe Rabaya Toma
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Khalid Been Badruzzaman Biplob
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Dr. Md. Sazzadur Rahman
Professor
Institute of Information Technology
Jahangirnagar University

DECLARATION

I hereby declare that this project has been completed under the supervision of **Musabbir Hasan Sammak**, Senior Lecturer, Department of Software Engineering, Daffodil International University. I also affirm that this project is my original work, submitted for the degree of B.SC. in Software Engineering, and neither the entire work nor any portion has been previously submitted for another degree at this or any other university.

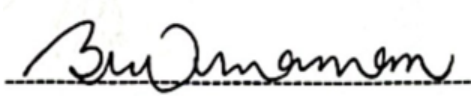
(Student's Signature)

Full Name : **Jamil Hasan Fahim**

ID Number : 2**11-35-688**

Date : 09/10/25

Certified By:



(Supervisor's Signature)

Full Name : **Musabbir Hasan Sammak**

Position : Lecturer (Sr. Scale)

Date : 09/10/25

ACKNOWLEDGEMENTS

First of all, I am grateful to The Almighty Allah for making my eligible to complete this project. Then I would like to thank my supervisor Musabbir Hasan Sammak, Lecturer (Sr. Scale) in the Department of Software Engineering. I am deeply grateful for his expert advice, sincere guidance, and valuable encouragement throughout this journey.
I am deeply grateful to everyone who contributed to this project. Their efforts and cooperation played a key role in confirming the results and added great value to the overall work.
I also want to thank all the faculty members of the Software Engineering Department for their help and support. Their valuable feedback throughout the process have been incredibly helpful.
Last but not least, I would like to thank to my parents, for their unconditional support, love and without this I would not have come this far.

DEDICATION

I therefore declare that I have done this project under the oversight of Musabbir Hossain Sammak, Lecturer (Sr. Scale), Department of Software Engineering, Daffodil International University. Also declare that neither entire record nor any portion of this record has been submitted somewhere else for my degree.

ABSTRACT

The **Algosikhi Learning Platform** is a web-based educational system designed to help students and self-learners master Data Structures and Algorithms (DSA) through interactive visualizations and real-time pseudocode execution. The platform offers a structured approach to learning, allowing users to explore theoretical explanations, run algorithm simulations, and observe pseudocode highlighting in sync with visualizations. It supports both guest users and registered users — while guest users can freely access all learning content, registered users gain access to personalized dashboards and automatic progress tracking. Built with a focus on clarity, simplicity, and engagement, Algosikhi eliminates the need for an admin panel, with updates managed manually by the developer. The project aims to make complex algorithmic concepts more understandable and enjoyable through intuitive design, smooth animations, and interactive learning features.

TABLE OF CONTENTS **Pages**

Chapter:1-Introduction

AlgoSikhi is an interactive learning platform designed to help students and aspiring programmers understand core data structures and algorithms through dynamic visualizations and structured theory. It combines theoretical explanations, step-by-step pseudocode panels, and real-time visual operations to offer a holistic understanding of algorithmic concepts.

The platform covers a wide range of topics including array manipulation, sorting/searching algorithms, STL containers (like vector,list, map, unordered map, set, unordered set, queue, priority queue, deque etc), There's also advanced operations like linked list, doubly circular linked list, circular queue graph, tree. One of the latest enhancements includes synchronized pseudocode highlighting during visual execution, which makes the learning experience more intuitive—especially for beginners who struggle to connect logic with code flow.

To promote effective learning, the platform also tracks user progress, offers login-based dashboards, and ensures users only mark topics complete after full interaction. Algosikhi is designed using HTML, CSS, and JavaScript with MongoDB and Express.js backend, making it lightweight and accessible across devices. This project addresses the gap in User-friendly algorithm learning tools and provides a practical solution for students to grasp DSA fundamentals with ease and clarity.

Problem Identification

Students often face difficulties in understanding complex algorithmic concepts by reading textbooks or theory-based tutorials alone. Many learners struggle to visualize how algorithms function step by step, especially sorting techniques, STL operations, and queue behaviors. Traditional learning platforms lack interactive and visual representations that bridge the gap between logic and execution. As a result, learners often feel overwhelmed, demotivated, and disconnected from real understanding, which affects their problem-solving confidence and coding performance.

Purpose and Justification

The Algosikhi project was developed to make algorithm learning more interactive, intuitive, and beginner-friendly. As someone who experienced the challenges of grasping algorithm flow and STL behaviors, I aimed to create a platform where users can learn with animated visualizations and pseudocode highlighting in real time. This project is particularly useful for school, college, or university students who want to strengthen their data structure and algorithm skills visually, instead of relying on dry theoretical explanations.

The motivation behind building this project was to help learners, including myself and peers, who are passionate about understanding how algorithms work practically. The platform simplifies abstract logic through step-by-step visuals, making it easier to grasp sorting algorithms, STL containers (like vector, list, map, set, queue), and their operations with minimal coding background.

Scope

Algosikhi is a web-based platform designed to visualize various algorithms and STL operations in an educational format. Its current scope includes:

1. **Array Operations:**
    - Creating and rendering arrays.
    - Visualizing sorting algorithms (Bubble, Selection, Insertion, Merge).
    - Interactive pseudocode rendering with dynamic line-by-line highlighting during execution.
2. **STL Container Operations:**
    - Visualization for containers like vector, list, map, set, queue, and priority_queue.
    - Support for basic to advanced functions like insert, erase, count, find, and iterator handling.
3. **User Progress Tracking:**
    - JWT-based login system with personalized dashboards.
    - Learning progress is auto-tracked and synced with the backend after visual completion.
4. **Interface Design:**
    - Clean layout divided into theory, pseudocode, and visualization sections.
    - Separate sections for each data structure with input fields and animation controls.
5. **Technical Stack:**
    - Frontend: HTML, CSS, JavaScript
    - Backend: Node.js, Express
    - Database: MongoDB
    - Authentication: JWT (JSON Web Token)

This project continues to expand with additional visualizations, improved animations, and more advanced topics planned for future integration.

1.1-Project Planning and Initiation

Feasibility Study (Step-by-Step)

In my Algosikhi Algorithm Visualization Platform, the feasibility study evaluates how practical and successful the project can be in helping students and learners understand core algorithmic concepts. A phased approach was used, where each step was carefully assessed before moving to the next. The study considered technical, educational, and usability factors to ensure the project would be beneficial and sustainable.

Phase 1: Preliminary Analysis & Project Scope

Definition:
Algosikhi is a web-based interactive platform focused on data structures and algorithm visualization. It combines theory, dynamic pseudocode, and animations to offer a complete learning experience.

Key Features:

- Theory + Pseudocode + Visualization layout for each algorithm.
- Interactive pseudocode highlighting during step-by-step execution.
- Sorting algorithms (Bubble, Selection, Insertion, Merge) with ascending/descending options.
- STL containers (Vector, Map, Set, Queue) with support for live operations.
- JWT-based login, personalized user dashboard, and progress tracking.
- Backend system for saving user learning progress.

Technology Stack:

- Frontend: HTML, CSS, JavaScript
- Backend: Node.js + Express
- Database: MongoDB
- Authentication: JWT

The platform is designed to be scalable, educational, and accessible to learners ranging from beginners to advanced programmers.

Phase 2: Market Feasibility Analysis (or Market Research)

The Algosikhi project targets the academic and self-learning market by identifying students' needs for visualization-based understanding of algorithms. Market research focused on understanding learners' pain points and aligning the platform's features accordingly.

Key Focus Areas:

1. **Target Audience:**
    - School, college, and university students aged 15–30.
    - Beginners in programming and DSA (Data Structures and Algorithms).
    - Visual learners who prefer animation over theoretical explanations.
2. **Market Trends:**
    - Rise in interactive learning platforms and visual-based education.
    - Growing popularity of web-based learning tools for competitive programming and coding interviews.
    - Integration of gamification, interactivity, and real-time progress tracking in education tools.
3. **Competitor Analysis:**
    - Platforms like VisuAlgo, GeeksforGeeks, and LeetCode have theoretical or code-based tools but limited real-time animations with pseudocode synchronization.
    - Most alternatives do not track personal progress or provide STL-specific visual explanations.
4. **Consumer Needs:**
    - Easy-to-understand algorithm animations.
    - Clear, line-by-line pseudocode explanation and highlighting.
    - A distraction-free, self-paced environment without needing to install software.
    - Progress tracking and dashboard to stay motivated.
5. **Pricing Strategy:**
    - Algosikhi is completely free to use for learners.

- The goal is educational impact, not monetization. However, potential future expansion may include certificate-based learning or institutional licensing if needed.

Phase 3: Technical Feasibility Analysis

Infrastructure:
The *Algosikhi* platform requires standard web development infrastructure, including a cloud-based hosting environment that supports Node.js for backend services, MongoDB for database storage, and static hosting for the frontend. Hosting is managed through platforms like Render or Vercel for efficient deployment and scalability. The system also supports HTTPS via SSL for secure data transmission.

Technology Stack:

- **Frontend:** HTML, CSS, JavaScript (vanilla JS for animations and interactivity)
- **Backend:** Node.js with Express.js
- **Database:** MongoDB for storing user data and progress
- **Authentication:** JWT-based token system
- The chosen technologies are lightweight, cost-effective, and ideal for scalable educational platforms. They allow real-time operations, maintainability, and high-performance delivery.

Integration with Third-Party Services (Optional/Future Scope):

- Although Algosikhi does not currently require payment or email services, the backend is designed to support integration with services like:
  - **Email Notifications:** Integration-ready with tools like Mailgun or Nodemailer for sending progress reports or user updates.
  - **Analytics:** Future use of tools like Google Analytics to monitor user interaction.
  - **Cloud Storage:** For storing generated user data and potentially exporting certificates or saved sessions.

Security Measures:

- User authentication is managed through **JWT (JSON Web Tokens)** ensuring protected access to dashboards and progress data.
- All communication is secured using **HTTPS (SSL encryption)**.
- **No sensitive data** such as passwords are stored in plaintext — instead, secure hashing (e.g., bycrypt) is used.
- Proper API route protection and validation measures are applied to prevent unauthorized access, tampering, or XSS attacks.

Phase 4: Financial Feasibility Analysis

Development Cost Consideration:
As a student-built academic and learning-focused platform, *Algosikhi* is developed using free and open-source technologies. This significantly reduces development costs.

- **Software Cost:** All core frameworks (HTML, CSS, JS, Node.js, MongoDB) are open-source and free to use.

- **Hosting Cost:** Initially deployed using Render (free tier) or alternatives like GitHub Pages + backend deployment platforms, which offer enough bandwidth for educational use.
- **Domain & SSL:** Optional custom domain can be registered at low cost; SSL is provided free through hosting platforms.

Hardware Requirements:
No special hardware (e.g., POS terminals or servers) is required. The project runs entirely online and is accessible through standard devices like smartphones, laptops, and desktops.

Maintenance & Scaling:
Maintenance costs are minimal since the project is built with a modular structure that allows easy updates. If usage grows significantly, backend and database plans can be upgraded gradually based on traffic and engagement levels.

Cost Efficiency & Market Viability:

- As a **free educational tool**, *Algosikhi* is cost-effective for learners and institutes.
- If expanded commercially, optional premium features (such as certificate generation, admin panels for institutions, or advanced tracking) can be introduced while keeping the base version free.
- The project's scalable architecture ensures it can grow with minimal additional cost.

Conclusion:
Algosikhi offers a technically feasible and economically sound solution for students and learners of algorithms. It minimizes development and operation costs while delivering maximum educational value. With increasing demand for visual and interactive learning tools, this project stands as a practical and beneficial initiative in both educational and personal skill development domains.

1.2-Target User Profile and Tentative Elicitation Process

Target User

The primary users of the **Algosikhi Algorithm Learning Platform** are students, beginners, and enthusiasts who want to learn and understand Data Structures and Algorithms visually. These users often struggle with theoretical understanding and seek a more interactive and visual approach to learning. Since no login is required, it is accessible to a wider audience.

The platform is especially useful for:

- University students learning DSA for the first time
- Competitive programming beginners
- Visual learners who prefer animations and pseudocode walkthroughs
- Self-learners who want to learn at their own pace without registration barriers

User Profile

Table : User Profile for Algosikhi (Learners/Students)

User Class	Note on Characteristics
Type of user	Learners, students, beginners in programming, self-learners
Age range	16-35 years
Frequency of use	Frequently — especially during learning sessions, before exams, or when exploring new topics
Mandatory	No registration needed to view content; optional login enables progress tracking
Computer experience	Basic to intermediate — able to use web browsers and interact with UI elements
Education	Secondary to university level (HSC, BSc in CSE/IT, or equivalent)
Goal	To understand algorithms step-by-step through visual animations and synced pseudocode
Language Skil	Proficient in English (Bengali support in future)
Number of users	Large (open access for all learners online)
Training	No formal training needed — platform is intuitive and guided
Others system use	Familiar with YouTube tutorials, educational apps like VisuAlgo or GeeksForGeeks
Way of working	Browse by topic → Read theory → View pseudocode → Run animations with highlights

Admin Profile

Table: User Profile for Algosikhi (Admin)

User Class	Note on Characteristics
Type of user	Admin
Age range	25-50 years
Frequency of use	Daily or multiple times a day to manage topics and customer info.
Mandatory	Yes, the admin must be actively involved in managing the platform's Backend and frontend.
Computer experience	Intermediate to advanced; skilled in navigating admin panels

Education	Secondary to university level (HSC, BSc in CSE/IT, or equivalent)
Goal	To maintain and manage the Algosikhi platform, ensure smooth visualization, handle inventory, and manage user information.
Language Skil	Proficient in English and Bengali.
Number of users	Small, limited to a few admins (1-5 people)
Training	Admins must undergo training on the system's admin panel, theory, pseudocode and visualization management (frontend) and backend response handling.
Others system use	Admin panels for managing online learning platforms, educational apps like VisuAlgo or GeeksForGeeks
Way of working	Admins work with the frontend and backend of the website, adding/editing topics, managing theories and visualizations and handling user inquiries.

Elicitation Process

1. **Informal Interviews**
   Spoke with students and peers to understand their struggles in learning DSA, especially visual understanding and language barriers.
2. **Observation**
   Observed usage patterns and difficulties while students were using similar platforms like Visulgo or trying to understand algorithms through textbooks.
3. **Feedback & Iteration**
   Based on personal and peer feedback, iteratively improved pseudocode formatting, visualization animations, and structure.
4. **Prototype Testing**
   Developed early versions and shared with students for usability feedback to improve layout, flow, and visualisation clarity.
5. **Admin Panel Requirement**
   Initially, no admin system was considered, as the platform was maintained manually. However, later feedback and system scalability needs highlighted the importance of an admin panel**.** This allows administrators to manage registered users, view their topic completion progress, and update user information (username, email, password, and topic completions) when necessary.

1.3-System Requirements

Hardware Requirements

Client (User) Requirements:

- **Desktop/Laptop:**
  - Processor: 2.0 GHz or higher
  - RAM: 4 GB or more

    - Operating System: Windows, macOS, or Linux

- **Mobile Devices:**
    - Smartphones with at least 4-inch screens
    - RAM: Minimum 2 GB
    - Internet: 5 Mbps or higher for smooth animations

Backup:

- Local or cloud backup for the project files (used manually by the creator)

Software Requirements

Frontend:

- Compatible with major browsers (Chrome, Firefox, Edge)
- Responsive layout for desktop, tablet, and mobile
- Intuitive and accessible UI
- Animated algorithm visualizations
- Pseudocode rendering and line-by-line highlighting

Backend (if any for user progress):

- Node.js with Express.js
- MongoDB for storing user progress (optional but currently used)

Other Software Features:

- Efficient image and script loading
- Minimal HTTP requests for smooth UX
- Lightweight structure, easy to maintain manually

Constraints and Dependencies

Constraints:

- **Internet Connectivity:** Required for accessing the online platform
- **Login System for Progress Tracking:** The platform is freely accessible for all users, but login is required only for those who wish to track their learning progress. JWT (JSON Web Tokens) are used for secure user authentication, and only minimal user data is collected to enable this feature.
- **Admin Panel Limitation:** Admins can view and update user data (username, email, password, topic completion). However, content updates (adding new algorithms, sections, or theory) still require manual changes by the creator in the system code.

