Jagan Institute of Management Studies

(3, Institutional Area, Sector-5, Rohini, Delhi-110085)

Affiliated to Guru Gobind Singh Indraprastha University

MINOR PROJECT - III

ON

"SSH(SORT AND SEARCH HUB)"

Submitted in partial fulfilment of requirement of

"Master of Computer Application" (MCA)

Paper Code – **MCA-169**

SUBMITTED TO:

Dr. Deepti Sharma
Associate Professor
JIMS, Rohini, Sector-5

SUBMITTED BY:

Name: Divyam Parhwal
Enrollment No.: 01214004423
MCA (Section – A) (1st Shift)
3rd Semester

ACKNOWLEDGEMENT

I would like to take this opportunity to thank The Guru Gobind Singh Indraprastha University (GGSIPU) for having projects as a part of the MCA curriculum. Many people at the university have influenced the shape and content of this project and supported us through it.

I express my sincere gratitude to Dr. Deepti Sharma for mentoring me in this project. She has been an inspiration and role model for this topic. Her guidance and active support have made it possible for me to work on the assignment.

I would like to extend my thanks to the members involved in making this project successful and without whom this project would not have been possible. Also, I would like to express sincere gratitude to our parents and a load of thanks to our friends for helping us throughout the course of the project.

Divyam Parhwal
Enrollment No.: 01214004423

CERTIFICATE

This is to certify that Divyam Parhwal student of MCA Shift -1 as undertaken Minor Project titled **"SSH(SORT AND SEARCH HUB)"** completed under my supervision and guidance in partial fulfilment of the requirement for the award of degree of Master of Computer Applications (MCA) as a part of the curriculum in Guru Gobind Singh Indraprastha University, New Delhi- 110078. To the best of my knowledge and belief, the data and information presented by the student in the Minor Project has not been submitted earlier.

Dr. Deepti Sharma

Associate Professor

JIMS Rohini, New Delhi

TABLE OF CONTENTS

ABSTRACT

The Sort and Search Hub (SSH) is an innovative educational platform meticulously crafted to enhance the comprehension of sorting and searching algorithms through immersive interactive visualizations. Leveraging dynamic visual aids alongside succinct, well-structured tutorials, SSH provides an invaluable resource for students, educators, and professionals alike, fostering a deeper and more intuitive understanding of these fundamental computational processes. By demystifying the complex mechanisms of various algorithms, SSH bridges the gap between theoretical concepts and practical application, making abstract computational ideas more tangible and accessible. This platform offers visualization and concise tutorials for the following algorithms: Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, Merge Sort, Heap Sort, Tim Sort, Shell Sort, Binary Search, and Linear Search. This report delves into the development processes, pedagogical methodologies, and the substantial educational benefits of SSH, underscoring its significant role in advancing algorithmic literacy. Additionally, the report explores the technical implementation using Python and Flask, highlighting the platform's contribution to modern web development and interactive learning.

Keywords: Sorting Algorithms, Searching Algorithms, Data Visualization, Educational Platform, Python, Flask, Interactive Learning

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTON

In the field of computer science education, the comprehension of sorting and searching algorithms stands as a fundamental pillar. These algorithms are integral to a wide array of complex computational processes, underpinning the functionality of data structures, database systems, network routing protocols, and various applications that demand efficient data management and retrieval. The efficiency and performance of software often hinge on the effective implementation of these algorithms, making their understanding not just an academic exercise but a practical necessity in the professional landscape.

Sorting algorithms, such as Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, Merge Sort, Heap Sort, Tim Sort, and Shell Sort, are designed to arrange data in a specific order, which is crucial for optimizing search operations, enhancing data readability, and improving algorithmic efficiency. Each sorting algorithm has unique characteristics, time complexities, and operational paradigms, making them suitable for different types of data and application requirements. For instance, Quick Sort is renowned for its efficiency with average-case time complexity of O(n log n), while Merge Sort is appreciated for its stability and consistent performance across diverse datasets.

Searching algorithms, including Binary Search and Linear Search, are pivotal for locating specific data elements within a dataset. Binary Search, with its O(log n) time complexity, is highly efficient for sorted datasets, enabling rapid access to data. Linear Search, although less efficient with a time complexity of O(n), is versatile and simple to implement, applicable to both sorted and unsorted datasets.

Despite their importance, the abstract nature of these algorithms often presents significant learning challenges. Traditional teaching methods, which rely heavily on static code examples and theoretical explanations, may not effectively convey the dynamic behavior and operational intricacies of these algorithms. This abstraction can lead to gaps in understanding, particularly when learners are unable to visualize the step-by-step processes that these algorithms undertake.

To address these challenges, the Sort and Search Hub (SSH) has been conceived as an innovative educational platform. SSH leverages the power of interactive visualization to transform the learning experience, providing a tangible, real-time representation of algorithmic operations. By visualizing the inner workings of sorting and searching algorithms, SSH aims to demystify these complex processes, making them more accessible and easier to grasp for learners of all levels. This approach not only aids in the comprehension of the algorithms but also enhances the retention of the concepts by engaging multiple cognitive pathways through visual learning.

1.1 OBJECTIVE OF THE PROJECT:

The primary objective of SSH is to create an immersive and interactive learning environment where users can visualize and understand the intricate mechanics of sorting and searching algorithms. This platform aims to:

- Enhance comprehension by providing visual aids that illustrate the step-by-step execution of algorithms.
- Offer detailed tutorials and explanations to support the learning process.
- Serve as a refresher tool for professionals looking to revisit and reinforce their understanding of fundamental algorithms.
- Allow users to interact with the algorithms by inputting custom arrays and observing real-time changes, thereby fostering a deeper understanding of the concepts.
- Ensure accessibility and ease of use, catering to learners at different levels of proficiency in computer science.

1.2 SCOPE OF THE PROJECT:

The scope of SSH includes the development and deployment of a comprehensive web-based platform using Python, Flask, SQLite, CSS, JavaScript, and HTML. The platform will feature modules for visualizing a curated selection of sorting and searching algorithms, specifically:

- **Sorting Algorithms:** Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, Merge Sort, Heap Sort, Tim Sort, Shell Sort.
- **Searching Algorithms**: Binary Search, Linear Search.

Key features of the platform will include:

- **Interactive Visualization**: Users can input custom arrays and see real-time visual representations of the algorithms' operations. This feature is designed to be accessible only to registered and logged-in users to maintain data integrity and user engagement.
- **Tutorials and Explanations**: Each algorithm will be accompanied by detailed, step-by-step tutorials and explanatory notes to aid in the learning process. These tutorials will cover the logic, implementation details, time and space complexities, stability, and suitability for different types of data.

- **User Interface**: The platform will boast a user-friendly interface, ensuring that users can easily navigate and utilize the various features without requiring extensive technical knowledge.
- **Educational Tools**: Additional educational tools, such as quizzes and interactive exercises, may be incorporated to further enhance learning and retention.
- **Performance Metrics**: The platform will track user interactions and learning progress, providing insights into areas where learners may need additional support or practice.

By combining these elements, SSH aims to provide a holistic and engaging learning experience, making the study of sorting and searching algorithms not only more accessible but also more enjoyable.

1.3 PROBLEM STATEMENT:

The education of sorting and searching algorithms presents significant challenges that hinder effective learning and comprehension. Users, including students, educators, and professionals, encounter the following issues:

1. **Limited Comprehension**: Many learners struggle to grasp the fundamental principles of sorting and searching algorithms due to their abstract nature, which can lead to misunderstandings and a lack of confidence in applying these concepts.
2. **Lack of Engagement**: Traditional educational methods often rely on static materials, such as textbooks and code snippets, which fail to engage students or demonstrate the dynamic behaviour of algorithms in real-time, limiting their ability to visualize and understand algorithmic processes.
3. **Inadequate Learning Resources**: Existing platforms frequently do not provide interactive tools that allow users to experiment with algorithms, leading to a passive learning experience and diminishing opportunities for hands-on practice.
4. **Retention Challenges**: The absence of engaging and practical resources makes it difficult for learners to retain information about sorting and searching algorithms, resulting in knowledge gaps and a diminished ability to apply these concepts in real-world scenarios.
5. **Diverse Learning Needs**: Learners come from varying backgrounds and possess different levels of understanding, yet most current resources do not cater to these diverse learning styles and needs, leading to an ineffective one-size-fits-all approach.
6. **Desire for Practical Application**: There is a strong desire among learners to see how sorting and searching algorithms are applied in real-world situations, but many educational resources do not effectively demonstrate these applications, leaving learners with a theoretical understanding that lacks practical context.

The Sort and Search Hub (SSH) aims to address these problems by providing an innovative platform that combines interactive visualizations with comprehensive tutorials, fostering a deeper understanding of sorting and searching algorithms and enhancing overall algorithmic literacy.

1.4 PURPOSE OF THE PROJECT:

The primary purpose of the Sort and Search Hub (SSH) is to create a comprehensive and engaging educational platform that enhances the understanding of sorting and searching algorithms through interactive visualizations. By providing users with dynamic, real-time representations of algorithmic operations, SSH aims to transform the learning experience from abstract theory to practical understanding. The platform seeks to:

- Empower learners of all levels by providing visual aids that illustrate the step-by-step execution of algorithms, making complex concepts more accessible.
- Offer detailed tutorials that complement the visualizations, ensuring users can grasp the underlying logic and applicability of each algorithm.
- Serve as a valuable resource for both students and professionals, allowing users to reinforce their knowledge and revisit fundamental algorithms as needed.
- Facilitate hands-on learning by enabling users to input custom data sets and observe how different algorithms perform with varied inputs.
- Foster an inclusive learning environment that caters to diverse learning styles, thereby improving algorithmic literacy across a broad audience.

Through these initiatives, SSH aims to bridge the gap between theoretical concepts and practical applications in computer science, ultimately contributing to a deeper understanding of essential computational processes.

CHAPTER 2: SYSTEM ANALYSIS STUDY

2.1 INTRODUCTION

The Sort and Search Hub (SSH) was meticulously designed by identifying key educational and usability requirements, reflecting the evolving landscape of computer science education. Recognizing the significance of algorithm visualization as a pedagogical tool, SSH aims to bridge the gap between abstract theoretical concepts and practical understanding through dynamic, interactive learning experiences.

2.1.1 Functional Requirements:

- Algorithm Visualizer: Provides interactive, step-by-step visualizations for sorting and searching algorithms, allowing users to see how data structures change in real time.
- Tutorials and Complexity Analysis: Offers comprehensive explanations of each algorithm's logic, including best and worst-case complexity, and practical applications, fostering a deeper cognitive connection with the material.
- Custom Input: Users can input custom data sets, enabling hands-on learning and real-time adjustments, which is crucial for understanding algorithm performance under different conditions.
- User Management: Facilitates user authentication and tracks personalized learning progress, enabling learners to monitor their development over time.

2.1.2 Non-Functional Requirements:

- **Scalability:** The platform is designed to support a large number of concurrent users, ensuring an uninterrupted experience.
- **Responsiveness:** Real-time interactions and visualizations are prioritized to maintain user engagement.
- **Security:** Implements robust security measures to protect user data and maintain session integrity.

2.1.3 Constraints and Limitations:

- **Technical Constraints:** The platform may experience limited performance for very large data inputs due to the inherent complexity of visualizing extensive sorting and searching operations, which could impact real-time feedback.
- **User Interface Constraints:** New users, especially those unfamiliar with sorting and

searching terminology, may face a learning curve, highlighting the need for intuitive design and supportive onboarding materials to enhance user experience.

2.2 FEASIBILITY STUDY

The feasibility study is a critical step in the early stages of project development, aimed at assessing the technical, economical, and operational feasibility of the proposed system. The feasibility study for the project evaluates its viability in terms of technological readiness, economic viability, and operational effectiveness.

2.2.1 Technical Feasibility

Technical feasibility assesses whether it is technically possible to develop the software and whether the required technology is readily available. During the study phase, various areas of technical feasibility were examined:

- **Availability of Technology:** The frontend and backend development tools required for the SSH project, are HTML, CSS, JavaScript, Python, Flask, and SQLite, are widely available and accessible. These technologies are well-documented and supported by extensive online communities.
- **Compatibility with Hardware:** The platform is designed to work seamlessly with current hardware configurations, including servers, databases, and client devices. This compatibility ensures that users can access the platform from a variety of devices, including desktops, laptops, and tablets.
- **Scalability:** SSH is built with scalability in mind, allowing for the addition of new sorting and searching algorithms, enhanced user features, and the capacity to handle an increasing user base and data input over time without compromising performance.
- **Browser Support:** The frontend development tools utilized in the project are supported by major browsers, ensuring a consistent and reliable user experience across different platforms. This cross-browser compatibility is critical for user engagement and accessibility.
- **Availability of Programmers:** There is a substantial pool of programmers experienced in the technologies used for developing SSH, facilitating recruitment and resource allocation. This availability ensures that the development and maintenance of the platform can be adequately supported.
- **Availability of Backend Tools:** The backend tools and frameworks used in SSH, are Flask and SQLite, are open-source and compatible with modern system development practices. This ensures that the platform can be built and maintained cost-effectively.

2.2.2 Economical Feasibility

Economic feasibility assesses the economic factors associated with the project, including costs, benefits, and returns on investment. The SSH project is considered economically feasible due to the following factors:

- **Minimal Investment:** The project leverages freely available development tools and technologies, significantly reducing initial investment costs. This aspect is crucial for educational initiatives where funding may be limited.
- **Low Hosting Expenses:** Hosting expenses are kept minimal, with the potential to start on a low-cost or free hosting server. As the platform scales and attracts more users, transitioning to a more robust hosting solution will remain cost-effective.
- **Manageable Future Costs:** Future costs, such as domain registration fees and additional server capacity, are predictable and manageable. The budget can be adjusted to accommodate these expenses as the platform grows.
- **Potential for Revenue Generation:** Although SSH is primarily educational, there may be opportunities for monetization through premium features, sponsored content, or partnerships with educational institutions, increasing its economic viability.

2.2.3 Operational Feasibility

Operational feasibility assesses how well the proposed system addresses identified problems, capitalizes on opportunities, and meets requirements. For the SSH project, operational feasibility is ensured by:

- **User-Centric Design**: The platform is designed with a focus on user engagement, featuring intuitive navigation, interactive visualizations, and clear instructional content.
- **Integration with Learning Environments**: SSH can be integrated into existing educational frameworks or curricula, supporting instructors and learners in a seamless manner.
- **Adaptability to User Feedback**: SSH incorporates user feedback mechanisms, allowing continuous improvement based on actual user experiences and needs.
- **Comprehensive Support Resources**: Training materials and support forums are provided to assist users in maximizing the platform's educational benefits, ensuring smooth onboarding and usage.

By addressing these feasibility considerations, SSH is positioned to deliver a valuable educational tool that enhances understanding of sorting and searching algorithms, fostering a more engaging learning experience for users.

CHAPTER 3: REQUIREMENT ANALYSIS

3.1 Modules Involved in the SSH:

1. Algorithm Visualization Module:-

- **Real-time Visualization**: This module provides real-time visualization of various sorting and searching algorithms. Users can watch algorithms execute step-by-step, which helps in understanding their inner workings.
- **Dynamic Display of Operations**: The module displays key operations such as comparisons, swaps, and data rearrangements visually. Each operation is highlighted to allow users to follow along easily and comprehend how data structures change.
- **Interactive Controls**: Users can control the speed of visualization, pause, and resume the algorithm's execution, providing an interactive learning experience.
- **Support for Multiple Algorithms**: This module supports various algorithms, including Bubble Sort, Quick Sort, Merge Sort, Binary Search, and Linear Search, allowing users to compare different algorithms side by side.

2. Tutorial Module:-

- **Comprehensive Explanations**: Each algorithm's structure is explained in detail, including a breakdown of how the algorithm operates and the rationale behind its design.
- **Pseudocode and Complexity Analysis**: The module provides pseudocode for each algorithm alongside an analysis of best, average, and worst-case time and space complexities, aiding users in understanding the performance implications of different approaches.
- **Visual Aids and Examples**: To enhance learning, visual aids such as diagrams and example datasets are included to illustrate how the algorithms function under various scenarios.
- **Assessment Quizzes**: Users can take quizzes to assess their understanding of the algorithms covered in the tutorials, reinforcing their learning through practical questions.

3. User Management Module:-

- **User Registration and Login**: Users can create an account and log in to access personalized features. The registration process includes email verification to ensure valid user accounts.
- **User Profile Customization**: Users can customize their profiles, including preferences

for algorithm visualization settings, preferred algorithms, and notification settings.

- **Activity Tracking**: The module tracks user interactions with the platform, providing insights into which algorithms the user has explored and identifying areas for further improvement.
- **Feedback Mechanism**: Users can submit feedback regarding the tutorials and visualizations, helping to refine and improve the platform based on user experiences and suggestions.

3.2 SOFTWARE DEVELOPMENT LIFE CYCLE MODEL:

SSH employs the **Agile Model** for its development process, allowing for iterative improvements based on user feedback. The Agile approach supports continuous integration and delivery, ensuring that user needs are consistently met. Phases in Agile Development are:

1. **Requirement Analysis**:
    - **Gathering Requirements**: Ongoing interaction with users and stakeholders to gather and prioritize features and functionalities. This includes identifying the most requested algorithms for visualization and tutorial content.
    - **Documentation of Requirements**: The requirements are documented in a clear and concise manner, forming the basis for each development cycle. Key requirements include algorithm accuracy, visualization clarity, and user engagement features.
2. **Design**:
    - **Architectural Design**: High-level system architecture is defined, outlining how the various modules (visualization, tutorial, and user management) will interact and integrate.
    - **User Interface Design**: Focus on creating a user-friendly interface that promotes ease of use and accessibility, ensuring that users can easily navigate between different sections of the platform.
3. **Implementation and Unit Testing**:
    - **Development of Modules**: Each module is developed incrementally, allowing for early testing and feedback on individual components.
    - **Unit Testing**: Each module undergoes unit testing to ensure that individual functionalities work as intended, with particular focus on algorithm accuracy in the visualization module.

4. **Integration and System Testing**:

   - **Module Integration**: After individual modules are tested, they are integrated to form a complete system. Integration testing ensures that the modules work together seamlessly.
   - **System Testing**: The entire system is tested for usability, performance, and security to identify any issues before deployment.

5. **Operation and Maintenance**:

   - **Deployment**: Once the system passes all testing phases, it is deployed for user access.
   - **Continuous Monitoring and Feedback**: The platform is monitored for performance and user engagement, with ongoing collection of user feedback to inform future updates.
   - **Regular Updates**: The SSH platform will undergo regular updates based on user feedback and technological advancements, ensuring that it remains relevant and effective as an educational tool.

CHAPTER 4: SYSTEM DESIGN

The process of designing the Sort and Search Hub (SSH) involves defining its components or modules to satisfy specific requirements. This section presents various diagrams and visual representations that depict the design and creation process of the SSH platform.

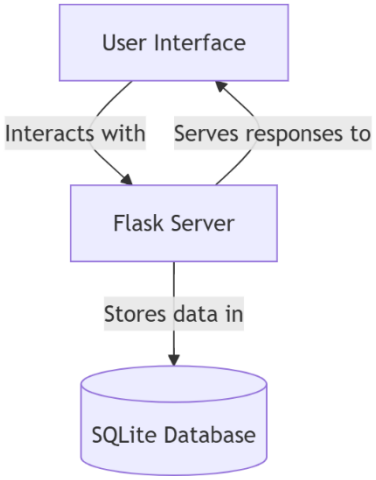
4.1 System Architecture:

SSH's architecture follows a three-tier structure, which ensures separation of concerns and facilitates scalability and maintenance.

