

# Algorithm Visualizer

## DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE AWARD OF THE DEGREE  
OF

## BACHELOR OF SCIENCE

## (HONS.)

## COMPUTER APPLICATIONS

By

Haris Moin

20CAB106

GK2110



DEPARTMENT OF COMPUTER SCIENCE  
ALIGARH MUSLIM UNIVERSITY  
ALIGARH (INDIA)

Year of submission: 2023

## **CERTIFICATE**

This is to certify that Haris Moin, student of B.Sc. Hons Computer Applications, Aligarh Muslim University is working on project Algorithm Visualizer under the able guidance and supervision of Dr. Faisal Anwer, Assistant Professor, Aligarh Muslim University. In this semester Haris Moin has worked on Analysis and Design part of the project Algorithm Visualizer and the rest stages will be carried out in the next Semester.

No part of this report has been reproduced from any other report and the contents are based on original research to the best of my knowledge.

Dated:

Signature of Supervisor

Dr. Faisal Anwer  
(Assistant Professor)

## ACKNOWLEDGMENT

I would firstly like to say Alhamdulillah (All praise be to Allah) who has blessed me with so much that I can't thank enough.

I express my sincere thanks to my senior project supervisors **Dr. Faisal Anwer, Assistant Professor, Department of Computer Science**, who guided me all along, till the completion of my project work, and helped me to get sufficient knowledge and all the necessary information regarding the project for a better understanding of the same. I also thank **Prof. Aasim Zafar, Chairman, Department of Computer Science**, for providing the support and guidance which helped me complete the project on time. I am extremely grateful and thankful for their immense support.

I am fortunate enough to get encouragement, help, and support from all Teaching staff members of the department of computer science and all the resources I was able to avail in department's Lab. My thanks and appreciations also go to my colleagues in developing the project and people who have willingly helped me out with their abilities.

Haris Moin

GK2110

20CAB106

# ABSTRACT

Algorithms are formulas or methods for solving a problem. They serve their purpose by executing a sequence of finite steps until a result is achieved. So, algorithms are basically a sequence of carefully planned and documented steps.

Problem-solving is much more efficient and with the vast variety of algorithms to choose from every user's needs are satisfied according to their different form of data. As mentioned, a large amount of algorithms have been developed over the course of its advent and no matter however much easier they make our tasks happen, their understanding is very complex with no good form of visualization technique present. Every algorithm is eventually written/represented in the form of pseudo code which is just interpretation of steps of the algorithm in a recognized language like English which can be understood at a glance but it may be time consuming, or an algorithm may be in the form of code written in some high-level programming language which can be much harder to understand.

So, in order to make the learning process of algorithms easier multiple techniques like flow-charts, animations and have been used by instructors and students from time to time. Multiple websites provide an enriched documentation of algorithms and their pictorial representation and some other proved step by step visualization. But, still most of them lack in features like for instance if big data is provided or data for the problem is changed or incorporation of complex algorithms for their visualization such as Greedy algorithms etc. This Web based online platform shall serve the purpose of providing its users (Educators/Students) with an enriched documentary of algorithms alongside their graphical visualization. Also, it shall possess an interactive IDE which alongside the algorithm and its visualization shall propose its code in a specific programming language. This would give the user the freedom of manipulating with the code and having a hands-on experience of what the results after this change would look like.

# Content

	Pages
<b>Certificate</b> .....	i
<b>Acknowledgment</b> .....	ii
<b>Abstract</b> .....	iii
<b>Chapter 1: INTRODUCTION</b> .....	1
1.1: Literature Survey .....	7
1.2: Present state of art and its shortcoming .....	7
1.3: Introduction of problem or work to be taken up.....	7
1.4: Broad outline of the work.....	7
<b>Chapter 2: PROBLEM FORMULATION</b> .....	7
2.1: Various aspects of the problem .....	7
2.2: Significance of the work.....	7
2.3: Proposed system/method of solution.....	7
<b>Chapter 3: SYSTEM ANALYSIS AND DESIGN</b> .....	7
3.1: Information collection .....	7
3.2: Requirement specification .....	7
3.3: Analysis and development of actual solution .....	7
3.4: Description of various Modules .....	7
3.5: Choice of language .....	7
3.6: Choice of system for implementation.....	7
<b>Chapter 4: SYSTEM IMPLEMENTATION AND TESTING</b> .....	7
4.1: Operating Environment .....	7
4.2: Input requirement .....	7
4.3: System testing.....	7
<b>Chapter 5: APPLICATION AND SCOPE</b> .....	7
5.1: Application areas .....	7
5.2: Scope of the work.....	7

Cont.....

<b>Chapter 6: CONCLUSION .....</b>	<b>7</b>
6.1: Advantages and special features of the system.....	7
6.2: Limitations.....	7
6.3: Future extension .....	7

## References & Bibliography

## Appendix

A1 : DFD/System flow diagram etc .....	7
A2 : UML diagrams.....	7
A3 : ER diagrams .....	7
A4 : Schema diagrams.....	7
A5 : Activity diagrams .....	7
A6 : Wireframe Diagrams .....	7
A7 : Program listing-codes .....	7
A8 : Input (Test) data/screen and Sample output/ Reports/Screens .....	7

# **Chapter 1:**

# **INTRODUCTION**

1.1 Literature Survey

1.2 Present state of art and its shortcoming

1.3 Introduction of problem or work to be taken up

1.4 Broad outline of the work

## 1.1 Literature Survey

Algorithm Visualizer: Its features and working by Barnini Goswami, Anushka Dhar, Akash Gupta, Antriksh Gupta from Computer Science and Engineering Department Krishna Engineering College Ghaziabad, India. Published in the International Journal of Institute of Electrical and Electronics Engineers (IEEE), 2021

The paper “Algorithm Visualizer: Its features and working” introduces a web-based application using java script and react, that uses interactive visualizations to help students learn about various algorithms, such as path-finding, sorting, and CPU scheduling algorithms. The use of this application can not only help students better understand these algorithms, but also provide an innovative and engaging way for educators to teach them.

The paper starts by discussing the importance of algorithms as a fundamental subject in the field of computer science, but also acknowledges that it can be a complicated and difficult concept to understand. It also depicts the importance of how pictorial representation are better than techniques used in computer science for better understanding of algorithms such as pseudo code and more.

The authors have made use of vanilla JavaScript and React JS frameworks which allowed them to build an effective and engaging application for visualizing algorithms. Vanilla JavaScript provided a strong foundation for creating interactive and dynamic visualizations, allowing the application to handle user input and control animations. Additionally, React JS's modular and reusable component-based approach made it easier for the authors to break down the application into smaller, more manageable parts, while also providing built-in support for managing the application state. This allowed for more efficient and responsive visualizations, as well as easier updates to the display based on changes in the underlying data.



# **Chapter 2:**

# **PROBLEM FORMULATION**

2.1 Various aspects of the problem

2.2 Significance of the work

2.3 Proposed system/method of solution

## **2.1 Various aspects of the problem**

### **Definition and Aspects**

Most of the times our experience of understanding how algorithms work and how they process data has limitations on the viewer, restricting them to only 2 dimensional pictures and diagrams which are inadequate and often leads them to losing interest in this subject. This effect has seemed to persist and new learners are always discouraged by the lack of a one-in-all platform that could provide them with all the graphical resources they need to have an in-depth knowledge of the same. Many attempts have been made to contribute in this regard but most of the web platforms available nowadays for better visualisation of algorithms and data structures are not optimized, they work for pre-defined fed data, doesn't show the flow of data alongside the working of the algorithm and most of them only contain a handful of visualisations available to the viewer such as all the sorting algorithms and some may have search algorithms etc.

These systems and applications lack many things among which most important can be the list/number of visualizations available and whether or not the user is able to provide an input and see the changes happen on his own data. Regardless of that, the work done in those early attempts have proved to be of great help in the completion and idea of this project.

## **2.2 Significance of the work**

The current applications and system pose many unwanted barriers in the process of learning some of which are –

- The platforms are not interactive, meaning they do not take into account of how the user wants a specific algorithm to work on a provided data by the user themselves this makes the user being alienated.
- There is not enough material to be viewed on the platform. At most what we get is the group of basic sorting and searching algorithms with a poor construction of their visualization.
- Apart from the visualization not many platforms provide a comprehensive study of these algorithms alongside their functioning which is very crucial in some scenarios. At most they present the user with a link of the very algorithm which directs them to another

website. While that may be enough but is still not a tidy way to compact things when creating a project and also an inconvenience to the visitors of the application.

Altogether these applications fail to give a good experience to the visitor and thus not fulfilling their purpose

## **2.3 Proposed System/ Method of Solution**

This project is more automated and interactive in comparison to its predecessors and it aims to fulfil all the gaps left in previous attempts by developers. It shall target the very problems faced in the current applications present which are, to name a few interactiveness with the user, the participation of users in contributing their own algorithms and optimization of almost all known and basic algorithms. This project will give an in-built IDE as its main interface which could be used to select a specific algorithm out of the many listed and view its visualization along with default or user fed data. Apart from this, the users will be able to create and edit their own contribution profile at the application which shall give the user permission to contribute and make posts regarding the same in order for the admin to approve them. If not for contribution purposes the visitor could just enjoy the IDE for exploring and experimenting purposes. The IDE would be comprised of three sections broadly which are –

- The Visual section
- The Categories of Algorithms
- The Code Section containing the code of present algorithm in a specific language which could be modified

This shall provide a great user interface and experience as everything being at one place and easy to navigate from. From the Main interface which is the IDE itself the user could sign up in a section provided which shall be available at the navigation bar, or if already rolled in the user could just log in, which is just another interface consisting the login section of the admin as well. The admin on the other hand has all command over who gets to be a contributor, what contribution gets approved and does it get to be displayed on the Visualizer and further he would have the ability to modify the data of the Visualizer in any way possible.

Key Points –

- More Convenient

- User Friendly and Interactive
- Security of Data
- Minimize Manual Work
- Less Paper work
- Greater Efficiency

# **Chapter 3:**

# **SYSTEM ANALYSIS AND DESIGN**

3.1 Information collection

3.2 Requirement specification

3.3 Analysis and development of actual solution

3.4 Description of various Modules

3.5 Choice of language

3.6 Choice of system for implementation

## **3.1 Information Collection**

The data on which the visualizer operates is based on its users input and the code that the admin and contributors contribute.

## **3.2 Requirement Specification**

### **3.2.1.1 Product Functions**

- The category of algorithms to choose from will show the desired visualization
- Separate Login and sign pages for Users, Admin and Contributors. Though, the interface for redirecting to the Login pages of Admin and Contributor shall be same.
- Visitors can become Contributors by simply signing up and creating their profile
- Contributors can then create posts regarding the admission of a specific a algorithm they think needs recognition.
  - This Post goes to the Admin via the Main system, who can either reject this approval or accept it based on either one of the decisions;
  - The algorithm gets added to alongside its visualization and the contributor gets notified about it.
- Shows the profile of the Admin and the Contributor which could be modified
- Admin could remove Contributors

### **3.2.1.2 User Classes and Characteristics**

In here the system admin, contributors and visitors are the system users. The Visitor could browse through the whole application and take advantage of all the resources provided, the visualizer could be invoked to get desired visualizations as needed, and the vast collection of algorithm and data structures provided could be explored. The user/visitor could become a contributor by signing up and creating a profile. This admission is approved by the admin in the manner of whether the user gets to stay in the list of contributors or not.

Default language is English

It is advised for all the users to have basic idea of operating(using) the system and should have an experience to work in the browser(internet).

### **3.2.1.3 Design and Implementation Constraints**

Some of the design and implementation constraints identified are listed below:

- Users are not allowed to log in until enrolled as a contributor
- Interface should be well versed and prepared for good user experience
- Visitor is allowed to have only one profile equivalent to their email id.

#### **3.2.1.4 Assumptions and Dependencies**

### **3.2.2 Requirement Specification**

#### **3.2.2.1 User Interfaces**

User Interface Design is concerned with the dialogue between a user and the computer. It is concerned with everything from starting the system or logging into the system to the eventually presentation of desired inputs and outputs.

The user interface of this system will have to be simple and clear. Most importantly, the information must be easy to read, easy to understand and accessible. The colour scheme should be appropriate to provide familiarity and there should be no contrast issues. It's required that interface should be user friendly and simple to use

The following steps are various guidelines for user Interface Design:

- The system user should always be aware of what to do next.
- The screen should be formatted so that various types of information, instructions and messages always appear in the same general display area.
- Message, instructions or information should be displayed long enough to allow the system user to read them.
- Default values for fields and answers to be entered by the user should be specified.
- A user should not be allowed to proceed without correcting an error.
- The system user should never get a fatal error

#### **3.2.2.2 Hardware Interfaces Client side:**

- Processor with 500Mhz clock speed.
- 512 Megabytes of RAM capacity.
- 1 Gigabytes of free Hard disk space.

### **3.2.2.3 Software Interfaces**

- Client side: Web Browser, Operating System (any), Internet connectivity.

### **3.2.2.4 Communications Interfaces**

- Client on Internet will be using HTTP/HTTPS Protocol

## **3.2.3 Performance Requirements**

Some Performance requirements identified is listed below:

- The database shall be able to accommodate a minimum of 10,000 records.
- The system shall support use of multiple users at a time.
- The system shall have a good response time

## **3.2.4 Safety and Security Requirements**

Some of the factors that are identified to protect the system from accidental or malicious access, use, modification, destruction, or disclosure are described below. Specific requirements in this area could include the need to:

- Keep specific log or history data sets.
- The software will include an error tracking log that will help the user understand what error occurred when the application crashed along with suggestions on how to prevent the error from occurring again
- Restrict communications between some areas of the program
- Check data integrity for critical variables
- Later version of the software will incorporate encryption techniques in the user/license authentication process.
- Assign certain functions to different modules
- Communication needs to be restricted when the application is validating the user or license.
- The system must automatically log out all users after a period of inactivity.



### 3.2.5 Software Quality Attributes

The system will possess some quality attributes to the users:

- **Reliability:**  
The system has the ability to work all the times without failures apart from network failure.
- **Correctness:**  
The application should be correct in terms of its functionality, calculations used internally and the navigation should be correct
- **Testability:**  
The system should be easy to test and find defects. If required, it should be easy to divide into different modules for testing.
- **Reusability:**  
Different code library classes should be generic enough to be easily used in different application modules. Divide the application into different modules so that modules can be reused across the application.
- **Robustness:**  
The System should work correctly and efficiently in all conditions.
- **Flexibility:**  
The application should be flexible and can run smoothly on any device, platform, or operating system.

### 3.2.6 : System Features

This section illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

#### 1 Login

##### 1.1 Description

A login page for Admin and Contributor to access specific functionalities of each of them.