Dependencies:

- Hosting infrastructure (e.g., Render, GitHub Pages)
- Optional MongoDB database (for user progress tracking)
- JavaScript libraries used for DOM manipulation and animations

- SSL certificate for HTTPS access and secure connections

1.4-Project Scheduling

Time Frame

Phase	Duration	Start Date	End Date
Phase 1: Planning & Requirement Gathering	2 weeks	01/01/2025	14/01/2025
Phase 2: Design & Prototyping	3 weeks	15/01/2025	08/02/2025
Phase 3: Backend Development	5 weeks	09/02/2025	16/03/2025
Phase 4: Frontend Development	5 weeks	17/03/2025	24/04/2025
Phase 5: Payment Integration & Testing	3 weeks	25/04/2025	17/05/2025
Phase 6: Admin Panel Development &Testing	3 weeks	18/06/2025	11/07/2025
Phase 7: User Testing & Quality Assurance	2 weeks	12/07/2025	27/07/2025

Chapter 2: Design and Implementation

The Algosikhi Learning Platform is a web-based system designed to help students and self-learners understand data structures and algorithms (DSA) through interactive visualisations and real-time pseudocode highlighting. Built with HTML, CSS, and JavaScript, the platform presents a clean and responsive user interface accessible across modern browsers such as Chrome, Firefox, and Edge.

The design emphasizes user experience, allowing learners to access theory, pseudocode, and step-by-step animations side-by-side without requiring login. However, for users who want to track their progress, the platform offers an optional login system with secure JWT-based authentication. The backend is implemented using Node.js, Express, and MongoDB for robust performance, with security practices like hashed passwords and token verification.

Visualization modules include core topics such as Array operations, Sorting algorithms, STL containers, and Queues. The pseudocode section dynamically updates based on the selected algorithm and highlights each line during execution, providing a powerful learning experience.

The system is also built for future scalability and maintainability. It uses modular scripts, clean UI patterns, and performance-focused practices to ensure a smooth and modern educational experience for learners at all levels.

2.1Functional requirements

FR01	Registration
Description	Users can optionally register to track their learning progress.
Stakeholder	**User**

FR02	Login/Logout
Description	Registered users can log in to their account and log out.
Stakeholder	**Admin, User**

Login as Admin:

FR03	View User Info
Description	Admin can see all kinds of users in the admin dashboard.
Stakeholder	**Admin**

FR04	Edit User Info
Description	Admin can edit all kinds of user information in the admin dashboard.
Stakeholder	**Admin**

FR05	Delete User Info
Description	Admin can delete a user or their information in the admin dashboard.
Stakeholder	**Admin**

Login as User:

FR06	Homepage Access
Description	Any user (logged in or not) can access the homepage, see the list of topics (Array, STL, Sorting, etc.).
Stakeholder	**User**

FR07	View Theory
Description	User can read theoretical explanations for each topic (e.g., sorting algorithm logic, vector operations).

Stakeholder	User

FR08	Pseudocode Display
Description	Users can view structured pseudocode corresponding to the selected operation.
Stakeholder	User

FR09	View Visualization
Description	Users can interactively view algorithm animations with step-by-step execution.
Stakeholder	User

FR10	Pseudocode Highlighting
Description	During visualization, each pseudocode line is dynamically highlighted to match the operation step.
Stakeholder	User

FR11	Topic Navigation
Description	Users can switch between different sections like Array, STL Containers, Queue, and Sorting using the UI.
Stakeholder	User

FR12	Progress Tracking
Description	Logged-in users can track completed subtopics (e.g., bubble sort, vector insert).
Stakeholder	User

FR13	Auto Progress Update
Description	Completion is marked when a user interacts with and completes a visualization (e.g., runs sort operation).
Stakeholder	User

FR14	Dashboard Access
Description	Logged-in users can view a progress dashboard with topic-wise completion stats and overall progress.
Stakeholder	User

FR15	Secure Authentication
Description	JWT-based secure login and storage of tokens in local storage to protect session data.
Stakeholder	**User**

FR16	Backend Communication
Description	The frontend communicates with the backend for login, registration, and progress updates via REST API.
Stakeholder	**User**

FR17	Error Handling
Description	Proper user feedback for invalid login, duplicate registration, or network errors.
Stakeholder	**User**

FR18	Public Access
Description	Users who don't want to log in can still browse, read theory, and use the visualizers freely.
Stakeholder	**User**

2.2 Non-Functional Requirements

To ensure quality, usability, and efficiency, the Algosikhi Learning Platform adheres to a set of non-functional requirements that cover performance, reliability, portability, and overall system behaviour. The goal is to provide a smooth learning experience for users accessing the platform across various devices and environments.

Performance

Performance is a core focus for the Algosikhi Learning Platform. The system is optimised for quick page loads, fast visualisation rendering, and minimal user wait times. This has been achieved through:

- Minimizing and optimizing JavaScript and CSS files.
- Compressing media and limiting unnecessary HTTP requests.
- Using client-side caching techniques to reduce reloading delays.
- Ensuring the platform remains responsive and functional across devices (desktop, tablet, mobile). The platform is capable of supporting multiple concurrent users accessing and interacting with visualizations simultaneously without lag.

Reliability

Reliability ensures that the learning platform is accessible and stable at all times. Key measures include:

- Error handling in frontend and backend to gracefully manage issues.
- Secure session handling through JWT-based authentication for logged-in users.
- Enforcing role-based access control, ensuring that only admins can access administrative functions. Passwords are hashed before storage, and sensitive actions (e.g., editing or deleting user data) are validated through secure server-side checks.
- Minimal downtime by deploying the application to reliable hosting platforms.
- Ensuring consistent experience during both peak and regular usage times. Even if progress tracking features are unavailable due to network issues, users can still access all learning modules freely.

Portability

The Algosikhi platform is designed to be portable and accessible on various operating systems and browsers. Features include:

- Full compatibility with all modern web browsers such as Chrome, Firefox, and Edge.
- Responsive design for cross-device usage — whether desktop, tablet, or mobile.
- Easy deployment and migration using standard hosting environments (e.g., Render, Vercel).
- Modular code architecture that allows further enhancement or deployment across other educational platforms if needed.

This ensures that the platform performs consistently across all user environments and screen sizes.

2.3System Design Overview

This section presents the high-level system design for the Algosikhi Learning Platform. It includes diagrams that illustrate user interactions, backend logic flow, data management, and functional processes. Though the system is not built on object-oriented principles, tools like use case, activity, and ER diagrams help visualize the architecture and behavior of the platform effectively.

Use Case Diagram

The Algosikhi Learning Platform is built primarily for **students**, **self-learners**, and **curious individuals** interested in mastering data structures and algorithms through interactive visualizations and real-time pseudocode. Users can access the platform freely without login, while logged-in users enjoy additional features like progress tracking and dashboards.

There are three types of users in the system:

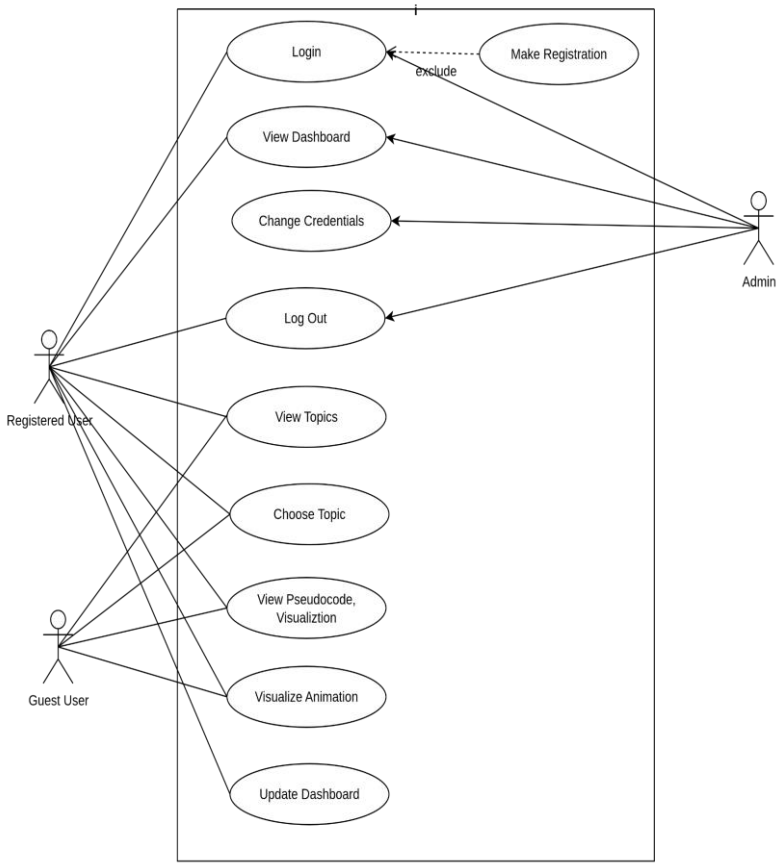
1. **Guest Users (Unregistered Visitors):**
    - Can access the homepage, browse topics, read theory, and run algorithm visualizations.
    - Have full access to visualizers and pseudocode interaction.
    - No data is stored or tracked for guests.

2. **Registered Users (With Login):**
    - Can log in securely using JWT-based authentication.
    - Can access a personal dashboard that shows their topic-wise progress.
    - Their interactions (e.g., completing sorting visualizations) are stored in the backend database to track learning completion.
3. **Admin Users:**
    - Can view all registered users and their learning progress.
    - Can update user information such as name, email, or password.
    - Can modify topic completion/uncompletion for users.
    - Can delete user accounts if necessary.

Key Use Case Scenarios:

- View theoretical explanation of algorithms.
- Select an algorithm topic (e.g., Bubble Sort, Vector Operations).
- Run a visualized simulation of the algorithm.
- Observe real-time pseudocode highlighting alongside the animation.
- (Optional) Log in or register to track completed subtopics.
- View learning dashboard (for registered users).
- Admins manage user data, progress records, and account credentials.

The system ensures a learning-focused, interactive, and structured experience. While students focus on their learning journey, admins ensure smooth management of users and progress records.



Use case Diagram

Case Description:

Make Registration

Use Case	**Make Registration**
Goal	A new user will be able to register into the system
Pre Condition	The user must have access to the registration page
Success End Condition	User account is created successfully
Failed End Condition	Registration fails due to input error or server issue
Primary Actors	Guest Users
Secondary Actors	System
Trigger	User clicks on "Register" button
Description/Main case scenario	1. User navigates to the registration page 2. Enters a valid username and password 3. Clicks the "Register" button 4. The system creates an account using JWT authentication logic 5. User is notified of successful registration 6. Redirect to login page
Alternative flows	1a. Server not responding 1a1. Try again later" 2a. Username/password field empty 2a1. All fields are required 3a. Passwords do not match 3a1. Passwords must match
Quality Requirements	Should handle errors gracefully and provide instant feedback to user

Login

Use Case	**Login**
Goal	Admin and registered user will be able to log into the system securely
Pre Condition	The user must already be registered or be an admin
Success End Condition	Actor is logged in and redirected to dashboard.
Failed End Condition	Login fails and error message is shown
Primary Actors	Admins, Registered User
Secondary Actors	System
Trigger	Actor clicks on "Login" button

Description/Main case scenario	1. Actor accesses the login option on the homepage 2. Actor enters valid username/email and password 3. System verifies the credentials using the backend 4. Actor is redirected to dashboard/homepage
Alternative flows	1a. Actor did not get the login option 1a1. “Server not responding” 2a. Actor provided invalid credentials 2a1. “Incorrect username or password”
Quality Requirements	Login process must be secure and fast with valid input and show proper error messages for incorrect inputs.

View Dashboard

Use Case	**View Dashboard**
Goal	Logged-in users or admins can view topic-wise progress
Pre Condition	Actor must be logged in
Success End Condition	Dashboard loads with accurate user data
Failed End Condition	Dashboard does not load or shows incorrect data
Primary Actors	Users, Admins
Secondary Actors	Backend Service
Trigger	Actor clicks on "Login" button and successfully logs in
Description/Main case scenario	1. Actor navigates to dashboard 2. System verifies JWT token 3. System fetches user’s progress data from database 4. Dashboard displays completed and pending topics
Alternative flows	1a. Token is invalid or expired: 1a1. Redirect to login page 2a. No progress data found: 2a1. Show empty dashboard with message "No progress yet" 3a. Server error while fetching data: 3a1. Show "Unable to load dashboard"
Quality Requirements	The dashboard should be fast, responsive, and accurately reflect learning progress

Change Credentials

Use Case	**Change Credentials**
Goal	Admins can update any user's credentials or topic completion status.
Pre Condition	Admin must be logged in
Success End Condition	Credentials or progress data are updated successfully in the system.
Failed End Condition	Update fails and an error message is displayed.
Primary Actors	Admin
Secondary Actors	Backend System
Trigger	Admin clicks on "Login" button and successfully logs in
Description/Main case scenario	1. System shows list of users in the dashboard 2. Admin selects one to edit. 3. Admin updates user credentials (username, email, password) and progress status (completed/uncompleted). 4. System validates input and updates database. 5. Confirmation message displayed.
Alternative flows	1a. Invalid input provided: 1a1. Show "Invalid input, please correct errors." 2a. Database update fails: 2a1. Show "Update failed, try again later." 3a. Unauthorized attempt: 3a1. Show "You are not authorized to perform this action."
Quality Requirements	Updates must be secure, validated, and restricted by user role.

Log out

Use Case	**Log out**
Goal	Logged-in actors can safely log out from the platform
Pre Condition	Actor must be logged in
Success End Condition	Actor is logged out and redirected to homepage
Failed End Condition	Logout process fails
Primary Actors	Admins, User
Trigger	Actor clicks the "Log Out" button

Description/Main case scenario	1. Actor clicks on "Log Out" 2. System removes token from local storage 3. Actor is redirected to homepage
Alternative flows	1a. Token is not cleared properly: 1a1. Show message "Unable to log out, please try again"
Quality Requirements	The logout function should work instantly and securely

View Topics

Use Case	**View Topics**
Goal	User can browse all available algorithm/data structure topics
Pre Condition	Homepage should be displayed
Success End Condition	Topics are loaded and visible
Failed End Condition	Topics fail to load
Primary Actors	Guest User, Registered User
Secondary Actors	
Trigger	User opens project
Description/Main case scenario	1. User visits topic section 2. All available topics are fetched and displayed 3. Each topic links to subtopics or visualization pages
Alternative flows	1a. Topic fetch fails: 1a1. Show message "Unable to load topics"
Quality Requirements	Topics should be displayed clearly and load quickly

Choose Topic

Use Case	**Choose Topic**
Goal	User selects a specific topic to view theory or visualization
Pre Condition	Selected topic must be loaded
Success End Condition	Selected topic is loaded
Failed End Condition	Topic fails to load
Primary Actors	Guest User, Registered User
Secondary Actors	
Trigger	User clicks on a specific topic e.g. “Array Sort”

Description/Main case scenario	1. User selects a topic from the list 2. System loads the relevant theory and visualization page
Alternative flows	1a. Page fails to load: 1a1. Show error "Topic could not be opened"
Quality Requirements	Smooth transition from topic list to selected topic view

View Pseudocode, Visualization section

Use Case	**View Pseudocode, Visualization**
Goal	User can see both the pseudocode and visual animation for a selected topic
Pre Condition	Topic must be selected
Success End Condition	Pseudocode and visualizer are loaded and visible
Failed End Condition	Either pseudocode or animation does not load
Primary Actors	Guest User, Registered User
Secondary Actors	
Trigger	Page loads after topic selection
Description/Main case scenario	1. User selects a topic 2. System displays corresponding pseudocode 3. User interacts with visualization (e.g., sorting steps) 4. Pseudocode highlights dynamically based on animation
Alternative flows	1a. Visualization fails: 1a1. Show "Visualizer could not load" 2a. Pseudocode does not highlight: 2a1. Show static pseudocode with notice
Quality Requirements	Both features should load quickly and remain synced during animation

Visualize Animation

Use Case	**Visualize Animation**
Goal	User selects a specific topic to view theory or visualization
Pre Condition	Pseudocode and visual interface must be loaded
Success End Condition	Animation runs correctly step by step
Failed End Condition	Animation crashes or doesn’t start
Primary Actors	Guest User, Registered User
Secondary Actors	
Trigger	User clicks "Start Visualization"

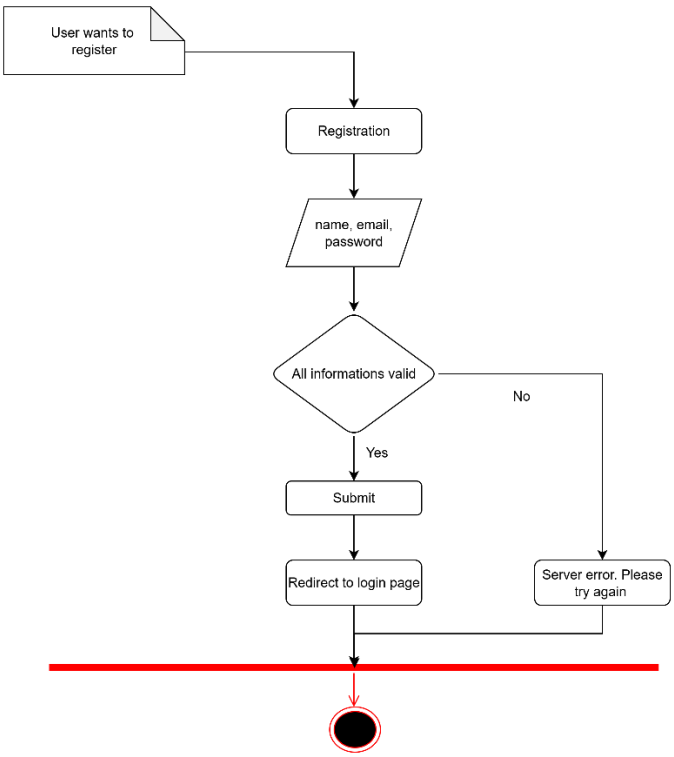
Description/Main case scenario	1. User clicks "Start" 2. Visual animation begins 3. Steps are highlighted in pseudocode as animation proceeds 4. User can pause, reset, or replay
Alternative flows	1a. Animation buttons don't respond: 1a1. Show message "Interaction failed, reload page"
Quality Requirements	Animations must be smooth, interactive, and synced with pseudocode