1. Presentation Layer: This layer is responsible for the user interface and is developed using HTML, CSS, and JavaScript. It interacts directly with the users, capturing their inputs and displaying the outputs generated by the application layer.

2. Application Layer: This layer handles the business logic and is implemented using Flask, a lightweight WSGI web application framework in Python. It processes the user inputs, executes the algorithmic logic, and generates the appropriate responses to be displayed on the user interface.

3. Data Layer: This layer is responsible for data storage and management. It uses an SQLite database to store user data, interaction history, and other relevant information. The database ensures data persistence and allows for efficient retrieval and manipulation of data.

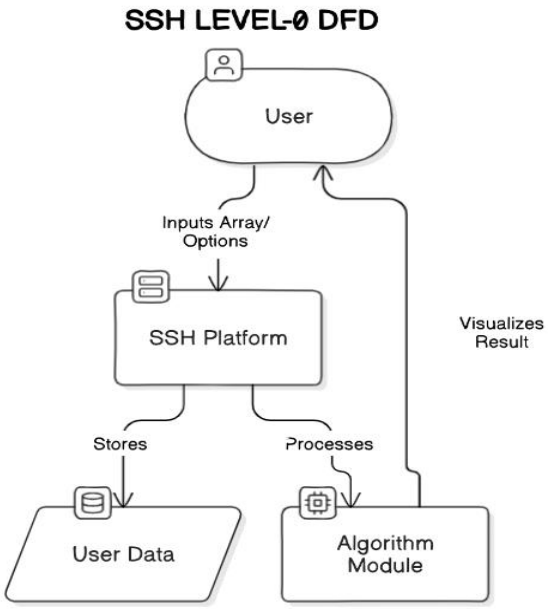


System Architecture Diagram

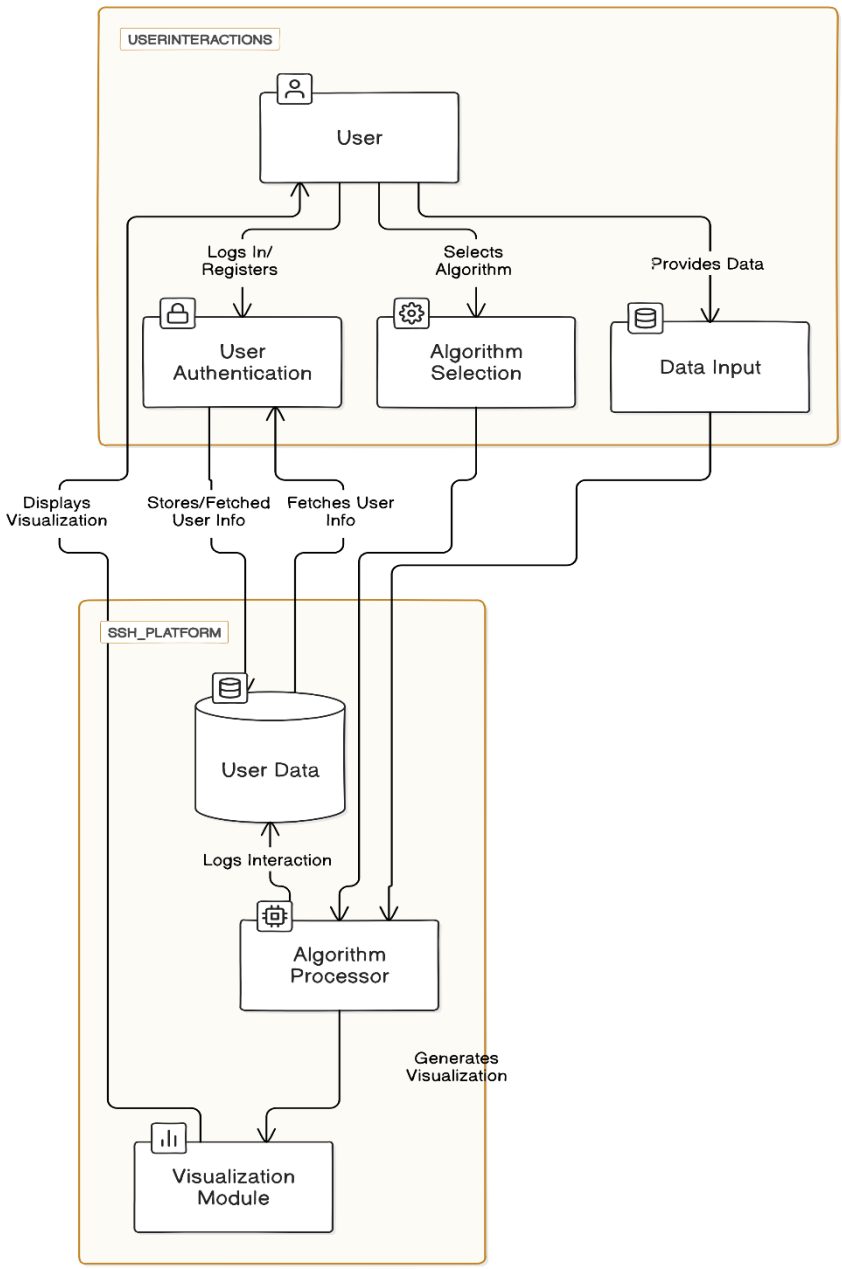
4.2 Data Flow Diagram:

Data Flow Diagrams (DFD) provide a graphical representation of the data flow within the system. They depict how data moves through the system, the processes that transform the data, and the storage locations.

4.2.1 DFD Level 0:

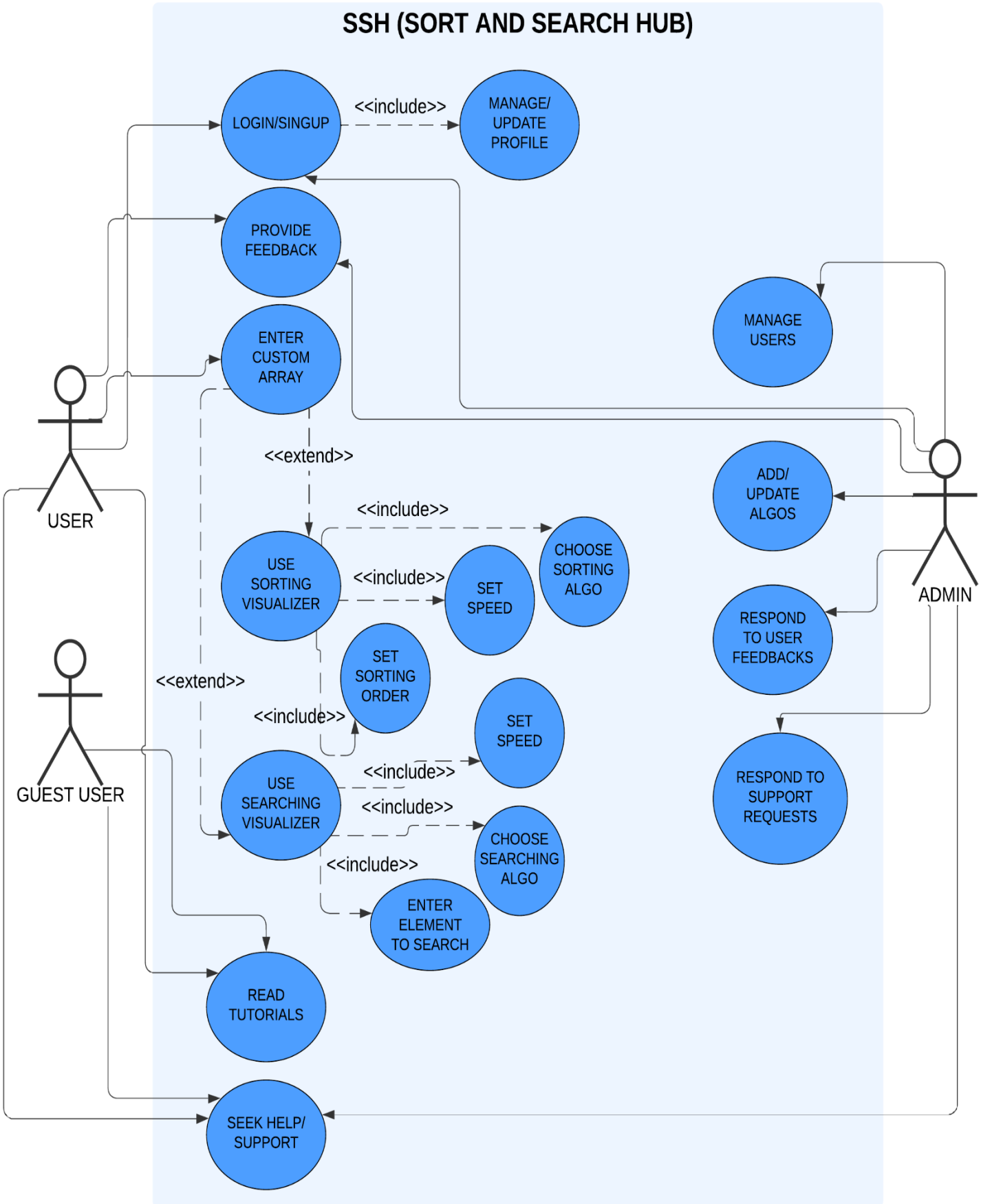


4.2.2 DFD Level 1:

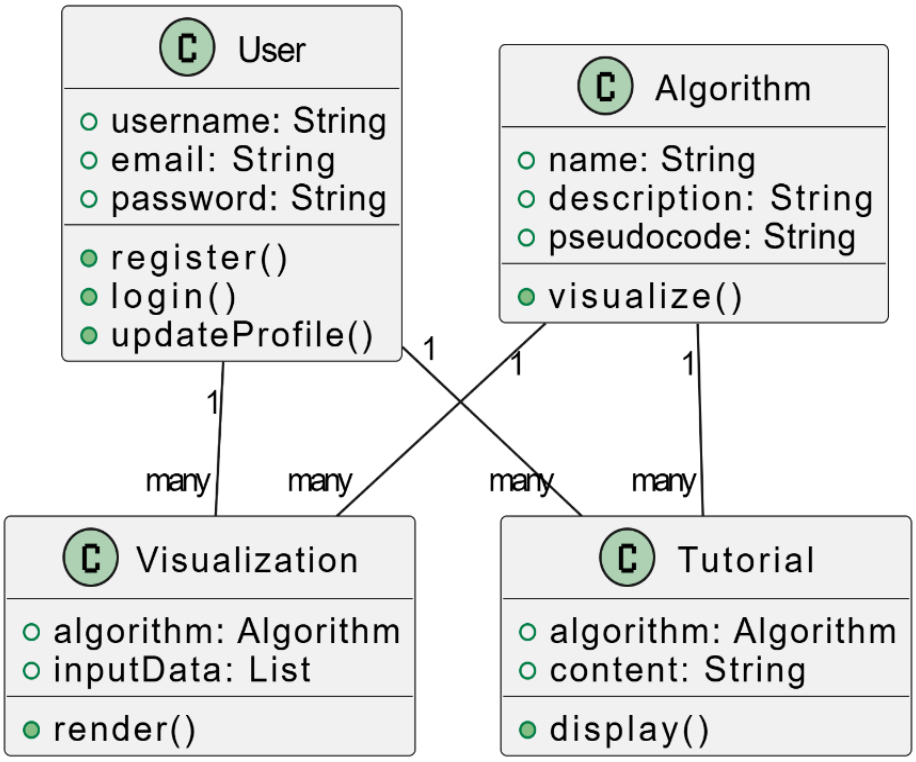


4.3 PHYSICAL DESIGN:

4.3.1 Use Case Diagram:

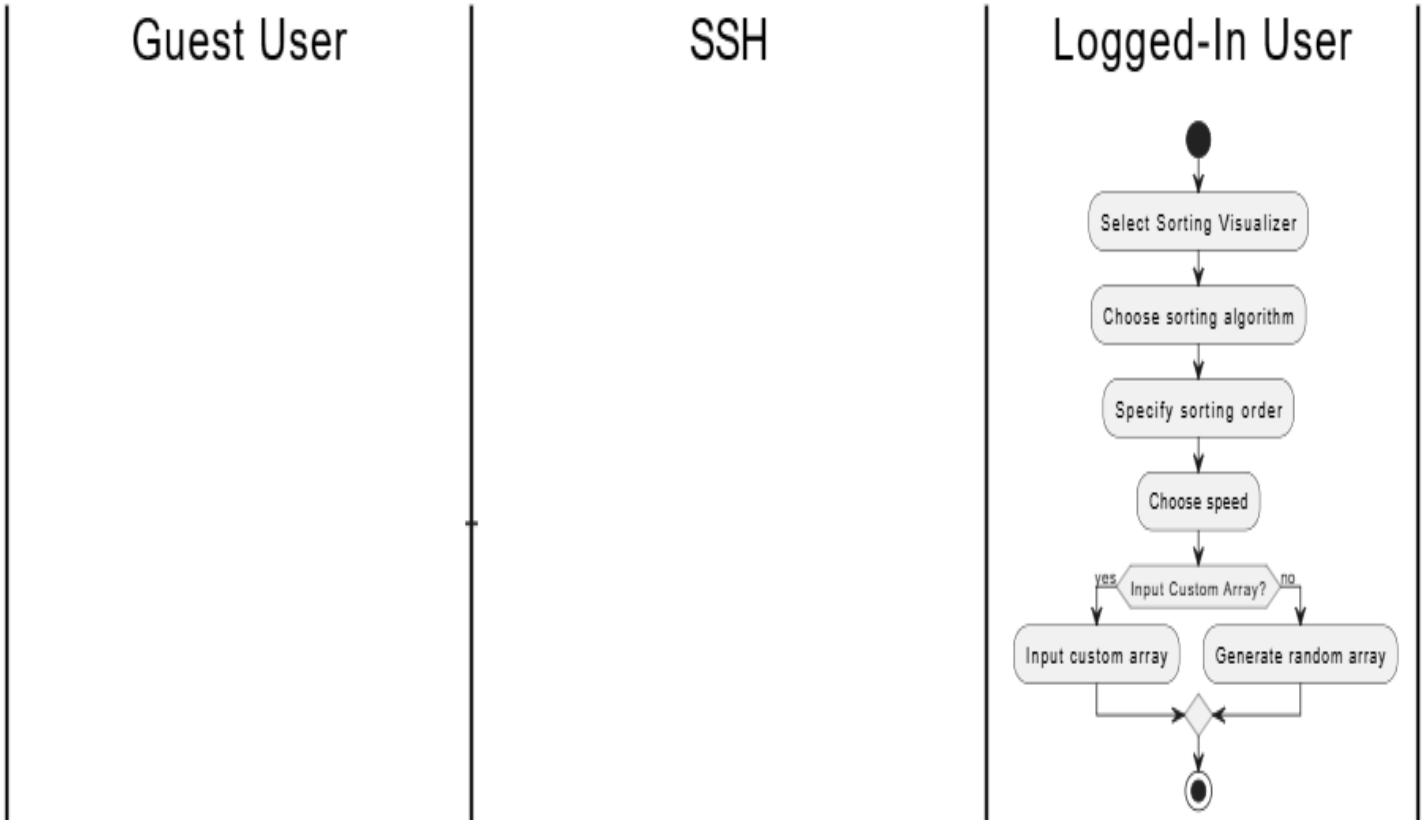


4.3.2 Class Diagram:

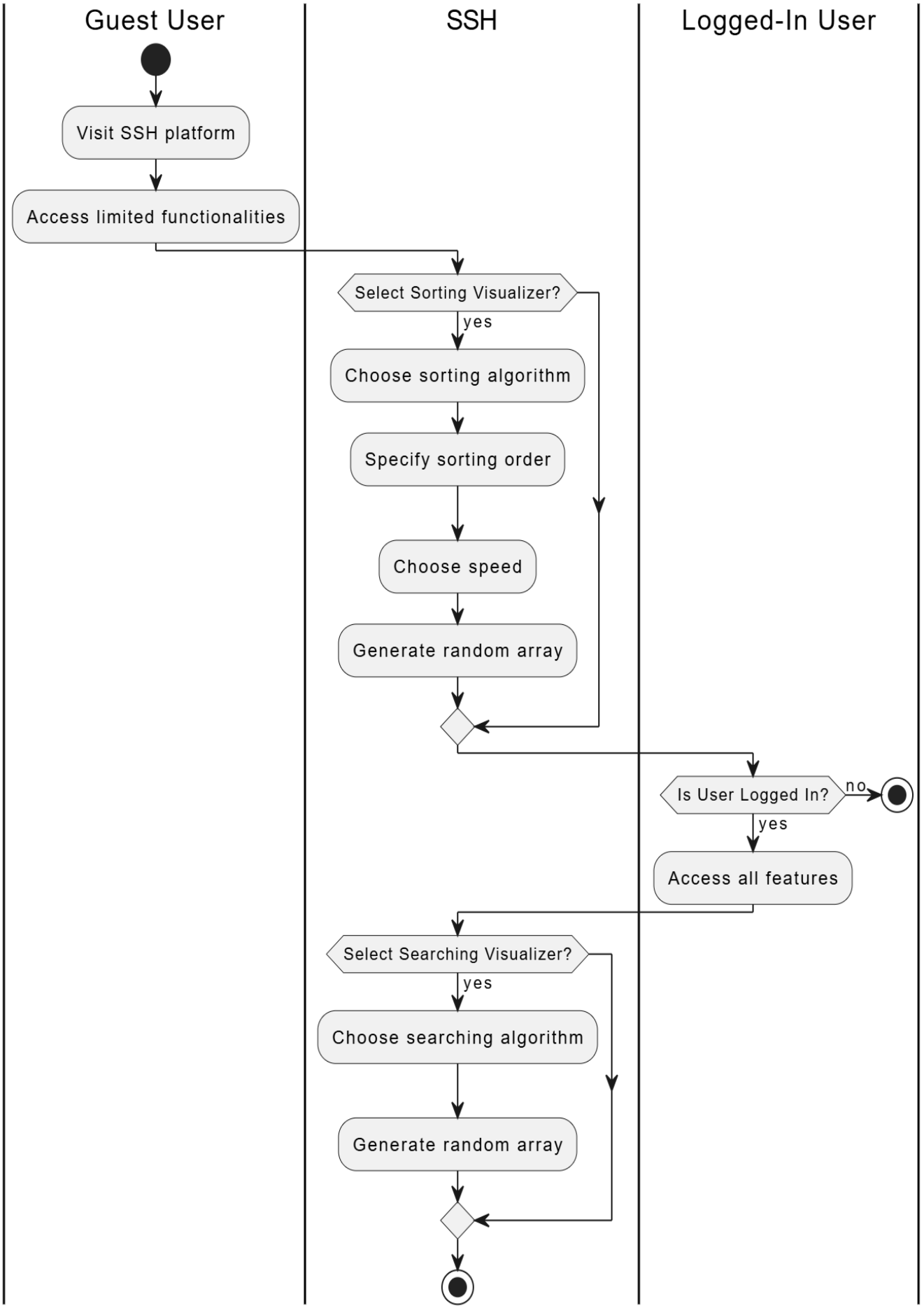


4.3.3 Activity Diagram:

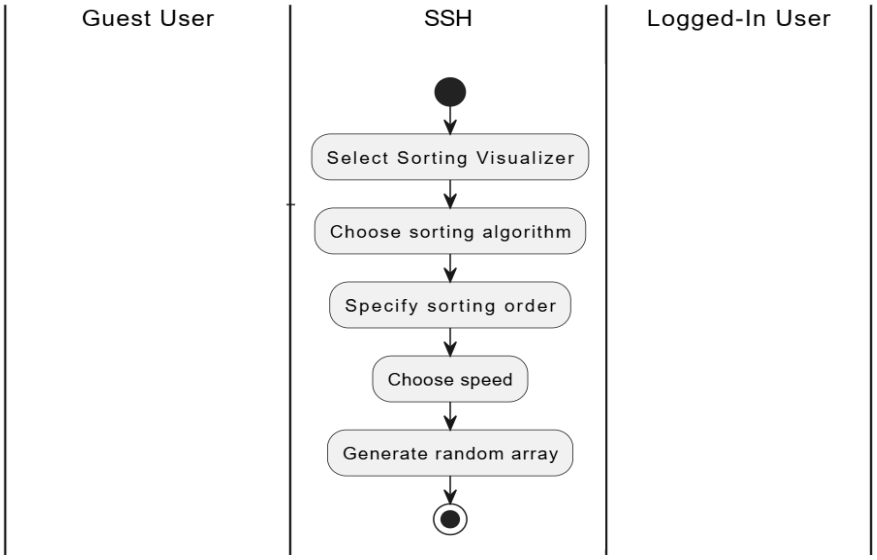
a) Sorting Visualizer for Logged-In Users Activity diagram:



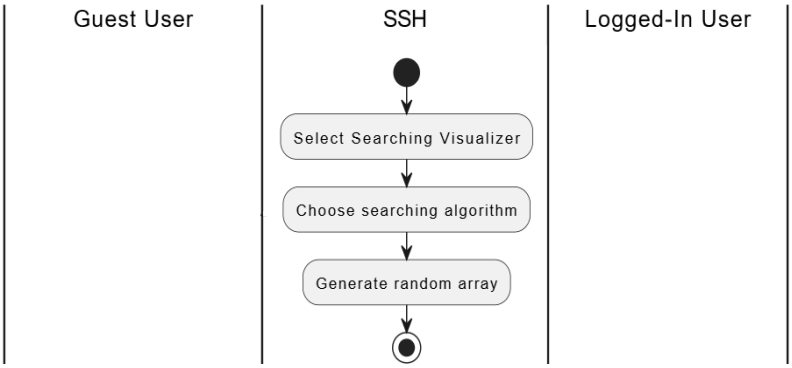
b) User Registration and Access Activity diagram:



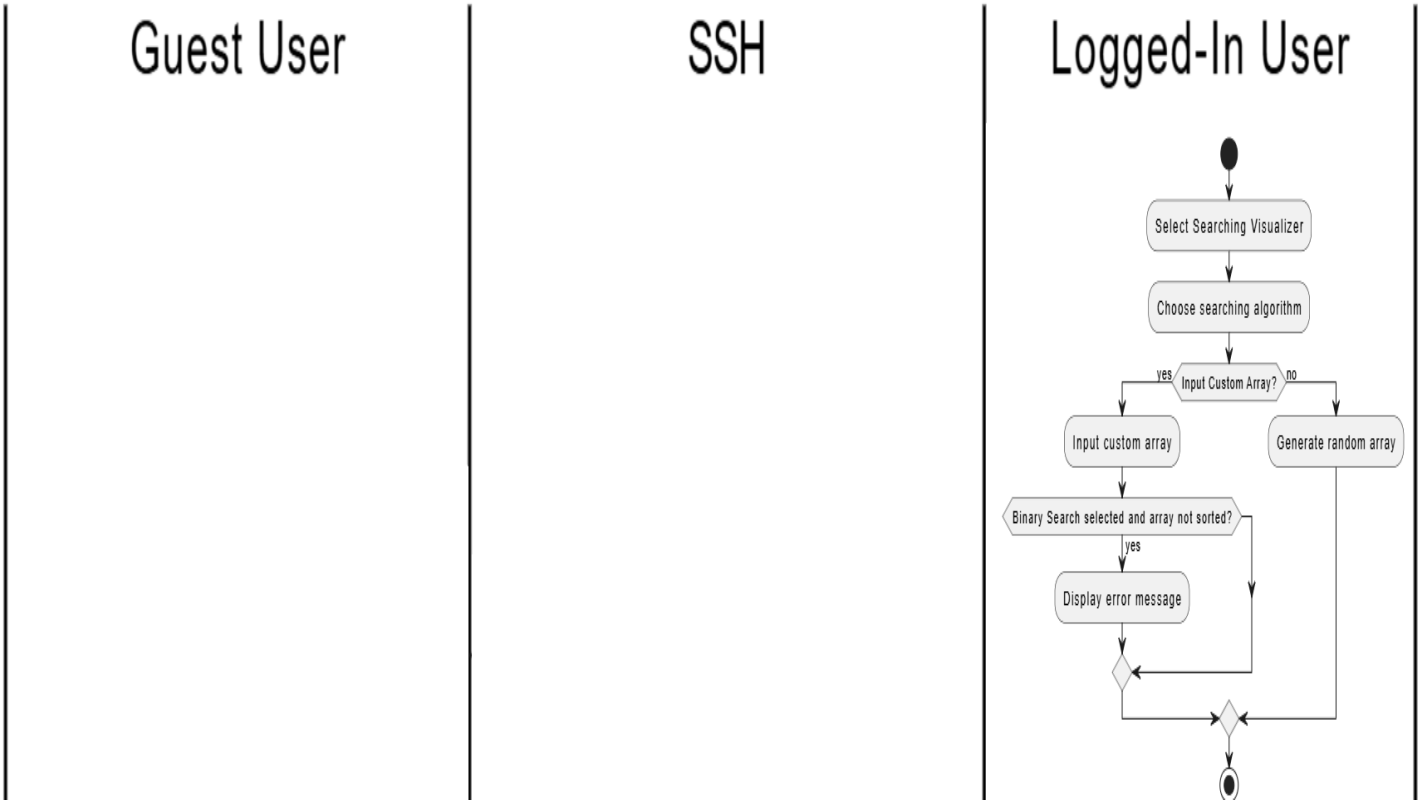
c) Sorting Visualizer for Guest Users Activity diagram:



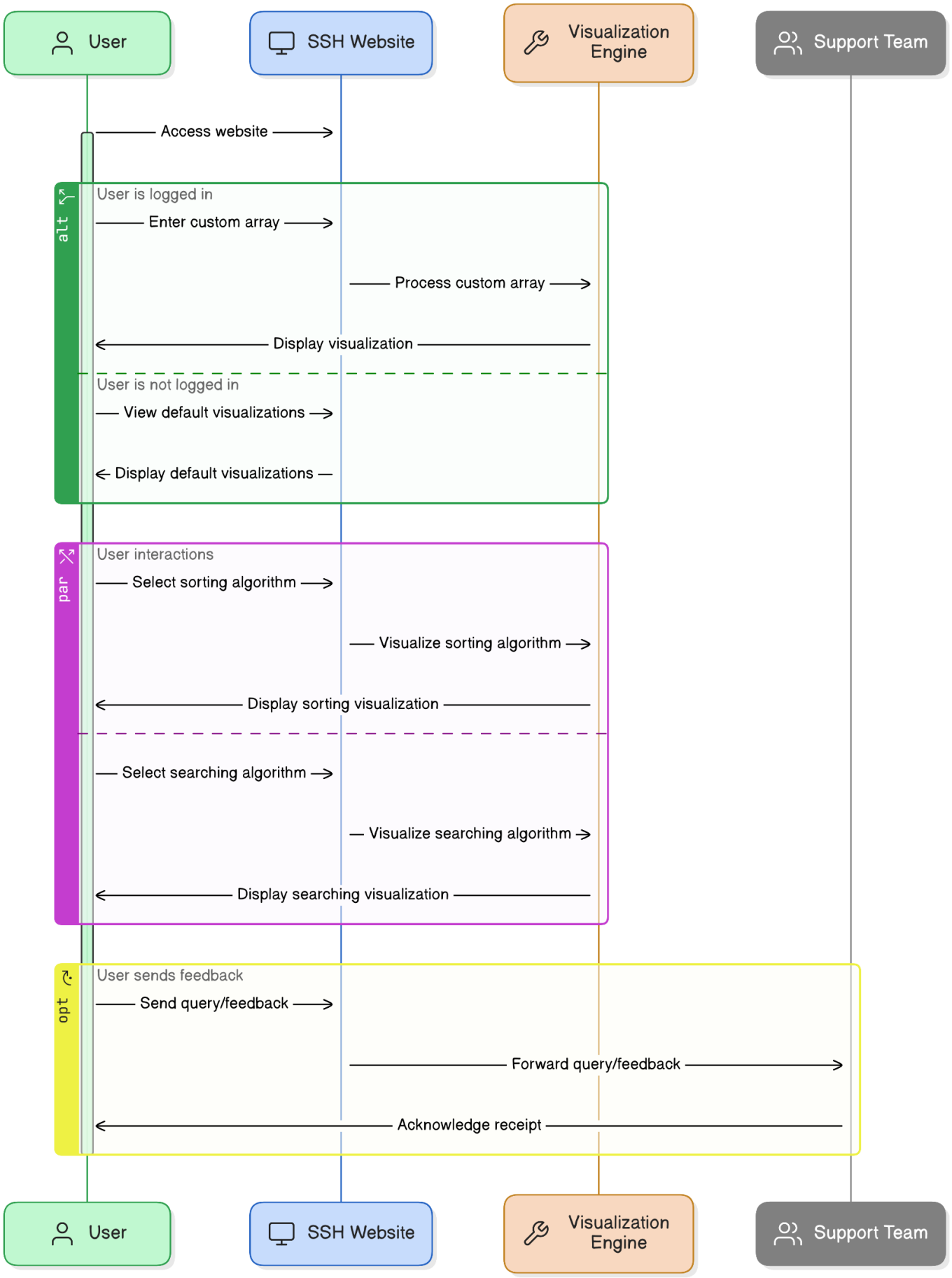
d) Searching Visualizer for Guest Users Activity diagram:



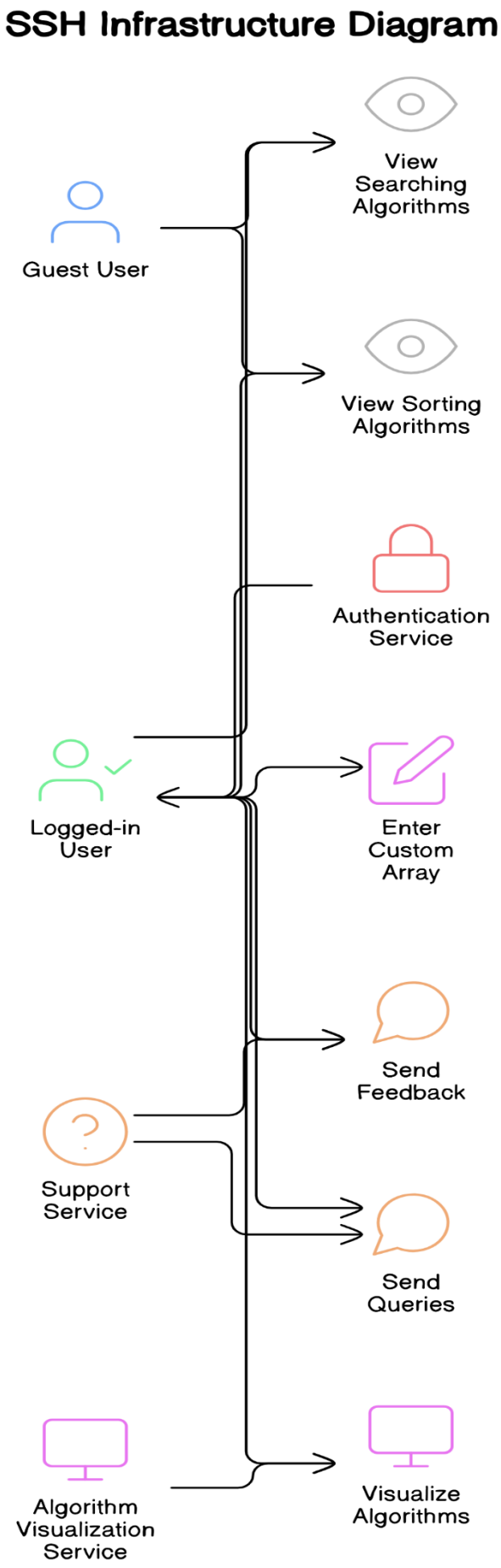
e) Searching Visualizer for Logged-In Users Activity diagram:



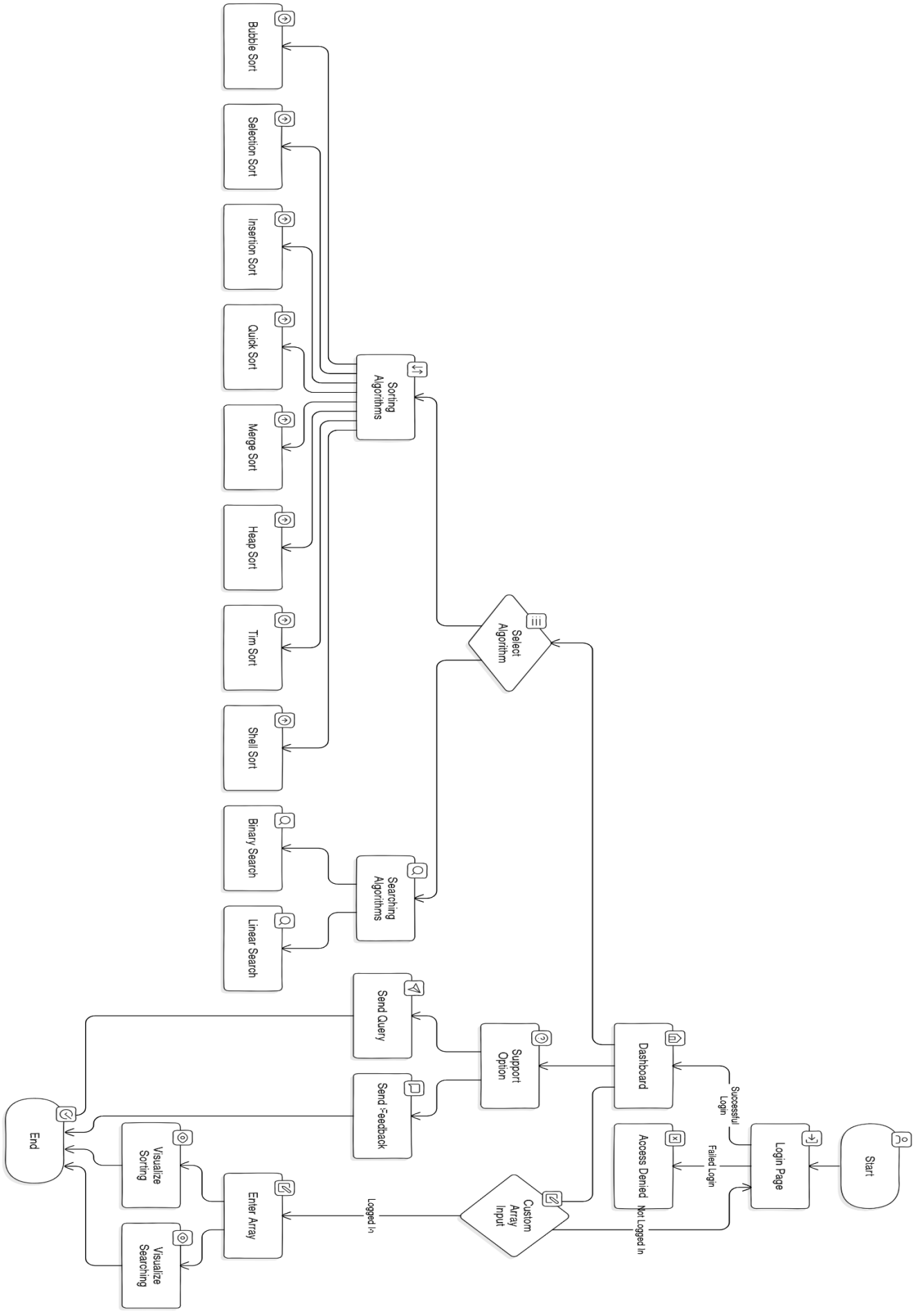
4.3.4 Sequence Diagram:



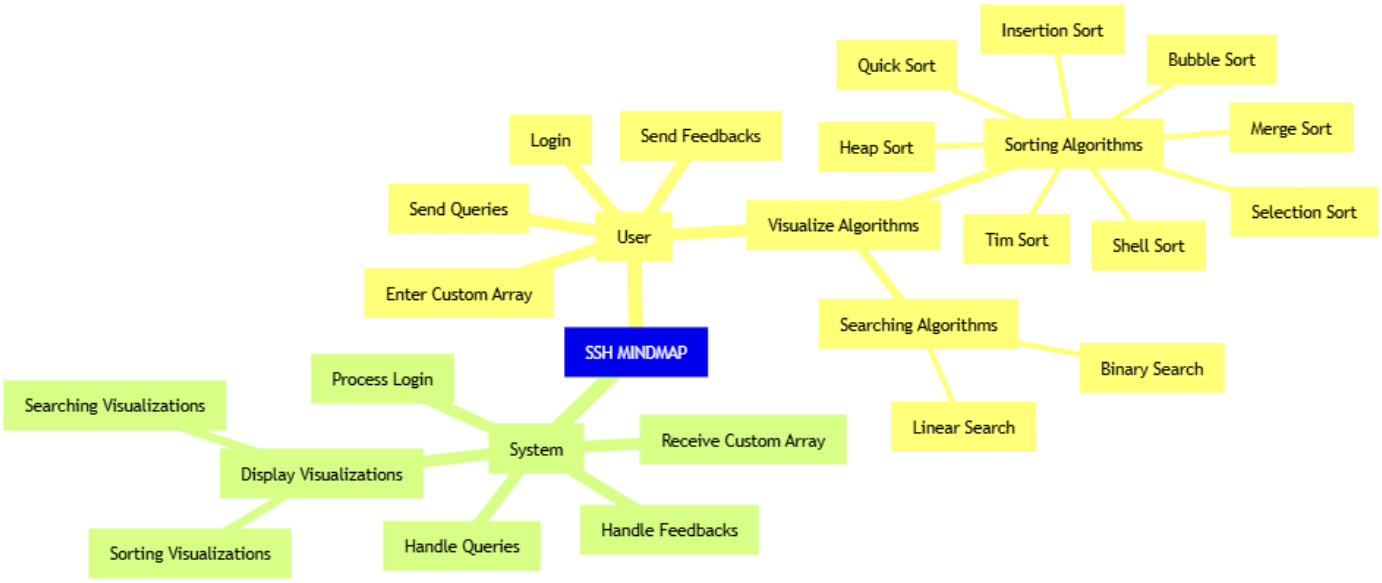
4.3.5 Infrastructure Diagram:



4.3.6 FLOWCHART FOR SSH:



4.3.7 MINDMAP:



4.4 Database Design:

4.4.1 Introduction:

An entity-relationship model (ER model) describes the data and relationships within the SSH (SSH Sort and Search Hub) platform, providing a foundation for its relational database structure. The primary components of the ER model for SSH are:

- **Entities**: Key objects within the system that represent significant data points. In SSH, these include:
    - **User**: Represents registered users of the platform, capturing their details like username, email, and preferences.
    - **Feedback**: Represents user feedback submitted to the platform, including the content of the feedback.
    - **Algorithm**: Represents various sorting and searching algorithms available for visualization.
- **Relationships**: Associations between entities that convey meaningful interactions. Examples in SSH include:
    - **Submits**: A relationship between User and Feedback, indicating that a user can submit feedback.
    - **Visualizes**: A relationship indicating that users can visualize different algorithms.
- **Attributes**: Characteristics that provide further detail about entities. Examples include:

- **User**: Attributes like firstname, lastname, username, and email.
- **Feedback**: Attributes such as content and timestamp.
- **Algorithm**: Attributes that define each algorithm's properties.