##### 1.2 Response

The User must need to enter the required username and password to access the main system

### 1.3 Functional Requirements

REQ-1: Everyone must have a unique Username

REQ-2: Contributor and Admin have different login pages

REQ-3: Login details must be saved into the database at the time of Register for future reference

## 2 Register

### 2.1 Description

A registration page for Admin and Contributor to access specific functionalities of each of them.

### 2.2 Response

The User must need to enter the details in order to Register

### 2.3 Functional Requirements

REQ-1: Everyone gets a username and password for login after registration

REQ-2: Only Contributor has registration pages

REQ-3: Registration details must be saved into the database at the time of Sign up for future reference

## 3 Search

### 3.1 Description

A database result for an algorithm so required

### 3.2 Response

The User must need to enter the name of the algorithm

### 3.3 Functional Requirements

REQ-1: Visitor, Contributor select the algorithm and get its detail

REQ-2: The Category of Algorithm gets shown if it falls under any such category

## 4 Visualize

### 4.1 Description

A chosen algorithm is fetched and is graphically shown on the interface along with its source code and its hierarchy.

### 4.2 Response

The User can view the result on the main interface after pressing run

#### 4.3 Functional Requirements

REQ-1: Every algorithm has a name and an id and the category to which it belongs

REQ-2: Visitor and Contributor must specify its name

REQ-3: The run button should be clicked in order for it to display

### 5 Record log

#### 5.1 Description

Maintain a record for the activities in the system by the users.

#### 5.3 Functional Requirements

REQ-1: Contributor's and admin login details must be saved

REQ-2: A record of which algorithm are fetched

### 6 Add Contribution

#### 6.1 Description

An admin functionality for adding a contribution to the algorithm database and modifying the visualizer.

#### 6.2 Response

The post/contribution gets added and an approval sent to its rightful contributor.

The contribution is made available on the visualizer

#### 6.3 Functional Requirements

REQ-1: The admin must have login details

REQ-2: The contributors data must be fetched and the id of contribution should be made available so as to take action

### 7 Remove Contributor

#### 7.1 Description

Admin's power to remove a contributor

#### 7.2 Response

The contributor along with their current email get removed from the system.

### 7.3 Functional Requirements

REQ-1: The login of admin

REQ-2: The unique id of the contributor

REQ-3: Notification sent to the user

## 8 Approve/ Disapprove Contribution

### 8.1 Description

The admin response to the post/contribution made by the User

### 8.2 Response

A notification sent on User's profile for their knowledge of whether their post has been approved/disapproved for being registered on the visualizer.

### 8.3 Functional Requirements

REQ-1: The admin logged in

REQ-2: The contributor id and details

REQ-3: The post/contribution id

REQ-4: Approval/Disapproval message being sent

## 9 Modify the Visualizer data

### 9.1 Description

Admin authority to modify the visualizer's database in order to delete, modify, add any algorithm and details of its visualization.

### 9.2 Response

The data get embedded in the system

### 9.3 Functional Requirements

REQ-1: Admin's unique id and password

REQ-2: Confirmation after modification

# **Chapter 4:**

# **SYSTEM IMPLEMENTATION AND TESTING**

4.1 Operating Environment

4.2 Input Requirements

4.3 System testing

## 4.1 Operating Environment

The server-side components of the software system will operate within a Linux operating system. The client-side components of the software system must operate in common web browser within Windows, Linux or mac operating system with minimum hardware requirements –

- Processor with 500Mhz clock speed,
- 512 Megabytes of RAM capacity
- 1 Gigabytes of free Hard disk space.

## 4.2 Input Requirements

The input requirements of the code snippet in the visualizer must be precise and in accordance with the syntax of the language which in this case should be python3 and above. Also, the user must take note of the size of their code, and roughly of what complexity could it reach. The requirements would conclude in,

- Syntax corrective measures should be applied with the response of the visualizer
- The complexity(space and time) should be taken into account before running the code as it could reach out of the reach of visualizer

Further taken in consideration the proper use of the application the user must take in account of human error to be present in the code contributed by other users. Therefore, contribution and corrections should be regularly made.

## 4.3 System Testing

System testing is a critical phase in the software development life cycle that involves testing the entire system as a whole to ensure that it meets the requirements and functions as expected. The following are some of the things that are included in system testing:

1. Functional testing: This involves testing the system's functionality against the requirements to ensure that it performs the intended tasks. This may include testing various scenarios and edge cases to ensure that the system responds correctly to different inputs.

The Importance of proper validation of data have taken into consideration. Various techniques have been implemented to ensure that the data entered into the system is accurate and complete. Front-end validation checks have been included that ensure that users enter the correct data format, length, and type. Additionally, Back-end validation is also done to ensure that data received from the client side is valid and meets the expected criteria.

2. Security testing: This involves testing the system's security features to ensure that it can protect user data and prevent unauthorized access. This may include testing the system's authentication and authorization mechanisms, encryption, and access controls.

The system is successfully tested against unauthorized access. The system will redirect anyone who tries to access the system without authorization to the login page or homepage (depends on the type of data tried to be accessed).

Even if user is logged in, he/she may only access data intended for them, on attempt of any other data, they will be logged out of the system and redirected to the homepage

3. Usability testing: This involves testing the system's user interface and user experience to ensure that it is intuitive, easy to use, and meets the needs of users. This may include testing the system's navigation, user input and feedback, and accessibility.

The system is tested by various users for the interface and user experience. As this is one of the aim of the project, the system is make very user friendly and easily accessible with proper notes and information provided where required.

4. Compatibility testing: This involves testing the system's compatibility with different hardware and software configurations, browsers, and devices. This ensures that the system can be accessed and used by a wide range of users.

As the system is hosted on the internet for anyone to access, it is compatible with all devices which can access internet through a browser and have internet connectivity.

Overall, system testing is a critical part of software development that ensures that the system is functional, secure, performant, and easy to use. It is important to perform thorough system testing to identify and address any issues before deploying the system to users.

# **Chapter 5:**

# **APPLICATION AND SCOPE**

5.1 Application areas

5.2 Scope of the work



## 5.1 Application Areas

The applications of this project are endless when considered in the field of education and learning. Visual learning is an effective way to engage learners and help them understand complex concepts. Research has shown that visual aids such as diagrams, charts, and animations can enhance comprehension and retention of information. Moreover, visual learning can be especially helpful for learners who are more visual or kinesthetic learners. This project can help learners better understand how algorithms work by showing them a step-by-step execution of the code.

This can be especially helpful for learners who struggle to understand abstract concepts, such as loops or recursion. By providing a visual representation of the code execution, learners can more easily identify errors, visualize how the algorithm works, and understand the flow of the code.

Moreover, this project has many potential applications. For example, it can be used in computer science classrooms to help students understand programming concepts. It can also be used in coding bootcamps and online courses to provide a more engaging and interactive learning experience. Additionally, your project can be used by educators in other fields, such as math or science, to help students visualize complex concepts.

Furthermore, this project can help individuals who are learning to code on their own. Many beginners struggle with understanding how algorithms work and how to debug their code. The project can provide a helpful tool for beginners to test their code and see how it executes step by step. This can be especially helpful for those who are new to programming and may feel overwhelmed by the abstract nature of coding.

In summary, algorithm visualizer project has the potential to be an incredibly useful tool for educators and learners alike. It can help learners better understand programming concepts, identify errors in their code, and provide an engaging and interactive learning experience. Additionally, it has many potential applications beyond computer science classrooms and can be helpful for anyone learning to code or teaching complex concepts.

## 5.2 Scope of the work

The scope of algorithm visualizer is vast, as it has the potential to be used in many different contexts and by many different types of users. Here are some of the areas where algorithm visualizer can be applied:

1. **Education:** Algorithm visualizer can be used in educational institutions such as schools, colleges, and universities to help students learn programming concepts. Teachers can use the tool to create visual demonstrations of algorithms that students can follow along with, and students can use the tool to test their own code and see how it executes.
2. **Online Courses and Tutorials:** Algorithm visualizer can be used in online courses and tutorials to help learners understand complex programming concepts. Developers and educators can create video tutorials that include visualizations of code execution, or they can create interactive lessons that allow learners to test their own code and see how it executes.
3. **Coding Bootcamps:** Algorithm visualizer can be a valuable tool for coding bootcamps and other intensive programming training programs. By providing learners with a visual representation of code execution, instructors can help them better understand the concepts they are learning and identify errors in their code.
4. **Debugging:** Algorithm visualizer can be used as a debugging tool for developers to help them identify errors in their code. By stepping through the code execution, developers can pinpoint the source of the error and fix it more quickly.
5. **Algorithm Design and Analysis:** Algorithm visualizer can also be used to design and analyze algorithms. By visualizing how different algorithms execute, developers can compare their efficiency and determine which algorithm is best suited for a particular task.

Overall, the scope of algorithm visualizer is broad, and it has the potential to be a valuable tool for anyone involved in programming or computer science education.

# **Chapter 6:**

# **CONCLUSION**

6.1 Advantages and special features of the system

6.2 Limitations

6.3 Future extension

## 6.1 Advantages and special features of the system

It has several advantages that can help users learn programming concepts more easily, identify errors in their code more quickly, design more efficient algorithms, and have a more engaging and interactive learning experience. The few advantages are :

1. Better understanding of algorithms: By visualizing how algorithms execute, users can gain a better understanding of how they work. This can help users identify errors in their code, optimize algorithms for performance, and design new algorithms more effectively.
2. Improved learning outcomes: Visual aids have been shown to improve learning outcomes, especially for complex subjects like programming. An algorithm visualizer can help students grasp programming concepts more easily and engage them in the learning process.
3. Interactive learning: Algorithm visualizer provides an interactive learning experience. Users can step through the code execution at their own pace, pause and resume as needed, and experiment with different inputs to see how the algorithm behaves.
4. Error identification and debugging: Algorithm visualizer can help users identify errors in their code more quickly and easily. By visualizing the code execution step by step, users can pinpoint the source of the error and fix it more efficiently.
5. Efficient algorithm design: Algorithm visualizer can help users design more efficient algorithms. By comparing the execution of different algorithms, users can determine which one is best suited for a particular task and optimize it for performance.
6. Engagement: Visual aids like algorithm visualizer can make programming more engaging and fun. Users can see the code come to life, and this can motivate them to learn more and experiment with programming concepts.

The special features of this application include:

- Contribution: become a contributor by registering and becoming part of the algorithm visualizer
- Registration
- Login
- Logout
- Terminal

- Visual display
- Profile creation
- Profile Update
- Contribution update
- Contribution delete
- Admin dashboard access
- Admin login

## 6.2 Limitations

The limitations of this project are at its terminal execution side which executes the visualization works on the basis of a java script library named `visualization.bundle.js` and the library expects an argument of python code executed each line at a time which is further done by the intermediary CGI or WSGI that acts as a development stage for executing python code.

Some of the limitations include:

- The code argument in the terminal should not execute more than 1000 times, after which It will stop working and would want you to change your code.

## 6.3 Future extension

The scope of this project would include,

- Extending its features In more than one programming language such as, in order of their priority as JAVA, C++, JavaScript, etc
- Increasing the storage capability of the visualizer in order to accept code of greater space complexity.
- Adding a restriction on user basis of who can contribute and who cannot

## REFERENCES:



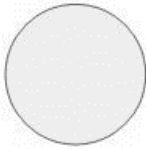


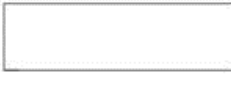


1. Roger S. Pressman & David Lowe (2009), Web Engineering: A Practitioner's Approach, McGraw-Hill. Retrieved September 11, 2022
2. SDLC - Agile Model. (n.d.-b). Retrieved September 25, 2022, from [https://www.tutorialspoint.com/sdlc/sdlc\\_agile\\_model.htm](https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm)
3. UML Activity Diagram Tutorial. (n.d.). Lucidchart. Retrieved October 27, 2022, from <https://www.lucidchart.com/pages/uml-activity-diagram>
4. "IEEE Guide for Software Requirements Specifications," in IEEE Std 830- 1984 , vol., no., pp.1-26, 10 Feb. 1984, doi: 10.1109/IEEESTD.1984.119205. a. Retrieved September 17, 2022 from [https://web.cs.dal.ca/~hawkey/3130/srs\\_template-ieee.doc](https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc)
5. Rajarman, V. (1989), Analysis and Design of Information Systems, Prentice Hall of India. . Retrieved September 21, 2022.
6. Barnini Goswami, Anushka Dhar, Akash Gupta, Antriksh Gupta(2021) Algorithm Visualizer: Its features and working 10.1109/UPCON52273.2021.9667586
7. Robin Nixon, (2014), Learning PHP, MySQL & JavaScript with JQUERY, CSS & HTML5, O'Reilly Media, Inc. Retrieved October 09, 2022.
8. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein(2009) Introduction to Algorithms (Third Edition)
9. UML Use Case Diagram Tutorial. (n.d.-b). Lucidchart. Retrieved October 25, 2022, from <https://www.lucidchart.com/pages/uml-use-case-diagram>

## APPENDIX

### A1 : DFD/System flow diagram etc.

Data flow diagram is the starting point of the design phase that functionally decomposes the requirements specification. A DFD consists of a series of bubbles joined by lines. The bubbles represent data transformation and the lines represent data flows in the system. A DFD describes what data flow rather than how they are processed, so it does not hardware, software and data structure. A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

The data flow diagram is a graphical description of a system's data and how to Process transform the data is known as Data Flow Diagram (DFD). Unlike details flow chart, DFDs don't supply detail descriptions of modules that graphically describe a system's data and how the data interact with the system.