Update Dashboard

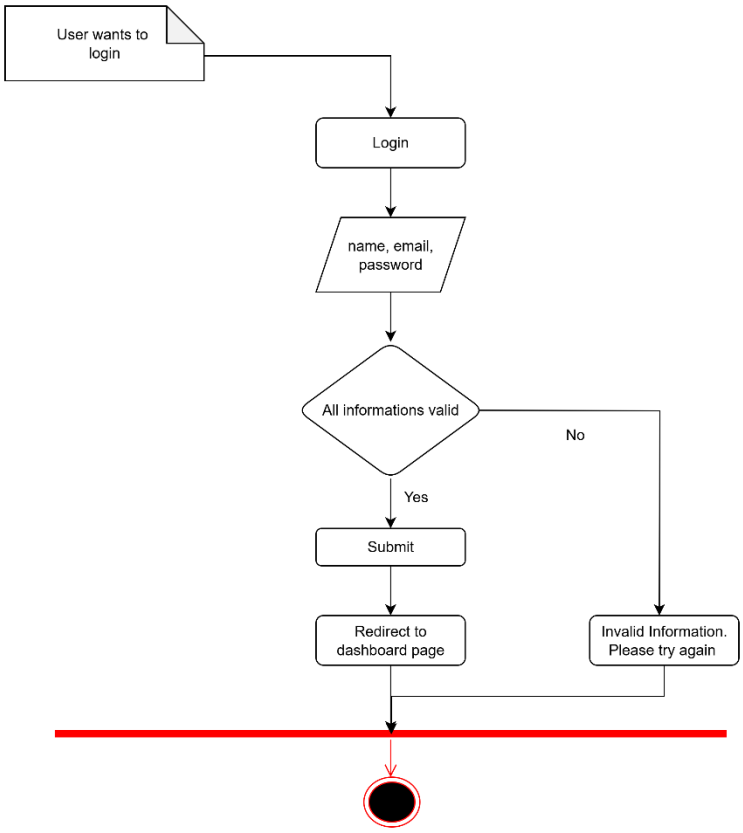
Use Case	**Update Dashboard**
Goal	When a logged-in user finishes a visualization, the dashboard updates learning progress
Pre Condition	User must be logged in and have completed a visualization
Success End Condition	Dashboard reflects new completion status
Failed End Condition	Dashboard does not update
Primary Actors	Registered User
Secondary Actors	Backend API
Trigger	Visualization is completed
Description/Main case scenario	1. User completes visualization 2. System detects completion and sends update to backend 3. Backend stores updated progress in database 4. Dashboard reflects updated completion
Alternative flows	1a. API call fails: 1a1. "Progress not saved, try again later" 2a. Token is invalid: 2a1. Redirect to login
Quality Requirements	Progress should only update after valid visualization interaction