4.4.2 Database Specification:

4.4.2.1 Table USER :-

Column Name	Type	Not Null	Primary Key	Default
id	INTEGER	YES	YES	DEFAULT None
firstname	varchar(150)	YES		DEFAULT None
lastname	varchar(150)	YES		DEFAULT None
username	varchar(150)	YES	UNIQUE	DEFAULT None
email	varchar(150)	YES	UNIQUE	DEFAULT None
country	varchar(100)	YES		DEFAULT None
gender	varchar(20)	YES		DEFAULT None
password	varchar(150)	YES		DEFAULT None

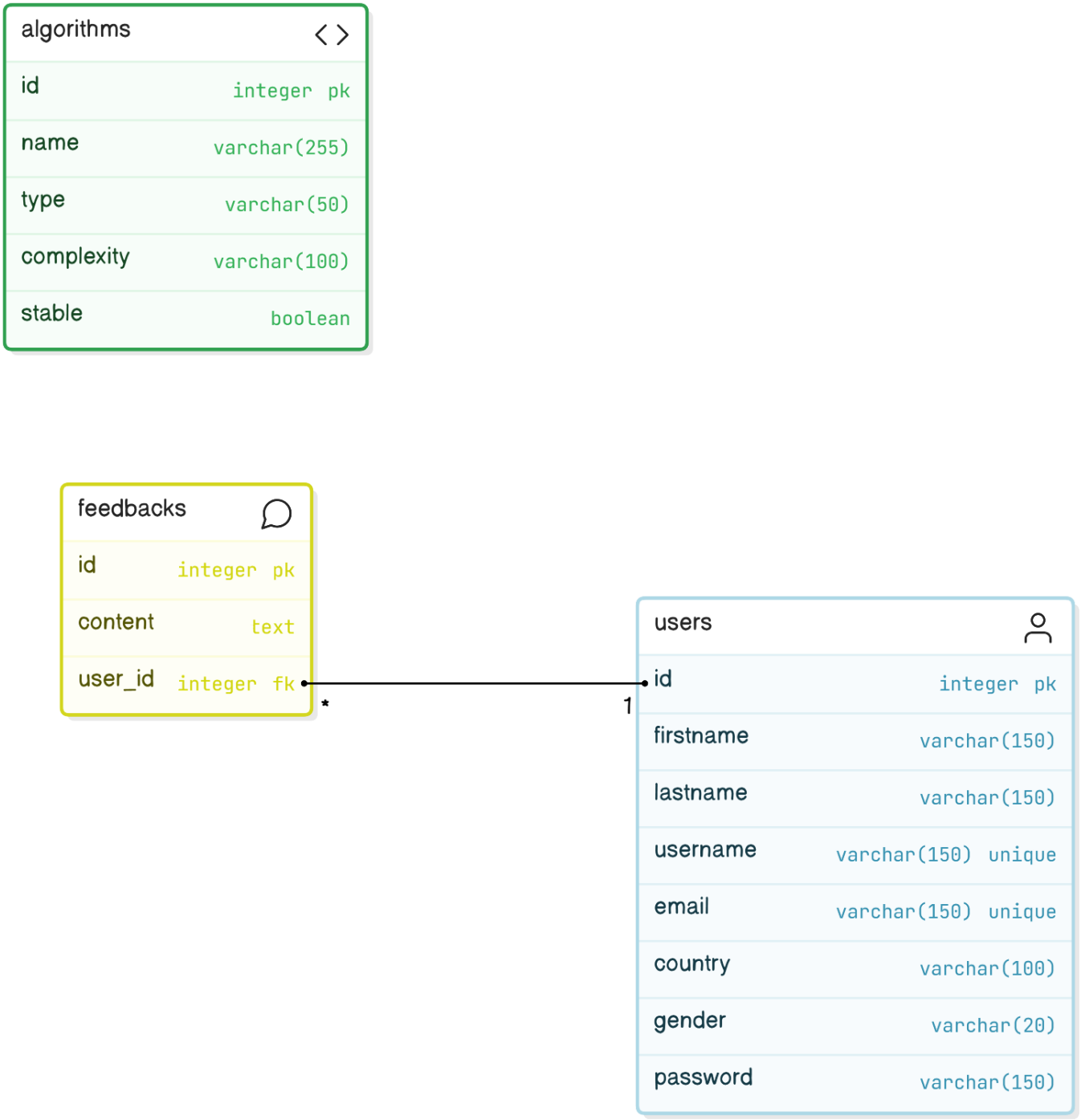
4.4.2.2 Table FEEDBACK :-

Column Name	Type	Not Null	Primary Key	Default
id	INTEGER	YES	YES	DEFAULT None
content	TEXT	YES		DEFAULT None
user_id	INTEGER	YES	FOREIGN KEY	DEFAULT None

4.4.2.3 Table ALGORITHM :-

Column Name	Type	Not Null	Primary Key	Default
id	INTEGER	YES	YES	DEFAULT None
name	varchar(255)	YES		DEFAULT None
type	varchar(50)	YES		DEFAULT None
complexity	varchar(100)	YES		DEFAULT None
stable	BOOLEAN	YES		DEFAULT None

4.4.3 ERD(ENTITY-RELATIONSHIP DIAGRAM):



CHAPTER 5: SYSTEM REQUIREMENT STUDY

5.1 Existing System:

The existing educational platforms for algorithm visualization often fall short in providing comprehensive interactivity and coverage of a wide range of algorithms. While platforms like VisuAlgo and Algomation offer valuable visualizations for commonly taught algorithms, they typically lack depth and engagement. These tools often present static visualizations that do not allow users to manipulate data or interact with the algorithms in meaningful ways. This limitation restricts users from exploring the algorithms at their own pace and verifying their understanding through experimentation.

There is a clear gap in current educational resources where platforms either offer only basic visualizations without interactivity or focus narrowly on a limited set of algorithms. This results in a learning experience that may not fully engage users or address their diverse learning needs. Additionally, existing resources often provide insufficient explanations linking visual representations to the underlying theoretical concepts, which hampers learners' understanding of the material.

Recognizing these challenges, the SSH Sort and Search Hub aims to create a more engaging and comprehensive learning experience. By emphasizing user interactivity, SSH allows users to input custom data, select from a broad range of algorithms, and visualize the sorting and searching processes in real-time. This dynamic approach encourages users to experiment, make predictions, and validate their learning through hands-on interaction with the algorithms. SSH not only visualizes algorithms but also integrates detailed tutorials that elucidate the reasoning behind each step, the theoretical foundations, and practical applications.

By addressing these gaps, SSH is positioned to provide a holistic educational platform that combines visual, textual, and interactive elements, facilitating deeper comprehension and retention of sorting and searching algorithms in the digital age.

5.2 Proposed System:

The proposed system for the SSH (SSH Sort and Search Hub) aims to deliver a robust educational platform designed to facilitate a comprehensive understanding of sorting and searching algorithms. The platform integrates real-time interactivity with algorithm visualizations, creating an engaging and intuitive user experience. Key features of the proposed system include:

- **Interactive Algorithm Visualizations**: Users will be able to interact with a wide range of sorting and searching algorithms, input custom data, and visualize the algorithm's operations in real-time. This interactivity fosters deeper engagement and enhances the learning experience.
- **Real-time Data Input and Validation**: The platform allows users to enter custom data for visualization purposes, equipped with built-in validation mechanisms to ensure the accuracy and integrity of the data. This minimizes errors and enhances the quality of the learning materials.
- **Immediate Feedback Mechanisms**: Upon completing algorithm visualizations, users will receive instant feedback regarding their inputs and the performance of the algorithms. This timely information enhances the learning process by allowing users to adjust their approaches based on outcomes.
- **Integrated Feedback System**: An integrated feedback mechanism will allow users to submit their insights and experiences regarding the platform's features, providing valuable data for continuous improvement. This ensures that the SSH platform evolves based on user needs and preferences.
- **Comprehensive Resource Accessibility**: The system will provide easy access to a wide range of educational resources, including tutorials, detailed explanations, and performance analytics. This holistic approach supports users in developing a well-rounded understanding of algorithms.

By implementing the proposed system, SSH aims to optimize educational workflows, improve user engagement, and enhance the overall learning experience. The integration of interactive visualizations, robust data management, and user feedback mechanisms will contribute to a more effective and responsive educational platform, ultimately promoting deeper understanding and retention of sorting and searching algorithms.

5.3 Objective of the System:

The primary objective of the SSH platform is to provide a comprehensive and user-friendly educational resource for learning sorting and searching algorithms. Key goals include:

- **User-Centric Experience:** Create an intuitive platform that allows users to navigate easily and engage deeply with algorithm visualizations, enhancing satisfaction and usability.
- **Comprehensive Educational Resource:** Offer in-depth insights into algorithms, including theoretical foundations, practical applications, and step-by-step tutorials for a solid understanding.

- **Time-Efficient Learning:** Streamline the learning process with automated visualizations and immediate feedback, saving users time and facilitating quicker comprehension.
- **Reduction of Manual Efforts:** Minimize manual effort through automation and real-time interaction, allowing users to focus on understanding and application.
- **Digitization of Educational Processes:** Automate key processes like algorithm selection and data visualization to enhance learning efficiency.
- **Holistic Management of Learning Resources:** Organize a comprehensive repository of algorithms, tutorials, and user feedback, ensuring accuracy and accessibility.

By achieving these objectives, SSH aims to set a new standard in educational platforms for algorithm visualization, driving engagement and success in digital learning.

5.4 System Specifications:

5.4.1 Hardware Requirements:

- **Minimum Hardware Requirements:**
  - Processor: Intel Atom® processor or similar (2 cores, 1.6 GHz or better)
  - RAM: 2 GB
  - Storage: 250 GB HDD or SSD
- **Recommended Hardware Requirements:**
  - Processor: Intel® Core™ processor or similar/better processor
  - RAM: 4 GB or more
  - Storage: 500 GB HDD or SSD for improved performance
  - Network: 1 Gbit/10 Gbit network capability
  - Disk Space: 5 GB of free disk space

5.4.2 Software Requirements:

- **Operating System:** Windows 7 or higher, macOS X 10.11 or higher, or Linux (RHEL 6/7, 64-bit; Ubuntu compatible)
- **Programming Language:** Python
- **Python Version:** 3.6.4 or later
- **Web Framework:** Flask 2.1.2
- **Database Management:** SQLite
- **Additional Tools:** Visual Studio Code or PyCharm for development
- **Email Handling:** Flask-Mail in combination with Elastic Mail SMTP for user feedback management

CHAPTER 6: SOFTWARE CODING, TESTING AND IMPLEMENTATION

6.1 INTRODUCTION

6.1.1 System Coding Environment and Standards Followed:

The coding phase of the SSH platform involves implementing various modules as specified in the design document. This process utilizes a high-level programming language, primarily Python, and follows a structured approach to ensure that each component functions as intended. The main goal is to translate the design specifications into functional code while adhering to best practices in software development.

In line with good software development practices, it is essential for programmers to adhere to well-defined coding standards. These standards foster consistency, readability, and maintainability across the codebase, which is particularly important for collaborative projects. Adhering to coding standards not only facilitates code reviews but also enhances the overall quality of the software.

Key coding standards for the SSH project include:

- **Limited Use of Global Variables:** Minimize the use of global variables to enhance code modularity and prevent unintended side effects.
- **Use of Standard Headers:** Employ standard headers for different modules to ensure clarity and organization within the code.
- **Naming Conventions:** Follow clear naming conventions for local variables, global variables, constants, and functions to promote understanding and maintainability.
- **Consistent Indentation:** Maintain proper indentation throughout the code to improve readability and structure.
- **Simplicity in Style:** Avoid overly complex coding styles to ensure that the code remains easy to understand for current and future developers.