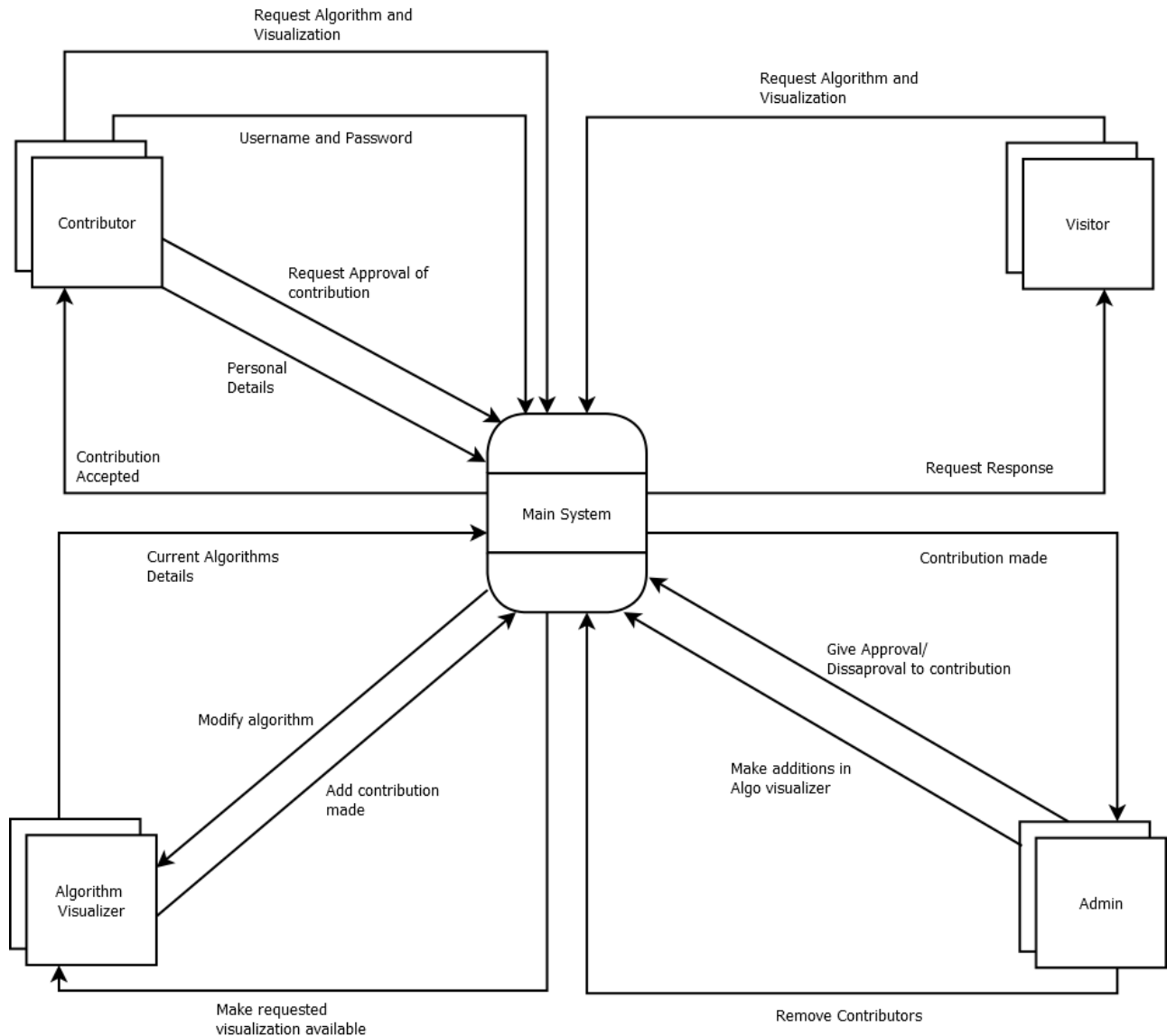
Notation	De Marco & Yourdon	Gane and Sarson
External Entity		
Process		
Data Store		
Data Flow		

A data flow diagram (DFD) is a significant modelling technique for analysing and constructing information processes. DFD literally means an illustration that explains the course or movement of information in a process. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a Process Model.



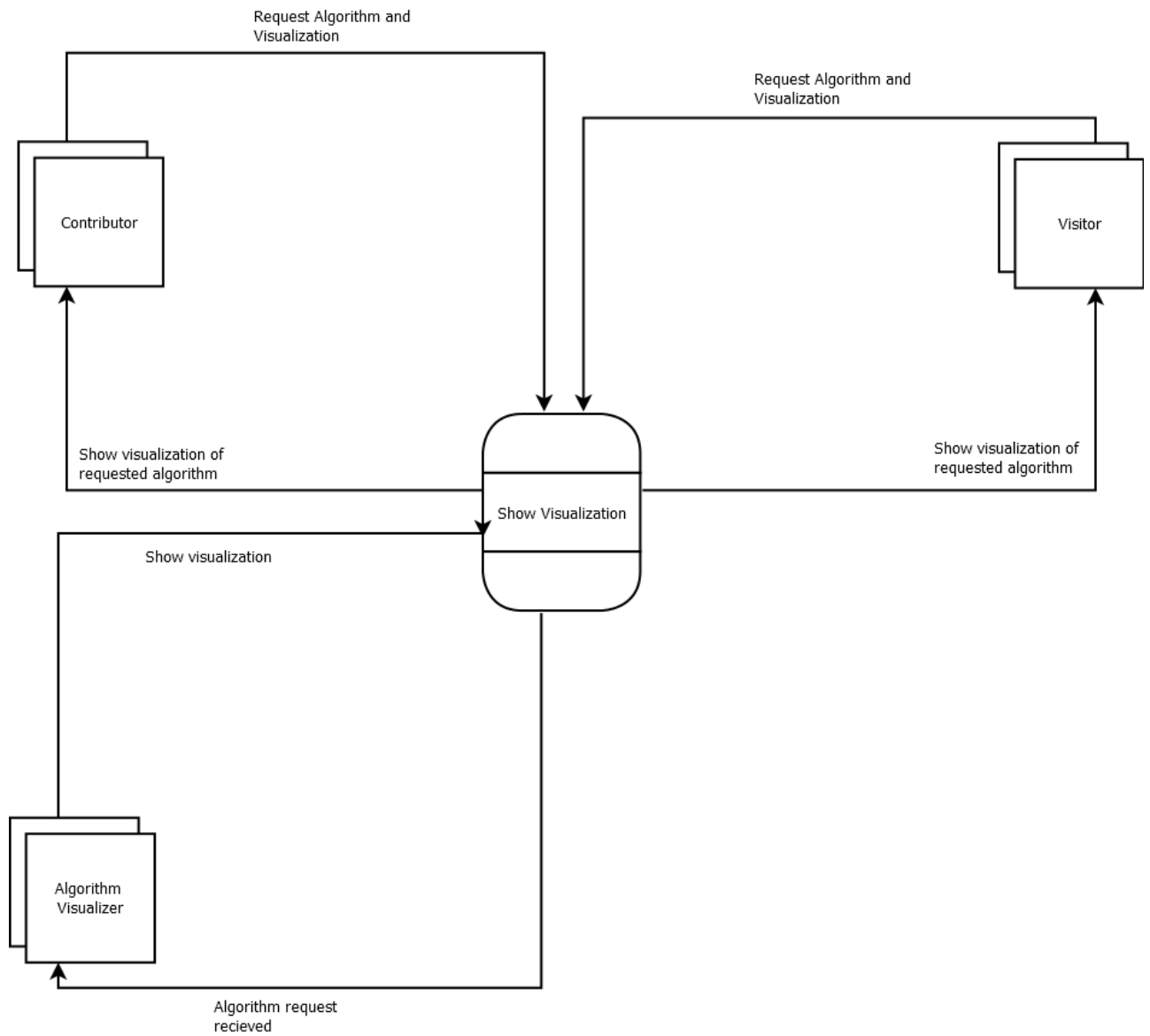
## Level 0 Context level DFD:

DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analysed or modelled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