Activity Diagram

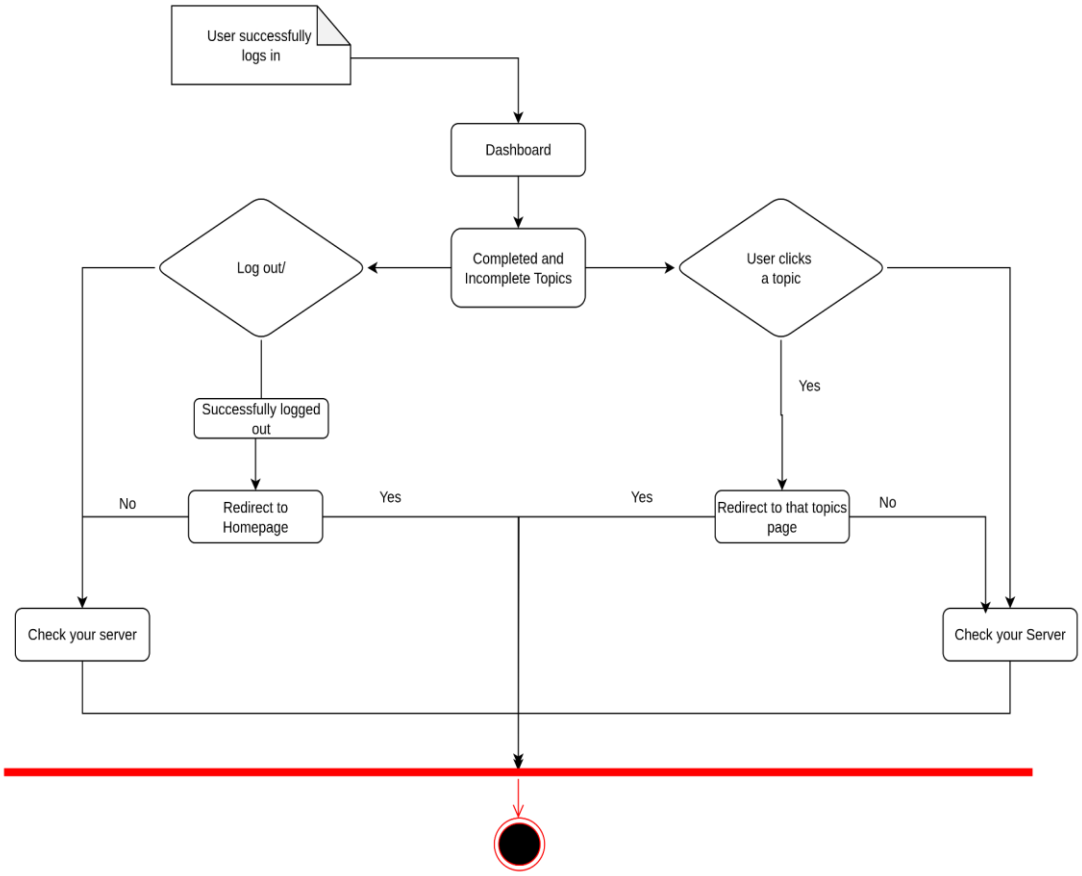
Make Registration



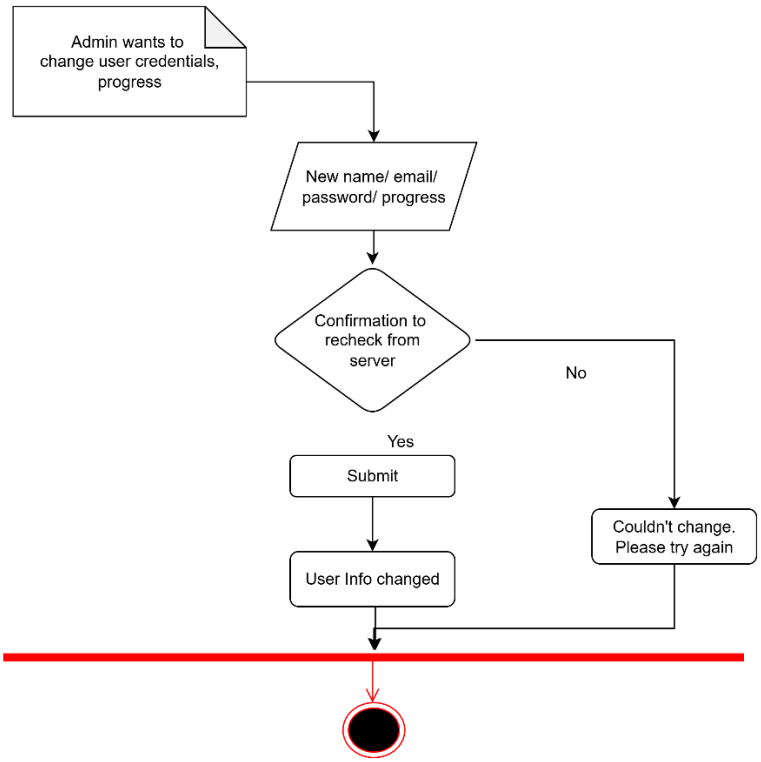
Login



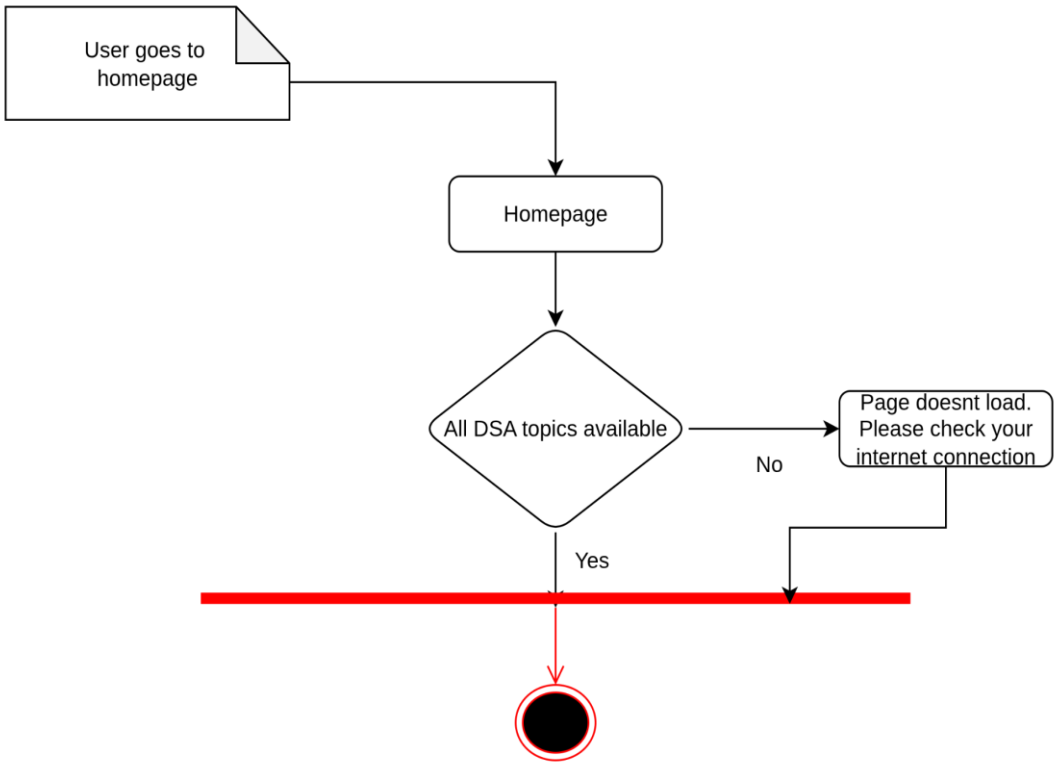
View Dashboard



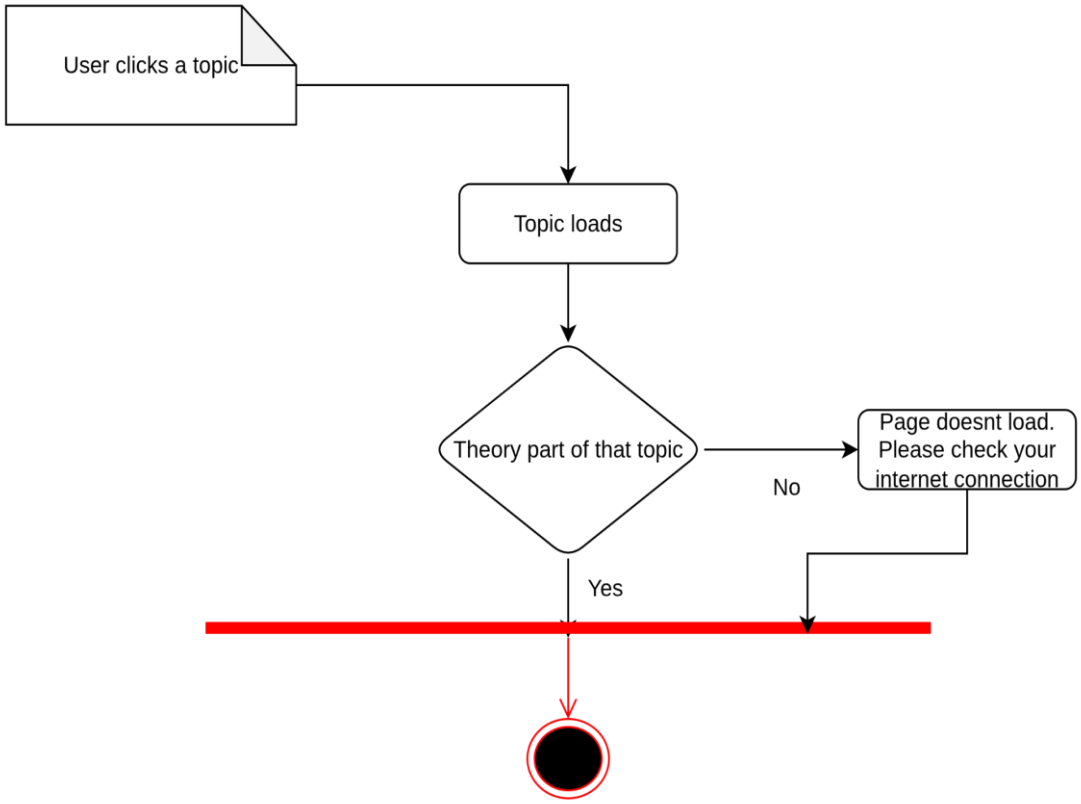
Change Credentials



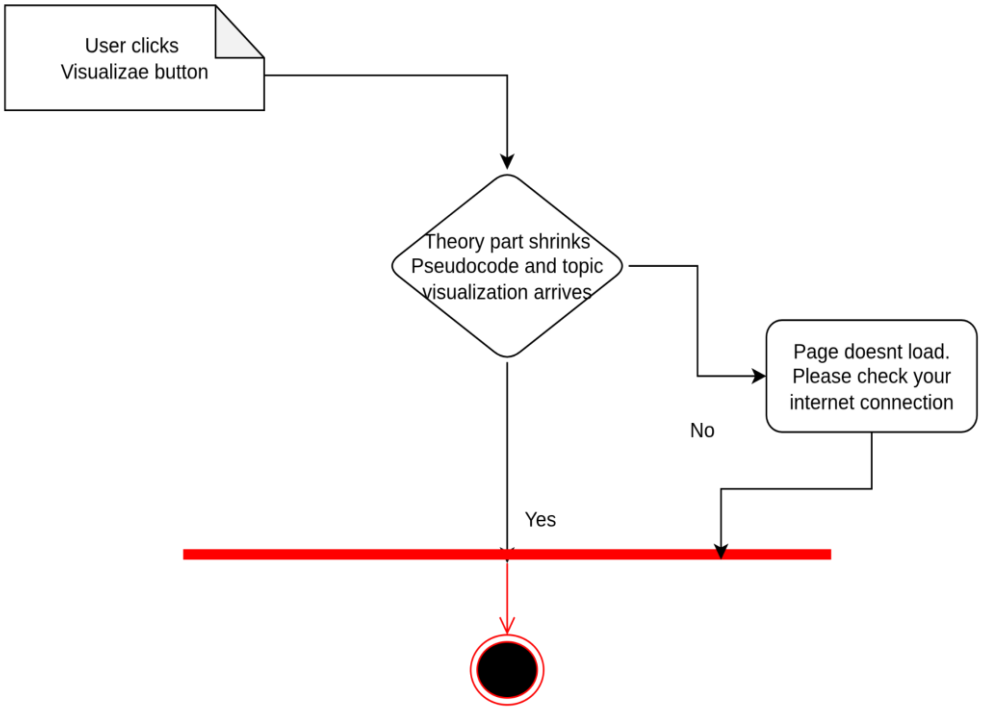
View Topics



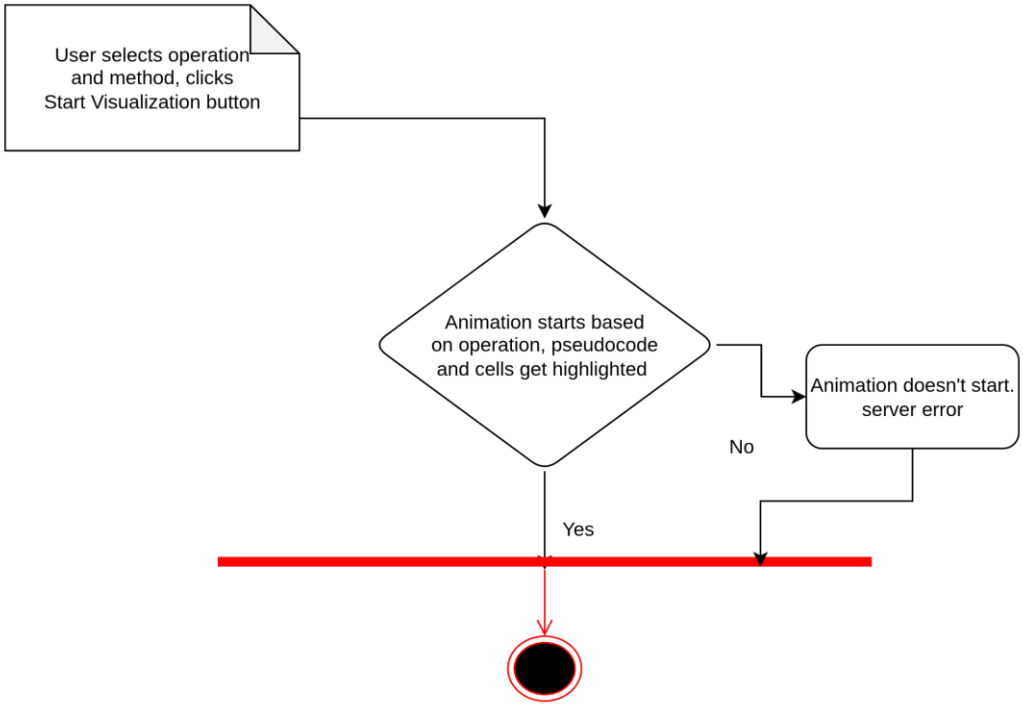
Choose Topic



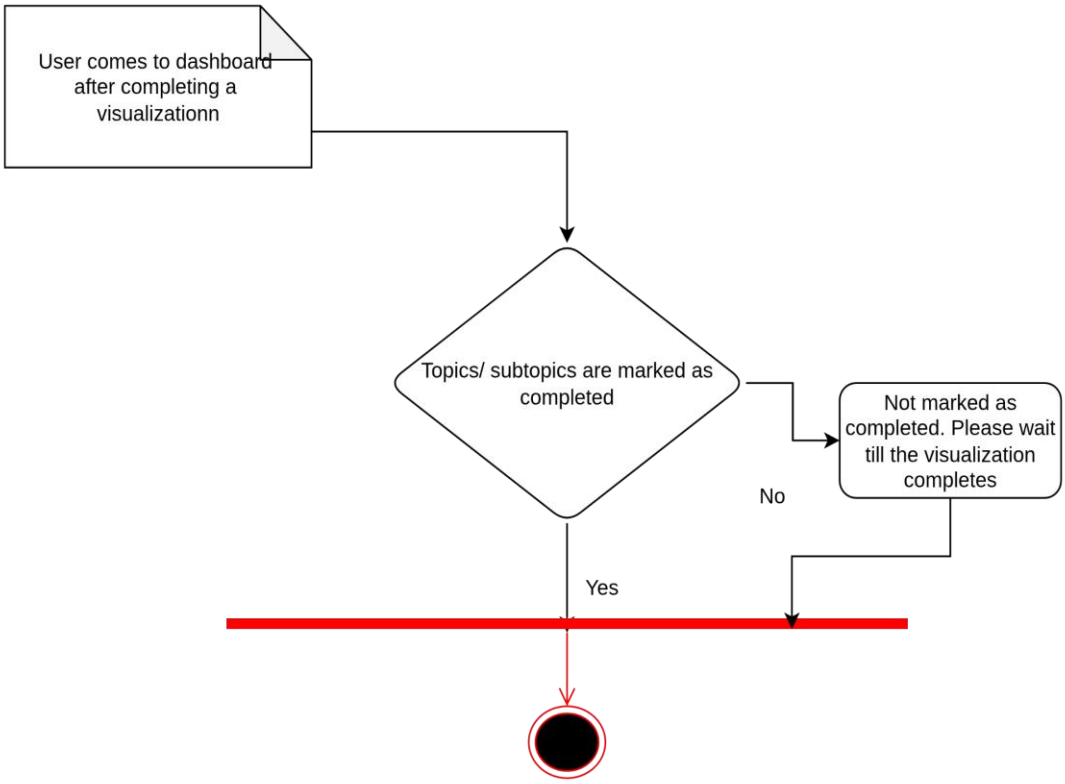
View pseudocode, visualization section



Visualize Animation

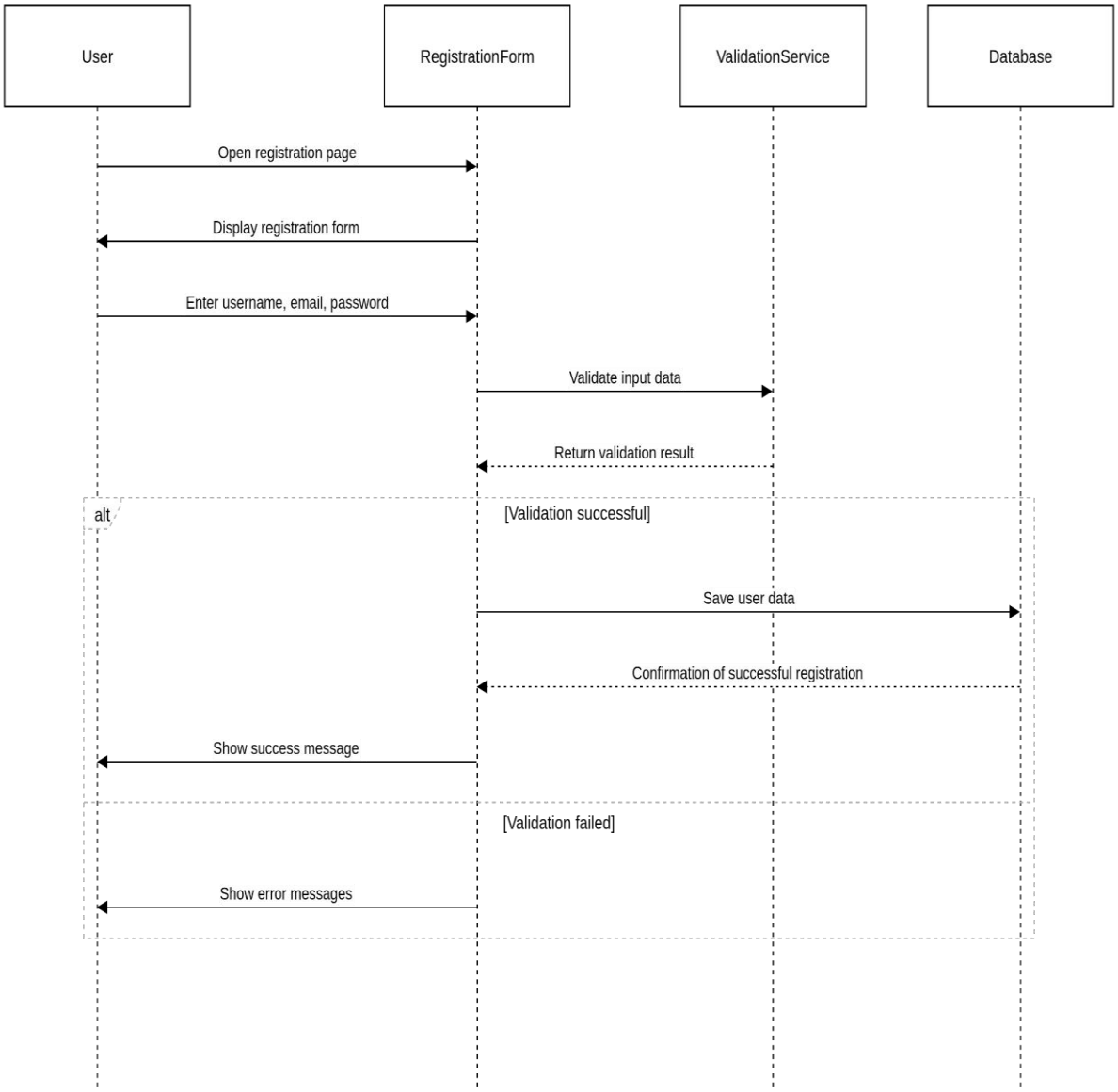


Update Dashboard

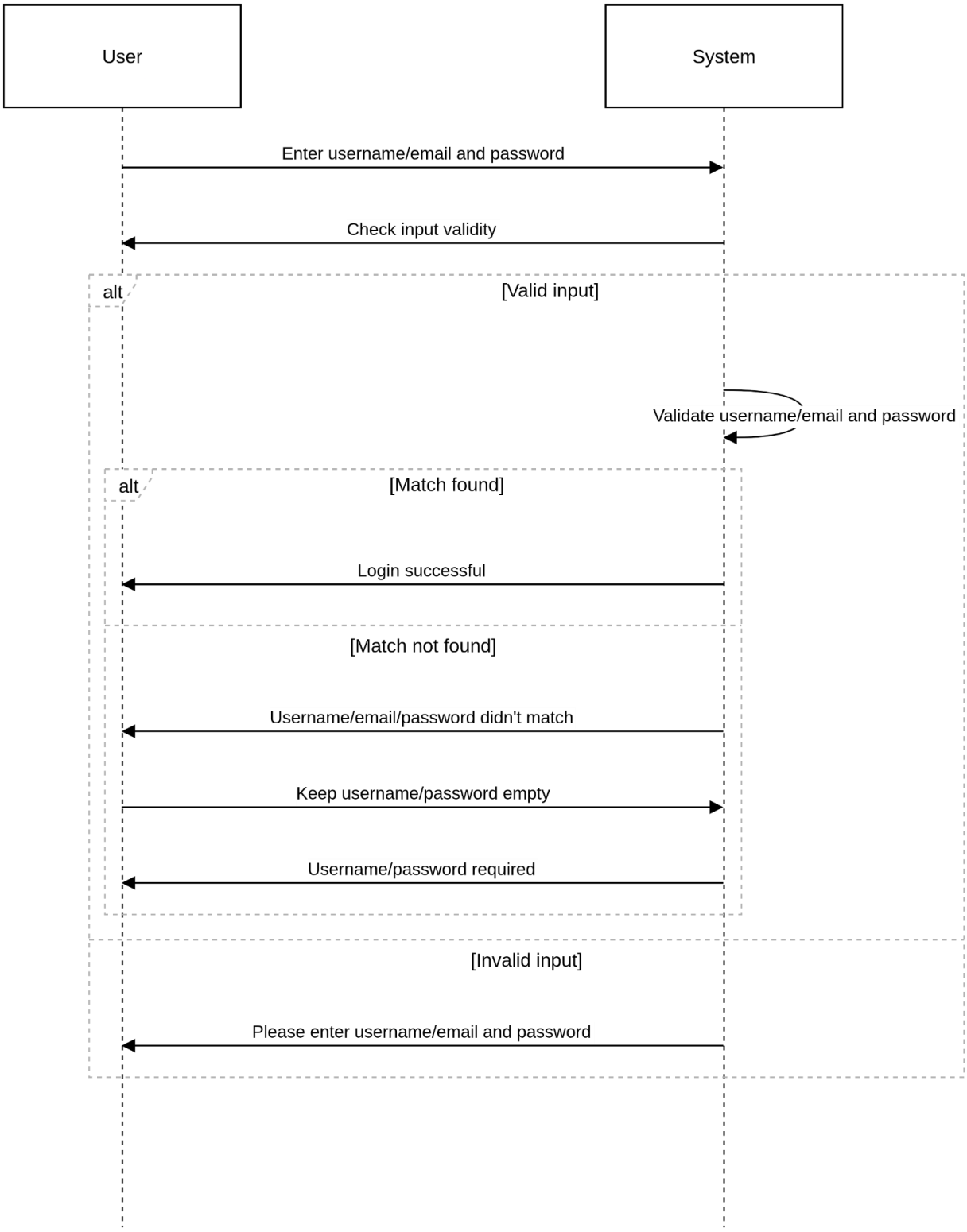


Sequence Diagram

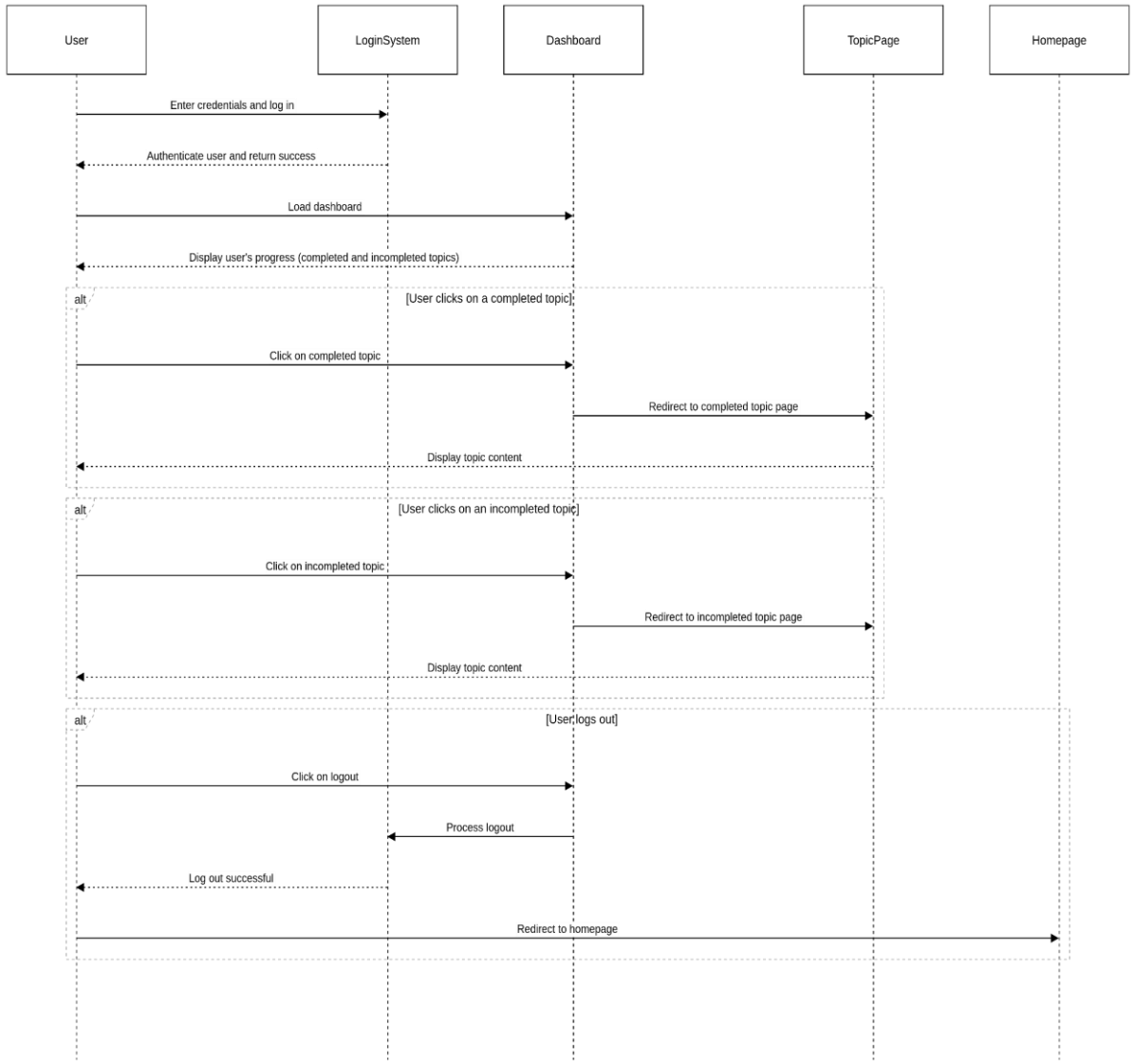
Make Registration



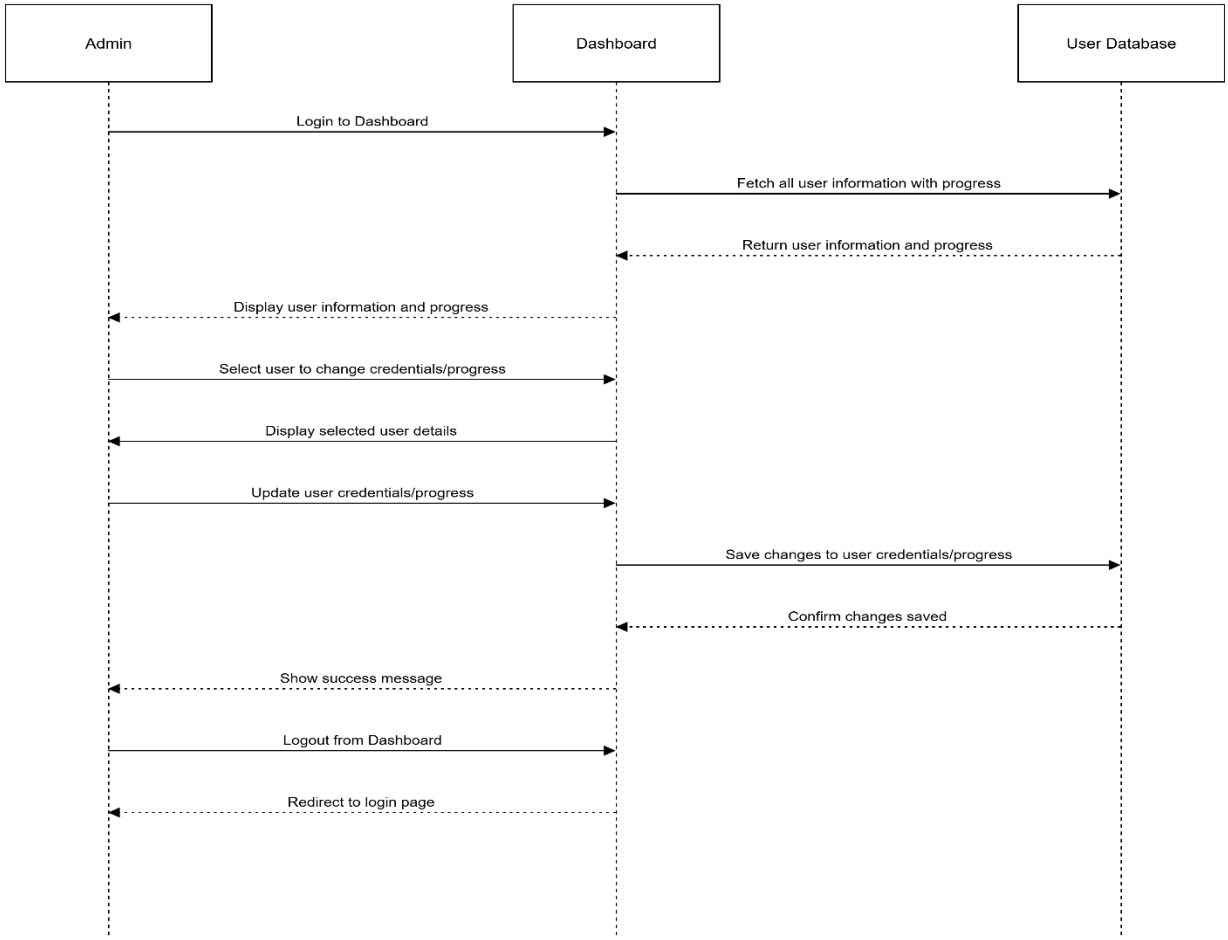
Login



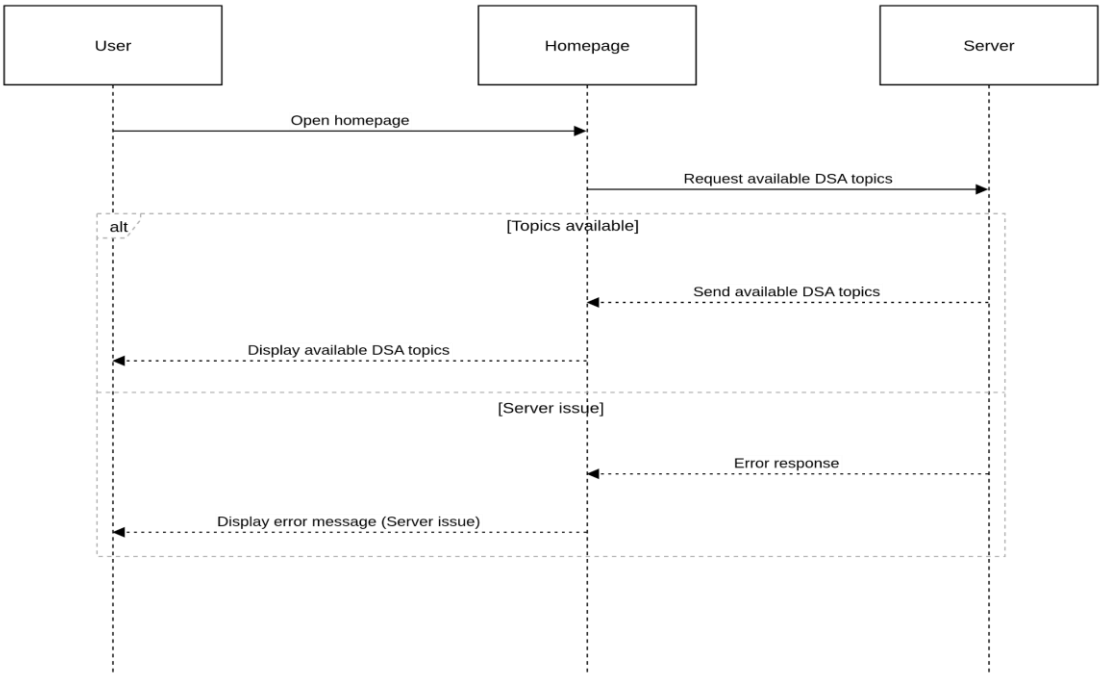
View Dashboard



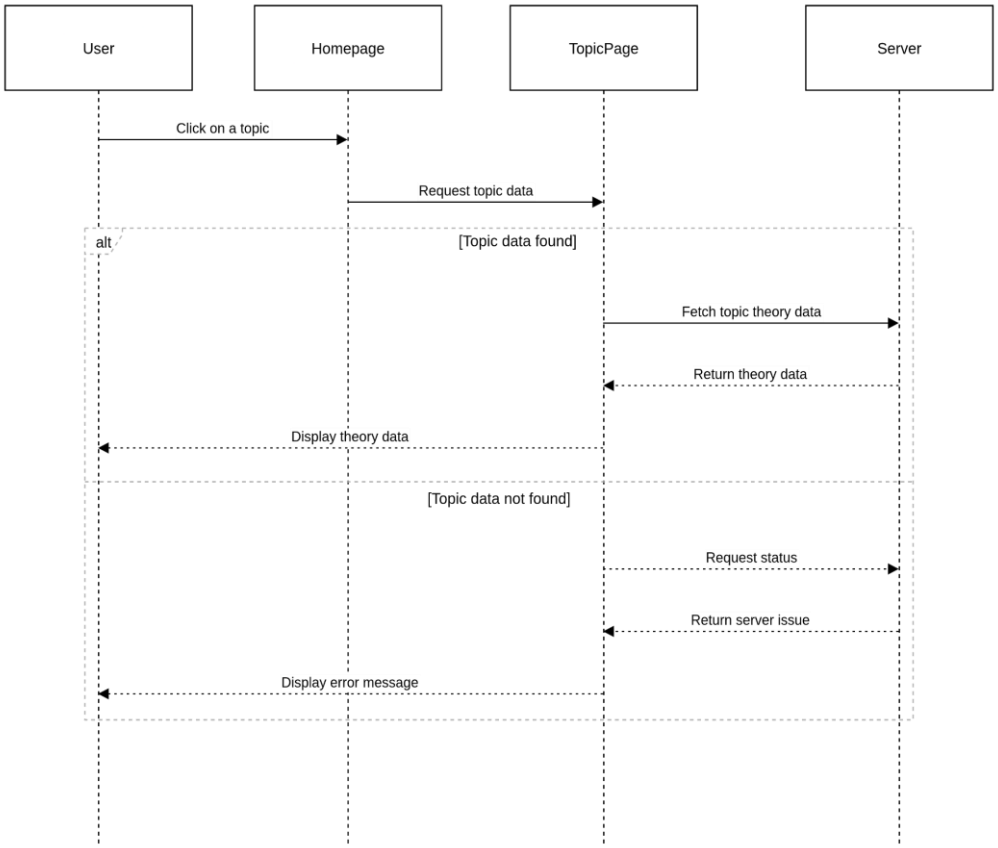
Change Credentials



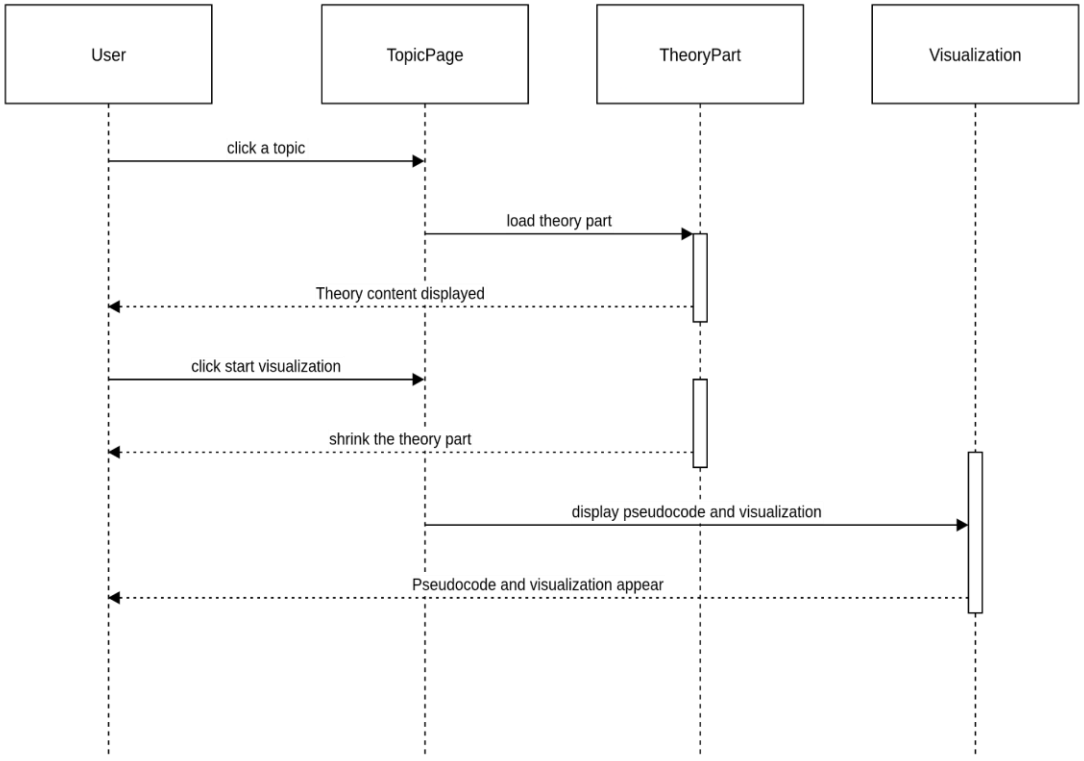
View Topics



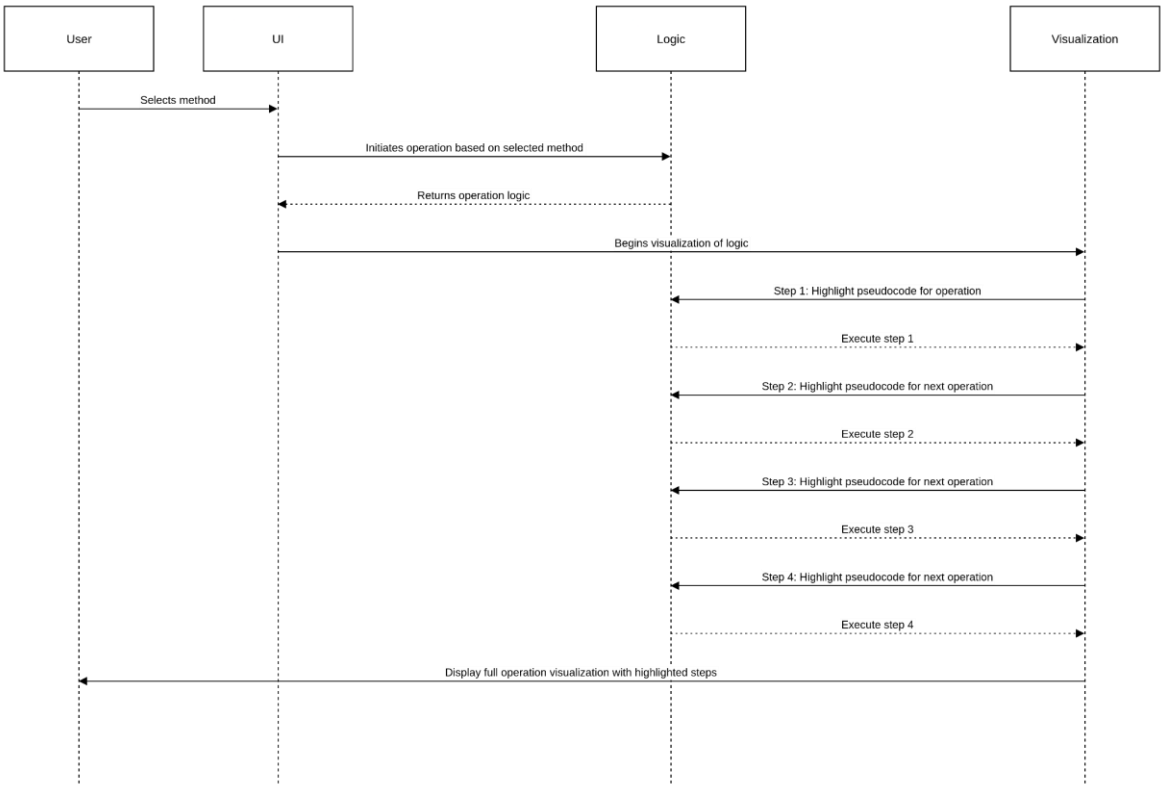
Choose Topics



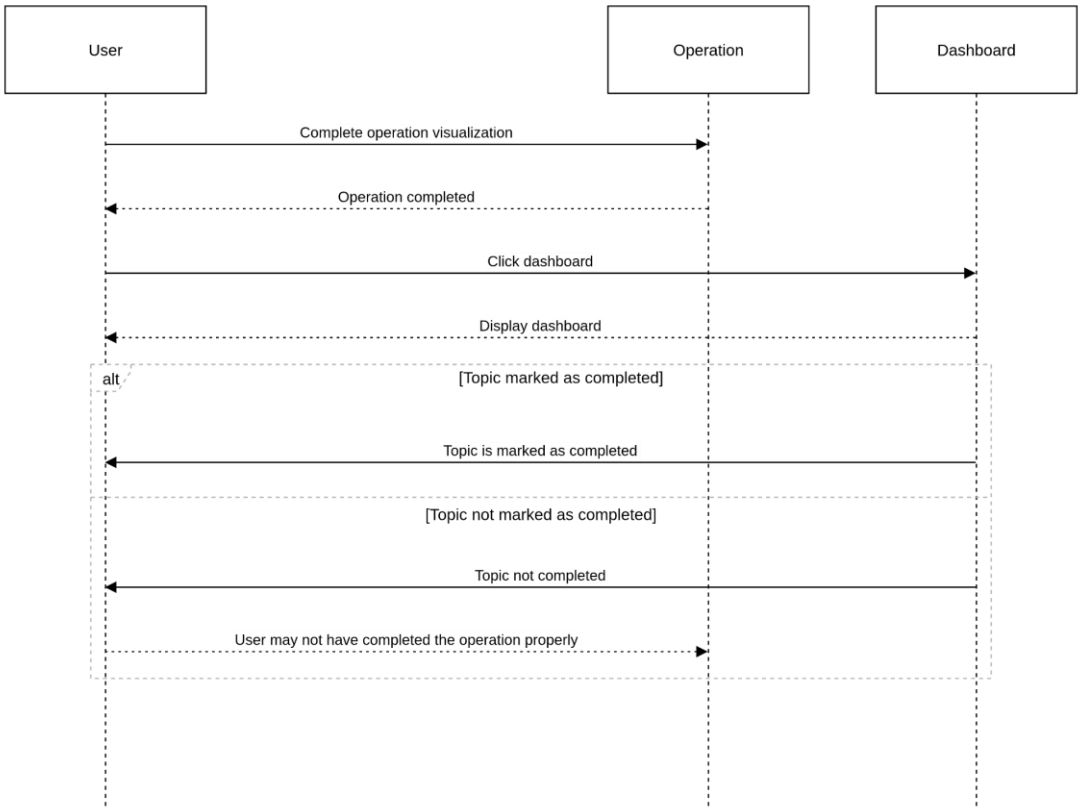
View Pseudocode, Visualization section



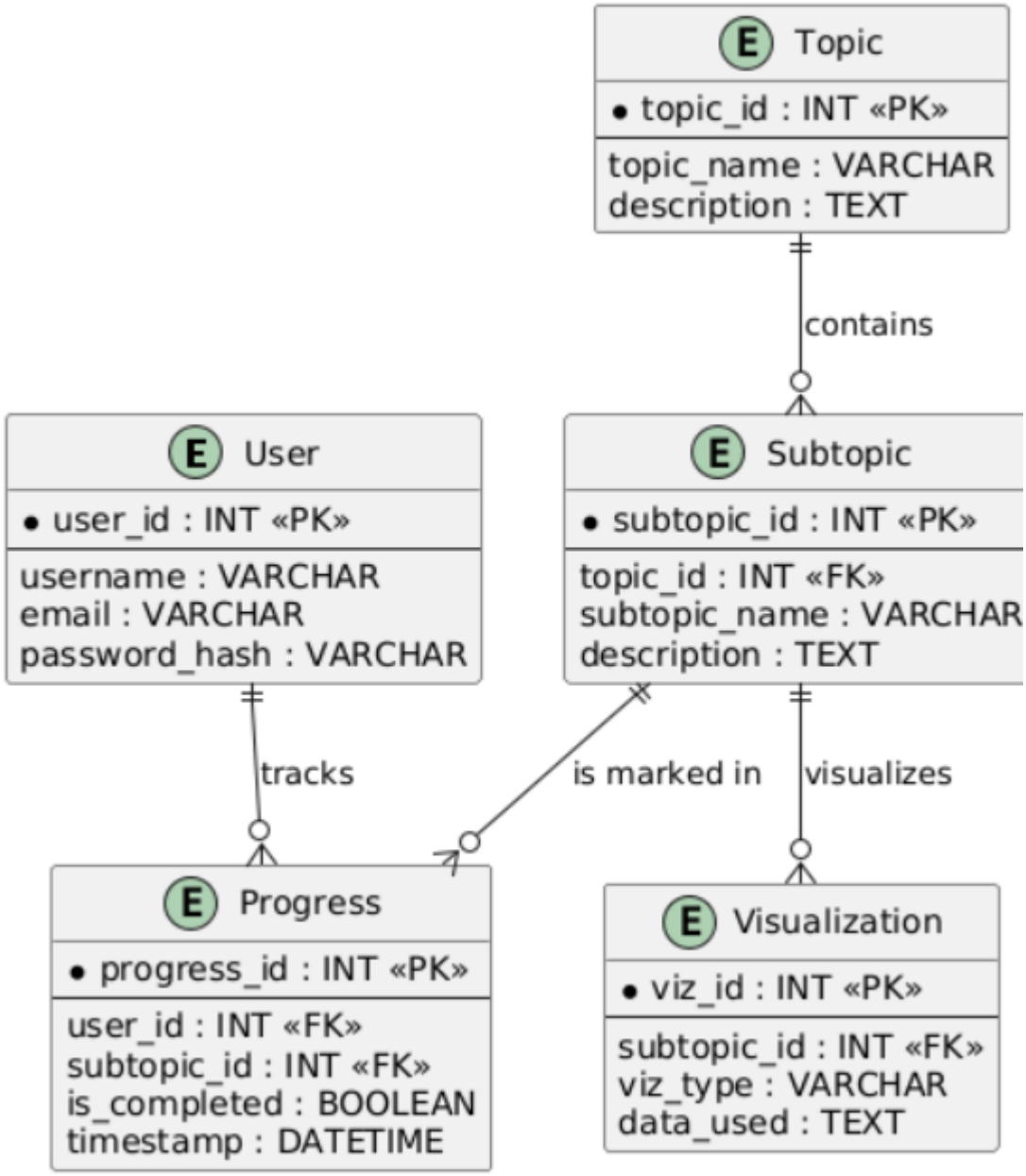
Visualize Animation



Update Dashboard



ER Diagram:



2.4 Coding: Appendix A

The Coding Appendix provides an overview of the core functionalities implemented in the Algosikhi learning platform. This includes sample code snippets responsible for operations like dynamic pseudocode rendering, user authentication with JWT, and interactive algorithm visualizations. Each snippet demonstrates how technologies like HTML, CSS, JavaScript, and Node.js were combined to deliver a smooth, secure, and educational experience. The platform ensures code readability, modularity, and performance for future scalability and maintenance.

All source code is available on GitHub:

Frontend: https://github.com/Fahimjh/AlgoSikhi

Backend: https://github.com/Fahimjh/AlgosikhiBackend

Chapter-3 Software Testing

Introduction

This chapter covers the **testing phase** of the Algosikhi learning platform. Testing is a critical part of the software development life cycle, ensuring that features work as expected, bugs are identified and fixed, and the platform is usable across devices.

The platform supports interactive algorithm learning and progress tracking. Core testing focused on ensuring that visualization tools, pseudocode synchronization, user login/logout, and learning progress storage function correctly. This chapter also discusses the test strategy, key test cases, pass/fail criteria, and efforts made to guarantee reliability, performance, and user satisfaction.

3.1 Testing Features

Features to be tested

a. User Registration
b. User Login
c. Visualization Browsing (Array, Sorting, STL, etc.)
d. Pseudocode Rendering and Highlighting
e. Interaction with Visualizations (e.g., Bubble Sort animation)
f. Progress Tracking
g. Token-based Authentication (JWT)
h. Dashboard Display (based on user data)
i. Logout Functionality

3.2 Testing Strategies

Test Approach

For the Algosikhi learning platform, a well-defined testing strategy was followed to validate all major functionalities. The types of testing include:

- **Functional Testing**: Ensuring each feature works correctly — such as login, pseudocode synchronization, animation triggers, and dashboard loading.
- **Usability Testing**: Verifying that the user interface is easy to navigate for all levels of learners.
- **Performance Testing**: Making sure that visualizations load smoothly and operations like animations run without lag.
- **Integration Testing**: Ensuring that the backend and frontend communicate properly — e.g., verifying progress save/load via APIs.
- **Security Testing**: Validating JWT-based authentication and protection of user-specific progress data.
- **Regression Testing**: Rechecking all features after any new functionality (e.g., new algorithm added) to ensure nothing else breaks.

Pass/Fail Criteria

- **Functional Testing**: A feature passes if it performs the expected operation and handles edge cases without crashing. It fails if it misbehaves, crashes, or gives incorrect results.
- **Performance**: Platform should load within 2–3 seconds on standard broadband, and visualizations should animate smoothly without frame drops.
- **Security**: JWT tokens must be correctly validated, and no unauthorized access should be possible to user data or dashboards.
- **Usability**: Interface should be intuitive and understandable. Any confusing behavior or misleading design fails the test.

Any feature failing a test is documented, debugged, fixed, and retested before platform release.

3.3 System testing (Test cases with report)

Table-1: Test case report for User Registration

Test Case: 3.3.1		**Test Case Name:** User Registration					
System: Algosikhi		**Subsystem:** User Authentication					
Designed by: User		**Design Date:** 12/07/2025					
Executed by: User		**Execution Date:** 13/07/2025					
Description: The user registers providing valid registration information.							
Pre-condition: The user accesses the registration page.							
Step	**Name**	**Email**	**Password**	**Retype password**	**Response**	**Pass/ Fail**	**Comment**
1	User	user@gmail.com	12345678	12345678	Registration successful	Pass	Registration is successful with valid input.