6.1.2 Sample Code Layouts:

app.py file:

```
from flask import Flask, render_template, request, jsonify, redirect, url_for, flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, login_user, login_required, logout_user, current_user, UserMixin
from flask_bcrypt import Bcrypt
from flask_mail import Mail, Message
import os
import secrets
```

```
import re
app = Flask(__name__)
app.config['SECRET_KEY'] = secrets.token_hex(16)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
login_manager = LoginManager()
login_manager.login_view = 'login'
login_manager.init_app(app)
bcrypt = Bcrypt(app)
class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    firstname = db.Column(db.String(150), nullable=False)
    lastname = db.Column(db.String(150), nullable=False)
    username = db.Column(db.String(150), unique=True, nullable=False)
    email = db.Column(db.String(150), unique=True, nullable=False)
    country = db.Column(db.String(100), nullable=False)
    gender = db.Column(db.String(20), nullable=False)
    password = db.Column(db.String(150), nullable=False)
    feedbacks = db.relationship('Feedback', backref='author', lazy=True)
class Feedback(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.Text, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
with app.app_context():
    db.create_all()
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))
def is_valid_password(password):
    return (
        len(password) >= 12 and
        re.search(r'[A-Z]', password) and
        re.search(r'[a-z]', password) and
        re.search(r'\d', password) and
        re.search(r'[!@#$%^&*(),.?":{}|<>]', password)
    )
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/sorting')
def sorting_page():
    return render_template('sorting.html')
@app.route('/searching')
def searching_page():
    return render_template('searching.html')
app.config.update(
    MAIL_SERVER='smtp.elasticemail.com',
    MAIL_PORT=2525,
    MAIL_USE_TLS=True,
    MAIL_USE_SSL=False,
    MAIL_USERNAME='contactthessh@gmail.com',
    MAIL_PASSWORD='6EC602E55B61C26529531CEF3DBD6C50E855'
)
mail = Mail(app)
@app.route('/support', methods=['GET', 'POST'])
def support():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        phone = request.form['phone']
        message_body = request.form['message']
        msg = Message(
            subject=f"New Enquiry from {name}",
            sender=app.config['MAIL_USERNAME'],
            recipients=["contactthessh@gmail.com"],
            reply_to=email
```

```
        )
        msg.html = f"""
            <h3>New Enquiry</h3>
            <p><strong>Name:</strong> {name}</p>
            <p><strong>Email:</strong> {email}</p>
            <p><strong>Phone:</strong> {phone}</p>
            <p><strong>Message:</strong></p>
            <p>{message_body}</p>
        """
        try:
            mail.send(msg)
            flash("Message Sent Successfully", "success")
        except Exception as e:
            flash(f"Failed to send message: {str(e)}", "danger")
        return redirect(url_for('support'))
    return render_template('support.html')
@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('index'))
    if request.method == 'POST':
        firstname = request.form.get('firstname').strip()
        lastname = request.form.get('lastname').strip()
        country = request.form.get('country').strip()
        gender = request.form.get('gender').strip()
        username = request.form.get('username').strip()
        email = request.form.get('email').strip()
        password = request.form.get('password').strip()
        confirm_password = request.form.get('confirm_password').strip()
        if not all([firstname, lastname, country, gender, username, email, password, confirm_password]):
            flash('Please fill out all fields.', 'danger')
            return redirect(url_for('register'))
        if password != confirm_password:
            flash('Passwords do not match.', 'danger')
            return redirect(url_for('register'))
        if not is_valid_password(password):
            flash('Password must be at least 12 characters long and include uppercase, lowercase, number, and special character.', 'danger')
            return redirect(url_for('register'))
        if User.query.filter_by(username=username).first():
            flash('Username already exists.', 'danger')
            return redirect(url_for('register'))
        if User.query.filter_by(email=email).first():
            flash('Email already registered.', 'danger')
            return redirect(url_for('register'))
        hashed_password = bcrypt.generate_password_hash(password).decode('utf-8')
        new_user = User(
            firstname=firstname,
            lastname=lastname,
            country=country,
            gender=gender,
            username=username,
            email=email,
            password=hashed_password
        )
        db.session.add(new_user)
        db.session.commit()
        flash('Registration successful! You can now log in.', 'success')
        return redirect(url_for('login'))
    return render_template('register.html')
@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('index'))
    if request.method == 'POST':
        username_or_email = request.form.get('username_or_email').strip()
        password = request.form.get('password').strip()
        user = User.query.filter(
```

```
            (User.username == username_or_email) | (User.email == username_or_email)
        ).first()
        if not user or not bcrypt.check_password_hash(user.password, password):
            flash('Invalid credentials. Please try again.', 'danger')
            return redirect(url_for('login'))
        login_user(user)
        flash('Logged in successfully!', 'success')
        return redirect(url_for('index'))
    return render_template('login.html')
@app.route('/logout')
@login_required
def logout():
    logout_user()
    flash('You have been logged out.', 'info')
    return redirect(url_for('index'))
@app.route('/profile', methods=['GET', 'POST'])
@login_required
def profile():
    if request.method == 'POST':
        firstname = request.form.get('firstname').strip()
        lastname = request.form.get('lastname').strip()
        country = request.form.get('country').strip()
        gender = request.form.get('gender').strip()
        username = request.form.get('username').strip()
        email = request.form.get('email').strip()
        old_password = request.form.get('old_password').strip()
        new_password = request.form.get('new_password').strip()
        confirm_new_password = request.form.get('confirm_new_password').strip()
        if not all([firstname, lastname, country, gender, username, email]):
            flash('First name, Last name, Country, Gender, Username, and Email cannot be empty.', 'danger')
            return redirect(url_for('profile'))
        if new_password or confirm_new_password:
            if not old_password:
                flash('Please enter your current password to update your password.', 'danger')
                return redirect(url_for('profile'))
            if not bcrypt.check_password_hash(current_user.password, old_password):
                flash('Current password is incorrect.', 'danger')
                return redirect(url_for('profile'))
            if new_password != confirm_new_password:
                flash('New passwords do not match.', 'danger')
                return redirect(url_for('profile'))
            if not is_valid_password(new_password):
                flash('New password must be at least 12 characters long and include uppercase, lowercase, number, and special character.', 'danger')
                return redirect(url_for('profile'))
            current_user.password = bcrypt.generate_password_hash(new_password).decode('utf-8')
        if User.query.filter(User.username == username, User.id != current_user.id).first():
            flash('Username already taken.', 'danger')
            return redirect(url_for('profile'))
        if User.query.filter(User.email == email, User.id != current_user.id).first():
            flash('Email already registered.', 'danger')
            return redirect(url_for('profile'))
        current_user.firstname = firstname
        current_user.lastname = lastname
        current_user.country = country
        current_user.gender = gender
        current_user.username = username
        current_user.email = email
        db.session.commit()
        flash('Profile updated successfully.', 'success')
        return redirect(url_for('profile'))
    return render_template('profile.html')
@app.route('/feedback', methods=['GET', 'POST'])
@login_required
def feedback():
    if request.method == 'POST':
        content = request.form.get('content').strip()
        if not content:
```

```
            flash('Feedback cannot be empty.', 'danger')
            return redirect(url_for('feedback'))
        new_feedback = Feedback(content=content, author=current_user)
        db.session.add(new_feedback)
        db.session.commit()
        flash('Thank you for your feedback!', 'success')
        return redirect(url_for('feedback'))
    feedbacks = Feedback.query.filter_by(user_id=current_user.id).all()
    return render_template('feedback.html', feedbacks=feedbacks)
@app.route('/sort', methods=['POST'])
def sort_route():
    data = request.json
    array = data.get('array', [])
    algorithm = data.get('algorithm', '')
    ascending = data.get('ascending', True)
    if algorithm not in ['bubble', 'selection', 'insertion', 'quick', 'merge', 'heap', 'tim', 'shell']:
        return jsonify({'error': 'Invalid sorting algorithm'}), 400
    try:
        steps = []
        if algorithm == 'bubble':
            steps = bubble_sort(array, ascending)
        elif algorithm == 'selection':
            steps = selection_sort(array, ascending)
        elif algorithm == 'insertion':
            steps = insertion_sort(array, ascending)
        elif algorithm == 'quick':
            steps = quick_sort(array, ascending)
        elif algorithm == 'merge':
            steps = merge_sort(array, ascending)
        elif algorithm == 'heap':
            steps = heap_sort(array, ascending)
        elif algorithm == 'tim':
            steps = tim_sort(array, ascending)
        elif algorithm == 'shell':
            steps = shell_sort(array, ascending)
    except Exception as e:
        return jsonify({'error': str(e)}), 500
    return jsonify({'steps': steps})
def bubble_sort(array, ascending=True):
    steps = []
    n = len(array)
    for i in range(n):
        for j in range(0, n-i-1):
            steps.append(([x for x in array], [j, j+1]))
            if (array[j] > array[j+1] and ascending) or (array[j] < array[j+1] and not ascending):
                array[j], array[j+1] = array[j+1], array[j]
                steps.append(([x for x in array], [j, j+1]))
    return steps
def selection_sort(array, ascending=True):
    steps = []
    n = len(array)
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            steps.append(([x for x in array], [min_idx, j]))
            if (array[j] < array[min_idx] and ascending) or (array[j] > array[min_idx] and not ascending):
                min_idx = j
        if min_idx != i:
            array[i], array[min_idx] = array[min_idx], array[i]
            steps.append(([x for x in array], [i, min_idx]))
    return steps
def insertion_sort(array, ascending=True):
    steps = []
    for i in range(1, len(array)):
        key = array[i]
        j = i -1
        while j >=0 and ((array[j] > key and ascending) or (array[j] < key and not ascending)):
```

```
            steps.append(([x for x in array], [j, j+1]))
            array[j +1] = array[j]
            j -=1
        array[j +1] = key
        steps.append(([x for x in array], [j +1]))
    return steps
def quick_sort(array, ascending=True):
    steps = []
    def _quick_sort(arr, low, high):
        if low < high:
            pi = partition(arr, low, high)
            _quick_sort(arr, low, pi -1)
            _quick_sort(arr, pi +1, high)
    def partition(arr, low, high):
        pivot = arr[high]
        i = low -1
        for j in range(low, high):
            steps.append(([x for x in arr], [j, high]))
            if (arr[j] < pivot and ascending) or (arr[j] > pivot and not ascending):
                i +=1
                arr[i], arr[j] = arr[j], arr[i]
                steps.append(([x for x in arr], [i, j]))
        arr[i+1], arr[high] = arr[high], arr[i+1]
        steps.append(([x for x in arr], [i+1, high]))
        return i+1
    _quick_sort(array, 0, len(array)-1)
    return steps
def merge_sort(array, ascending=True):
    steps = []
    def _merge_sort(arr, l, r):
        if l < r:
            m = (l + r) //2
            _merge_sort(arr, l, m)
            _merge_sort(arr, m+1, r)
            merge(arr, l, m, r)
    def merge(arr, l, m, r):
        n1 = m -l +1
        n2 = r -m
        L = arr[l:m+1]
        R = arr[m+1:r+1]
        i = j =0
        k = l
        while i < n1 and j < n2:
            steps.append(([x for x in arr], [k]))
            if (L[i] <= R[j] and ascending) or (L[i] >= R[j] and not ascending):
                arr[k] = L[i]
                i +=1
            else:
                arr[k] = R[j]
                j +=1
            k +=1
            steps.append(([x for x in arr], [k-1]))
        while i < n1:
            arr[k] = L[i]
            steps.append(([x for x in arr], [k]))
            i +=1
            k +=1
            steps.append(([x for x in arr], [k-1]))
        while j < n2:
            arr[k] = R[j]
            steps.append(([x for x in arr], [k]))
            j +=1
            k +=1
            steps.append(([x for x in arr], [k-1]))
    _merge_sort(array, 0, len(array)-1)
    return steps
def heap_sort(array, ascending=True):
```

```
    steps = []
    n = len(array)
    def heapify(arr, n, i):
        largest_smallest = i
        l = 2 * i +1
        r = 2 * i +2
        if l < n:
            steps.append(([x for x in arr], [i, l]))
            if (arr[l] > arr[largest_smallest] and ascending) or (arr[l] < arr[largest_smallest] and not ascending):
                largest_smallest = l
        if r < n:
            steps.append(([x for x in arr], [largest_smallest, r]))
            if (arr[r] > arr[largest_smallest] and ascending) or (arr[r] < arr[largest_smallest] and not ascending):
                largest_smallest = r
        if largest_smallest != i:
            arr[i], arr[largest_smallest] = arr[largest_smallest], arr[i]
            steps.append(([x for x in arr], [i, largest_smallest]))
            heapify(arr, n, largest_smallest)
    for i in range(n//2 -1, -1, -1):
        heapify(array, n, i)
    for i in range(n-1, 0, -1):
        array[i], array[0] = array[0], array[i]
        steps.append(([x for x in array], [0, i]))
        heapify(array, i, 0)
    return steps
def tim_sort(array, ascending=True):
    steps = []
    min_run = 32
    def insertion_sort_tim(arr, left, right):
        for i in range(left + 1, right):
            key = arr[i]
            j = i - 1
            while j >= left and ((arr[j] > key and ascending) or (arr[j] < key and not ascending)):
                steps.append(([x for x in array], [j, j + 1]))
                arr[j + 1] = arr[j]
                j -= 1
            arr[j + 1] = key
            steps.append(([x for x in array], [j + 1]))
    for start in range(0, len(array), min_run):
        end = min(start + min_run, len(array))
        insertion_sort_tim(array, start, end)
    size = min_run
    while size < len(array):
        for left in range(0, len(array), size * 2):
            mid = min(left + size, len(array))
            right = min((left + size * 2), len(array))
            if mid < right:
                merged = merge(array[left:mid], array[mid:right], ascending)
                array[left:right] = merged
                steps.append(([x for x in array], list(range(left, right))))
        size *= 2
    return steps
def merge(left, right, ascending=True):
    merged = []
    i = j = 0
    while i < len(left) and j < len(right):
        if (ascending and left[i] <= right[j]) or (not ascending and left[i] >= right[j]):
            merged.append(left[i])
            i += 1
        else:
            merged.append(right[j])
            j += 1
    merged.extend(left[i:])
    merged.extend(right[j:])
    return merged
def shell_sort(array, ascending=True):
    steps = []
```

```
    n = len(array)
    gap = n // 2
    while gap > 0:
        for i in range(gap, n):
            temp = array[i]
            j = i
            while j >= gap and ((ascending and array[j - gap] > temp) or (not ascending and array[j - gap] < temp)):
                steps.append(([x for x in array], [j - gap, j]))
                array[j] = array[j - gap]
                j -= gap
            array[j] = temp
            steps.append(([x for x in array], [j]))
        gap //= 2
    return steps
@app.route('/bubblesort')
def bubblesort():
    return render_template('bubblesort.html')
@app.route('/selectionsort')
def selectionsort():
    return render_template('selectionsort.html')
@app.route('/insertionsort')
def insertionsort():
    return render_template('insertionsort.html')
@app.route('/quicksort')
def quicksort():
    return render_template('quicksort.html')
@app.route('/mergesort')
def mergesort():
    return render_template('mergesort.html')
@app.route('/heapsort')
def heapsort():
    return render_template('heapsort.html')
@app.route('/timsort')
def timsort():
    return render_template('timsort.html')
@app.route('/shellsort')
def shellsort():
    return render_template('shellsort.html')
@app.route('/binarysearch')
def binarysearch():
    return render_template('binarysearch.html')
@app.route('/linearsearch')
def linearsearch():
    return render_template('linearsearch.html')
@app.route('/terms')
def terms():
    return render_template('terms.html')
@app.route('/aboutus')
def aboutus():
    return render_template('aboutus.html')
if __name__ == '__main__':
    app.run(debug=True)
```

style.css file:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
    background-image: url('/static/images/back.jpeg');
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
    min-height: 100vh;
}
```

```
.container {
    width: 90%;
    max-width: 1200px;
    margin: auto;
    overflow: hidden;
    padding: 20px;
    background-color: #fff;
    box-shadow: 0 0 10px rgba(0,0,0,0.4);
    margin-top: 20px;
    margin-bottom: 20px;
}
h1, h2, h3 {
    color: #333;
}
p {
    color: #555;
}
button, .button {
    display: inline-block;
    padding: 10px 20px;
    background-color: #007BFF;
    color: #fff;
    border: none;
    cursor: pointer;
    text-decoration: none;
    border-radius: 4px;
    margin-top: 10px;
}
button:hover, .button:hover {
    background-color: #0056b3;
}
.form-group {
    margin-bottom: 15px;
}
.form-group label {
    display: block;
    color: #333;
    margin-bottom: 5px;
}
.form-group input[type="text"],
.form-group input[type="email"],
.form-group input[type="password"],
.form-group select,
.form-group textarea {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
}
textarea {
    resize: vertical;
}
.alert {
    padding: 15px;
    margin-bottom: 20px;
    border-radius: 4px;
}
.alert.success {
    background-color: #d4edda;
    color: #155724;
}
.alert.danger {
    background-color: #f8d7da;
    color: #721c24;
}
.alert.info {
    background-color: #d1ecf1;
```

```
    color: #0c5460;
}
header {
    width: 100%;
    background-color: #002472;
    padding: 20px 0;
    color: #ffffff;
    position: sticky;
    top: 0;
    z-index: 1000;
}
.header-container {
    display: flex;
    align-items: center;
    justify-content: space-between;
    width: 90%;
    max-width: 1200px;
    margin: auto;
}
.logo-image {
    height: 80px;
    margin-right: 10px;
}
header h1 {
    margin: 0;
    font-size: 24px;
    color: #ffffff;
}
nav {
    display: flex;
    align-items: center;
}
nav a {
    color: #ffffff;
    text-decoration: none;
    margin-left: 20px;
    font-size: 18px;
    transition: color 0.3s;
    display: flex;
    align-items: center;
}
nav a:hover {
    color: #FFD700;
}
.profile-section {
    display: flex;
    align-items: center;
    margin-left: 20px;
}
.profile-image {
    height: 30px;
    width: 30px;
    border-radius: 50%;
    margin-right: 5px;
}
.flash-messages {
    width: 90%;
    max-width: 1200px;
    margin: 20px auto;
}
.flash-messages .alert {
    border-radius: 4px;
}
.main-content {
    min-height: calc(100vh - 120px);
    padding: 20px 0;
}
```

```
.container {
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 80%;
    max-width: 1200px;
    padding: 20px;
    box-sizing: border-box;
    border-radius: 10px;
    background-color: #ffffff;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    margin: 0 auto;
}
.controls {
    margin-bottom: 20px;
    text-align: center;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
}
.controls button, .controls select, .controls input[type="text"] {
    margin: 5px;
    padding: 10px;
    font-size: 16px;
    border: 1px solid #000000;
    border-radius: 5px;
    background-color: #002472;
    color: #ffffff;
    cursor: pointer;
    transition: background-color 0.3s, color 0.3s, border-color 0.3s;
}
.controls label {
    margin: 5px;
    padding: 15px;
    font-size: 18px;
    border: 1px solid #000000;
    border-radius: 5px;
    background-color: #002472;
    color: #ffffff;
    display: inline-block;
    transition: background-color 0.3s, color 0.3s, border-color 0.3s;
    text-align: center;
}
.controls button:hover, .controls select:hover, .controls input[type="text"]:hover, .controls label:hover {
    background-color: #006ee4;
    color: #ffffff;
}
.controls input[type="text"] {
    border-color: #ced4da;
    background-color: #ffffff;
    color: #000000;
}
.controls input[type="text"]:focus {
    outline: none;
    border-color: #007BFF;
}
.array-container {
    display: flex;
    align-items: flex-end;
    justify-content: center;
    height: 80%;
    width: 100%;
    margin: 10px 0;
    border: 4px solid #000000;
    position: relative;
    background-color: #ffffff;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
```

```
    overflow: hidden;
}
.bar {
    background-color: var(--default-color, #007BFF);
    transition: background-color 0.3s, height 0.3s;
    margin: 0 2px;
    text-align: center;
    color: #ffffff;
    font-size: 12px;
    min-height: 20px;
    max-height: 100%;
    transition: height 0.3s, background-color 0.3s;
    flex-grow: 1;
    min-width: 20px;
}
.bar:hover {
    background-color: #0056b3;
}
.code-container {
    width: 98%;
    margin-top: 30px;
    padding: 20px;
    background-color: #f1f1f1;
    border-left: 6px solid #007BFF;
    border-radius: 5px;
    overflow: auto;
}
.code-container h2 {
    margin-bottom: 10px;
    color: #333333;
}
#algorithmCode, #codeDisplay {
    background-color: #ffff00;
    color: #000000;
    padding: 15px;
    border-radius: 5px;
    font-family: 'Courier New', Courier, monospace;
    font-size: 14px;
    border: 4px solid #000000;
    font-weight: bold;
    white-space: pre-wrap;
    word-wrap: break-word;
    max-height: 400px;
    overflow: auto;
}
@media (max-width: 768px) {
    .controls {
        flex-direction: column;
        align-items: stretch;
    }
    nav a {
        margin-left: 10px;
        font-size: 16px;
    }
    .array-container {
        height: 200px;
    }
    #algorithmCode, #codeDisplay {
        font-size: 12px;
    }
}
.main-content .features {
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
}
.main-content .feature {
```

```
    flex: 1;
    min-width: 300px;
    background-color: #f9f9f9;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.05);
}
.main-content .feature h3 {
    margin-top: 0;
}
.main-content .feature p {
    color: #555;
}
.feedback-list {
    list-style-type: disc;
    padding-left: 20px;
}
.form-group input[type="text"],
.form-group input[type="email"],
.form-group input[type="password"],
.form-group select,
.form-group textarea {
    width: calc(100% - 20px);
    padding: 10px;
    margin: 5px 0;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}
.learn-algorithms {
    margin-top: 40px;
}
.learn-algorithms h2 {
    text-align: center;
    color: #333333;
    margin-bottom: 20px;
}
.algorithm-images {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 20px;
    margin-bottom: 40px;
    margin-right:     20px;
}
.algorithm-images img {
    width: 200px;
    height: auto;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.2);
    transition: transform 0.3s;
}
.algorithm-images img:hover {
    transform: scale(1.05);
}
.footer {
    background-color: #002472;
    color: #ffffff;
    padding: 20px 0;
    text-align: center;
    position: relative;
    bottom: 0;
    width: 100%;
}
.footer p {
    margin: 0;
}
```

sorting.js file:

```
let array = [];
let sorting = false;
function generateArray() {
    array = [];
    for (let i = 0; i < 20; i++) {
        array.push(Math.floor(Math.random() * 100) + 1);
    }
    displayArray();
}
function takeUserInput() {
    const userInput = document.getElementById("userInput").value;
    if (userInput) {
        array = userInput.split(",").map(num => parseInt(num.trim(), 10));
        displayArray();
    } else {
        alert("Please enter numbers separated by commas.");
    }
}
function displayArray() {
    const arrayContainer = document.getElementById("arrayContainer");
    arrayContainer.innerHTML = '';
    array.forEach((num) => {
        const bar = document.createElement("div");
        bar.classList.add("bar");
        bar.style.height = `${num * 4}px`;
        bar.textContent = num;
        arrayContainer.appendChild(bar);
    });
}
function sleep(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}
function displayAlgorithmCode() {
    const algorithm = document.getElementById("sortSelect").value;
    const codeContainer = document.getElementById("algorithmCode");
    if (codeSnippets[algorithm]) {
        codeContainer.textContent = codeSnippets[algorithm];
    } else {
        codeContainer.textContent = "// Algorithm code not available.";
    }
    hljs.highlightElement(codeContainer);
}
async function startSort() {
    if (sorting) return;
    sorting = true;
    displayAlgorithmCode();
    const algorithm = document.getElementById("sortSelect").value;
    const ascending = document.getElementById("orderSelect").value === 'ascending';
    const speed = document.getElementById("speedSelect").value;
    let speedFactor = 150;
    if (speed === 'fast') speedFactor = 100;
    else if (speed === 'medium') speedFactor = 250;
    else if (speed === 'slow') speedFactor = 400;
    const response = await fetch('/sort', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ array, algorithm, ascending })
    });
    const data = await response.json();
    if (data.error) {
        alert(data.error);
        sorting = false;
```

```
        return;
    }
    for (const step of data.steps) {
        array = step[0];
        displayArray();
        if (step[1] && step[1].length > 0) {
            const bars = document.getElementsByClassName("bar");
            step[1].forEach(idx => {
                if (bars[idx]) {
                    bars[idx].style.backgroundColor = 'red';
                }
            });
            await sleep(speedFactor);
            step[1].forEach(idx => {
                if (bars[idx]) {
                    bars[idx].style.backgroundColor = '#007BFF';
                }
            });
        } else {
            await sleep(speedFactor);
        }
    }
    sorting = false;
}
generateArray();
displayAlgorithmCode();
```

searching.js file:

```
const DEFAULT_COLOR = '#007BFF';
const COMPARE_COLOR = '#FF5733';
const FOUND_COLOR = '#28A745';
const NOT_FOUND_COLOR = '#DC3545';
let array = [];
let searching = false;
const linearSearchCode = `function linearSearch(arr, target) {
    for (let i = 0; i < arr.length; i++) {
        if (arr[i] === target) {
            return i;
        }
    }
    return -1;
}`;
const binarySearchCode = `function binarySearch(arr, target) {
    let left = 0;
    let right = arr.length - 1;
    while (left <= right) {
        let mid = Math.floor((left + right) / 2);
        if (arr[mid] === target) return mid;
        else if (arr[mid] < target) left = mid + 1;
        else right = mid - 1;
    }
    return -1;
}`;
function generateRandomArray() {
    array = [];
    for (let i = 0; i < 20; i++) {
        array.push(Math.floor(Math.random() * 100) + 1);
    }
    displayArray();
    displayCode(document.getElementById("searchSelect").value);
}
function generateSortedRandomArray() {
    array = [];
    for (let i = 0; i < 20; i++) {
        array.push(Math.floor(Math.random() * 100) + 1);
```