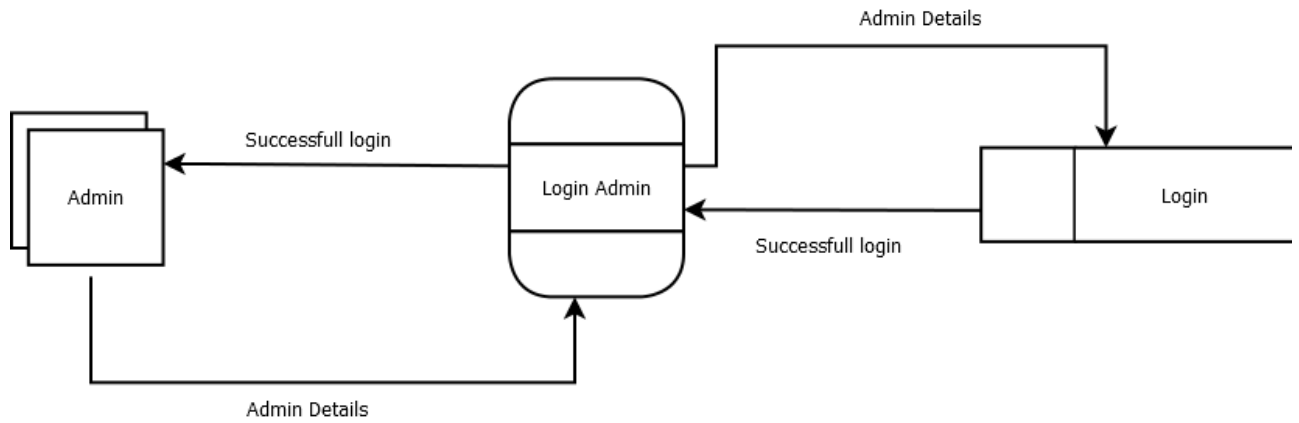


## DFD for different functionalities:

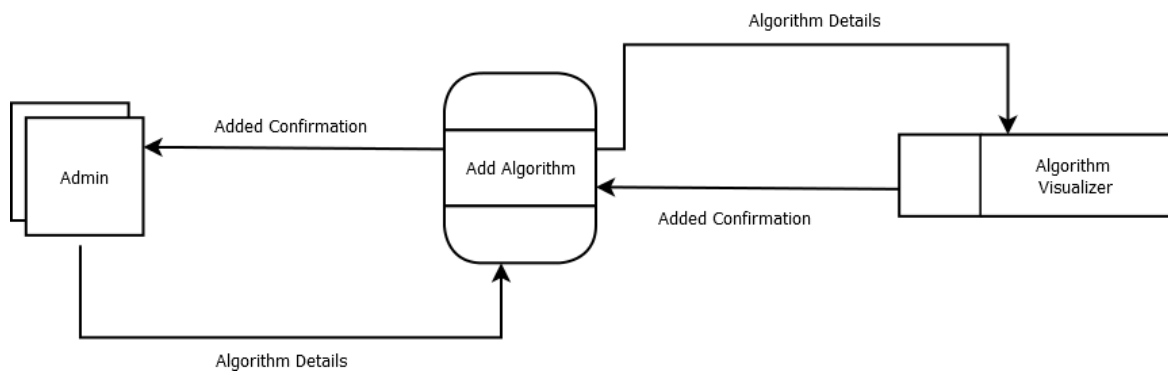
- Show Visualization



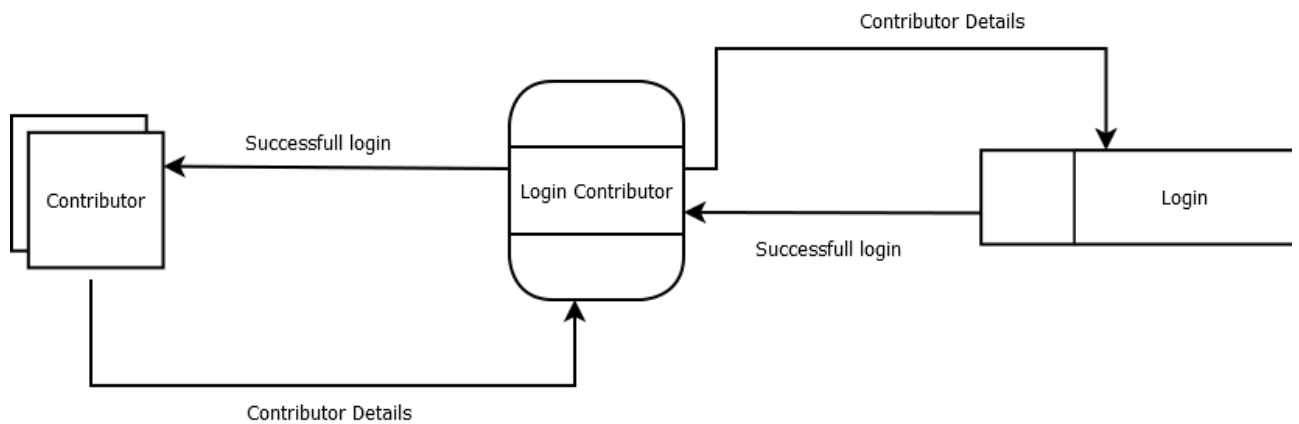
- Login Admin



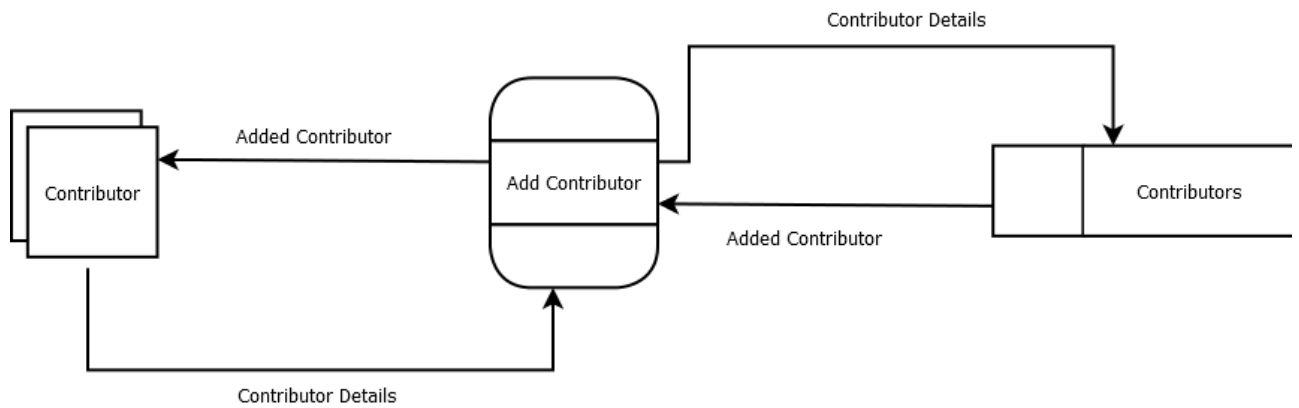
- Add Algorithm



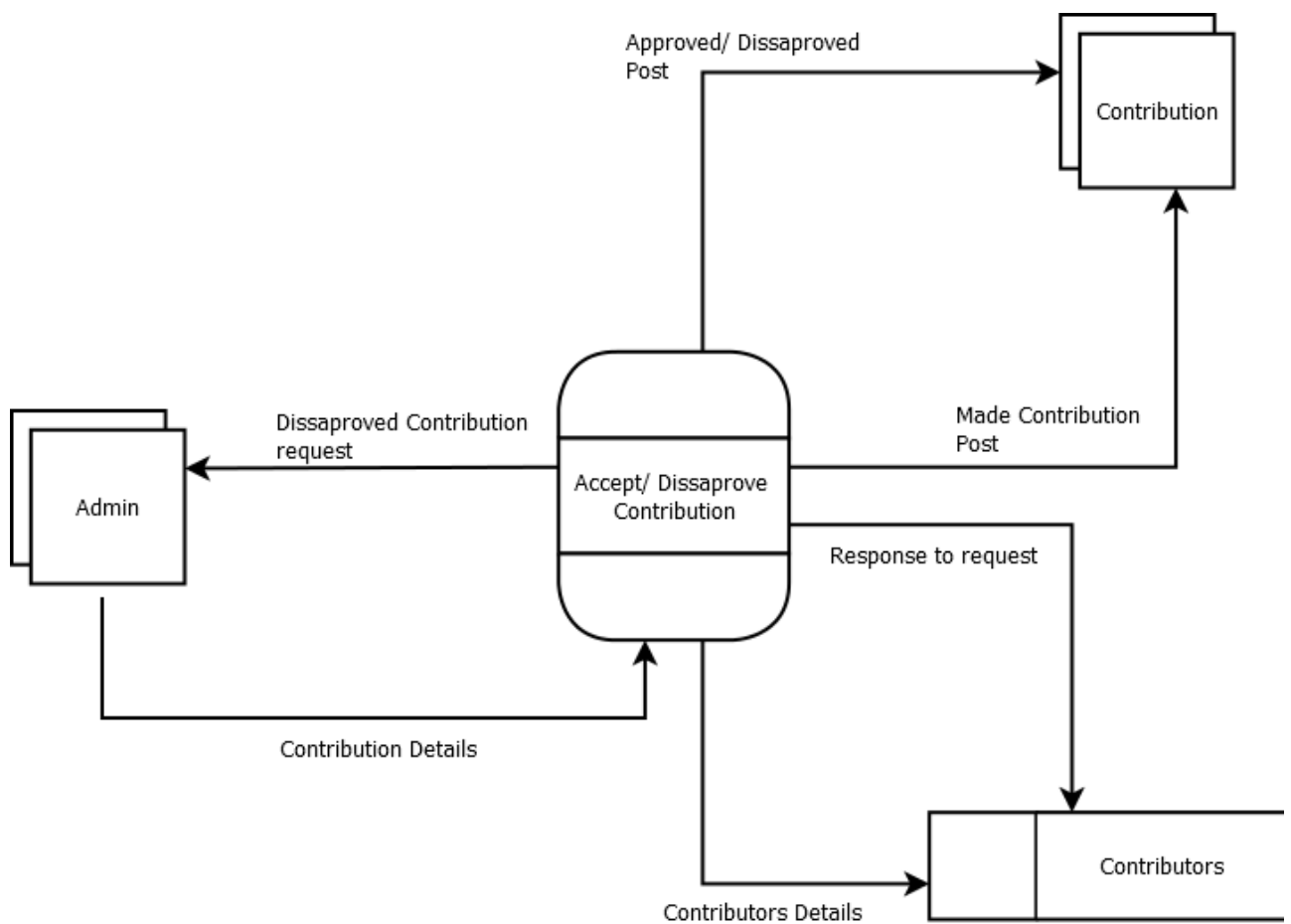
- Login Contributor



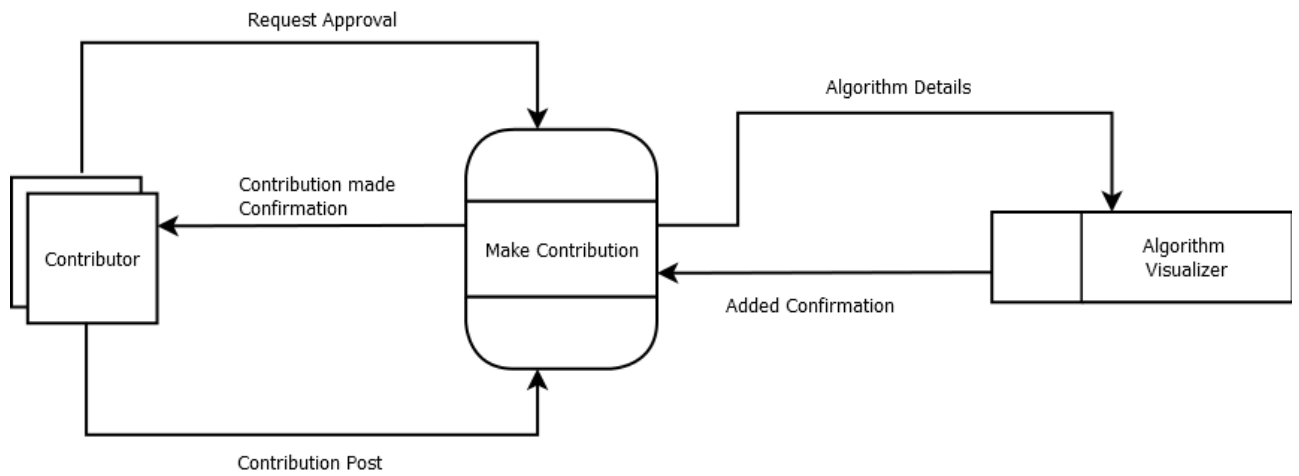
- Add Contribution



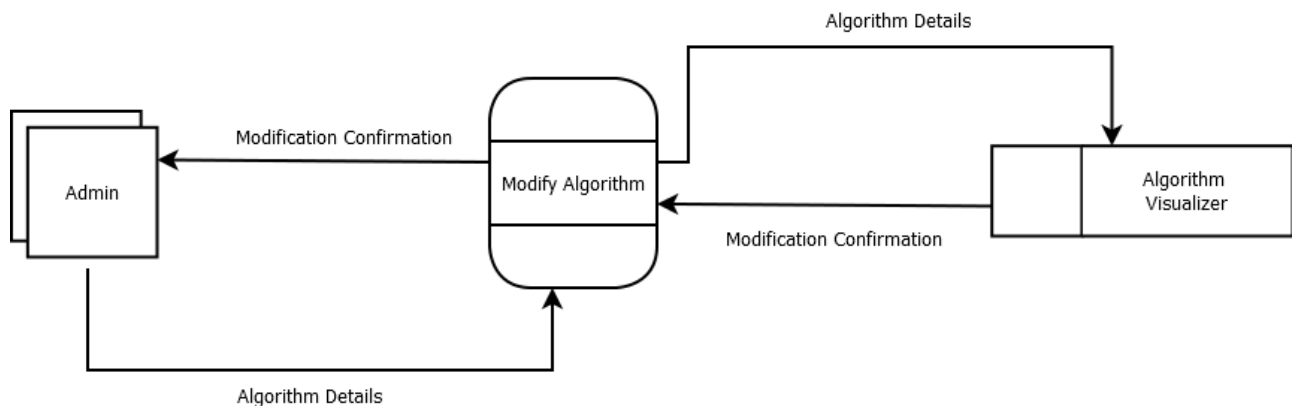
- Accept/ Disapprove Contribution



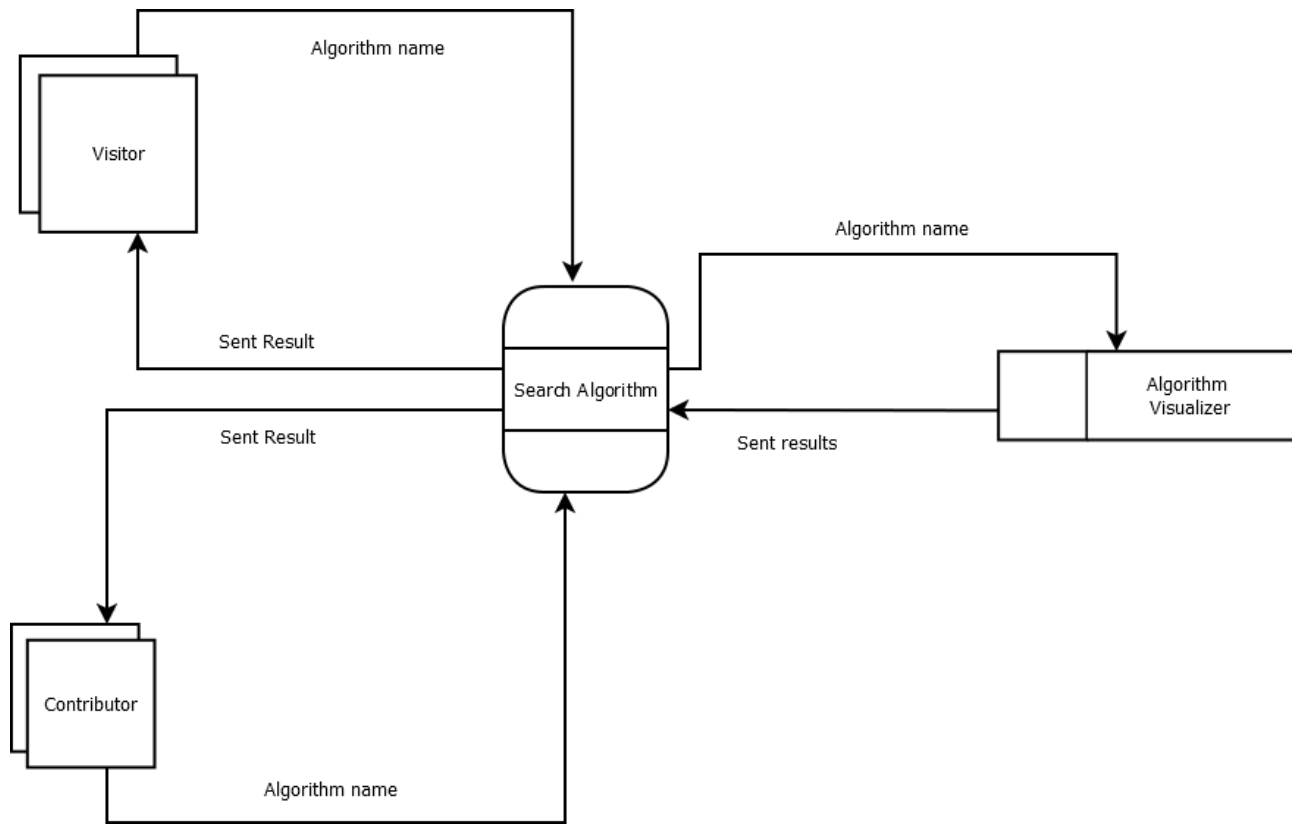
- Make Contribution



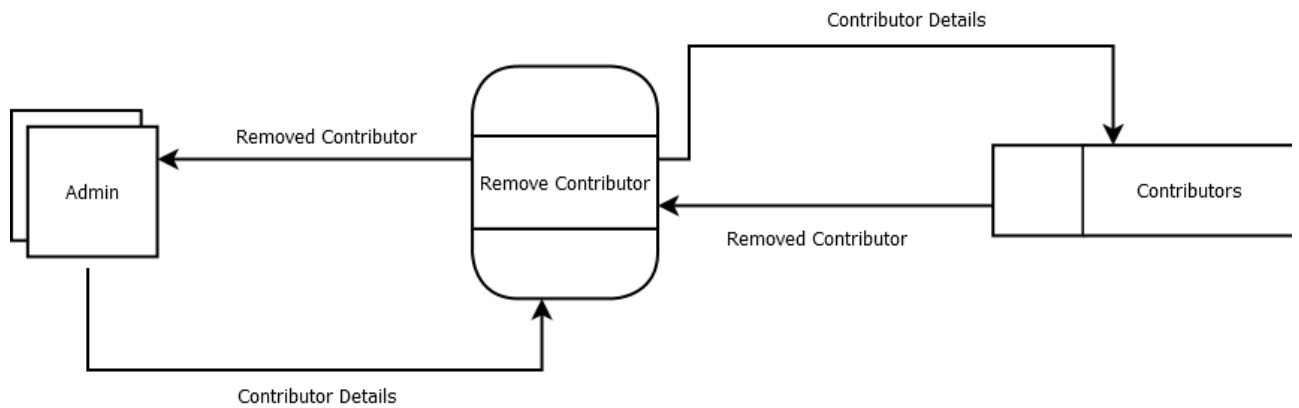
- Modify Algorithm



- Search Algorithm



- Remove Contribution



## A2 : UML Designs

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

Use case diagram components include:

**Actors:** The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.

**System:** A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.

**Goals:** The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modelling toolkit that you can use to build your diagrams. Use cases are represented with a labelled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modelled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

UML use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system

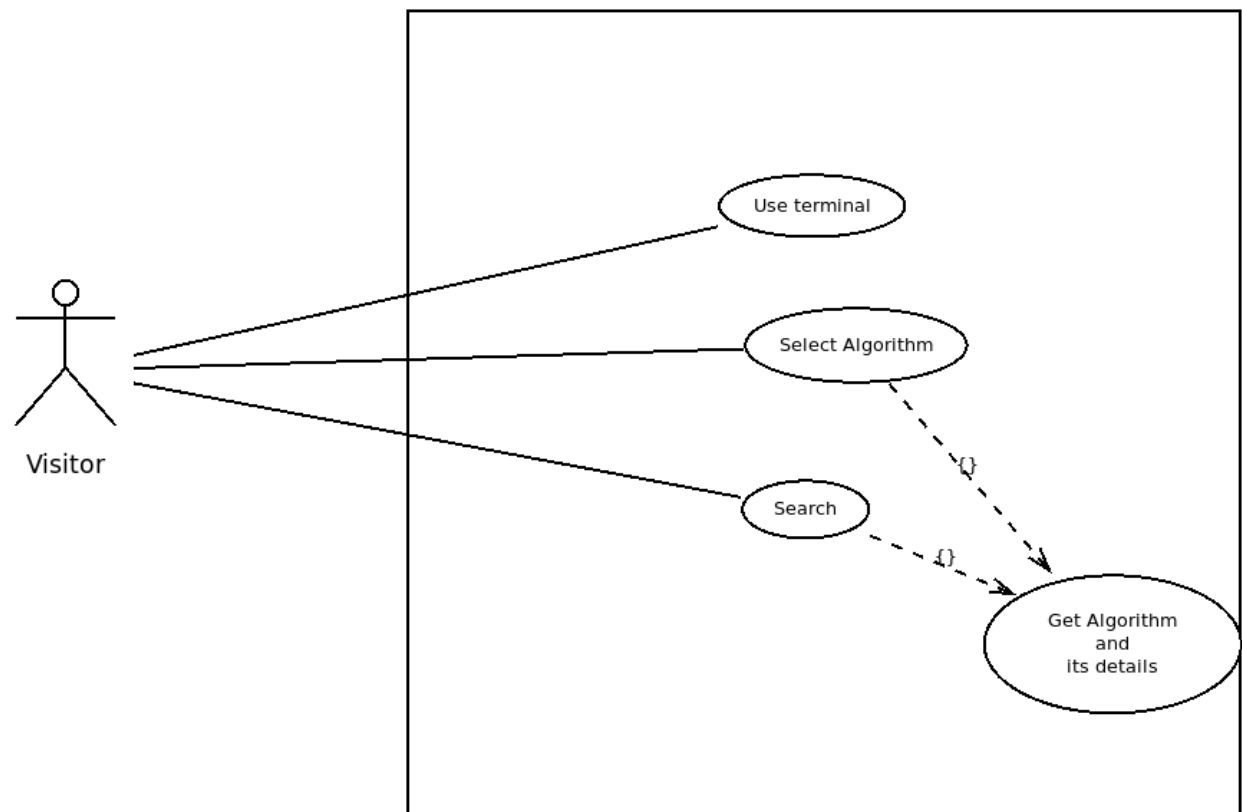
- Specifying the context and requirements of a system
- Modelling the basic flow of events in a use case