2	UseR	user@gmail.com	12345678	12354678	Name is Invalid	Fail	Write your name again
3	User	user@gmail.com	12345678	12345698	Password missmatch	Fail	Please write your password
4	User	usor@gmail.com	12345678	12345678	Email is not valid	Fail	Please write right email
Post-condition: The user is successfully registered with valid information, and the process is complete.							

Table-2: Test case report for User/Admin Login

Test Case: 3.3.2			**Test Case Name:** User Login			
System: Algosikhi			**Subsystem:** User Authentication			
Designed by: User			**Design Date:** 14/07/2025			
Executed by: User			**Execution Date:** 15/07/2025			
Description: The user logs into the system to access their personalized dashboard and track learning progress						
Pre-condition: The user must be registered in the system.						
Step	**Name**	**Email**	**Password**	**Response**	**Pass/ Fail**	**Comment**
1	User	user@gmail.com	1234568	Login successful	Pass	Login worked with valid data
2	Admin	admin@gmail.com	12345678	Emaill not found	Fail	Email doesn't exist
3	User	user@gmail.com	adfdfds	Incorrect Password	Fail	Password is incorrect
4	User		1234568	Email is not valid	Fail	Email input is empty
5	User	user@gmail.com		Password required	Fail	Password input is empty
Post-condition: User is successfully registered with valid information, and the process is complete.						

Table-3: Test case report for View Dashboard

Test Case: 3.3.3				**Test Case Name:** View Dashboard	
System: Algosikhi				**Sub System:** Dashboard	
Designed by: User				**Design Date:** 16/07/25	
Executed by: User				**Execution Date:** 17/07/25	
Description: The user views their personal learning dashboard, while the admin views all users' dashboards and progress details.					
Pre Condition: The user or admin must be logged in into the system with admin privileges.					
Steps	**Role**	**Action**	**Expected Response**	**Pass/Fail**	**Comment**
1	User	Open Dashboard	Display personal topics, subtopics, and progress bar	Pass	User sees only their own progress
2	User	Open Dashboard without completing any topic	Dashboard shows 0% progress	Pass	Works correctly with no progress
3	Admin	Open Dashboard	Display all registered users with progress stats	Pass	Admin sees user list and progress
4	Admin	Open Dashboard with no user registered	"No users found" message	Pass	Handles empty data gracefully
Post-condition: User sees their own learning progress; Admin sees all users and their respective topic progress.					

Table-4: Test case report for Change Credentials

Test Case: 3.3.4			**Test Case Name:** Change Credentials		
System: Algosikhi			**Sub System:** Admin Pamel		
Designed by: User			**Design Date:** 18/07/25		
Executed by: User			**Execution Date:** 19/07/25		
Description: The admin modifies a registered user's credentials (username, email, password) or topic progress.					
Pre Condition: The admin must be logged in into the system with admin privileges.					
Steps	**Target User**	**Change Requested**	**Response**	**Pass/Fail**	**Comment**
1	User1@gmail.com	Email → newuser1@gmail.com	Email updated successfully	Pass	User email updated
2	User2@gmail.com	Password → newPass123	Password updated successfully	Pass	User password updated
3	User3@gmail.com	Account Deletion	Account delated successfully	Pass	User account delated
4	User4@gmail.com	Change Username →""	Username required	Fail	Username not updated
5	User5@gmail.com	Mark topic→Array Sort	Topic completion updated	Pass	Topic status updated
6	User6@gmail.com	Unmark topic →Vector Intro	Topic uncompletion updated	Pass	Topic status updated
Post-condition: User credentials or topic completion data are updated in the database as per admin changes.					

Table-5: Test case report for View Topic

Test case:3.3.5		**Test case name:** Topic Browsing			
System:Algosikhi		**Subsystem:** Visualization Interface			
Designed By: User		**Design Date**: 20/07/25			
Executed By: User		**Execution Date**: 21/07/25			
Description: The user browses available algorithm and data structure topics.					
Pre-condition: The user must access the homepage of the website.					
Steps	**Name**	**Action**	**Response**	**Pass/Fail**	**Comment**
1	User	Open Homepage	Homepage loaded with topics	Pass	Topics visible without login
2	User	Click on "Array Sort"	Array Sorting theory section loads	Pass	Navigation is working
3	User	Click on "Visualize Array Sort"	Theory section shrinks, pseudocode and visualization section appears	Pass	All sections opens correctly
4	User	Click on "Vector Sort"	Vector Sorting theory section loads	Pass	Navigation is working
5	User	Click on "Visualize Vector Sort	Theory section shrinks, pseudocode and visualization section appears	Pass	All sections opens correctly
6	User	Click on "list Operation"	List Operation theory section loads	Pass	Navigation is working
7	User	Click on "Visualize list Operations"	Theory section shrinks, pseudocode and visualization section appears	Pass	All sections opens correctly
8	User	Click a broken link	404 not found	Fail	Link needs to be fixed
Post-condition: The user can explore different topics and access their theory, pseudocode and visualization pages without logging in.					