```
    }
    array.sort((a, b) => a - b);
    displayArray();
    displayCode(document.getElementById("searchSelect").value);
}
function takeUserInput() {
    const userInput = document.getElementById("userArrayInput").value;
    if (userInput) {
        array = userInput.split(",").map(num => parseInt(num.trim(), 10)).filter(num => !isNaN(num));
        displayArray();
        displayCode(document.getElementById("searchSelect").value);
    }
}
function displayArray(highlightIndices = [], color = COMPARE_COLOR) {
    const arrayContainer = document.getElementById("arrayContainer");
    arrayContainer.innerHTML = '';
    array.forEach((num, idx) => {
        const bar = document.createElement("div");
        bar.classList.add("bar");
        bar.style.height = `${num * 4}px`;
        bar.textContent = num;
        if (highlightIndices.includes(idx)) {
            bar.style.backgroundColor = color;
        } else {
            bar.style.backgroundColor = DEFAULT_COLOR;
        }
        arrayContainer.appendChild(bar);
    });
}
function displayCode(algorithm) {
    const codeDisplay = document.getElementById("codeDisplay");
    if (algorithm === 'linear') {
        codeDisplay.textContent = linearSearchCode;
    } else if (algorithm === 'binary') {
        codeDisplay.textContent = binarySearchCode;
    } else {
        codeDisplay.textContent = "// Code not available for the selected algorithm.";
    }
    hljs.highlightElement(codeDisplay);
}
function sleep(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}
async function startSearch() {
    if (searching) return;
    searching = true;
    const algorithm = document.getElementById("searchSelect").value;
    const speed = document.getElementById("speedSelect").value;
    let speedFactor = 250;
    if (speed === 'fast') speedFactor = 100;
    else if (speed === 'slow') speedFactor = 400;
    const targetInput = document.getElementById("searchInput").value;
    if (targetInput === '') {
        alert("Please enter a number to search.");
        searching = false;
        return;
    }
    let target = parseInt(targetInput, 10);
    if (isNaN(target)) {
        alert("Please enter a valid number.");
        searching = false;
        return;
    }
    if (algorithm === 'binary') {
        if (!isSorted(array)) {
            alert("The array must be sorted to execute binary search.");
            searching = false;
```

```
            return;
        }
    }
    const response = await mockSearch(array, target, algorithm);
    if (response.error) {
        alert(response.error);
        searching = false;
        return;
    }
    for (let i = 0; i < response.steps.length; i++) {
        const step = response.steps[i];
        array = step.array;
        displayArray(step.indices, COMPARE_COLOR);
        await sleep(speedFactor);
        if (response.found && i === response.steps.length - 1) {
            displayArray([response.index], FOUND_COLOR);
            await sleep(speedFactor);
        }
    }
    if (!response.found) {
        await sleep(speedFactor);
        alert("Element not found in the array.");
    }
    searching = false;
}
function isSorted(arr) {
    return arr.every((val, i, array) => !i || array[i - 1] <= val) ||
           arr.every((val, i, array) => !i || array[i - 1] >= val);
}
async function mockSearch(array, target, algorithm) {
    const steps = [];
    let foundIndex = -1;
    if (algorithm === 'linear') {
        for (let i = 0; i < array.length; i++) {
            steps.push({ array: [...array], indices: [i] });
            if (array[i] === target) {
                foundIndex = i;
                break;
            }
            await sleep(50);
        }
    } else if (algorithm === 'binary') {
        let left = 0;
        let right = array.length - 1;
        while (left <= right) {
            let mid = Math.floor((left + right) / 2);
            steps.push({ array: [...array], indices: [mid] });
            if (array[mid] === target) {
                foundIndex = mid;
                break;
            } else if (array[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
            await sleep(50);
        }
    }
    if (foundIndex === -1) {
        return { steps, found: false };
    } else {
        return { steps, found: true, index: foundIndex };
    }
}
function generateArray() {
    const algorithm = document.getElementById("searchSelect").value;
    if (algorithm === 'binary') {
```

```
    generateSortedRandomArray();
  } else {
    generateRandomArray();
  }
}
document.getElementById("searchSelect").addEventListener("change", function() {
  const selectedAlgorithm = this.value;
  displayCode(selectedAlgorithm);
  generateArray();
});
generateArray();
```

base.html file:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>SSH (SORT AND SEARCH HUB) - {% block title %}{% endblock %}</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/highlight.js/11.7.0/styles/default.min.css">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/highlight.js/11.7.0/highlight.min.js"></script>
  <script>hljs.highlightAll();</script>
</head>
<body>
  {% include 'header.html' %}
  <div class="main-content">
    {% with messages = get_flashed_messages(with_categories=true) %}
      {% if messages %}
        <div class="flash-messages">
          {% for category, message in messages %}
            <div class="alert {{ category }}">{{ message }}</div>
          {% endfor %}
        </div>
      {% endif %}
    {% endwith %}
    {% block content %}{% endblock %}
  </div>
  {% include 'footer.html' %}
</body>
</html>
```

feedback.html file:

```
{% extends "base.html" %}
{% block title %}Feedback{% endblock %}
{% block content %}
<div class="container">
  <h2>Feedback</h2>
  <form method="POST" action="{{ url_for('feedback') }}">
    <div class="form-group">
      <label for="content">Your Feedback:</label>
      <textarea id="content" name="content" rows="4" required></textarea>
    </div>
    <button type="submit" class="button">Submit Feedback</button>
  </form>
  <h3>Your Feedbacks:</h3>
  {% if feedbacks %}
    <ul class="feedback-list">
      {% for fb in feedbacks %}
        <li>{{ fb.content }}</li>
      {% endfor %}
    </ul>
  {% else %}
```

```
    <p>You haven't submitted any feedback yet.</p>
  {% endif %}
</div>
{% endblock %}
```

header.html file:

```
<header>
  <div class="header-container">
    <div class="logo">
      <img src="{{ url_for('static', filename='images/website_logo.png') }}" alt="Website Logo" class="logo-image">
    </div>
    <h1>SSH (Sort and Search Hub)</h1>
    <nav>
      <a href="{{ url_for('index') }}">Home</a>
      <a href="{{ url_for('sorting_page') }}">Sorting Visualizer</a>
      <a href="{{ url_for('searching_page') }}">Searching Visualizer</a>
      {% if current_user.is_authenticated %}
        <a href="{{ url_for('feedback') }}">Feedback</a>
        <div class="profile-section">
          <a href="{{ url_for('profile') }}">
            <img src="{{ url_for('static', filename='images/my_profile.png') }}" alt="My Profile" class="profile-image">
            My Profile
          </a>
        </div>
        <a href="{{ url_for('logout') }}">Logout</a>
      {% else %}
        <a href="{{ url_for('login') }}">Login</a>
        <a href="{{ url_for('register') }}">Register</a>
      {% endif %}
    </nav>
  </div>
</header>
```

index.html file:

```
{% extends "base.html" %}
{% block title %}Home{% endblock %}
{% block content %}
<div class="container">
  <h2>Welcome to SSH (Sort and Search Hub)</h2>
  <p>
    SSH is an interactive platform designed to help you understand and visualize various sorting and searching algorithms.
    Whether you're a student trying to grasp algorithmic concepts or a professional looking to refresh your knowledge,
    SSH provides an intuitive interface to explore and analyze the inner workings of these fundamental algorithms.
  </p>
  <div class="features">
    <div class="feature">
      <h3>Sorting Visualizer</h3>
      <p>
        Visualize different sorting algorithms in action. See how Bubble Sort, Quick Sort, Merge Sort, and others operate on your data.
        Customize the dataset and control the visualization speed to enhance your learning experience.
      </p>
      <a href="{{ url_for('sorting_page') }}" class="button">Explore Sorting Visualizer</a>
    </div>
    <div class="feature">
      <h3>Searching Visualizer</h3>
      <p>
        Understand how searching algorithms like Binary Search and Linear Search traverse data structures to find your target element.
        Watch step-by-step animations and compare the efficiency of each method.
      </p>
      <a href="{{ url_for('searching_page') }}" class="button">Explore Searching Visualizer</a>
    </div>
  </div>  <div class="learn-algorithms">
    <h2>Learn About Different Sorting and Searching Algorithms</h2>
```

```
    <div class="algorithm-images">
      <div class="algorithm">
        <a href="{{ url_for('bubblesort') }}">
          <img src="{{ url_for('static', filename='images/bubble_sort.png') }}" alt="Bubble Sort">
          <p>Bubble Sort</p>
        </a>
      </div>
      <div class="algorithm">
        <a href="{{ url_for('selectionsort') }}">
          <img src="{{ url_for('static', filename='images/selection_sort.png') }}" alt="Selection Sort">
          <p>Selection Sort</p>
        </a>
      </div>
      <div class="algorithm">
        <a href="{{ url_for('insertionsort') }}">
          <img src="{{ url_for('static', filename='images/insertion_sort.png') }}" alt="Insertion Sort">
          <p>Insertion Sort</p>
        </a>
      </div>
      <div class="algorithm">
        <a href="{{ url_for('quicksort') }}">
          <img src="{{ url_for('static', filename='images/quick_sort.png') }}" alt="Quick Sort">
          <p>Quick Sort</p>
        </a>
      </div>
      <div class="algorithm">
        <a href="{{ url_for('mergesort') }}">
          <img src="{{ url_for('static', filename='images/merge_sort.png') }}" alt="Merge Sort">
          <p>Merge Sort</p>
        </a>
      </div>
    </div>
    <div class="algorithm-images">
      <div class="algorithm">
        <a href="{{ url_for('heapsort') }}">
          <img src="{{ url_for('static', filename='images/heap_sort.png') }}" alt="Heap Sort">
          <p>Heap Sort</p>
        </a>
      </div>
      <div class="algorithm">
        <a href="{{ url_for('timsort') }}">
          <img src="{{ url_for('static', filename='images/tim_sort.png') }}" alt="Tim Sort">
          <p>Tim Sort</p>
        </a>
      </div>
      <div class="algorithm">
        <a href="{{ url_for('shellsort') }}">
          <img src="{{ url_for('static', filename='images/shell_sort.png') }}" alt="Shell Sort">
          <p>Shell Sort</p>
        </a>
      </div>
      <div class="algorithm">
        <a href="{{ url_for('binarysearch') }}">
          <img src="{{ url_for('static', filename='images/binary_search.png') }}" alt="Binary Search">
          <p>Binary Search</p>
        </a>
      </div>
      <div class="algorithm">
        <a href="{{ url_for('linearsearch') }}">
          <img src="{{ url_for('static', filename='images/linear_search.png') }}" alt="Linear Search">
          <p>Linear Search</p>
        </a>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

support.html file:

```
{% extends "base.html" %}
{% block title %}Support Portal{% endblock %}
{% block content %}
<body>
  <div class="container">
    <h2>Support Portal</h2>
    <!-- Display Flash Messages -->
    {% with messages = get_flashed_messages(with_categories=true) %}
      {% if messages %}
        {% for category, message in messages %}
          <div class="alert alert-{{ category }}">{{ message }}</div>
        {% endfor %}
      {% endif %}
    {% endwith %}
    <div class="contact-section">
      <div class="contact-card">
        <h2>Contact Information</h2>
        <div class="contact-info">
          <div>
            <b>Address:</b> 3, Institutional Area, Sector-5, Rohini (Near Rithala Metro Station), Delhi-110085
          </div>
          <div>
            <b>Phone Number:</b> (999)-089-504
          </div>
          <div>
            <b>Email:</b> contactthessh@gmail.com
          </div>
        </div>
      </div>
      <div class="contact-card">
        <h2>Send Your Query</h2>
        <div class="form-container">
          <form method="POST" action="{{ url_for('support') }}">
            <label for="name">Your Name</label>
            <input type="text" id="name" name="name" placeholder="Enter your name" required>
            <label for="email">Email</label>
            <input type="email" id="email" name="email" placeholder="Enter your email address" required>
            <label for="phone">Phone Number</label>
            <input type="text" id="phone" name="phone" placeholder="Enter your phone number" required>
            <label for="message">Message</label>
            <textarea id="message" name="message" rows="4" placeholder="How can we help you?" required></textarea>
            <button type="submit">Send</button>
          </form>
        </div>
      </div>
    </div>
  </div>
</body>
{% endblock %}
```

sorting.html file:

```
{% extends "base.html" %}
{% block title %}Sorting Visualizer{% endblock %}
{% block content %}
  <div class="container">
    <div class="controls">
      <button onclick="generateArray()">Generate Random Array</button>
      {% if current_user.is_authenticated %}
        <button onclick="takeUserInput()">Take User Input</button>
        <input type="text" id="userInput" placeholder="Enter comma separated no.">
      {% else %}
        <button onclick="alert('Please log in to use custom array input.')">Take User Input</button>
```

```
            <input type="text" id="userInput" placeholder="Login to enter custom array" disabled>
        {% endif %}
        <label for="sortSelect">Select Appropriate Sorting Algorithm:</label>
        <select id="sortSelect">
            <option value="bubble">Bubble Sort</option>
            <option value="selection">Selection Sort</option>
            <option value="insertion">Insertion Sort</option>
            <option value="quick">Quick Sort</option>
            <option value="merge">Merge Sort</option>
            <option value="heap">Heap Sort</option>
            <option value="tim">Tim Sort</option>
            <option value="shell">Shell Sort</option>
        </select>
        <label for="orderSelect">Sorting Order:</label>
        <select id="orderSelect">
            <option value="ascending">Ascending</option>
            <option value="descending">Descending</option>
        </select>
        <label for="speedSelect">Speed:</label>
        <select id="speedSelect">
            <option value="fast">Fast</option>
            <option value="medium">Medium</option>
            <option value="slow">Slow</option>
        </select>
        <button onclick="startSort()">Start Sorting</button>
    </div>
    <div id="arrayContainer" class="array-container"></div>
    <div id="codeContainer" class="code-container">
        <h2>Algorithm Code</h2>
        <pre id="algorithmCode" class="language-python">// Select a sorting algorithm to view its code here.</pre>
    </div>
</div>
<script src="{{ url_for('static', filename='sorting.js') }}"></script>
<script src="{{ url_for('static', filename='codeSortingSnippets.js') }}"></script>
{% endblock %}
```

searching.html file:

```
{% extends "base.html" %}
{% block title %}Searching Visualizer{% endblock %}
{% block content %}
<div class="container">
    <div class="controls">
        <button onclick="generateArray()">Generate Random Array</button>
        {% if current_user.is_authenticated %}
            <button onclick="takeUserInput()">Take User Input</button>
            <input type="text" id="userArrayInput" placeholder="Enter comma separated no.">
        {% else %}
            <button onclick="alert('Please log in to use custom array input.')">Take User Input</button>
            <input type="text" id="userArrayInput" placeholder="Login to enter custom array" disabled>
        {% endif %}
        <label for="searchSelect">Select Searching Algorithm:</label>
        <select id="searchSelect">
            <option value="linear">Linear Search</option>
            <option value="binary">Binary Search</option>
        </select>
        <label for="searchInput">Value to Search:</label>
        <input type="text" id="searchInput" placeholder="Enter value to search">
        <label for="speedSelect">Speed:</label>
        <select id="speedSelect">
            <option value="slow">Slow</option>
            <option value="medium">Medium</option>
            <option value="fast">Fast</option>
        </select>
        <button onclick="startSearch()">Start Searching</button>
    </div>
```

```
    <div id="arrayContainer" class="array-container"></div>
    <div id="codeContainer" class="code-container">
      <h2>Algorithm Code</h2>
      <pre><code id="codeDisplay" class="language-javascript">// Algorithm code will appear here</code></pre>
    </div>
  </div>
  <script src="{{ url_for('static', filename='searching.js') }}"></script>
{% endblock %}
```

footer.html file:

```
<footer>
  <div class="footer-content">
    <div class="footer-section footer-left">
      <div class="footer-logo">
        <img src="{{ url_for('static', filename='images/website_logo.png') }}" alt="WEBSITE LOGO">
      </div>
    </div>
    <div class="footer-section footer-center">
      <div class="footer-links">
        <h2>Quick Links</h2>
        <a href="http://127.0.0.1:5000/">Home</a>
        <a href="{{ url_for('aboutus') }}">About Us</a>
        <a href="{{ url_for('support') }}">Support</a>
        <a href="{{ url_for('terms') }}">Terms and Privacy Policy</a>
        <a href="{{ url_for('sorting_page') }}">Sorting Visualizer</a>
        <a href="{{ url_for('searching_page') }}">Searching Visualizer</a>
      </div>
    </div>
    <div class="footer-section footer-right">
      <div class="contact-us">
        <h2>Contact Us</h2>
        <p>Address: 3, Institutional Area, Sector-5, Rohini (Near Rithala Metro Station), Delhi-110085</p>
        <p>Phone Number: (999)-089-504</p>
        <p>Email: contactthessh@gmail.com</p>
        <p>Hours: Available 24/7</p>
      </div>
      <div class="follow-us">
        <h2>Follow Us</h2>
        <div class="social-icons">
          <a href="https://www.facebook.com/"><img src="{{ url_for('static', filename='images/fb_logo.png') }}" alt="Facebook" /></a>
          <a href="https://twitter.com/?lang=en"><img src="{{ url_for('static', filename='images/x_logo.png') }}" alt="X aka Twitter" /></a>
          <a href="https://www.youtube.com/"><img src="{{ url_for('static', filename='images/yt_logo.png') }}" alt="YouTube" /></a>
          <a href="https://www.instagram.com/"><img src="{{ url_for('static', filename='images/insta_logo.png') }}" alt="Instagram" /></a>
          <a href="https://in.linkedin.com/"><img src="{{ url_for('static', filename='images/linkedin_logo.png') }}" alt="LinkedIn" /></a>
        </div>
      </div>
    </div>
  </div>
  <div class="footer-container">
    <p>&copy; 2024 SSH (Sort and Search Hub). All rights reserved.</p>
  </div>
</footer>
```

6.2 TESTING

6.2.1 OVERVIEW & APPROACH

Testing is a crucial phase in the SSH project aimed at identifying errors in the sorting and searching functionalities, ensuring that the system performs as expected. It involves executing various algorithms with the goal of uncovering issues and ensuring the reliability

of the visualizers. Testing will occur throughout all stages of development, starting from the initial specification of requirements. Test cases will be created to evaluate the algorithms and the user interfaces using normal input data. The results will provide insights into the software's quality and can lead to necessary modifications, ensuring accurate and efficient implementation before the system goes live.

6.2.2 Unit Testing

Unit testing in the SSH project focuses on verifying individual components of the sorting and searching algorithms. Each algorithm is tested in isolation to ensure its correctness and efficiency. This level of testing is crucial for identifying and resolving issues early in the development process.

6.2.3 Component Integration Testing

Integration testing involves combining various tested components of the SSH project, particularly the sorting and searching visualizers. The testing process will focus on:

- Verifying the interaction between the sorting and searching algorithms.
- Assessing the overall functionality of the user interfaces when different components are integrated.

6.2.4 System Testing

After integration testing, the entire SSH system will be tested as a cohesive unit. Test cases will be derived from functional specifications that define how users interact with the sorting and searching features. Key attributes evaluated will include:

- **Performance:** Assessing how quickly sorting and searching operations are performed.
- **Usability:** Ensuring the interface is intuitive and user-friendly.
- **Reliability:** Verifying that algorithms produce consistent results.
- **Security:** Ensuring user data is handled securely, particularly during authentication.

6.2.5 Acceptance Testing

After system testing was complete, the system was handed over to the training section. This phase signifies the transition of ownership from the development team to the users. Acceptance testing will be focused on:

- Verifying that the sorting and searching features meet user requirements and expectations.
- Ensuring that the user interface aligns with administrative procedures and provides a seamless experience.

6.2.6 Test Cases:

6.2.6.1 User Authentication Testing:

Test Case ID	Description	Expected Result	Actual Result	Remark
TC01	Valid employee ID and password entered	Login Successful	Login Successful	PASS
TC02	Invalid employee ID with correct password	Login Unsuccessful	Login Unsuccessful	PASS
TC03	Correct employee ID with incorrect password	Login Unsuccessful	Login Unsuccessful	PASS
TC04	Both fields empty	Popup message: All Fields are required!	Popup message: All Fields are required!	PASS
TC05	Successful password reset with valid employee ID	Start forget password	Start forget password	PASS

6.2.6.2 Sorting Visualizer Testing:

Test Case ID	Test Scenario	Input	Expected Outcome	Result
TC_SORT_01	Sort an array in ascending order	Array: [5, 3, 8, 1]	Sorted Array: [1, 3, 5, 8]	Pass
TC_SORT_02	Sort an array in descending order	Array: [5, 3, 8, 1]	Sorted Array: [8, 5, 3, 1]	Pass
TC_SORT_03	Attempt to sort an empty array	Array: []	Error message: "Array is empty"	Pass
TC_SORT_04	Generate a random array	N/A	Random Array generated	Pass
TC_SORT_05	Validate color grading during sort	Array: [5, 3, 8, 1]	Colors change appropriately during sort	Pass
TC_SORT_06	Sort using a custom array from user input	Array: [12, 4, 5, 9, 1]	Sorted Array: [1, 4, 5, 9, 12]	Pass
TC_SORT_07	Verify functionality for guest users	Array: [7, 3, 2]	Access granted, sorting works as expected	Pass
TC_SORT_08	Verify functionality for verified users	Array: [10, 5, 6]	Access granted, sorting works as expected	Pass

6.2.6.3 Searching Visualizer Testing:

Test Case ID	Test Scenario	Input	Expected Outcome	Result
TC_SEARCH_01	Perform linear search on sorted array	Array: [1, 3, 5, 8], Target: 5	Index: 2	Pass
TC_SEARCH_02	Perform binary search on sorted array	Array: [1, 3, 5, 8], Target: 3	Index: 1	Pass
TC_SEARCH_03	Search for non-existent value	Array: [1, 3, 5, 8], Target: 6	Error message: "Value not found"	Pass
TC_SEARCH_04	Attempt to search in an unsorted array	Array: [5, 3, 8, 1], Target: 5	Error message: "Array must be sorted"	Pass
TC_SEARCH_05	Validate color changes during search	Array: [1, 3, 5, 8], Target: 5	Colors change appropriately during search	Pass
TC_SEARCH_06	Verify functionality for guest users	Array: [1, 2, 3], Target: 2	Index: 1	Pass