The notation for a use case diagram is pretty straightforward and doesn't involve as many types of symbols as other UML diagrams. You can use this guide to learn how to draw a use case diagram if you need a refresher. Here are all the shapes you will be able to find:

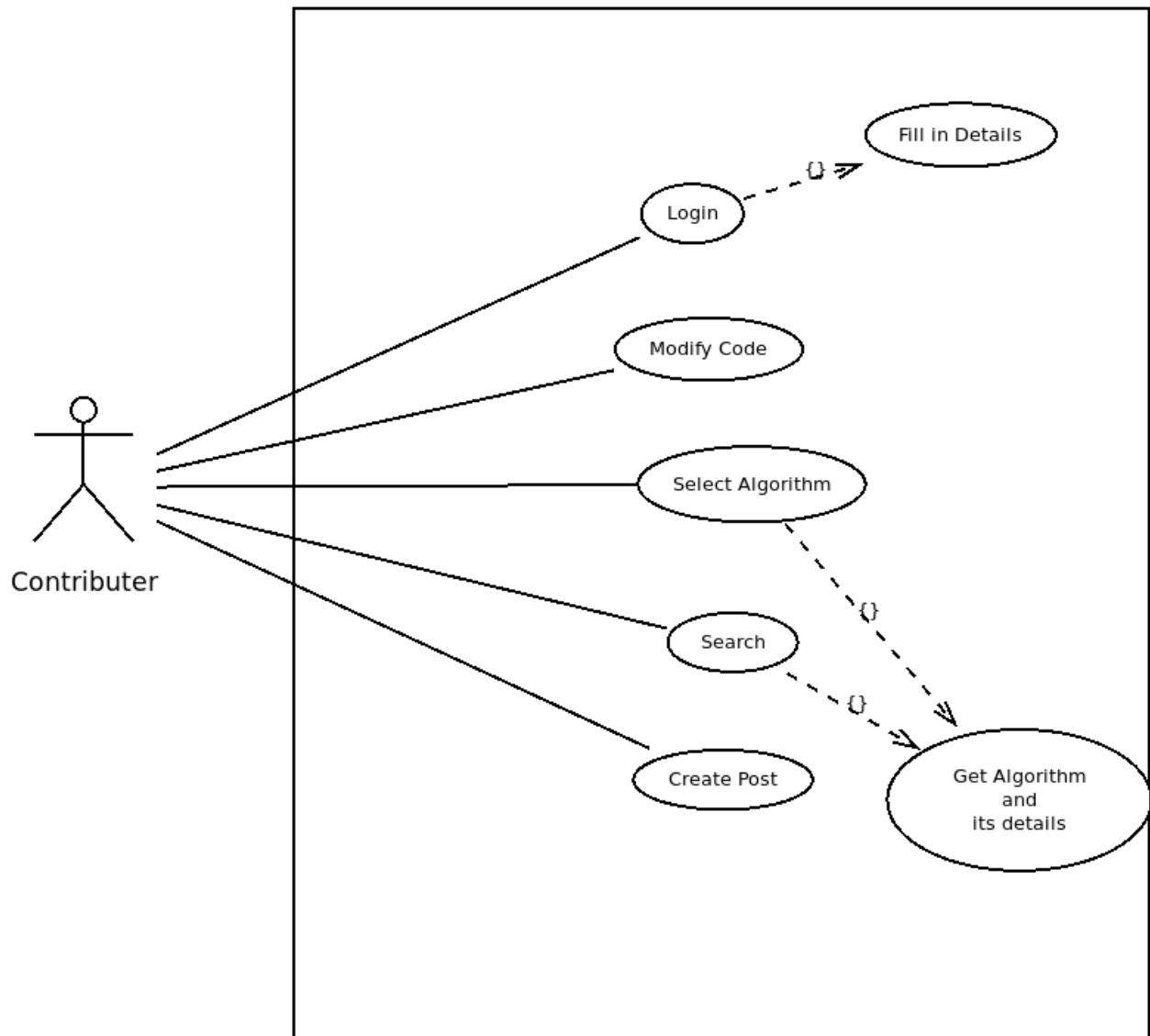
- Use cases: Horizontally shaped ovals that represent the different uses that a user might have.
- Actors: Stick figures that represent the people actually employing the use cases.
- Associations: A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
- System boundary boxes: A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.
- Packages: A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders



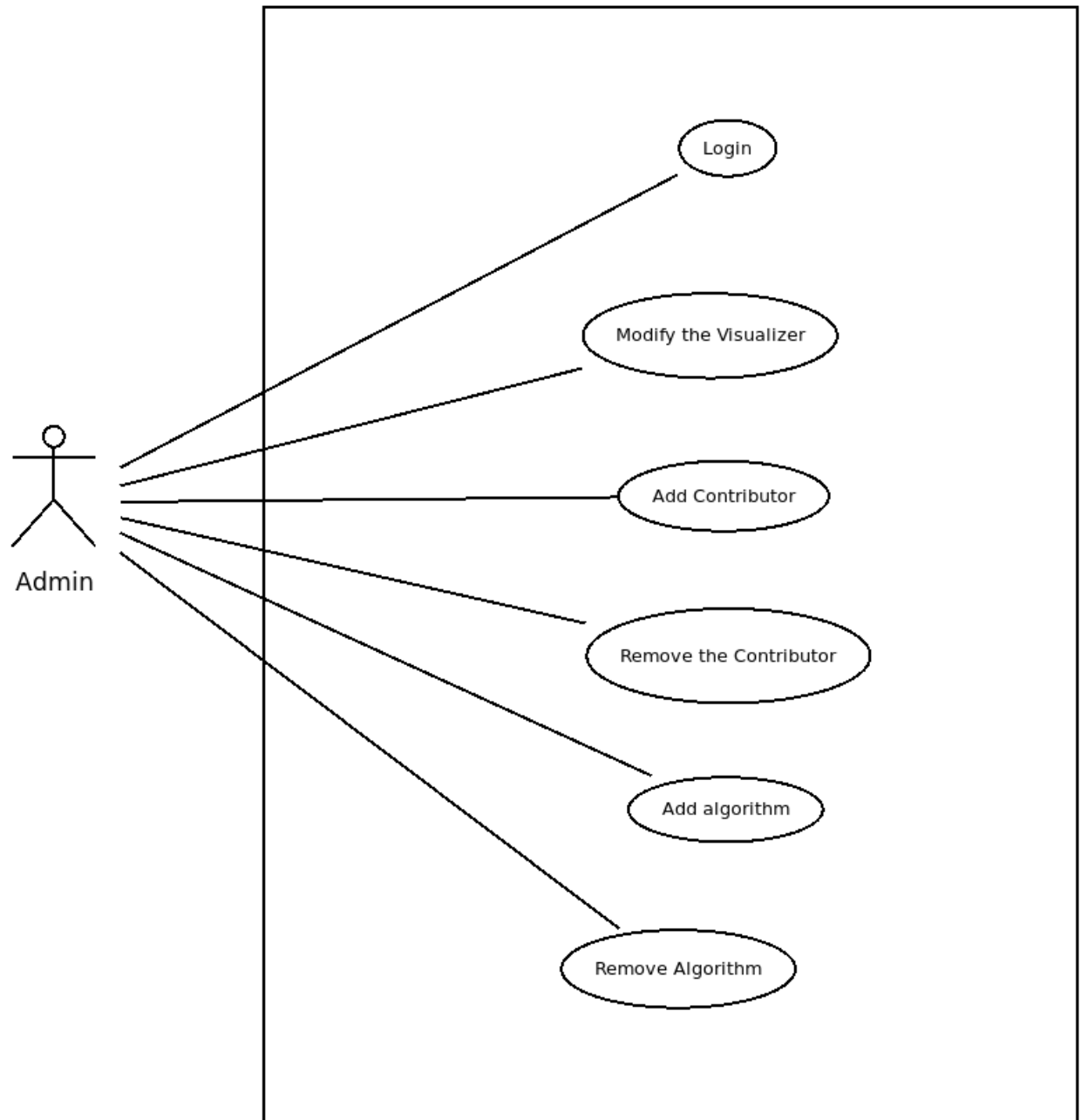
## Visitor USE-CASE



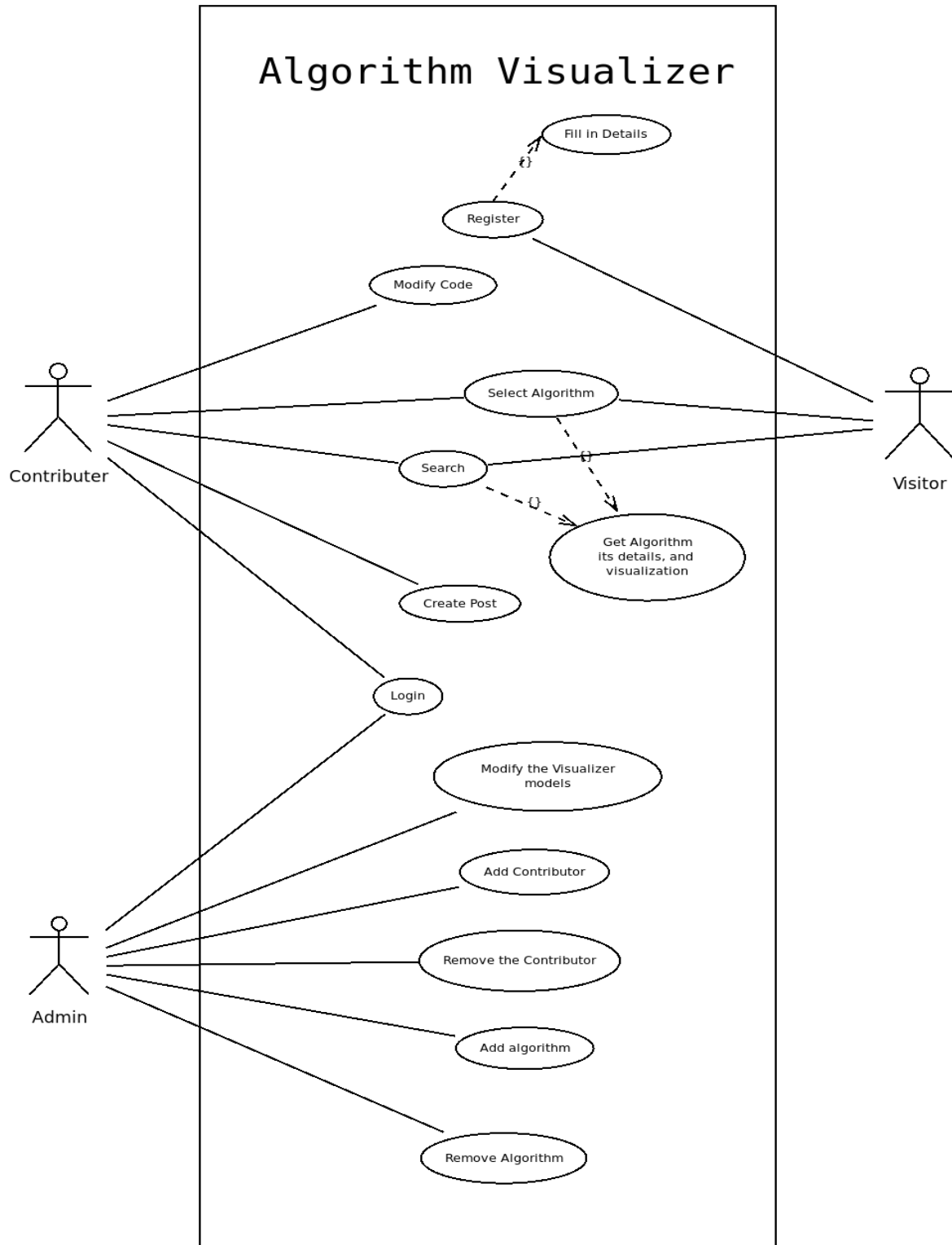
## Contributor USE-CASE



## Admin USE-CASE



## Algorithm Visualizer – USE CASE



### **A3 : ER Diagram**

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships. ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

Reasons for using the ER Diagram

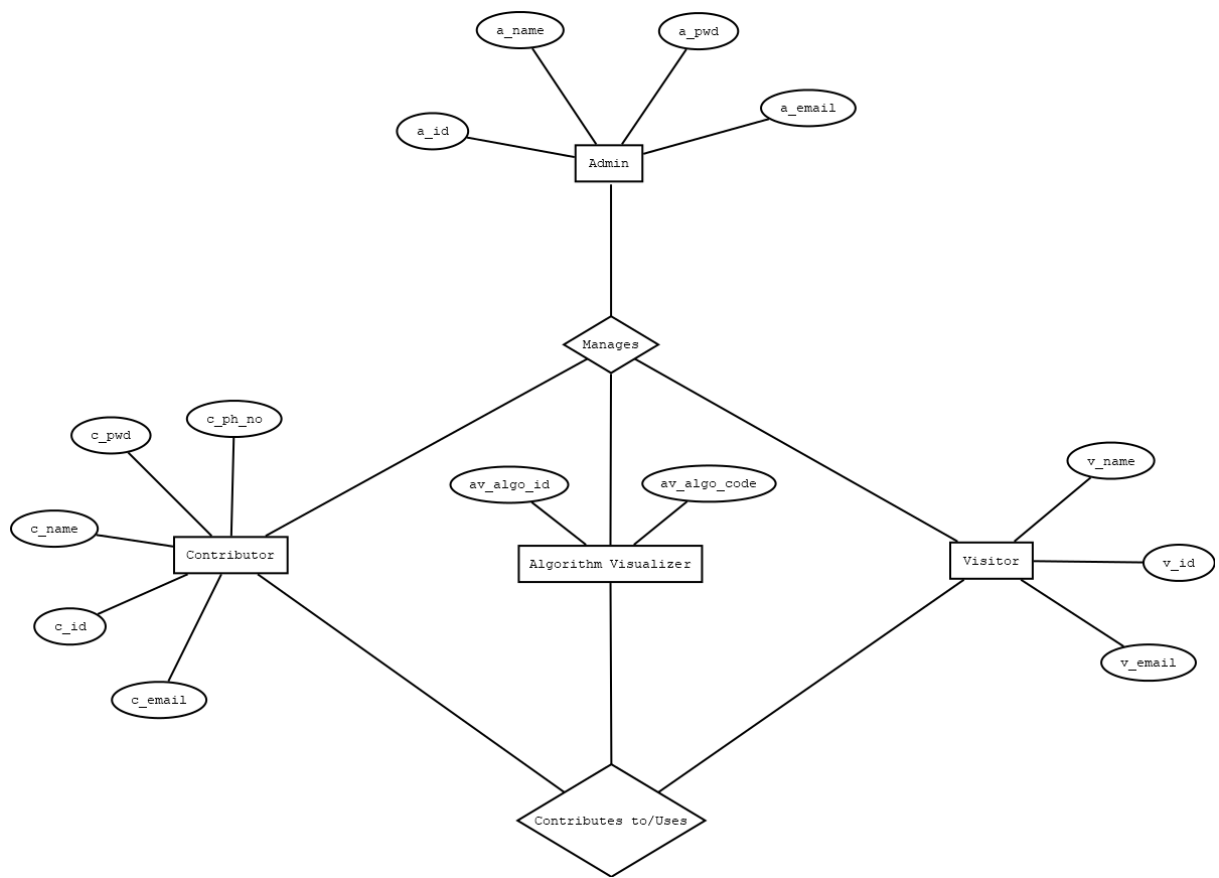
- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ERD Diagram allows you to communicate with the logical structure of the database to users