Table-6: Test case report for Learning Progress Tracking

Test case:3.3.6		**Test case name:** Learning Progress Tracking			
System: Algosikhi		**Subsystem:** Progress Management			
Designed By: User		**Design Date**: 22/07/25			
Executed By: User		**Execution Date**: 22/07/25			
Description: The user interacts with a visualization (e.g., runs a sorting animation), and the system tracks the topic as completed.					
Pre-condition: User must be logged in and have accessed a topic's (Array Sort for this test case) visualization page.					
Steps	**Name**	**Action**	**Response**	**Pass/Fail**	**Comment**
1	User	Click on "Visualize Array Sort"	Theory section shrinks, pseudocode and visualization section appears	Pass	Visualizatio n ready
2	User	Select a sorting Operation, Sorting Method, clicks Sort button	Array Sorting theory section loads	Pass	Interaction detected
3	System	Checks completion flag	Completion marked in db	Pass	Marked as completed successfully
4	User	Skips animation, goes to other page	Not marked Complete	Fail	Worked as intended
Post-condition: The topic is marked as "completed" only when the user interacts with the visualizer, ensuring accurate progress tracking.					

Table-7: Test case report for View Learning Progress History

Test case:3.3.6	**Test case name:** View Learning History
System: Algosikhi	**Subsystem:** Progress Dashboard
Designed By: User	**Design Date**: 23/07/25
Executed By: User	**Execution Date**: 23/07/25

Description: The user views their learning history and completed topics on the dashboard.					
Pre-condition: User must be logged in and have completed at least one topic.					
Steps	**Name**	**Action**	**Response**	**Pass/Fail**	**Comment**
1	User	Login	Dashboard Visible	Pass	User Authenticated
2	User	Clicks Dashboard	List of completed topics shown	Pass	Progress loaded correctly
3	System	Fetches data	Shows correct number of topics	Pass	Accurate data from db
4	User	Clicks on a completed/uncompleted topic	REdirect to that topics page	Pass	Navigation works
Post-condition: User successfully views completed topics and can navigate to any for review.					

Table-8: Test case report for Topic Navigation and Visualization Access

Test case:3.3.8	**Test case name:** Topic Navigation and Visualization Access
System:Algosikhi	**Subsystem:** Visualization Access
Designed By: User	**Design Date**: 23/07/25
Executed By: User	**Execution Date**: 24/07/25
Description: User can browse and access any available topics without login	
Pre-condition: None (Open access)	

Steps	**Name**	**Action**	**Response**	**Pass/Fail**	**Comment**
1	Visitor	Enter homepage	Homepage DSA topics appears	Pass	Works for all users
2	Visitor	Clicks on Array Sort	Array Sorting theory section loads	Pass	Proper redirection
3	Visitor	Click Array Sort Visualization	Theory section shrinks, pseudocode and visualization section appears	Pass	Proper redirection

4	Visitor	Click Sort (Visualization section)	Sorting algorithm and pseudocode visualization loads	Pass	No login required
Post-condition: Visitor can view and access any topic's theory and visualization freely.					

Table-9: Test case report for Visualization Interaction

Test case:3.3.9		**Test case name:** Visualization Interaction			
System:Algosikhi		**Subsystem:** Visual Animation			
Designed By: User		**Design Date**: 24/07/25			
Executed By: User		**Execution Date**: 25/07/25			
Description: Tests the execution of algorithm visualization and pseudocode highlighting.					
Pre-condition: Any topics visualization page loaded.					
Steps	**Name**	**Action**	**Response**	**Pass/Fail**	**Comment**
1	User	Clicks start visualization	Animation runs	Pass	Smooth visual playback
3	System	Syncs pseudocode	Highlights step by step	Pass	Synchronized successfully
4	User	Clicks reset	Animation resets	Passl	Proper reset
Post-condition: Visualization runs smoothly and pseudocode is synced during execution.					

Table-10: Test case report for Progress Update in backend

Test case:3.3.9	**Test case name:** Progress Update in backend
System:Algosikhi	**Subsystem:** Backend Progress API
Designed By: User	**Design Date**: 26/07/25
Executed By: User	**Execution Date**: 27/07/25
Description: When the user finishes interacting with a topic, the backend updates their learning status.	

Pre-condition: Logged-in user, completed interaction.					
Steps	**Name**	**Action**	**Response**	**Pass/Fail**	**Comment**
1	User	Completes Merge Sort visualization	Sends POST request to API	Pass	Triggered correctly
3	System	Saves to MongoDB	Returns success status	Pass	Stored accurately
4	User	Refreshes dashboard	Completion badge updated	Fail	Reflects updated state
Post-condition: Learning progress is accurately saved and reflected in the user dashboard.					

Chapter 4: Deployment and Maintenance

Introduction

This chapter describes the **deployment** and **maintenance** phases of the **Algosikhi learning platform**. Deployment refers to the process of launching the platform to the public, making it accessible to users via the web. Maintenance ensures that the system remains functional, up-to-date, and responsive to user needs after the initial launch.