TC_SEARCH_07	Verify functionality for verified users	Array: [1, 2, 3], Target: 1	Index: 0	Pass
TC_SEARCH_08	Search using a custom array from user input	Array: [7, 5, 1, 3], Target: 3	Index: 3	Pass

6.2.6.4 Support Portal Testing:

Test Case ID	Test Scenario	Input	Expected Outcome	Result
TC_SUPPORT_01	Access support portal	User logged in	Support portal loads successfully	Pass
TC_SUPPORT_02	Submit a support request	Request: "Issue with login"	Confirmation message sent	Pass
TC_SUPPORT_03	Submit support request with missing details	Request: ""	Error message: "Details required"	Pass
TC_SUPPORT_04	View support ticket status	Ticket ID: 123	Status displayed	Pass
TC_SUPPORT_05	Close support ticket	Ticket ID: 123	Confirmation message: "Ticket closed"	Pass

6.2.6.5 Feedback Portal Testing:

Test Case ID	Test Scenario	Input	Expected Outcome	Result
TC_FEEDBACK_01	Submit feedback	User ID: 1, Feedback: "Great app!"	Feedback submitted successfully	Pass
TC_FEEDBACK_02	Submit feedback with empty content	User ID: 1, Feedback: ""	Error message: "Feedback cannot be empty"	Pass
TC_FEEDBACK_03	View all feedback	User ID: 1	All feedback displayed	Pass
TC_FEEDBACK_04	Delete feedback	Feedback ID: 1	Feedback deleted successfully	Pass
TC_FEEDBACK_05	Edit feedback	Feedback ID: 1, New Feedback: "Awesome!"	Feedback updated successfully	Pass

CHAPTER 7: SCREENSHOTS

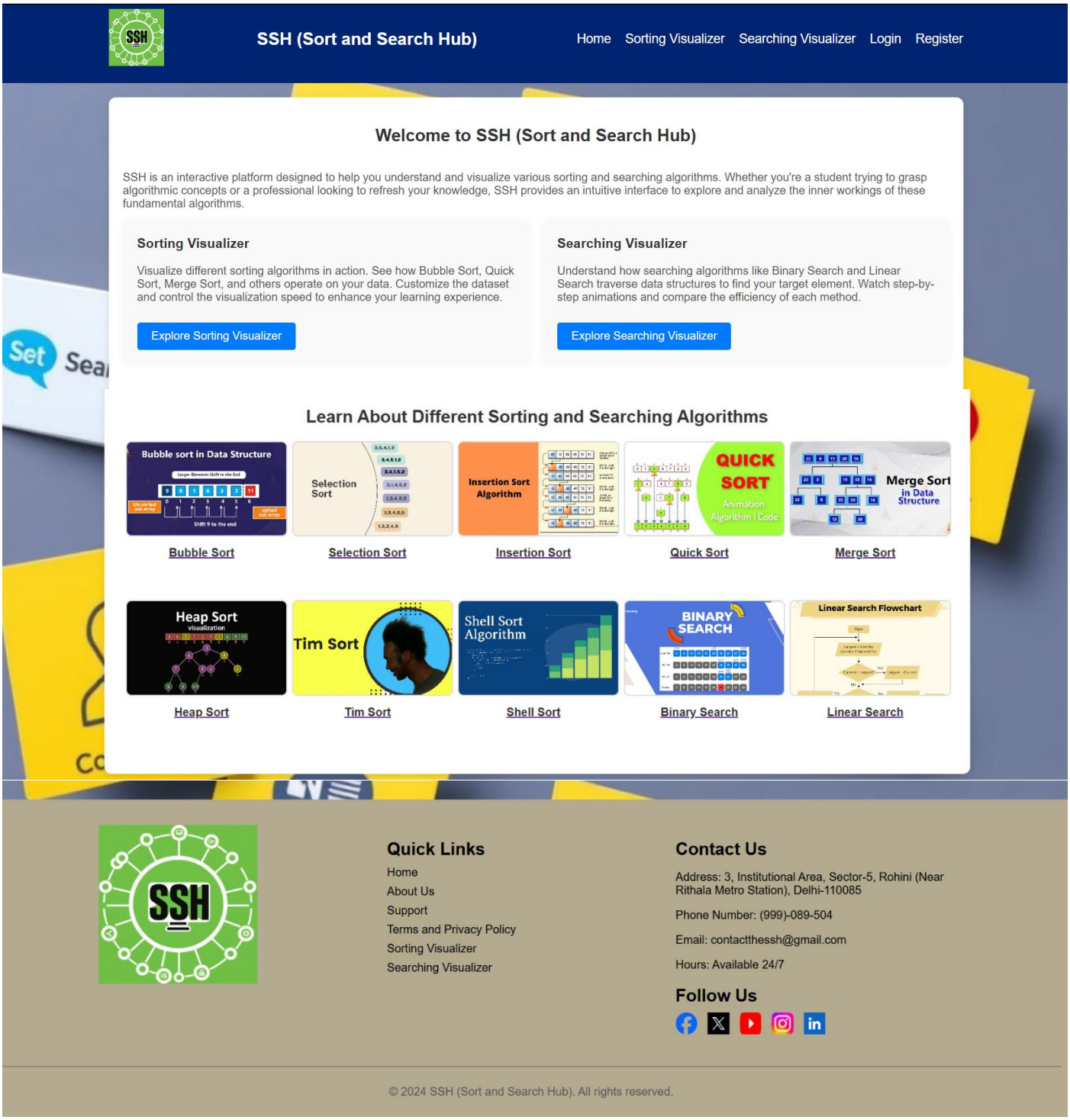


Figure 7.1 Home Page

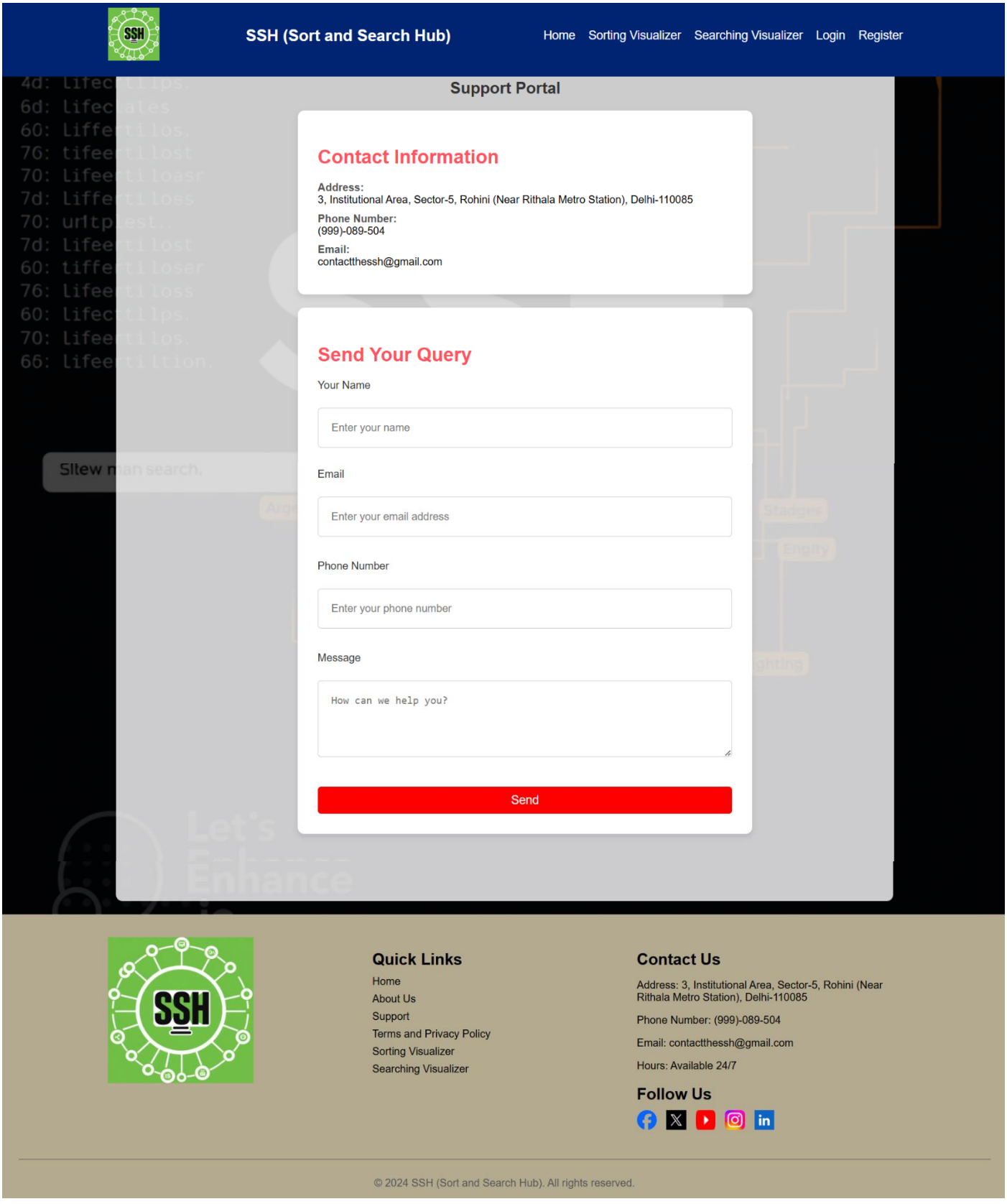


Figure 7.2 Support Portal

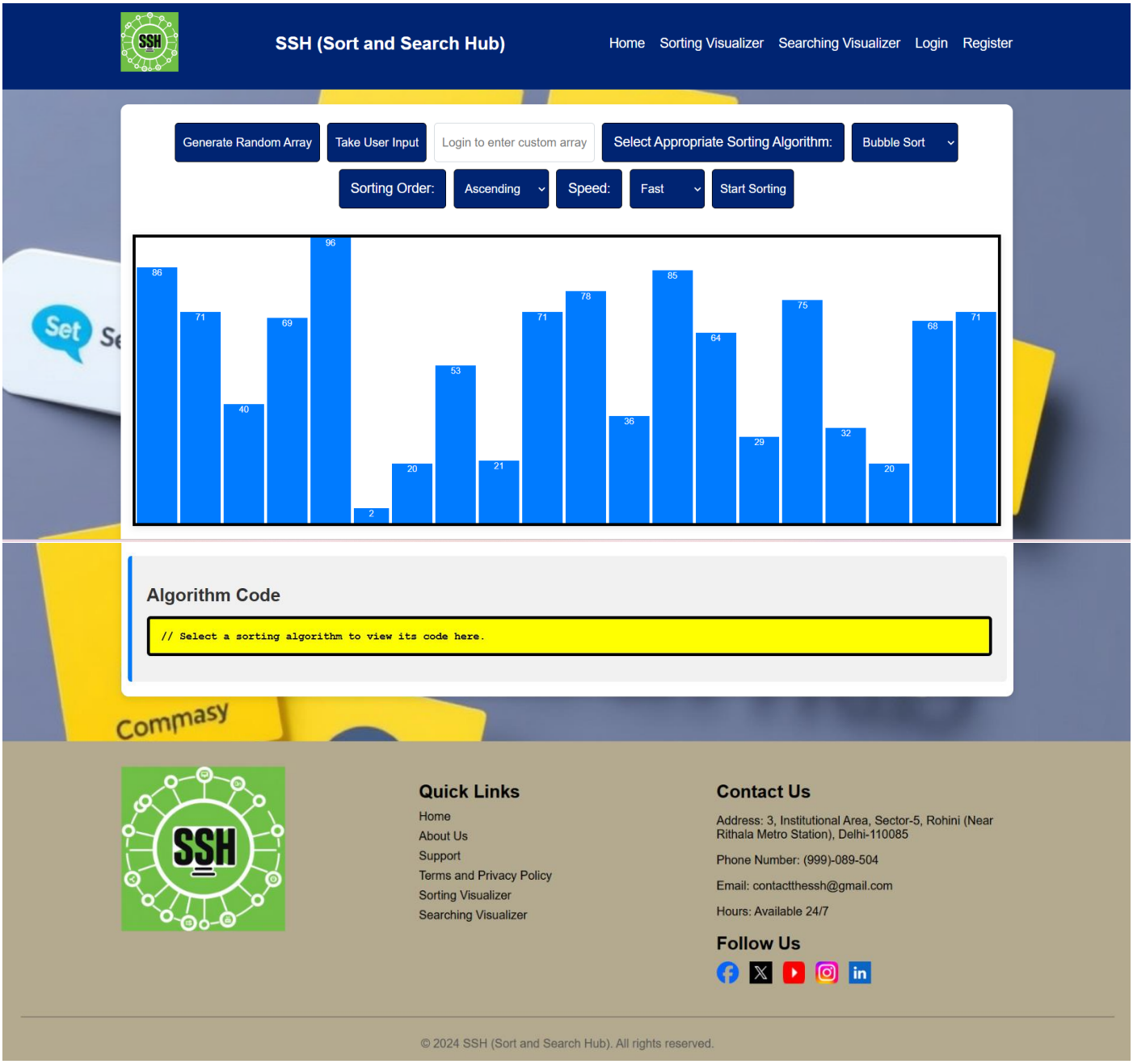


Figure 7.3 Sorting Visualizer

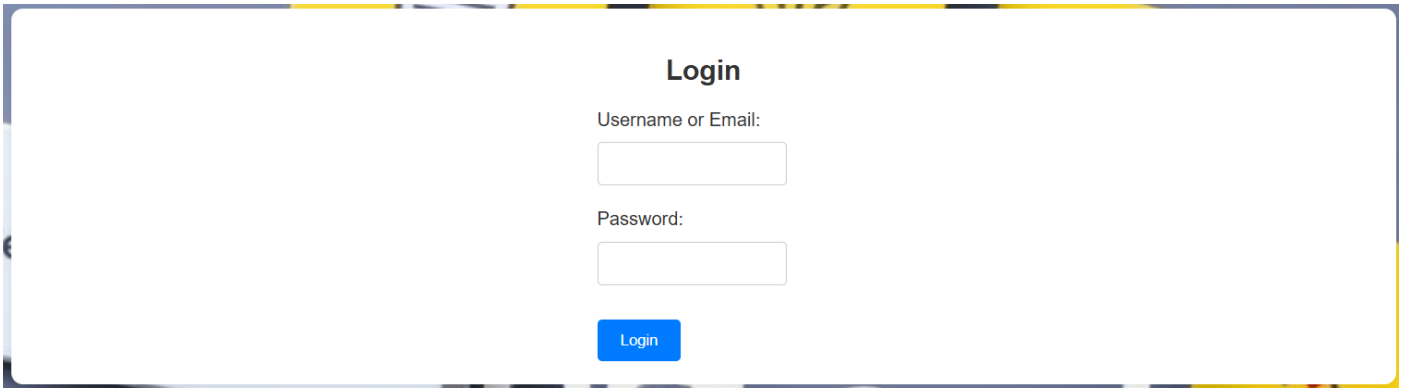


Figure 7.4 Login Form

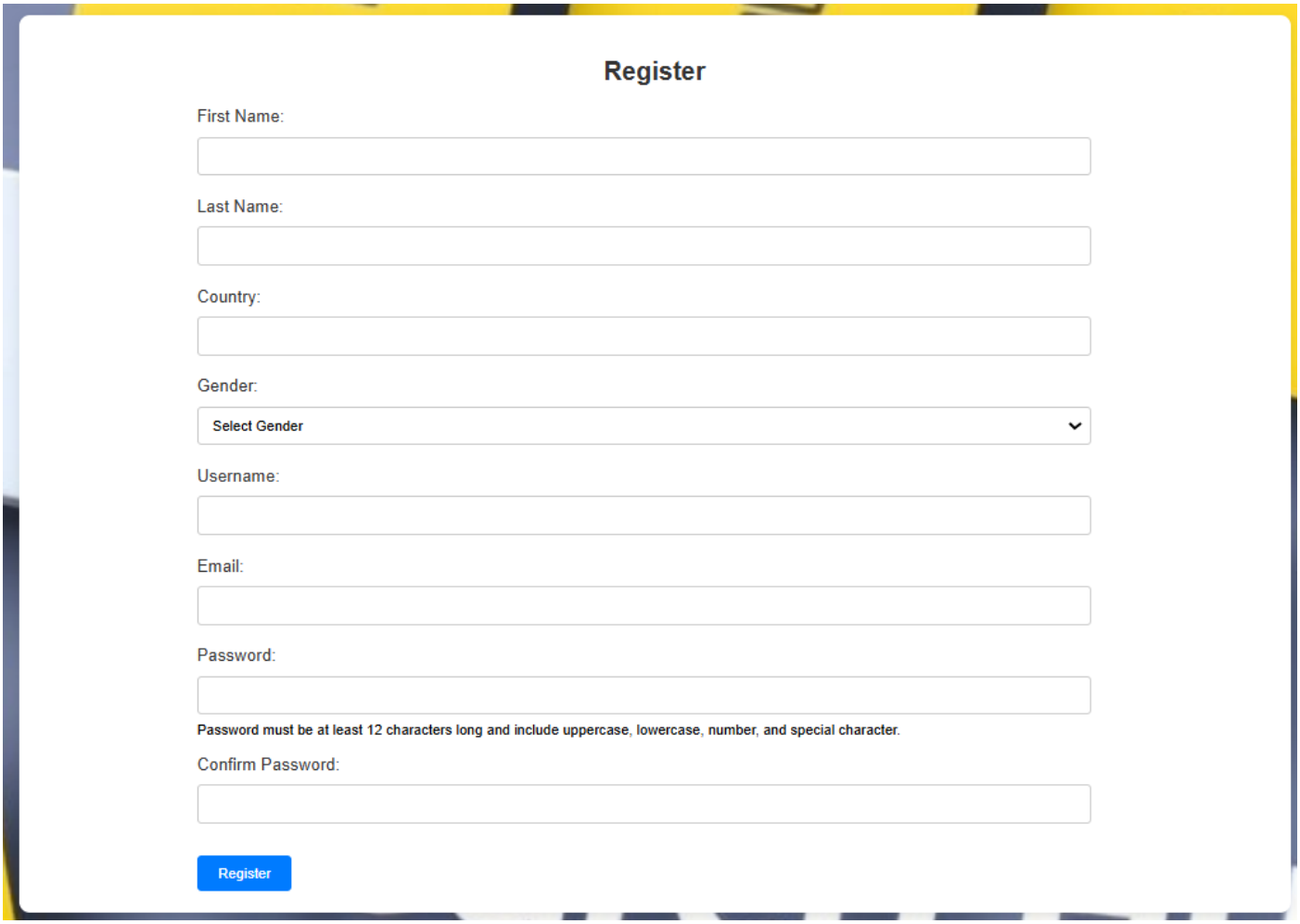


Figure 7.5 Registeration Form

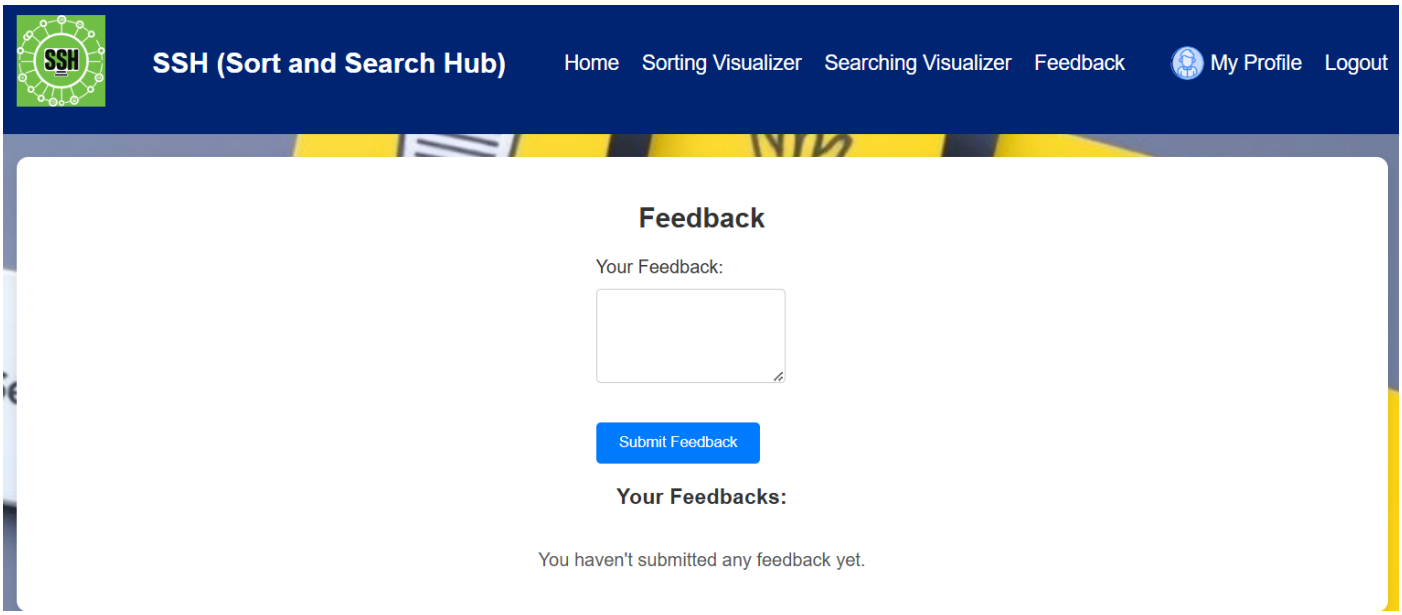


Figure 7.6 Feedback Form

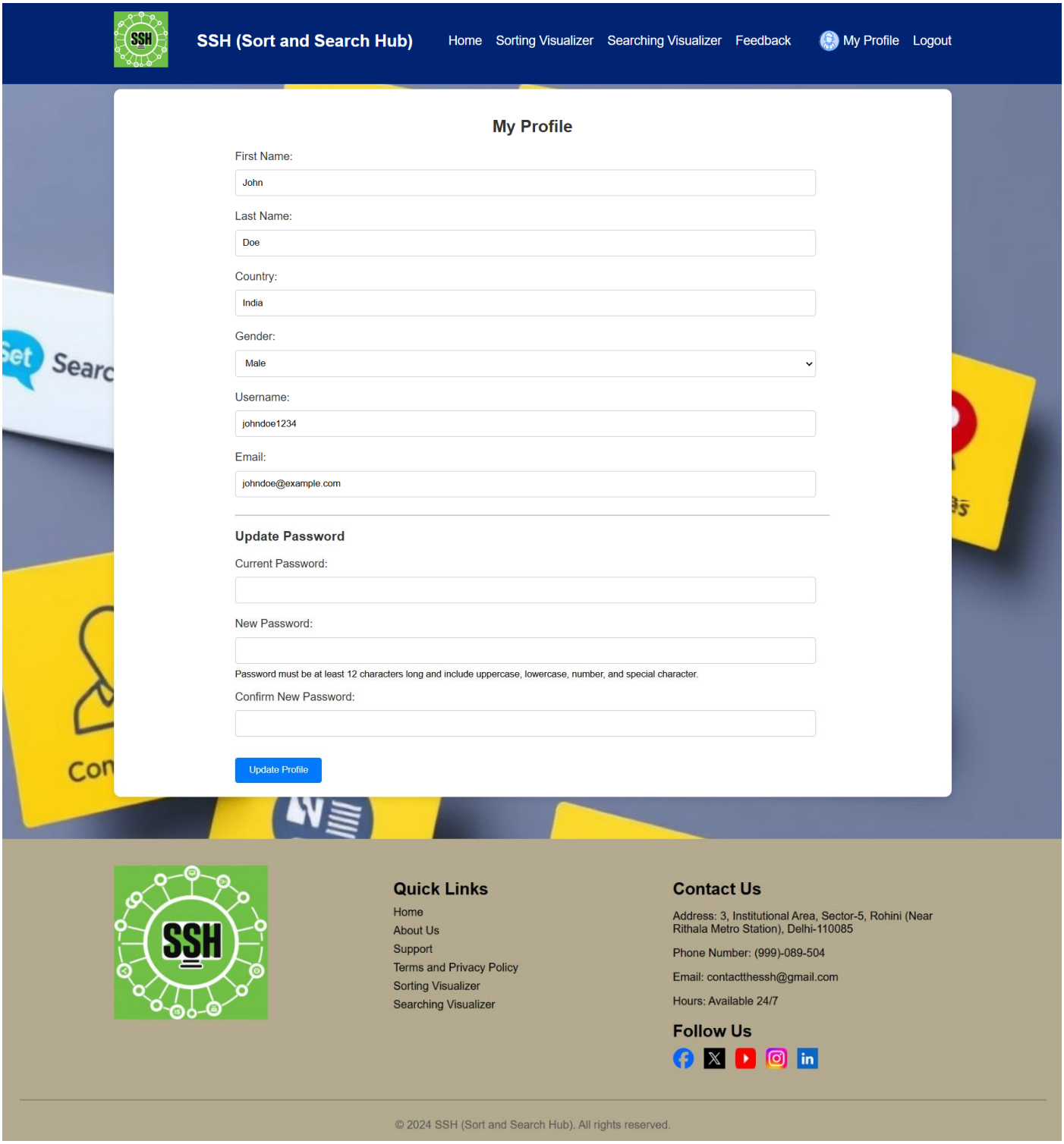
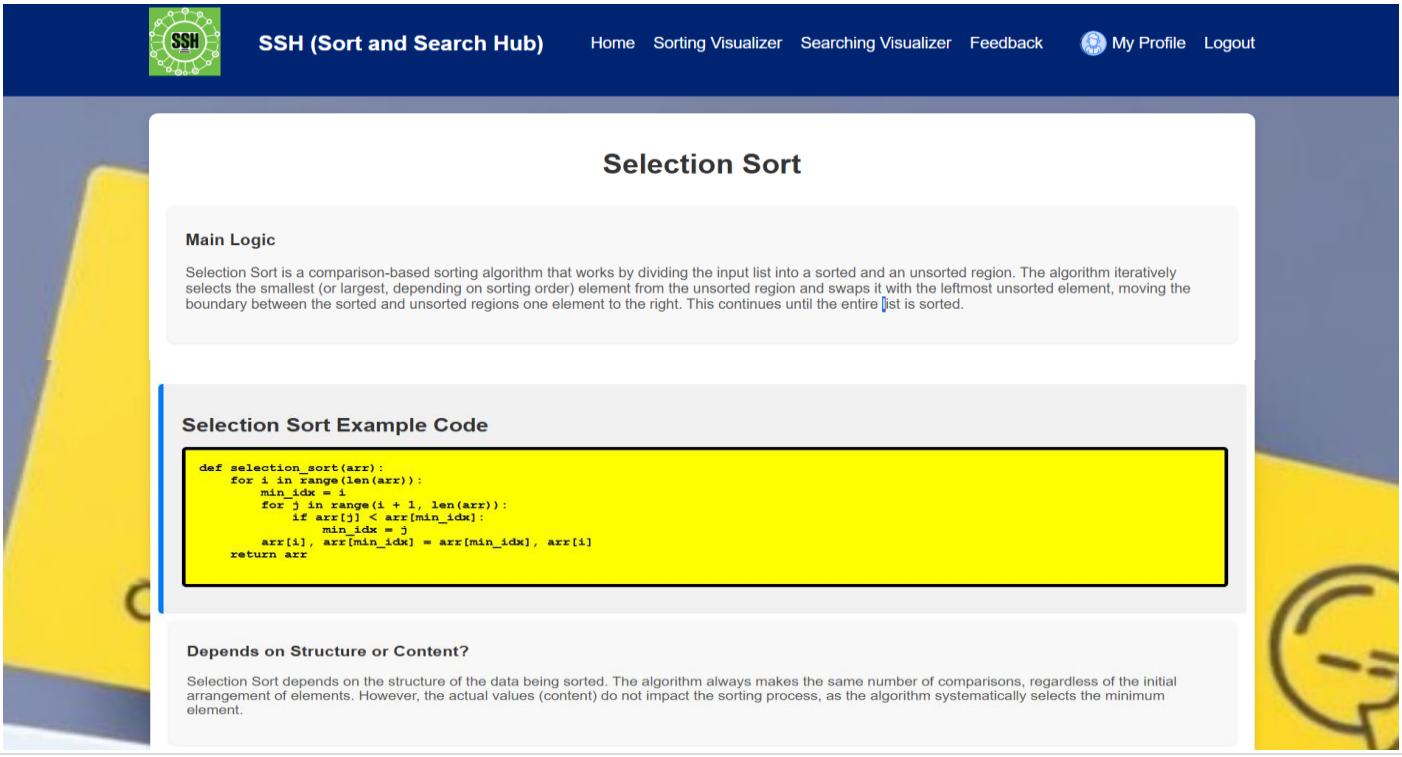


Figure 7.7 My Profile Updation Form

Figure 7.8 Searching Visualizer



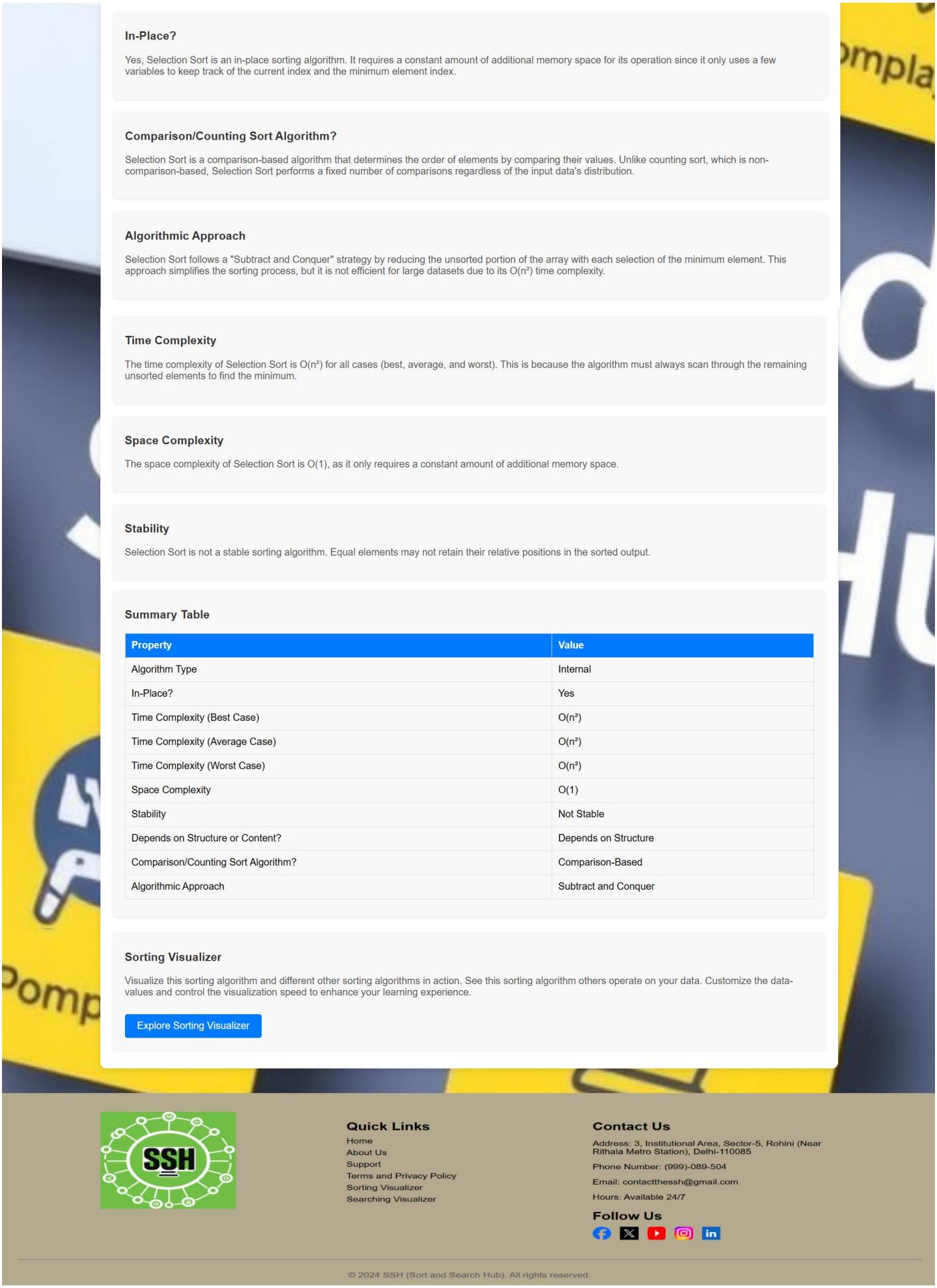


Figure 7.9 Algorithm Tutorial Pages Example (Selection Sort Tutorial Page)

CHAPTER 8: CONCLUSION

The development of the SSH Sort and Search Hub (SSH) represents a significant advancement in the visualization and understanding of sorting and searching algorithms. The project aimed to create an interactive and comprehensive platform that addresses the educational and practical needs of users seeking to learn and apply these algorithms effectively. Throughout the development lifecycle, these objectives were diligently pursued, resulting in the successful implementation of a robust system capable of meeting user needs.

The primary goal of SSH was to provide users with a seamless interface for visualizing and understanding sorting and searching algorithms. This objective has been achieved through the integration of real-time data visualization, user-friendly interfaces, and interactive features. By enabling users to input custom arrays and generate random arrays, and by offering different speed settings for visualizations, the system facilitates a deep understanding of algorithmic processes and enhances overall learning experiences.

Furthermore, the system's communication capabilities have been enhanced to foster collaboration among stakeholders. The feedback portal allows users to share their experiences and suggestions, which contributes to continuous improvement of the platform. The support portal ensures users can quickly resolve any issues they encounter, facilitating prompt resolution and effective communication. These features contribute to a more streamlined and transparent user experience.

While the system has successfully met its initial objectives, it is recognized that there is always room for improvement. Flexibility and adaptability are essential for accommodating future modifications and enhancements. Therefore, the system has been designed with modularity in mind, allowing for seamless integration of new functionalities and scalability to meet evolving user needs.

In terms of user-friendliness, efforts have been made to ensure a smooth and intuitive user experience. Despite the complexity of the system's underlying processes, the user interface has been optimized for simplicity and ease of use. This focus on user-centric design enhances user satisfaction and promotes widespread adoption of the platform.

In summary, SSH Sort and Search Hub represents a significant step forward in optimizing the learning and application of sorting and searching algorithms. By addressing key challenges and leveraging technology to streamline operations, the system empowers users to make informed decisions, improves communication and collaboration, and ultimately contributes to the overall success of the educational experience. Continued refinement and adaptation will be necessary to ensure the system's continued effectiveness and relevance in meeting the dynamic needs of users in the educational and professional landscape.

CHAPTER 9: BIBLIOGRAPHY

1. https://stackoverflow.com/
2. https://www.showwcase.com/
3. https://discord.com/
4. https://youtu.be/_ELCMngbM0E?si=JIBLRJfEwT-WkhUN
5. https://youtu.be/z4USlooVXG0?si=wCpfI-bHBcoK9fjP
6. https://youtu.be/aCafYfkhOpY?si=aJs8F3yC0VhroaZb
7. https://youtu.be/UxTwFMZ4r5k?si=BpzOlsihK_2LRHY_
8. https://youtu.be/CYvb3sqAI8E?si=4OEHKhdsPC5sx3jZ
9. https://youtu.be/WtU8g6d_C5A?si=5JAxdXnk_kW4AIFn
10. https://youtu.be/Qr4QMBUPxWo?si=kOK-734RMalsC55R
11. https://youtu.be/nOkpTwPvDTg?si=jv7yx3jSc06Hhxmh
12. https://github.com/clementmihailescu/Sorting-Visualizer
13. https://www.algoexpert.io/product
14. https://youtu.be/pFXYym4Wbkc?si=UBlFvAsCXch9aOLb
15. https://youtu.be/twRidO-_vqQ?si=czDc6Gyr4qIhtTUC
16. Kerren and J. T. Stasko. (2002) Chapter 1 Algorithm Animation. In: Diehl S. (eds) Software Visualization. Lecture Notes in Computer Science, vol 2269. Springer, Berlin, Heidelberg.
17. Hadi Sutopo, “SELECTION SORTING ALGORITHM VISUALIZATION USING FLASH”,The International Journal of Multimedia & Its Applications (IJMA) Vol.3, No.1, February 2011
18. Reif and T. Orehovacki ,“ViSA: Visualization of Sorting Algorithms”, Conference: 35th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2012), Opatija, Croatia, May 21-25, 2012, Volume: Computers in Education
19. Clement Mihailescu, “Sorting visualizer” ,2016, https://www.youtube.com/watch?v=pFXYym4Wbkc&t=903s
20. Faria, Brian, "Visualizing Sorting Algorithms" (2017). Honors Projects Overview. 127.
21. Slavomir Simonak, “Increasing the Engagement Level in Algorithms and Data Structures Course by Driving Algorithm Visualizations” September 2020, Informatica 44(3)
22. Hungarian (Küküllőmenti legényes) folk dance showing sorting by dance, https:// www.youtube.com/user/AlgoRythmics

---