Entity Relationship Diagram Symbols & Notations mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes. There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

Following are the main components and its symbols in ER Diagrams:

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipse:** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes

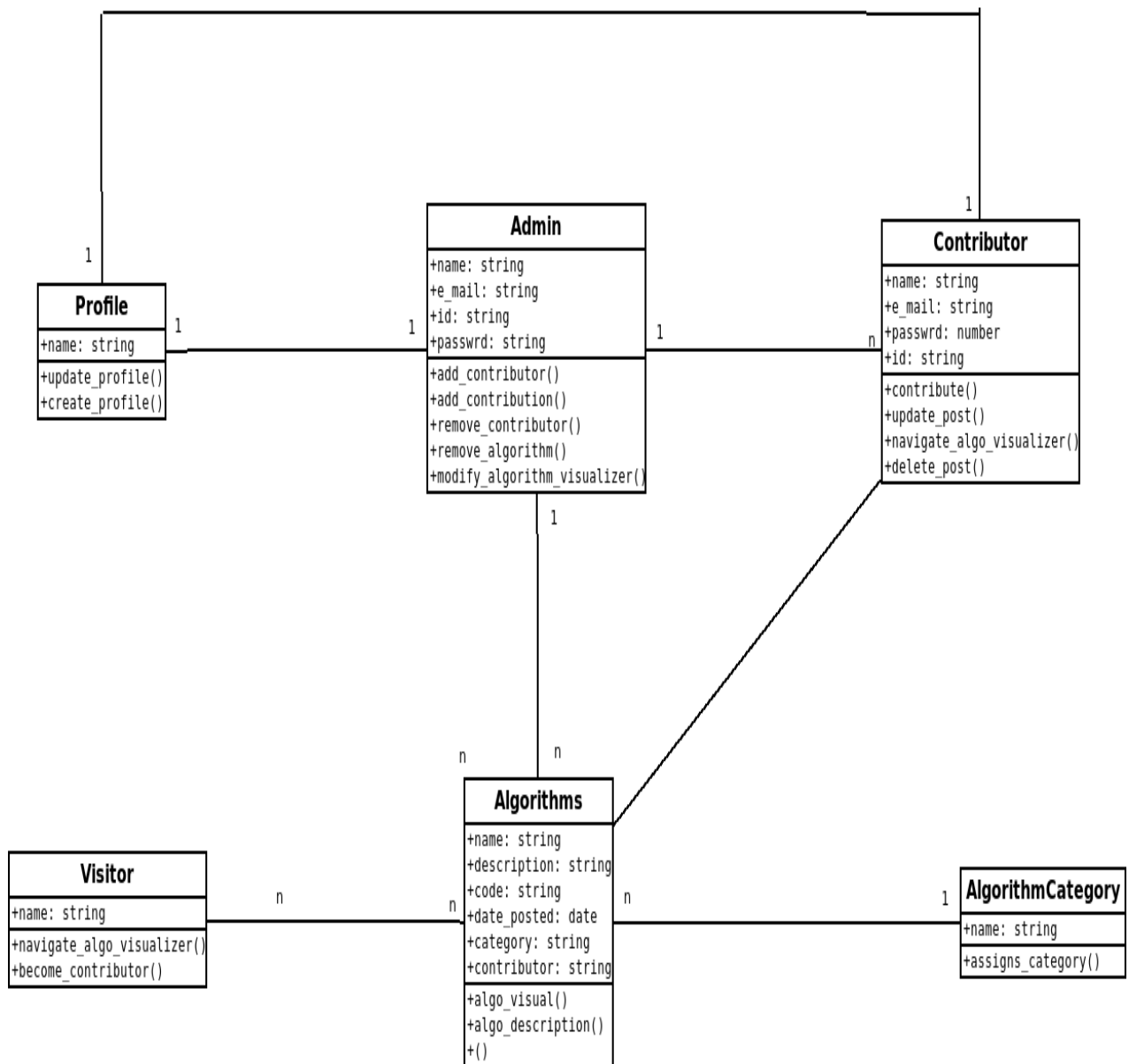




## A4 : Schema Diagrams

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.





## A5 : Activity Diagrams

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modeling how a collection of use cases coordinate to represent business workflows. Before you begin making an activity diagram, you should first understand its makeup. Some of the most common components of an activity diagram include:

**Action:** A step in the activity wherein the users or software perform a given task. In Lucid chart, actions are symbolized with round-edged rectangles.

**Decision node:** A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.

**Control flows:** Another name for the connectors that show the flow between steps in the diagram.

**Start node:** Symbolizes the beginning of the activity. The start node is represented by a black circle. **End node:** Represents the final step in the activity. The end node is represented by an outlined black circle.

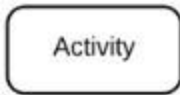
Activity diagrams present a number of benefits to users. We consider creating an activity diagram to:

- Demonstrate the logic of an algorithm.
- Describe the steps performed in a UML use case.
- Illustrate a business process or workflow between users and the system.
- Simplify and improve any process by clarifying complicated use cases.
- Model software architecture elements, such as method, function, and operation. Symbols, their Names and Description;



Start symbol

Represents the beginning of a process or workflow in an activity diagram. It can be used by itself or with a note symbol that explains the starting point.



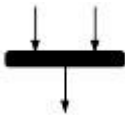
Activity symbol

Indicates the activities that make up a modeled process. These symbols, which include short descriptions within the shape, are the main building blocks of an activity diagram.



Connector symbol

Shows the directional flow, or control flow, of the activity. An incoming arrow starts a step of an activity; once the step is completed, the flow continues with the outgoing arrow.



Joint symbol/  
Synchronization bar

Combines two concurrent activities and re-introduces them to a flow where only one activity occurs at a time. Represented with a thick vertical or horizontal line.



Fork symbol

Splits a single activity flow into two concurrent activities. Symbolized with multiple arrowed lines from a join.



Decision symbol

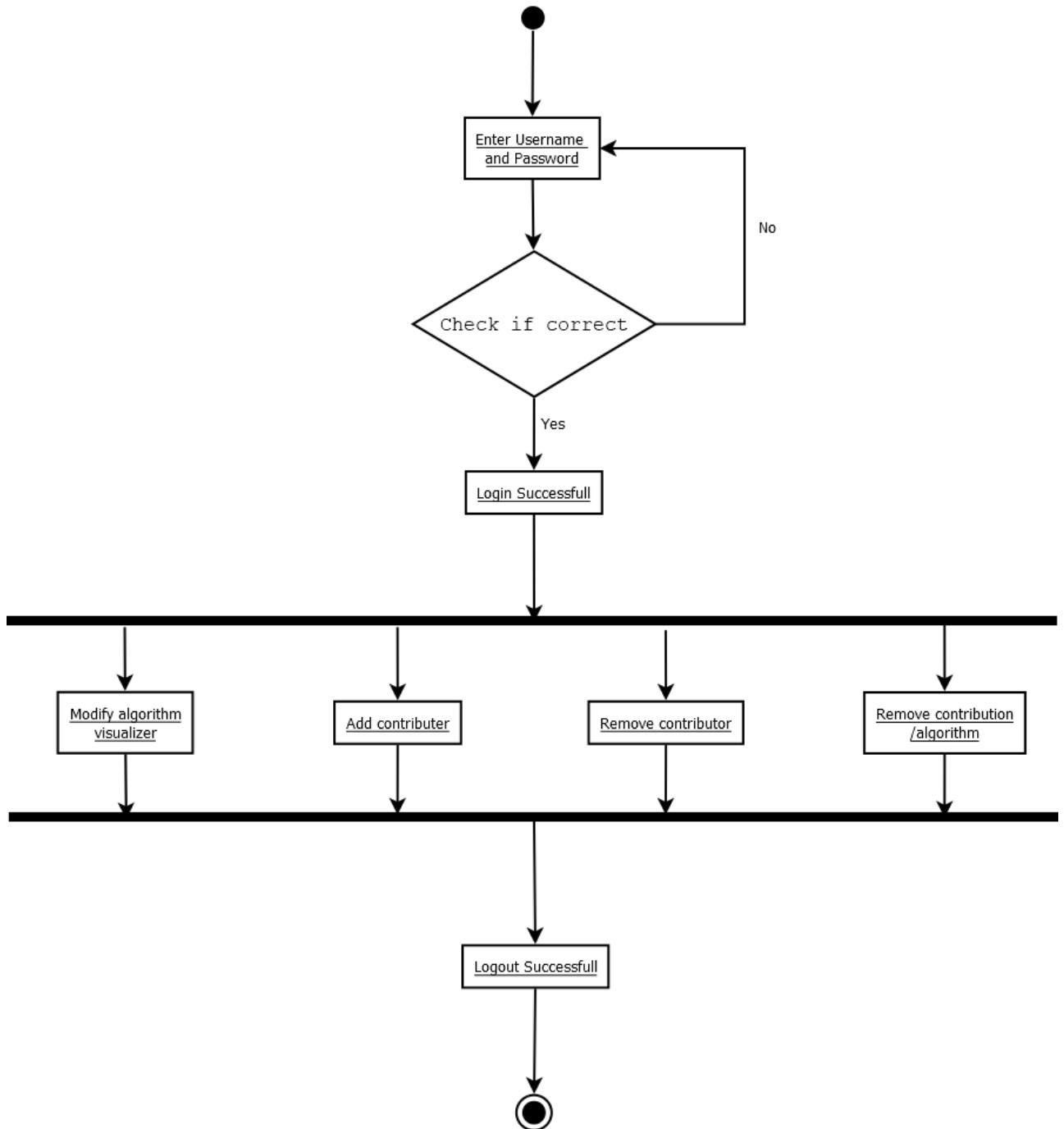
Represents a decision and always has at least two paths branching out with condition text to allow users to view options. This symbol represents the branching or merging of various flows with the symbol acting as a frame or container.



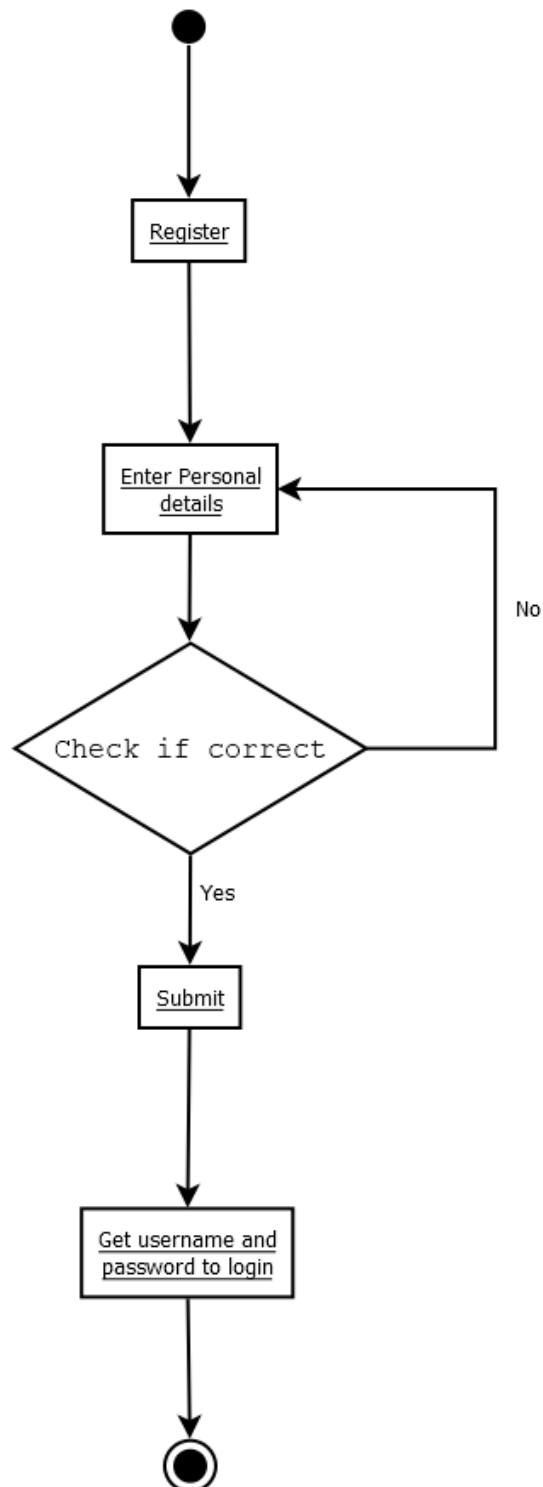
End symbol

Marks the end state of an activity and represents the completion of all flows of a process.

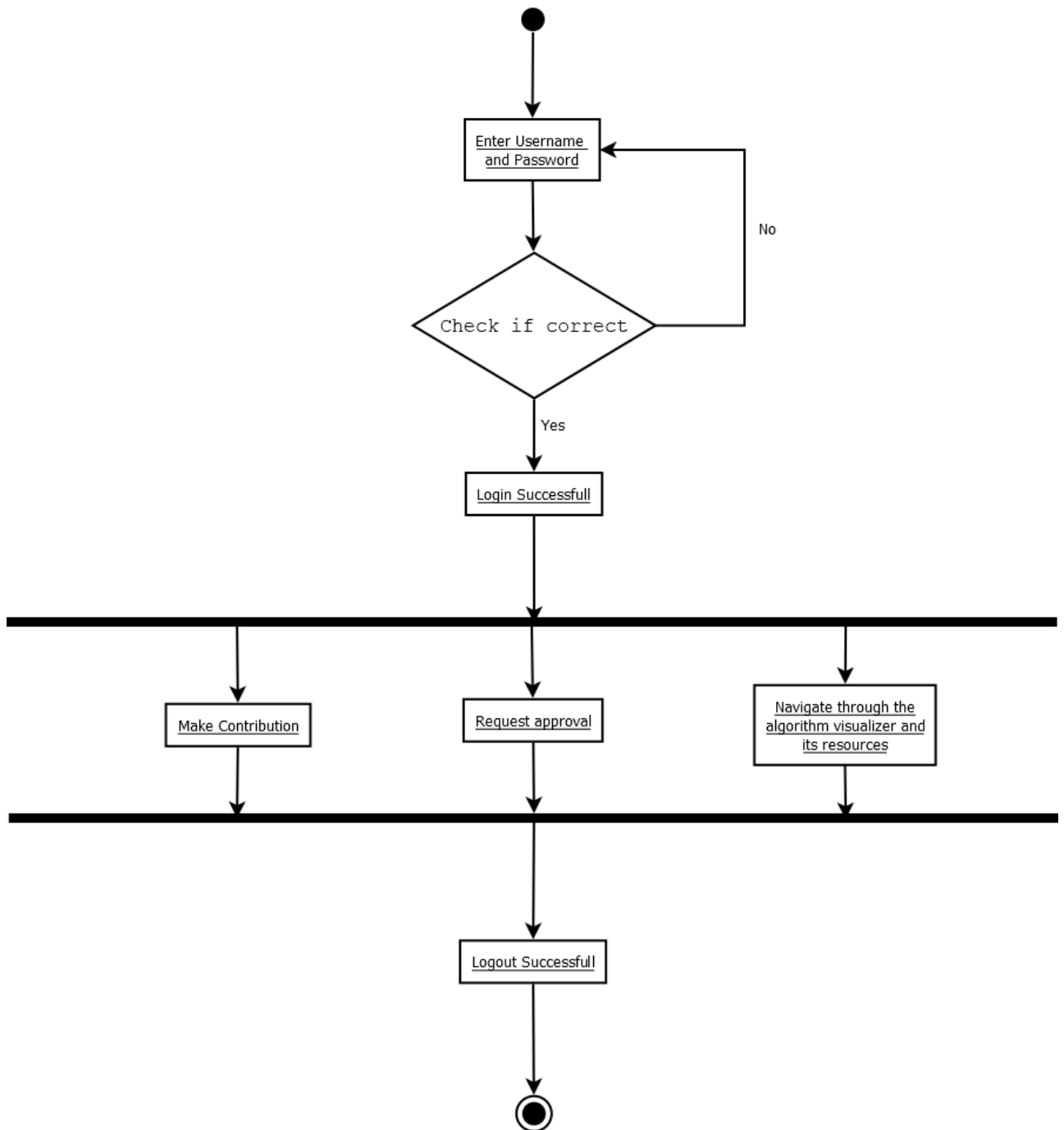
## Admin Activity Diagram



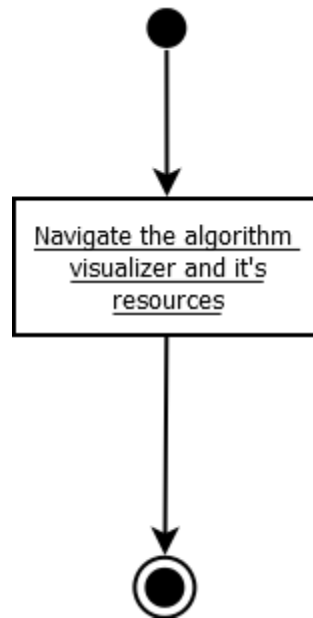
## Contributor Register Activity diagram



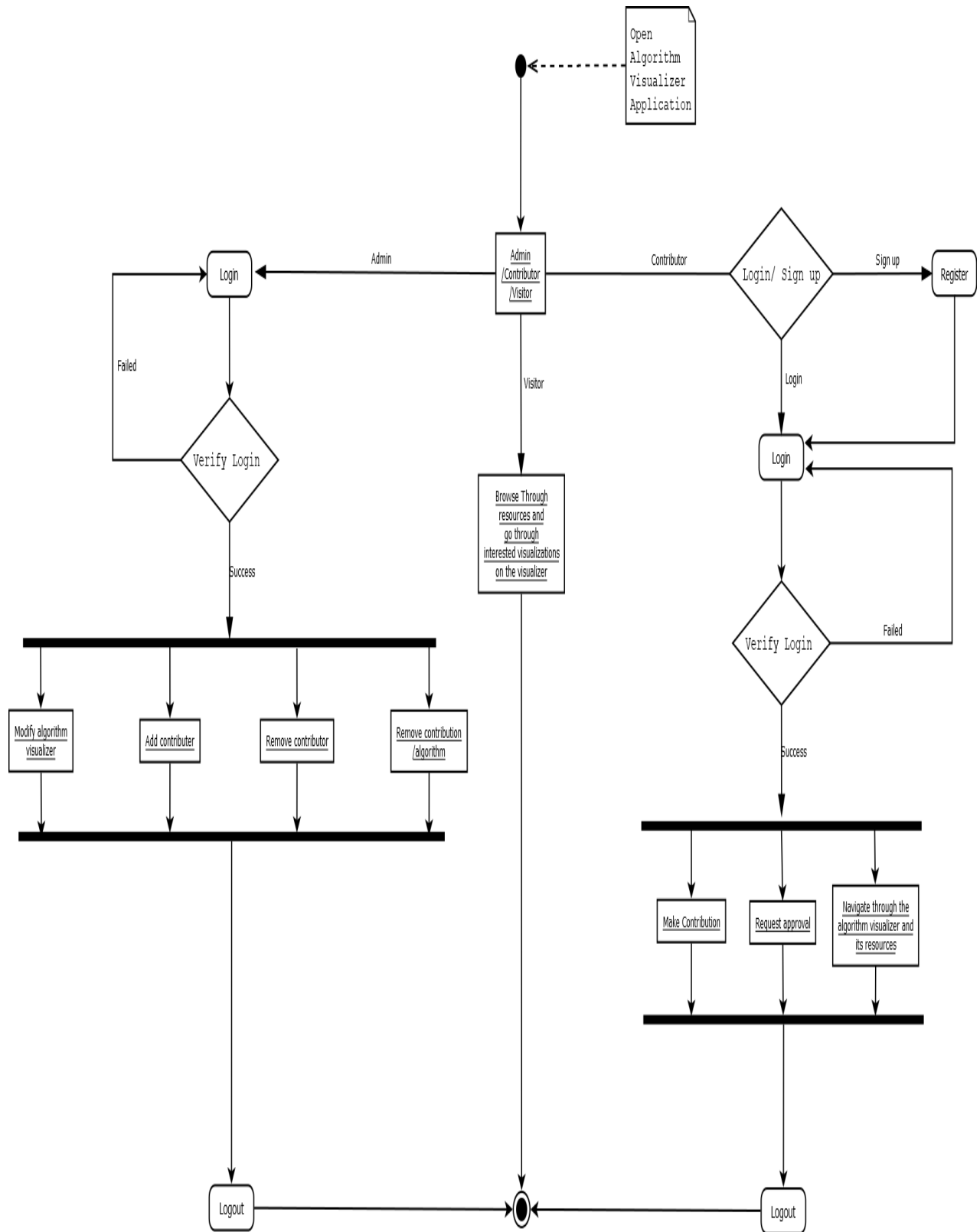
## Contributor Login Activity Diagram



## Visitor Activity Diagram



## Full Activity Diagram

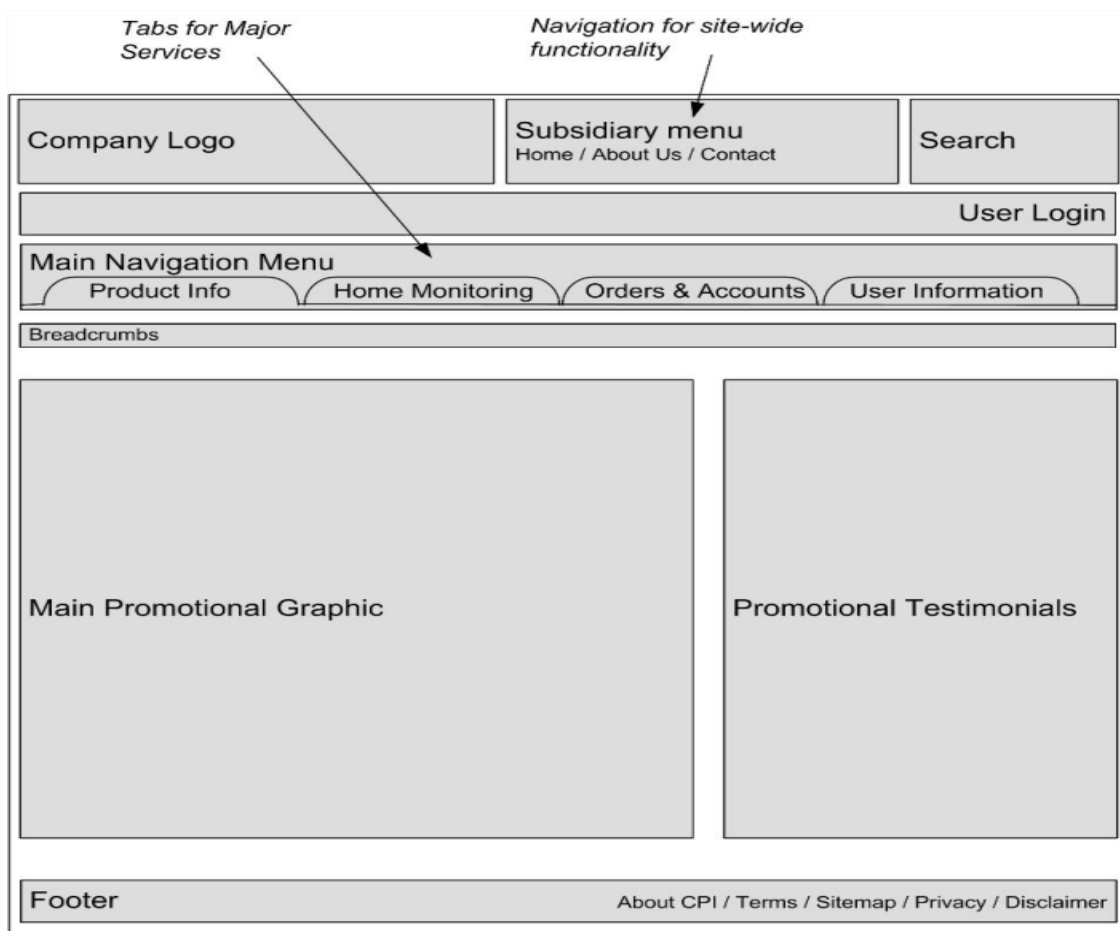




## A6 : Wireframe Diagrams

A wireframe model is a graphical model of how a website should look like before the coding process begins. Buttons, text and image placements are agreed upon before a single line of code is written. The wireframe also shows how the user will be able to navigate through the website as the links on the wireframe work. Wireframes can also be known as UX models/ UX designs. The common factor is that the design of the website/ software is agreed upon 100% before any code is written to prevent having to re-right code/ throw away code written due a change in specification.

- Conceptual layout of pages
- Captures core information and navigational elements.
- Supports both information design and interaction design.



## Main Interface

AV

AboutScopeDevelopersLogin/ Sign-up

Algorithms

### Algorithm Visualizer

### About

Images of  
the process of  
Development

### Scope

Developers Section

## The Visualizer



## The Editor



## Login



The login form is contained within a light gray rectangular box. At the top left of this box is the label "Login". Below the label are two horizontal input lines. Centered below the input lines is a rectangular button labeled "Register".

AV

About Scope Developers Login/ Sign-up

Login

Register

## Sign up



The sign up form is contained within a light gray rectangular box. At the top left of this box is the label "Sign up". Below the label are five horizontal input lines. There is no button visible on this form.

AV

About Scope Developers Login/ Sign-up

Sign up

## Algorithms Description pages

AV

AboutScopeDevelopersLogin/ Sign-up

Algorithm 1

AV

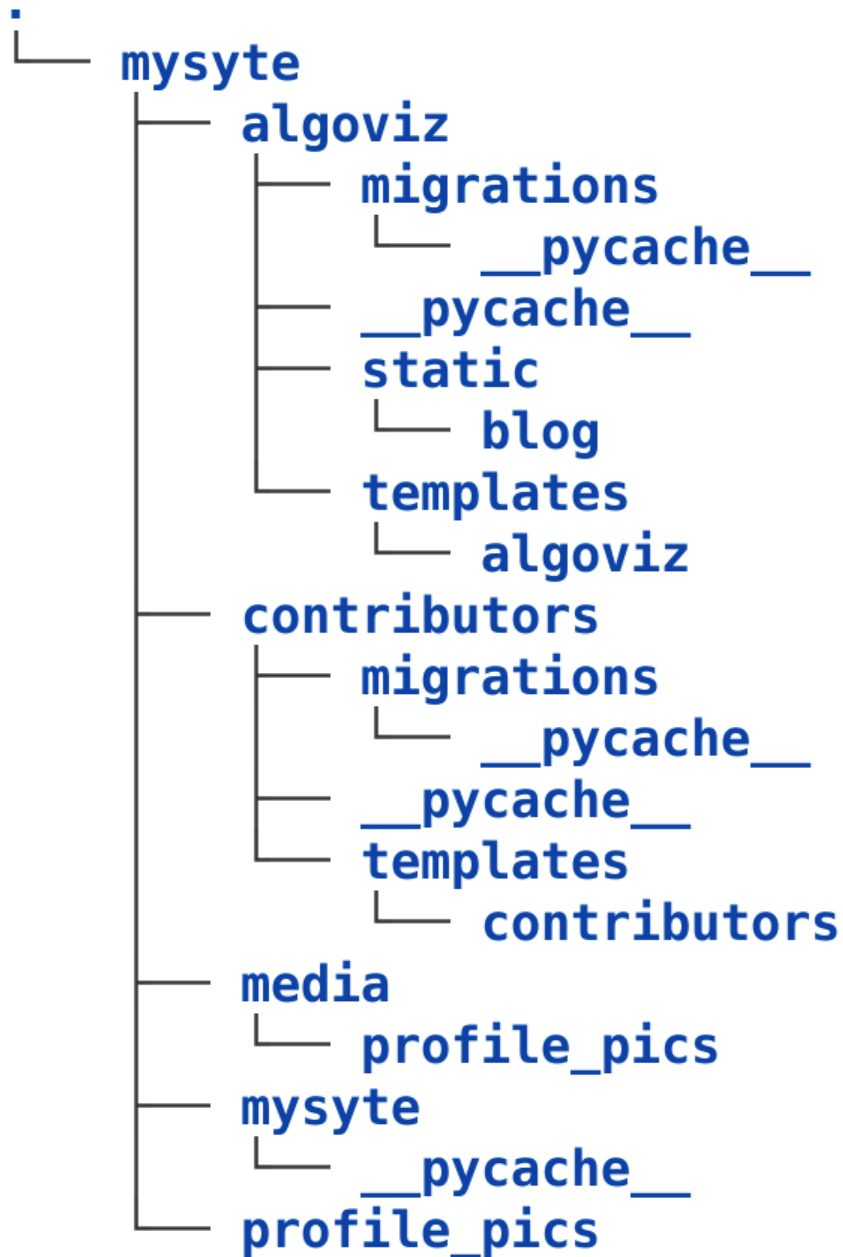
AboutScopeDevelopersLogin/ Sign-up

Algorithm 1



## A7 : Program listing-codes

# directory of project



20 directories

## # urls.py – algoviz

```
from django.urls import path
from .views import (
    AlgoListView,
    AlgoDetailView,
    AlgoCreateView,
    AlgoUpdateView,
    AlgoDeleteView,
)
from . import views

urlpatterns = [
    path('', views.home, name='algoviz-home'),
    path('about/', views.about, name='algoviz-about'),
    # path('terminal/', views.terminal, name='algoviz-terminal'),
    path('docs/', AlgoListView.as_view(), name='algoviz-docs'),
    path('algorithm/<int:pk>', AlgoDetailView.as_view(), name='algorithm-details'),
    path('algorithm/<int:pk>/update/', AlgoUpdateView.as_view(), name='algorithm-update'),
    path('algorithm/<int:pk>/delete/', AlgoDeleteView.as_view(), name='algorithm-delete'),
    path('algorithm/new/', AlgoCreateView.as_view(), name='algorithm-create'),
]
```

## # views.py - algoviz

```
from django.shortcuts import render
from .models import Algorithm
from django.contrib.auth.mixins import LoginRequiredMixin, UserPassesTestMixin
from django.views.generic import (
    ListView,
    DetailView,
    CreateView,
    UpdateView,
    DeleteView
)
from .models import Algorithm, AlgorithmCategory

def home(request):
    return render(request, 'algoviz/home.html')

def about(request):
    return render(request, 'algoviz/about.html')

# def terminal(request):
#     return render(request, 'algoviz/terminal.html')

def docs(request):
    context = {
        'algorithms': Algorithm.objects.all(),
        'algorithmcategory': AlgorithmCategory.objects.all(),
    }
    return render(request, 'algoviz/docs.html', context)
```

```

class AlgoListView(ListView):
    model = Algorithm
    template_name = 'algoviz/docs.html' # <app>/<model>_<viewtype>.html
    context_object_name = 'algorithms'
    ordering = ['-date_posted']

class AlgoDetailView(DetailView):
    model = Algorithm

class AlgoCreateView(LoginRequiredMixin, CreateView):
    model = Algorithm
    fields = ['name', 'description', 'code', 'category']

    def form_valid(self, form):
        form.instance.contributor = self.request.user
        return super().form_valid(form)

class AlgoUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
    model = Algorithm
    fields = ['name', 'description', 'code', 'category']

    def form_valid(self, form):
        form.instance.contributor = self.request.user
        return super().form_valid(form)

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.contributor:
            return True
        return False

class AlgoDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
    model = Algorithm
    success_url = '/'

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.contributor:
            return True
        return False

def about(request):
    return render(request, 'algoviz/about.html', {'title': 'About'})

```



## # models.py -algoviz

```
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User
from django.urls import reverse

class AlgorithmCategory(models.Model):
    name = models.CharField(max_length=50)

    def __str__(self):
        return self.name

class Algorithm(models.Model):
    name = models.CharField(max_length=50)
    description = models.TextField()
    code = models.TextField()
    date_posted = models.DateTimeField(default=timezone.now)
    category = models.ForeignKey(AlgorithmCategory, on_delete=models.CASCADE)
    contributor = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('algorithm-details', kwargs={'pk': self.pk})
```

## # models.py – contributors

```
from django.db import models
from django.contrib.auth.models import User
from PIL import Image

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    image = models.ImageField(default='default.jpg', upload_to='profile_pics')

    def __str__(self):
        return f'{self.user.username} Profile'

    def save(self, *args, **kwargs):
        super().save(*args, **kwargs)

        img = Image.open(self.image.path)

        if img.height > 300 or img.width > 300:
            output_size = (300, 300)
            img.thumbnail(output_size)
            img.save(self.image.path)
```

## # views.py – contributors

```
from django.shortcuts import render, redirect
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from .forms import UserRegisterForm, UserUpdateForm, ProfileUpdateForm
from algoviz.models import Algorithm

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            # username = form.cleaned_data.get('username')
            messages.success(request, f'Your account has been created! You are now able to log in')
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'contributors/register.html', {'form': form})
```

```
@login_required
def profile(request):
    if request.method == 'POST':
        u_form = UserUpdateForm(request.POST, instance=request.user)
        p_form = ProfileUpdateForm(request.POST,
                                   request.FILES,
                                   instance=request.user.profile)

        if u_form.is_valid() and p_form.is_valid():
            u_form.save()
            p_form.save()
            messages.success(request, f'Your account has been updated!')
            return redirect('profile')

    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form = ProfileUpdateForm(instance=request.user.profile)

    context = {
        'u_form': u_form,
        'p_form': p_form,
        'algorithms': Algorithm.objects.all()
    }

    return render(request, 'contributors/profile.html', context)
```

## # forms.py - contributors

```
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from .models import Profile

class UserRegisterForm(UserCreationForm):
    email = forms.EmailField()

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']

class UserUpdateForm(forms.ModelForm):
    email = forms.EmailField()

    class Meta:
        model = User
        fields = ['username', 'email']

class ProfileUpdateForm(forms.ModelForm):
    class Meta:
        model = Profile
        fields = ['image']
```

## # signals.py – contributors

```
from django.db.models.signals import post_save
from django.contrib.auth.models import User
from django.dispatch import receiver
from .models import Profile

@receiver(post_save, sender=User)
def create_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=User.objects.get(username=instance))

@receiver(post_save, sender=User)
def save_profile(sender, instance, **kwargs):
    instance.profile.save()
```

## # urls.py – mysyte

```
from django.contrib import admin
from django.contrib.auth import views as auth_views
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from contributors import views as user_views
from django.views.generic import RedirectView

urlpatterns = [
    path('admin/', admin.site.urls),
    path('myapp/',
         RedirectView.as_view(url='http://localhost:5000/visualize.html#mode=edit')),
    path('register/', user_views.register, name='register'),
    path('profile/', user_views.profile, name='profile'),
    path('login/',
         auth_views.LoginView.as_view(template_name='contributors/login.html'),
         name='login'),
    path('logout/',
         auth_views.LogoutView.as_view(template_name='contributors/logout.html'),
         name='logout'),
    path('', include('algoviz.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## # installed apps in settings.py file

```
# Application definition

INSTALLED_APPS = [
    'algoviz.apps.AlgovizConfig',
    'contributors.apps.ContributorsConfig',
    'crispy_forms',
    'crispy_bootstrap4',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

## # configuration of media and static files

```
# Application definition
```

```
INSTALLED_APPS = [  
    'algoviz.apps.AlgovizConfig',  
    'contributors.apps.ContributorsConfig',  
    'crispy_forms',  
    'crispy_bootstrap4',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

```
from django.conf import settings  
from django.db import migrations, models  
import django.db.models.deletion  
import django.utils.timezone
```

```
class Migration(migrations.Migration):
```

```
    initial = True
```

```
    dependencies = [  
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),  
    ]
```

```
    operations = [  
        migrations.CreateModel(  
            name='AlgorithmCategory',  
            fields=[  
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),  
                ('name', models.CharField(max_length=50)),  
            ],  
        ),  
    ]
```

```
from django.conf import settings  
from django.db import migrations, models  
import django.db.models.deletion  
import django.utils.timezone
```

```
class Migration(migrations.Migration):
```

```
    initial = True
```

```
    dependencies = [  
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),  
    ]
```

## A8 : Input (Test) data/screen ad Sample output/ Reports/Screens

### # Home Page

Algorithm Visualizer


Home

About

Terminal

Algorithms

User



**Learn The Easy Way**

Go Straight to the Terminal

An interactive code visualizing application equipped with loads of algorithms.

ALGORITHM VISUALIZER

Add your code snippet here

```
while code in Python 3.8:
    # Add algorithm here...
    for i in range(len(arr)):
        if arr[i] > arr[i+1]:
            swap(&arr[i], &arr[i+1])
```

ALGORITHM VISUALIZER

Add your code snippet here

```
Python 3.8
# Add algorithm here...
for i in range(len(arr)):
    # Add algorithm here...
```

Algorithm Visualizer

Home

About

Terminal

Algorithms

harismoin

```
A = 1;
WHILE(A < 5)
    DO Print A;
    A = A + 1;
ENDWHILE;
```

Pseudo Codes could be tough



Might get lost in Flow Charts

Contact the Developer

Email:

haris88@gmail.com

Phone:

+91 7865432672

Github Repository:




AlgoViz © 2023 All Rights Reserved.

## # About page

Algorithm Visualizer

Home About Terminal Algorithms

 User ▾

# About Us

Algorithm Visualizer is an interactive web-based platform enriched with lots of algorithms and code snippets contributed by its users which range in a categories such as Greedy algorithms, Divide and Conquer algorithms and many more. This platform serves its users with the hands-on visual representation of these algorithms through the means of a javascript library named visualize.bundle.js which is most famously used for building flow charts and roadmaps by multiple websites also making there contribution to this cause of learning through the means of graphical representation as *"a picture speaks more than a thousand words"*.

Become a registered user & [Contribute](#)

Sign Up

# # Terminal

Algorithm Visualizer

HomeAboutTerminalAlgorithms

User

ALGORITHM VISUALIZER

Add your code snippet here

Write code in Python 3.6

```
1 def search(arr, x):
2     for i in range(len(arr)):
3         if arr[i] == x:
4             return i
5     return -1
6
7 arr = [1,2,3,6,2,7,4]
8 x = 6
9 print('The element ', x, 'is present at:', search(arr,x), ' position')
```

Visualize Execution

# Output

Algorithm Visualizer

HomeAboutTerminalAlgorithms

User

ALGORITHM VISUALIZER

Add your code snippet here

Python 3.6

```
1 def search(arr, x):
2     for i in range(len(arr)):
3         if arr[i] == x:
4             return i
5     return -1
6
7 arr = [1,2,3,6,2,7,4]
8 x = 6
9 print('The element ', x, 'is present at:', search(arr,x))
```

→ line that has just executed  
→ next line to execute

Click a line of code to set a breakpoint; use the Back and Forward buttons to jump there.

<< First < Back Program terminated Forward > Last >>

Print output (drag lower right corner to resize)

The element 6 is present at: 3 position

Frames

Objects

Global frame

search imported object

arr

x 6

list

0 1 2 3 4 5 6


1 2 3 6 2 7 4



## # Algorithm Docs

Algorithm Visualizer

HomeAboutTerminalAlgorithms

 harismoin

Search by category:

All algorithms

Search

April 14, 2023

### Tower of Hanoi

Tower of Hanoi is a mathematical puzzle where we have three rods (A, B, and C) and N disks. Initially, all the disks are stacked in decreasing value of diameter i.e., the smallest disk is placed on the top and they are on rod A. The objective of the puzzle is to move the entire stack to another rod (here considered C), obeying the following simple rules:

- Only one disk can be moved at a time.
- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
- No disk may be placed on top of a smaller disk.

Worst Case Time Complexity:  $O(2^N)$   
Best Case Time Complexity:  $O(2^N)$

April 14, 2023


### Selection Sort

Selection sort is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list. The algorithm repeatedly selects the smallest (or largest) element from the unsorted portion of the list and swaps it with the first element of the unsorted portion. This process is repeated for the remaining unsorted portion of the list until the entire list is sorted.

## # Algorithm Description

Algorithm Visualizer

HomeAboutTerminalAlgorithms

 harismoin

April 14, 2023

UpdateDelete

### Binary Search

Binary Search is a searching algorithm used in a sorted array by repeatedly dividing the search interval in half.

Worst-case time complexity:  $O(\log n)$   
Best-case time complexity:  $O(1)$

```
def binarySearch(arr, l, r, x):  
  
    if r >= l:  
  
        mid = l + (r - l) // 2  
  
        if arr[mid] == x:  
            return mid  
  
        elif arr[mid] > x:  
            return binarySearch(arr, l, mid-1, x)
```

Visualize


## # User Profile

Algorithm Visualizer

HomeAboutTerminalAlgorithms

harismoin

Dashboard  
Contribute  
Profile  
Log Out



harismoin

harismoin88@gmail.com

Profile Info

Username\*

harismoin

Required: 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

Email\*

harismoin88@gmail.com

Image\*

Currently: [profile\\_pics/i3232937\\_1073987026026779\\_3685797144781879523\\_n\\_zWr5TQA.jpg](#)

Change:

Choose File

No file chosen

Update

## # Admin Dashboard

Django administration

WELCOME, HARIS. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

ALGOVIZ

Algorithm categorys

+ Add

Change

Algorithms

+ Add

Change

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

Change

Users

+ Add

Change

CONTRIBUTORS

Profiles

+ Add

Change

Recent actions

My actions

✖ Merge Sort2

Algorithm

✖ Merge Sort2

Algorithm

+ Others

Algorithm category

✖ testuser7

User

✖ testuser6

User

✖ testuser5

User

✖ testuser4

User

+ iharismoin Profile

Profile

✖ testuser9

User


✖ testuser7

User

## # Login page

Algorithm Visualizer

HomeAboutTerminalAlgorithms

 User ▾

Login

Username\*

Password\*


Login Now

Already Have An Account? [Sign Up](#)

## # Register as contributor

Algorithm Visualizer

HomeAboutTerminalAlgorithms

 User ▾

Become a Contributor

Username\*

Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

Email\*

Password\*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation\*

Enter the same password as before, for verification.


Sign Up

Already Have An Account? [Sign In](#)

## # Post Algorithm

Algorithm Visualizer

Home About Terminal Algorithms

 harismoin ▾

Algorithm Post

Name\*

Description\*

Code\*

Category\*

----- ▾

Contribute

Guidelines

Follow the given rules in order to contribute:

- Choose a descriptive name

- Write a brief overview of your code. It is recommended to provide the complexities with them as well

- Select the category under which your code might reside


- Provide the code with proper indentations and correct syntax. If any of these are not fulfilled, you might need to update the post.

- Make sure to keep your code compact.

## # Logout

Algorithm Visualizer

Home About Terminal Algorithms

 User ▾

You have been logged out

[Log In Again](#)