Since the platform is built with an emphasis on educational accessibility and dynamic user engagement, it requires continuous enhancements, bug fixes, performance optimizations, and possible content or feature updates. This chapter also outlines the tools used for deployment and the strategies followed for long-term stability, aligning with Agile practices and the **Software Release Life Cycle (SRLC)** model.

Following the SRLC (Software Release Life Cycle)

The development and release of the **Algosikhi learning platform** followed a structured approach using the **Software Release Life Cycle (SRLC)** model. This model helps ensure quality, stability, and timely delivery of features across the platform.

Planning Phase

In this initial stage, the core goals were defined — such as creating a platform to help students understand algorithms through theory, pseudocode, and step-by-step visualization. Requirements were collected regarding user interaction, visualization triggers, pseudocode highlighting, and progress tracking.

> **Agile Note**: Weekly sprints and iterative brainstorming sessions were held to prioritize feature sets like dashboard, sorting visualizations, STL operations, etc.

Development Phase

Development was done using **HTML, CSS, JavaScript** on the frontend, and **Node.js, Express.js, MongoDB** for the backend. The **JWT** authentication system was implemented to allow user-specific learning progress tracking. Backend APIs were created to update and retrieve user progress dynamically. Agile stand-ups were used to track progress and iterate based on challenges or feedback from testers.

Testing Phase

Various functional and non-functional test cases were executed (as documented in Chapter 3) to ensure smooth visualizations, accurate backend progress updates, and responsive interaction between the UI and logic. Testing was done in units, with features like login, progress tracking, and visualization syncing thoroughly verified.

> Usability testing was emphasized to check how new users interacted with topics, especially regarding ease of understanding algorithms visually.

Deployment Phase

The platform was deployed to a hosting service like **Render**, **Vercel**, or **GitHub Pages (frontend)** and **MongoDB Atlas (database)**. The backend was hosted on a service supporting **Node.js servers**, and APIs were secured using CORS and token verification. Login is optional, but necessary for users who wish to track their learning.

> Domains and SSL certificates were managed using hosting tools (free SSL from hosting platforms like Render or Vercel).

Maintenance Phase

Ongoing support includes:

- Fixing bugs reported by users or discovered during testing
- Updating topics or adding new visualizations (like BFS/DFS, Graphs, etc.)
- Monitoring database performance and API reliability
- Making manual changes when necessary (since there's no admin panel)
- Adding new features such as bookmarking, progress analytics, or language toggle (in future updates)

Feedback is gathered manually from users to determine priority features for future iterations.

Benefits of SRLC in Algosikhi

By following the SRLC process, the development of **Algosikhi** was systematic and efficient. Every phase was executed with a clear objective:

- Planning helped define the platform's educational goal
- Development ensured core functionalities were stable and user-friendly
- Testing ensured a smooth user experience and backend accuracy
- Deployment made the system live for global access
- Maintenance keeps the platform evolving based on real feedback

This structured yet Agile approach minimized bugs, improved learning engagement, and allows the platform to remain relevant to learners who want interactive DSA education.

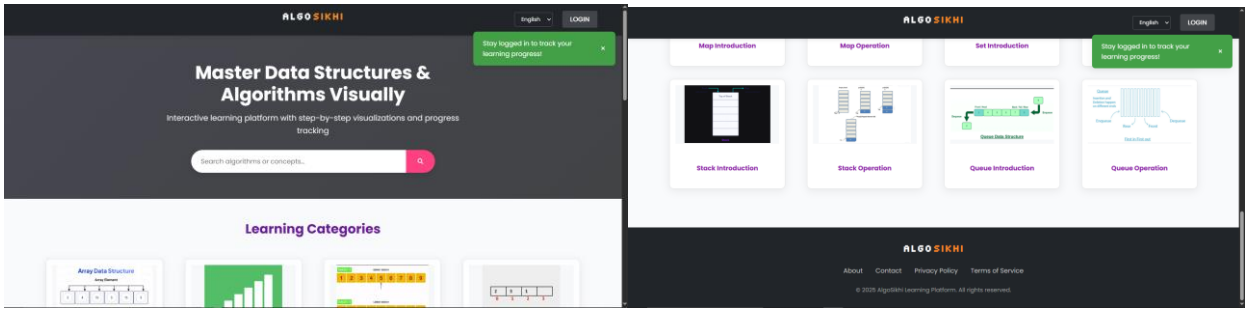
Chapter 5: User Manual

5.1 Introduction

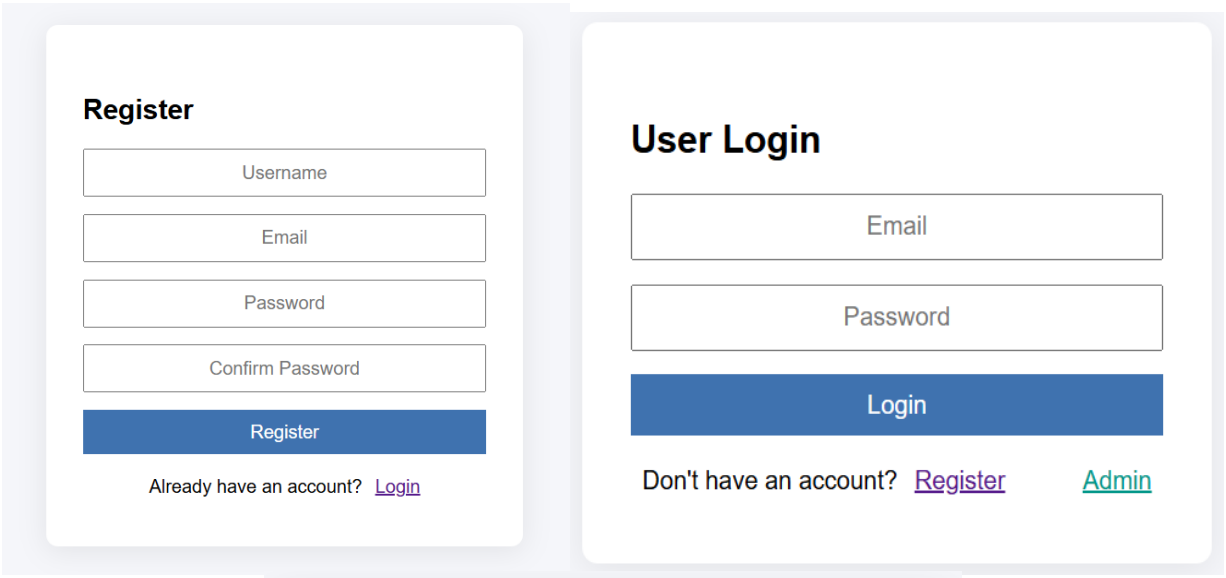
This chapter provides a complete **user manual** for navigating and using the **Algosikhi learning platform**. It includes **step-by-step instructions** for core features such as browsing topics, visualizing algorithms, tracking learning progress, and accessing dashboards. This guide is intended to help new users quickly understand how to use the system for an effective and interactive learning experience.

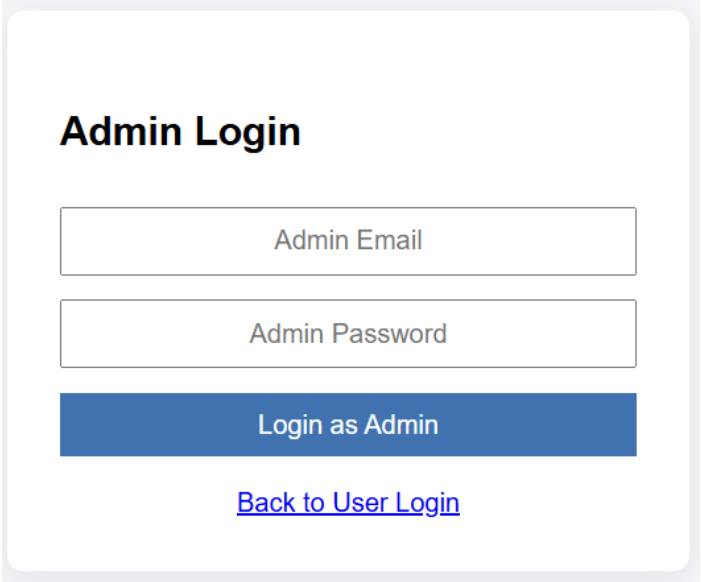
5.2 Key User Activities and Screenshots

5.2.1 Homepage Overview: Explain that from the homepage, users can start exploring topics without logging in, and how the homepage updates to show "Dashboard" when logged in.

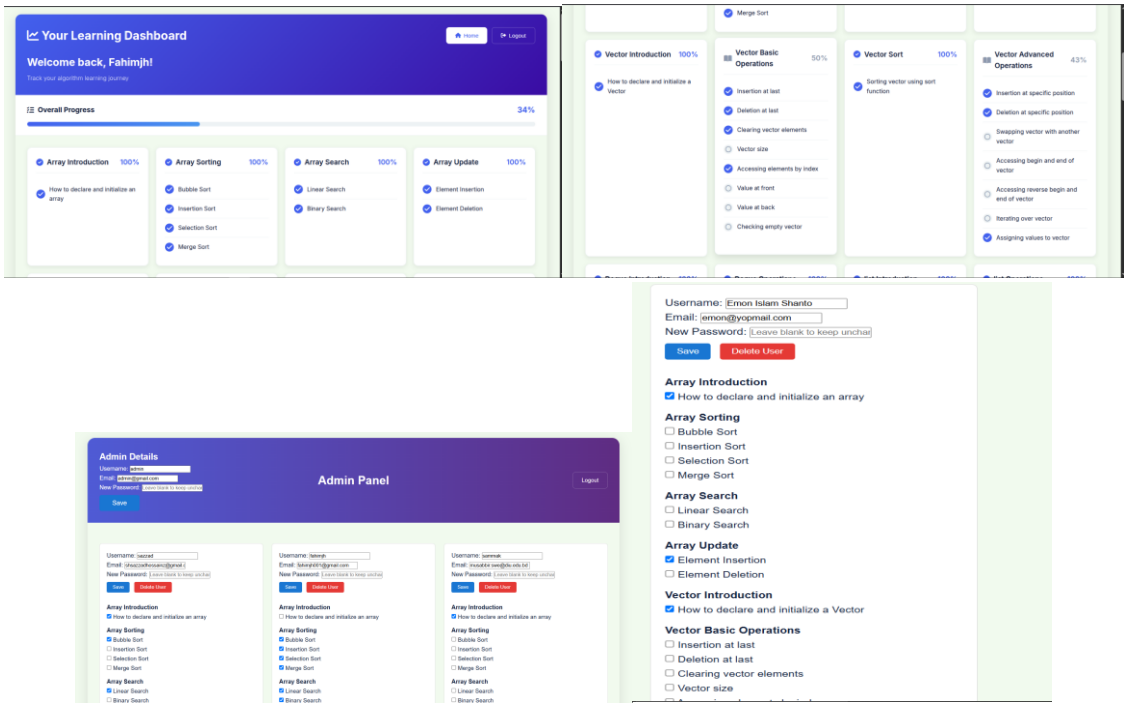


5.2.2 Login and Registration: Show how users can register/login if they want to track their learning progress. Mention how data is stored securely using JWT and MongoDB.

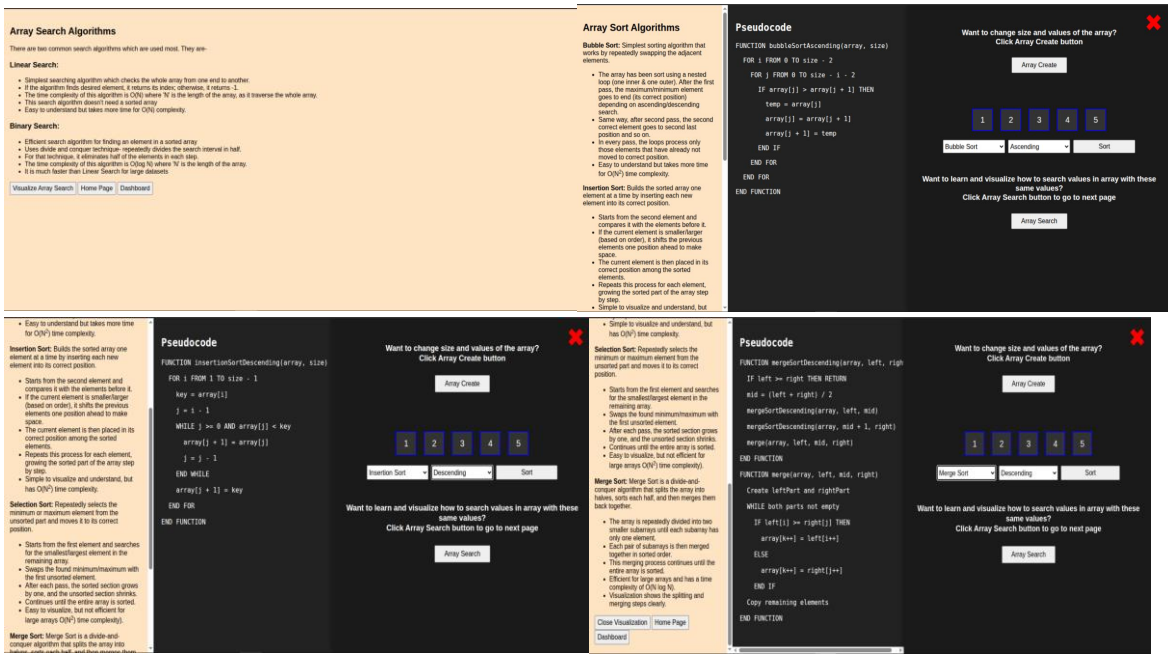




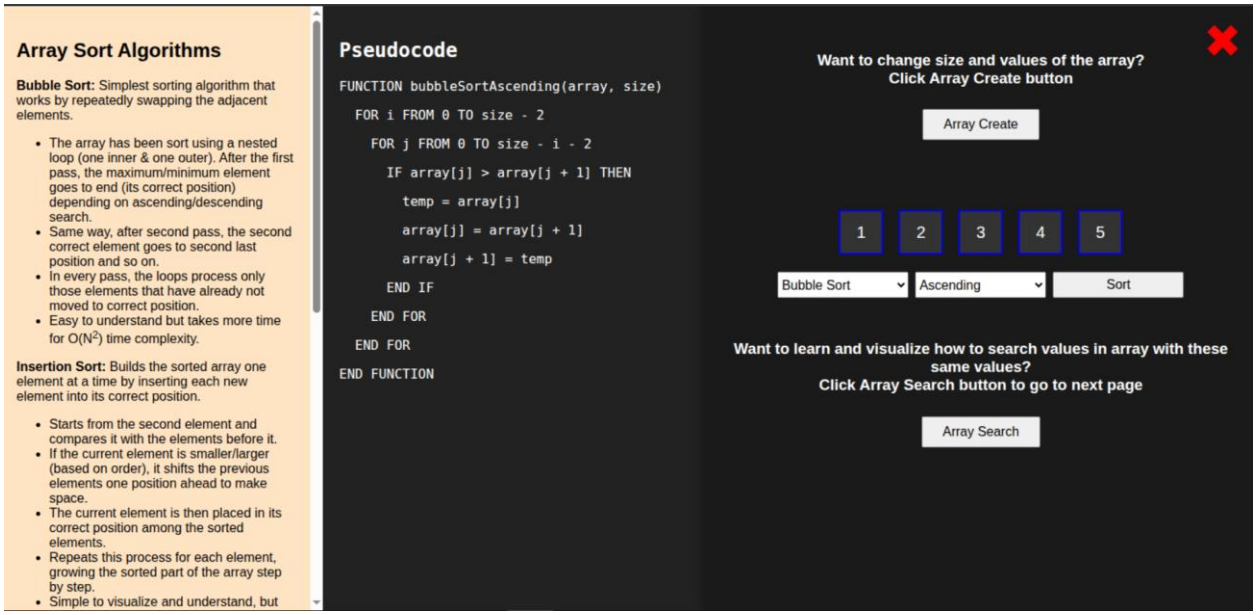
5.2.3 Dashboard Overview: User dashboard after login, showing progress tracking per topic and overall progress bar.

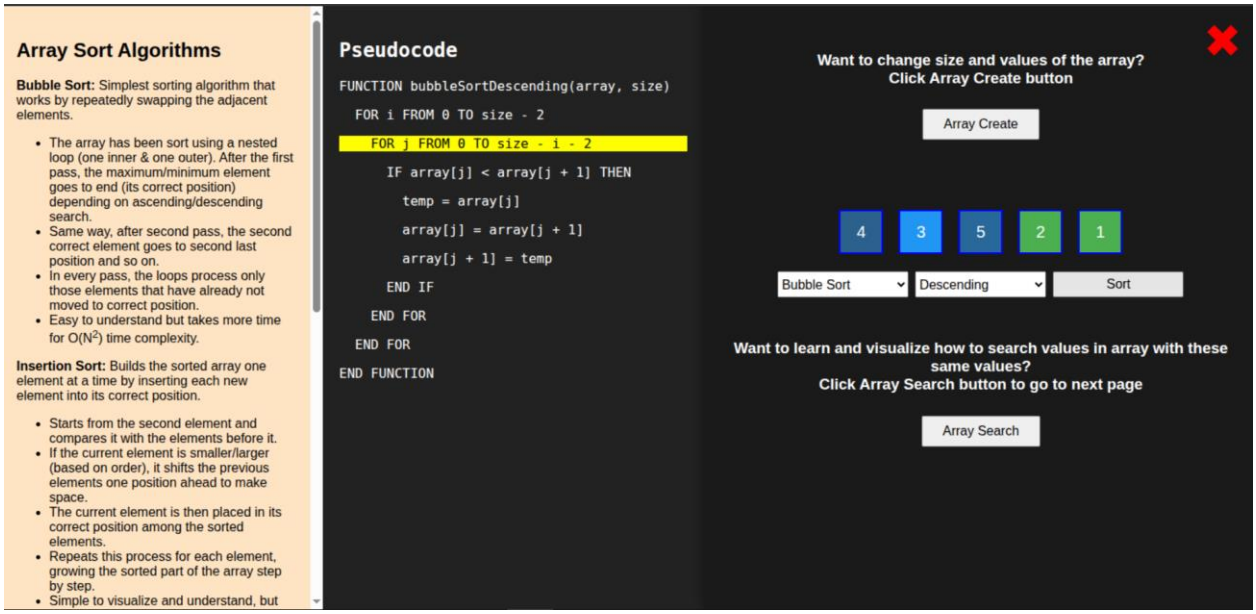


5.2.4 Theory, Pseudocode and visualization Section: Help the user understand how the interface is organized into theory, pseudocode, and visual interaction. Mention that pseudocode changes dynamically based on selected algorithm and order.

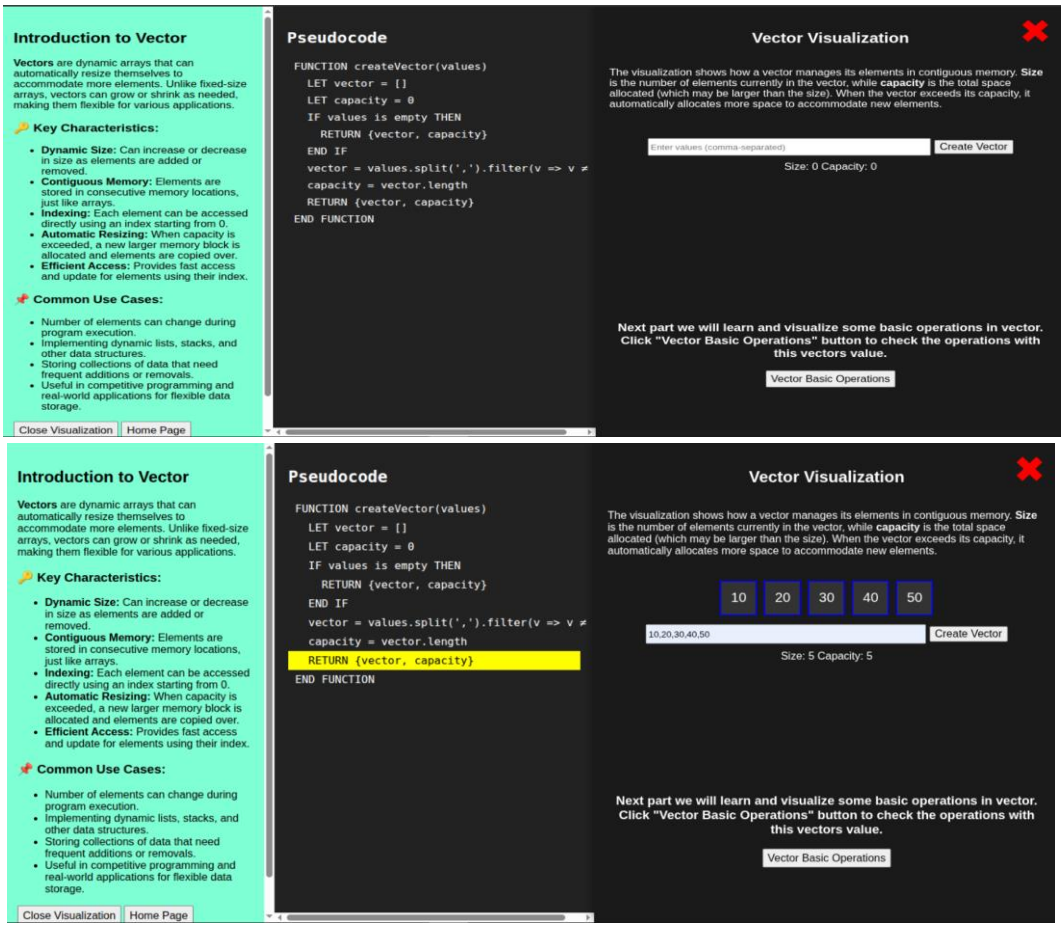


5.2.5 Visualization Page Example (Array Sorting): Explains how users select a sorting algorithm, click start, see step-by-step animation with highlighted pseudocode. Show how this interaction triggers progress tracking.

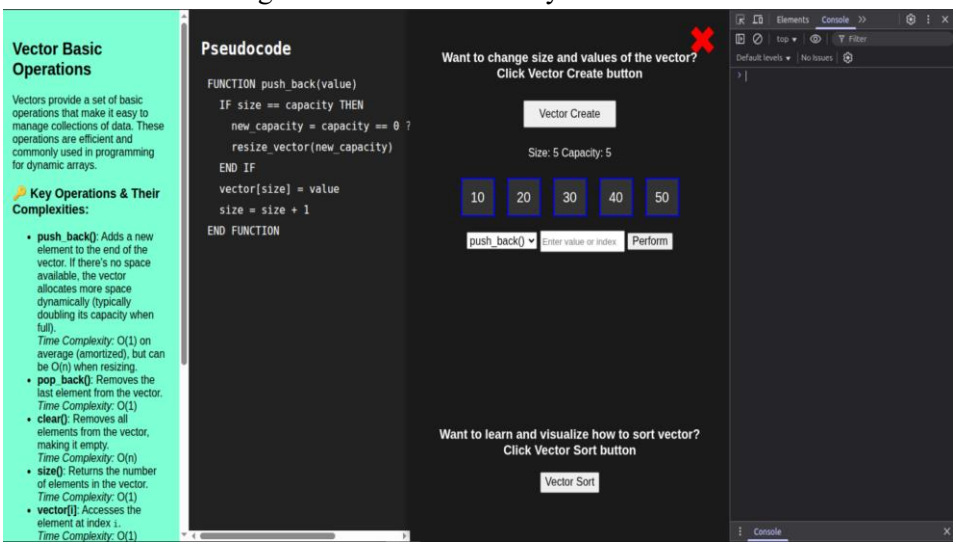


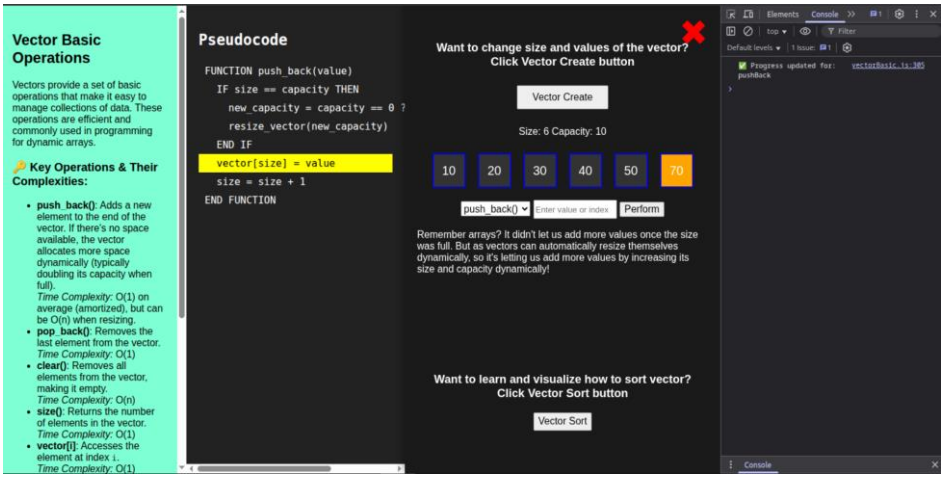


5.2.6 STL Operation Example (e.g., Vector, Map, Set): Show how STL container operations are visualized and animated to help learners understand internal behavior.

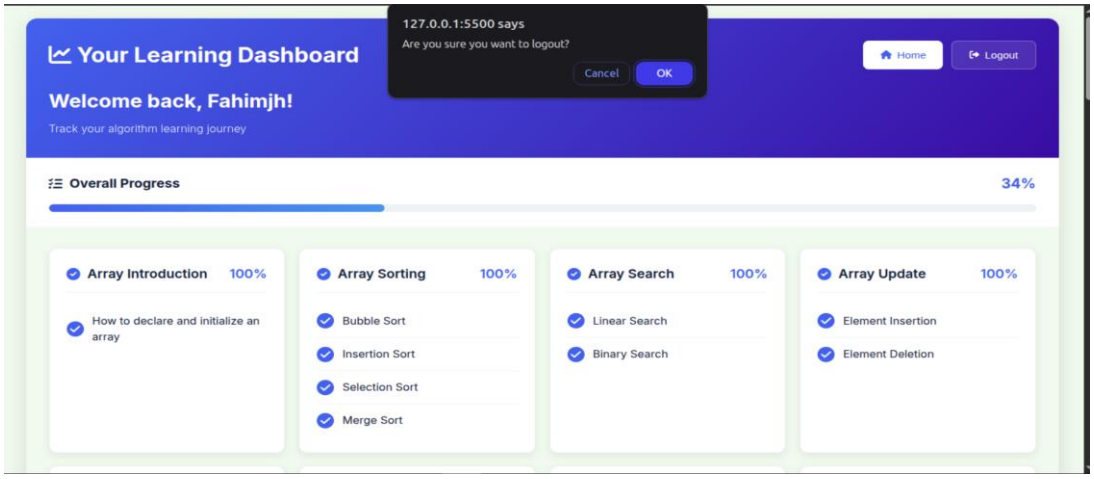


5.2.7 Learning Progress Update: Progress being updated after visualization finishes. Which helps users understand how their learning is tracked automatically.





5.2.8 Logout Process: Logout button or confirmation message. A simple explanation of how to securely log out to end their session.



Chapter 6: Project Summary

6.1 Introduction

Algosikhi is an educational web-based learning platform designed to help students understand algorithms and data structures interactively through theory, pseudocode, and live visualizations. The project enables both logged-in users (who want to track their progress) and visitors (who wish to learn freely) to explore topics like Array, STL Containers (Vector, Map, Set), Sorting, Queues, and more.

Unlike traditional learning websites, Algosikhi doesn't just provide static theory—it offers dynamic visualizations synced with pseudocode and real-time learning progress tracking. The platform ensures a smooth user experience by offering animated walkthroughs of operations such as sorting, vector manipulation, and STL containers, paired with clear, structured pseudocode. Logged-in users can track completed sections, view their learning stats on a dashboard, and soon will be able to test their understanding through quizzes.

The platform uses HTML, CSS, JavaScript (frontend), Node.js with Express and MongoDB (backend), and JWT for secure login authentication. It does **not include any admin panel**—all updates and additions are manually managed by the creator. Algosikhi is a strong step forward in bridging the gap between abstract theory and hands-on understanding in algorithm education.

6.2 Project Limitations

While Algosikhi covers a wide range of topics and supports real-time interaction, a few limitations currently exist due to time constraints and technical scope:

- **No Responsive Design Yet**: The platform is not optimized for mobile or tablet viewing. Currently, it works best on desktop browsers only.
- **No SSL/HTTPS Encryption**: The project does not yet use SSL encryption for secure HTTPS communication.
- **No Email Communication**: Although login/registration exists, features like password reset or email confirmation are not implemented.
- **No Multi-Language Support Yet**: All content is currently in English. Bengali support is planned.
- **No Quiz or Certification System Yet**: User knowledge isn't tested through MCQs or exams yet, and certifications are not issued.
- **No Teacher/Admin Role**: Currently, there is no interface for teachers or institutions to manage their students or content.
- **Manual Content Management**: There is no backend UI; all updates (content, visualizations) are added manually by the creator.

Despite these limitations, the core functionality—learning through visualization with progress tracking—is working reliably and effectively.

6.3 Project Summary

Algosikhi provides a modern and interactive way to learn algorithms through structured theory, animated visualization, and pseudocode interpretation. It allows learners to:

- Browse different algorithm categories and subtopics without login.
- Register and log in to track learning progress.
- View animations of sorting, STL operations, array manipulation, and queues.
- Access a dashboard that shows topic-wise completion and progress bar.
- Experience a theory-pseudocode-visualization layout that strengthens understanding.

Although features like responsive design, quizzes, and certification are not yet live, the project lays a solid foundation for interactive DSA education.

6.4 Future Work

Several advanced and impactful features are planned for future releases of Algosikhi:

1. **Responsive Design for Cross-Device Support**
    - Enable full compatibility on mobile and tablet screens for better accessibility.
    - Ensure cross-browser support for Chrome, Firefox, and Edge.

2. **Section-Based MCQ Quizzes**
    - After completing a full section (e.g., all topics under Array), a short quiz will be shown.
    - Quizzes will test user understanding through MCQs and track scores.

3. **Certification Exams**
    - Once a user finishes all topics and passes all section quizzes with >70%, a final exam on that topic becomes available.
    - If the user achieves more than **80% marks** on the final certification exam, they will receive a downloadable digital certificate signed by the platform creator.

4. **Bangla Language Support**
    - Add Bengali translations to theory, pseudocode, and visual content.
    - Allow users to switch between English and Bangla.

5. **Teacher Section for Institutions**
    - Teachers from universities can create accounts to:
        - Create and assign exams.
        - Track student progress.
        - Grade submissions and monitor engagement.
    - This makes the platform usable in formal academic settings.

6. **Other Ideas for the Future**
    - Dark mode toggle and accessibility features for visually impaired learners.
    - Improve backend automation for tracking quizzes, issuing certificates, and analytics.
    - Add complexity/time analysis after each visualization for deeper learning.
    - Integration of a "Learning Roadmap" or syllabus for beginner, intermediate, and advanced levels.

REFERENCES

1. GeekforGeeks. DSA and STL topics retrieved from

   https://www.geeksforgeeks.org/dsa/dsa-tutorial-learn-data-structures-and-algorithms/

2. W3schools. DSA and STL topics retrieved from

   https://www.w3schools.com/dsa/index.php

E-learning Platform References:

Project Idea and future work retrieved from-

1. https://visualgo.net/en

2. https://www.sololearn.com/en/

211-35-688

ORIGINALITY REPORT

11%	8%	1%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	4%
2	dspace.daffodilvarsity.edu.bd:8080 Internet Source	2%
3	Submitted to NCC Education Student Paper	1%
4	123dok.com Internet Source	<1%
5	Phenikaa University Publication	<1%
6	Submitted to UC, Irvine Student Paper	<1%
7	Submitted to Apex Australia Higher Education Student Paper	<1%
8	Submitted to Texas A & M University, Kingville Student Paper	<1%
9	landing.athabascau.ca Internet Source	<1%
10	Submitted to Higher Education Commission Pakistan Student Paper	<1%
11	Submitted to University of Moratuwa Student Paper	<1%

12	www.freelancer.com Internet Source	<1 %
13	Submitted to Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA) Student Paper	<1 %
14	Islam, Mohammad Rubyet. "Dynamic Prognostic Health Management for Response Time Based Remaining Useful Life Prediction of Software Systems", University of Maryland, College Park, 2023 Publication	<1 %
15	dokumen.tips Internet Source	<1 %
16	Areias, Davide Figueiredo. "Development of Cloud-Ready Platform with 3D Modeling and Energy Calculation Engine", Universidade de Coimbra (Portugal) Publication	<1 %
17	Submitted to University of Hertfordshire Student Paper	<1 %
18	www.jeppiaarinstitute.org Internet Source	<1 %
19	taikai.network Internet Source	<1 %
20	Submitted to The Robert Gordon University Student Paper	<1 %
21	ofcdesk.com.br Internet Source	<1 %
22	Submitted to De Montfort University Student Paper	<1 %

23	Submitted to Macquarie University Student Paper	<1%
24	repository.cuilahore.edu.pk Internet Source	<1%
25	www.researchgate.net Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

Account Clearance: