VIRTUAL ELECTRONICS LAB

Project Report on Virtual Electronics Lab

SUBMITTED BY

Nikhil Kumar - 21001570056
Vipin Kumar - 21001570094
Priyanshu - 21001570065

Under the guidance of
Dr. Vibha Gaur, Professor

2023-2024

Department of Computer Science

ACHARYA NARENDRA DEV COLLEGE

University of Delhi, Delhi

ACKNOWLEDGEMENT

On the successful completion of our project “**VIRTUAL ELECTRONIC LAB”**, we extend our deepest appreciation and gratitude to our esteemed guide, Mrs. Vibha Gaur, for her invaluable support, guidance, and mentorship throughout the duration of our software engineering project. Her expertise and insights have been instrumental in shaping the direction of our project, and her unwavering commitment to excellence has challenged us to perform at our best.

We would also like to express our heartfelt thanks to the faculty members of the Department of Computer Science for their encouragement and support. Their invaluable feedback, guidance, and suggestions have played a significant role in helping us improve the quality of our work.

We are deeply grateful for the opportunities provided to us, and we recognize that our success would not have been possible without the support and contributions of these individuals and institutions.-110019

Priyanshu Nikhil Kumar Vipin Kumar

ACHARYA NARENDRA DEV COLLEGE
(University of Delhi)

CERTIFICATE

This is to certify that **Priyanshu, Nikhil Kumar** and **Vipin Kumar** of Bachelor of Science (Hons) Computer Science from Acharya Narendra Dev College, University of Delhi, have successfully completed the project **"Virtual Electronics Lab"** during Semester IV under the supervision of **Prof. Vibha Gaur.**

Priyanshu	Nikhil Kumar	Vipin Kumar

Supervisor
Dr. Vibha Gaur, Professor
Department of Computer Science

Contents

Chapter-1

PROBLEM STATEMENT

Traditional physics labs can be expensive to set up and maintain, and may not offer enough time or resources to allow students to fully understand physics concepts. virtual electronics labs are an alternative that can be accessed from anywhere with an internet connection, and do not require expensive equipment or maintenance. Virtual labs can offer a wider variety of experiments that can better explore physics concepts.

However, there are challenges with virtual labs. The simulations need to accurately represent real-world scenarios, so that students can make correct conclusions. Some students may not have access to high-speed internet or advanced technology.

To overcome these challenges, we need to improve the accuracy of simulations and develop new technology to enhance the virtual lab experience. We also need to ensure that all students, regardless of their financial or geographic status, have equal access to virtual labs. By doing so, virtual electronics labs can be a powerful tool for improving the quality and accessibility of physics education.

In summary, virtual electronics labs offer a promising alternative to traditional labs. However, to make virtual labs effective in educating students, we must overcome the challenges they face. By improving the accuracy of simulations and ensuring equal access to virtual labs, we can make physics education more accessible and effective for all students.

- Limited access to physical labs
- Cost of maintaining physical labs
- Safety hazards associated with handling circuits
- Limited availability of lab equipment
- Need for high-quality lab simulations
- Need for flexible lab schedules

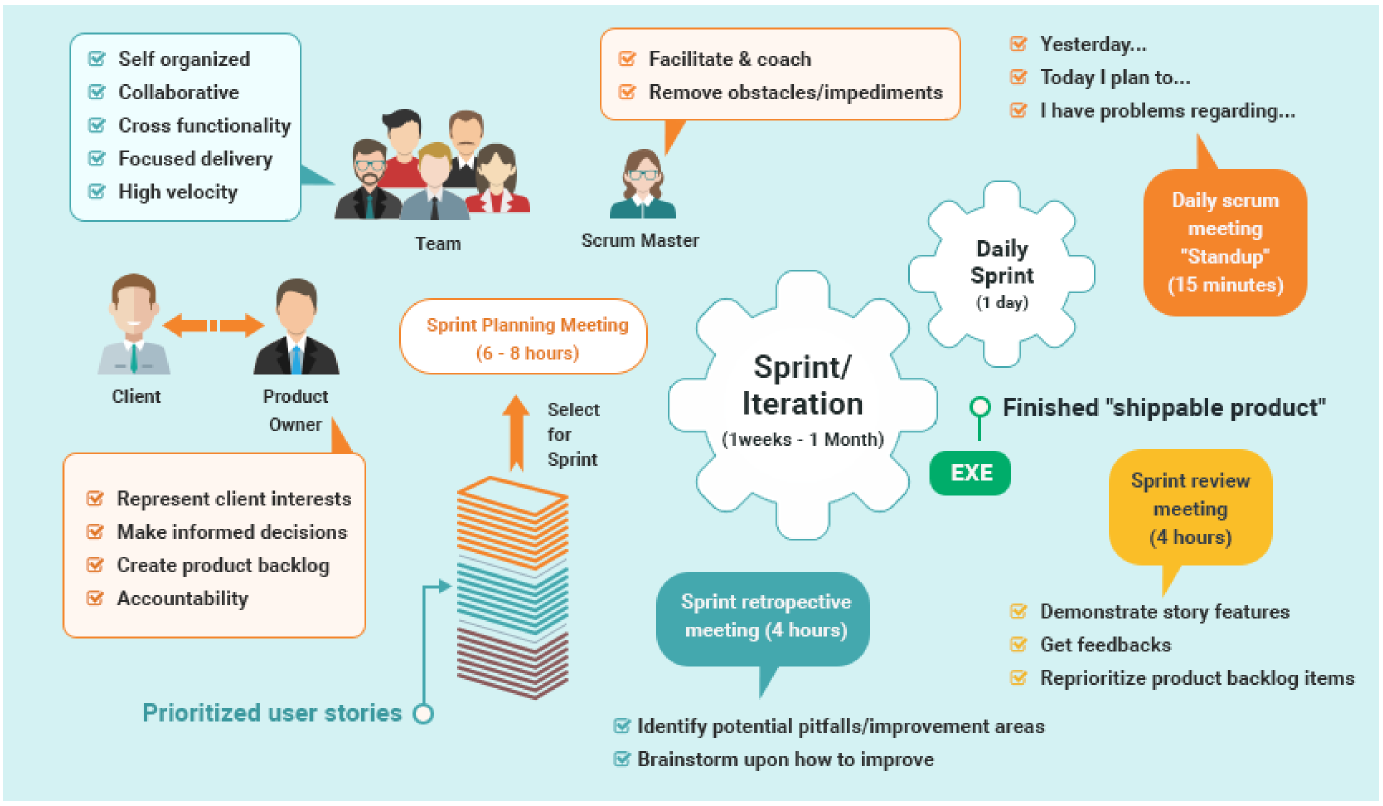
CHAPTER 02 - PROCESS MODEL

The best software process model for virtual electronics lab would be the agile development process model. The agile model is a flexible and iterative approach to software development that emphasizes collaboration, adaptability, and continuous improvement.

In the case of virtual electronics labs, the agile model would be particularly beneficial because it allows for frequent updates and improvements to the simulations, which can be based on feedback from students and teachers. This iterative approach would help ensure that the simulations are accurate and representative of real-world scenarios.

Additionally, the agile model would allow for a more collaborative approach to software development, which is essential in the case of virtual electronics labs. Collaboration between developers, educators, and students would help ensure that the simulations meet the needs of all students.

Overall, the agile model is the best software process model for virtual electronics labs because it provides a flexible and collaborative approach to software development that emphasizes continuous improvement and the ability to adapt to changing needs and feedback.


Self organized
Collaborative
Cross functionality
Focused delivery
High velocity
Team
Facilitate & coach
Remove obstacles/impediments
Scrum Master
Yesterday...
Today I plan to...
I have problems regarding...
Daily scrum meeting "Standup" (15 minutes)
Daily Sprint (1 day)
Client
Product Owner
Sprint Planning Meeting (6 - 8 hours)
Select for Sprint
Sprint/ Iteration (1weeks - 1 Month)
Finished "shippable product"
EXE
Represent client interests
Make informed decisions
Create product backlog
Accountability
Sprint review meeting (4 hours)
Sprint retropective meeting (4 hours)
Demonstrate story features
Get feedbacks
Reprioritize product backlog items
Prioritized user stories
Identify potential pitfalls/improvement areas
Brainstorm upon how to improve

CHAPTER 03 - REQUIREMENT ANALYSIS & MODELING

Software Requirement Analysis is the process of understanding and defining the needs and specifications of software that needs to be developed. It involves gathering and analyzing information about the user's requirements, constraints, and expectations for the software.

The goal of software requirement analysis is to create a clear and comprehensive set of requirements that will guide the development of the software. This involves breaking down the project into smaller, manageable pieces and defining the requirements for each piece. It also involves prioritizing the requirements based on their importance to the stakeholders and the business.

During the software requirement analysis process, the team may use a variety of tools and techniques to gather information and document the requirements. This may include interviews with stakeholders, surveys, use case diagrams, flowcharts, and other types of documentation.

Once the requirements have been identified and documented, they are typically reviewed and approved by stakeholders before the development process begins. This helps to ensure that everyone is on the same page and that the software will meet the needs and expectations of the stakeholders.

3.1 Data Flow Diagrams

It is a a visual representation of the system's data flow and helps to identify the sources of data, the processing steps, and the outputs generated by the system.

- Level 0 DFD
- Level 1 DFD

Level 0 DFD :- It is the simplest form of DFD which shows the system as a single process or function with inputs or outputs. The inputs and outputs are external to the system and are represented by rectangles called external entities. The process or function is represented by a circle.

The arrows in the level 0 DFD represent the flow of data between the external entities and the process or function.

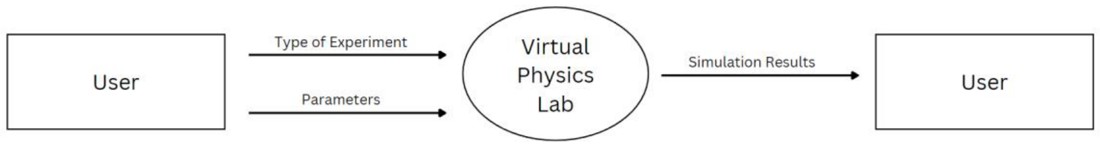


Fig 3.1

Level 1 DFD :- It is a more detailed version of level 0 DFD. It decomposes the process or function in the level 0 DFD into smaller processes and shows how they are interconnected.

Processes are represented by circles eg simulator, external entities are represented by rectangles eg user, data stores are represented by two parallel lines eg simulated circuit and data flows are represented by arrows eg displays circuit interface.

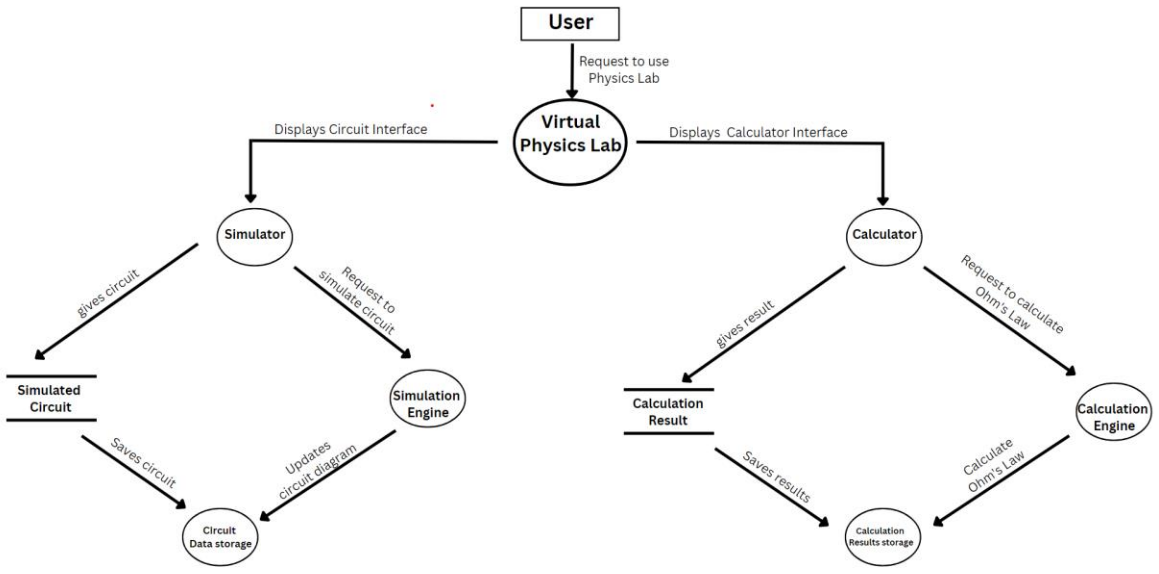

User
Request to use Physics Lab
Virtual Physics Lab
Displays Circuit Interface
Displays Calculator Interface
Simulator
Calculator
gives circuit
Request to simulate circuit
gives result
Request to calculate Ohm's Law
Simulated Circuit
Simulation Engine
Calculation Result
Calculation Engine
Saves circuit
Updates circuit diagram
Saves results
Calculate Ohm's Law
Circuit Data storage
Calculation Results storage

Fig 3.2

3.2 Data Dictionary

A data dictionary is a document or a repository that contains a description of the data objects used in a particular software application or system. It provides detailed information about each data element such as its name, description, data type, size, format, and relationship with other data elements. The data dictionary serves as a reference for developers, analysts, and end-users to understand the data used in the system and its characteristics. It helps in ensuring consistency and accuracy of the data and facilitates maintenance and modification of the system over time.

Field Name	**Data Type**	**Description**
User Id	Numeric	The unique identifier assigned to each user
Username	String	The username chosen by the user during registration
Password	String	The password chosen by the user during registration
Email	String	The email provided by the user during registration
Circuit Description	String	A brief description of the circuit
Circuit Diagram	Image	An image representing the circuit diagram
Components	String	A list of the components used in the circuit
Results	String	The simulation results for the circuit
Component Name	String	The name of the component
Component Description	String	A brief description of the component
Component Image	Image	An image representing the component
Component Type	String	The category of the component (eg. resistor, capacitor, transistor etc,)

3.3. Use Case Diagrams

A use case diagram is a graphical representation of the different ways in which users interact with a system or software. It shows the relationships between users (actors) and the system, and how they interact with each other. The diagram typically consists of actors, use cases, and the relationships between them.

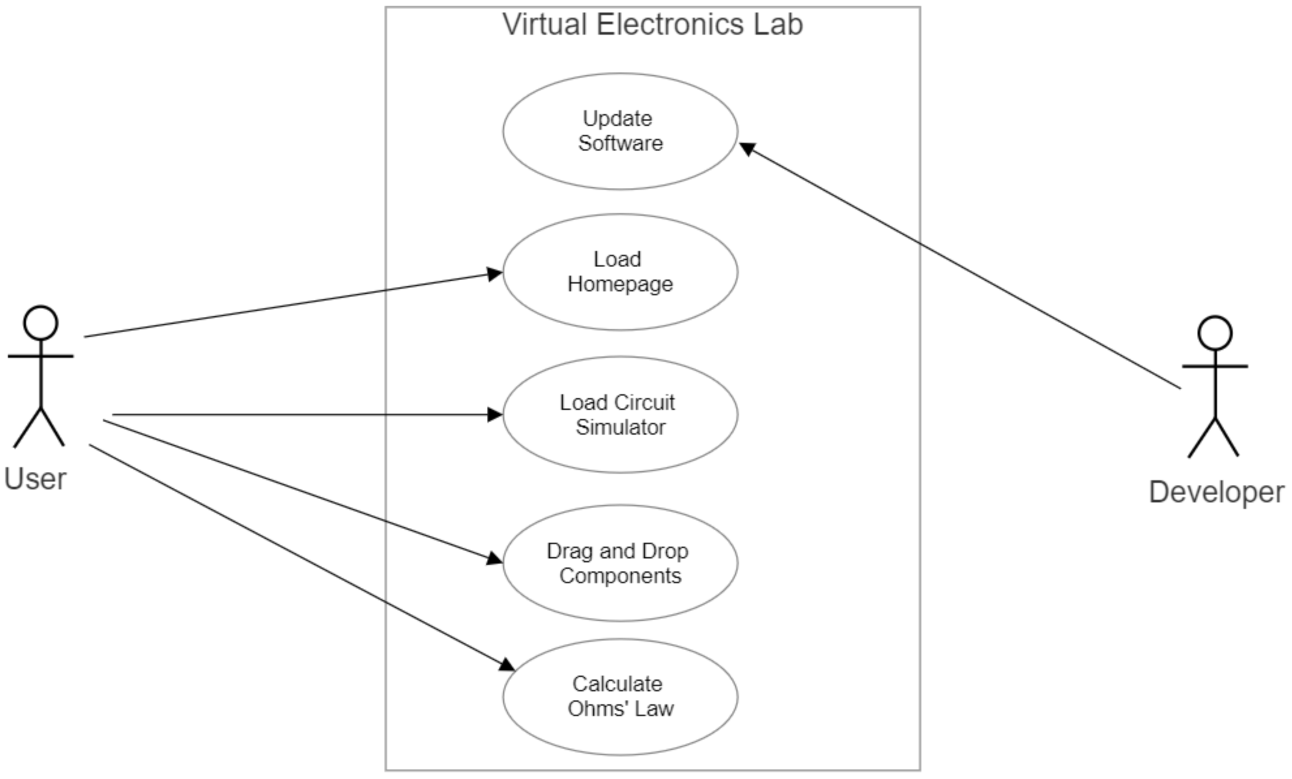


Fig 3.3

Actors in this use case diagram:-

1. User :- User can interact with this software. He/She can do following things with this software :-

a) Load Homepage
b) Load Circuit Simulator
c) Drag and Drop components
d) Calculate Ohm's Law

2. Developer :- Developer can make changes in the software like updating the software or adding new components and features.

3.4 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

CHAPTER 04 - SOFTWARE REQUIREMENT SPECIFICATION

Software Requirement Specification or SRS is a document that describes what a software system should do and how it should behave. It includes information about the system's features, functions, and performance requirements, as well as details about how the system should be designed, developed, and tested

4.1 Overall Description

The virtual electronics lab is a software system that aims to provide a digital platform for students and educators to learn and practice physics concepts related to electric circuits.

The system has two main features:

Circuit simulator with drag and drop features: This feature will allow users to create and simulate electric circuits using drag and drop components. The system should have a library of circuit components, including resistors, capacitors, and switches, that users can choose from to create their circuits.

Ohm's Law calculator: This feature will allow users to calculate the voltage, current, or resistance of a circuit using Ohm's Law formula. The system should have an interface that allows users to input the values of the two variables they know, and the system should calculate the third variable.

The system should be user-friendly and easy to navigate, with clear instructions and feedback for users. It should also be reliable and accurate in its calculations and simulations. The system should be developed using modern software development practices and should be tested thoroughly to ensure its quality and usability.

4.1.1 Product Functions

This virtual electronics lab is a web based system. The responsibility for the virtual electronics lab could vary depending on the organization or institution implementing it. In most cases, it would likely fall under the responsibility of the education or instructional technology department. This team would be responsible for creating and maintaining the software needed to run the virtual electronics lab.

The business purpose of the virtual electronics lab would be to provide students with a virtual environment where they can perform experiments and learn about physics concepts in a safe and controlled setting. This would save time, resources, and costs associated with traditional lab setups and provide a flexible and accessible learning experience for students.

The virtual electronics lab may have interfaces to other systems such as learning management systems (LMS), student information systems (SIS), and content management systems (CMS). These interfaces would allow for easy integration with other education technology tools and systems, providing a seamless experience for both teachers and students. For example, an LMS interface could allow teachers to assign virtual electronics lab experiments as homework or grade assignments completed in the virtual lab. A CMS interface could allow for easy creation and sharing of virtual lab experiments and materials.

4.1.2 User Characteristics

Users involved with a virtual electronics lab may include:

Students: Students are the primary users of the virtual electronics lab. They would typically have a background in physics and would use the lab to perform experiments and learn physics concepts. They may need some basic training on how to use the virtual lab software and equipment.

Teachers: Teachers are responsible for assigning lab experiments, guiding students through the experiments, and grading completed assignments. They would require training on how to integrate the virtual electronics lab into their teaching curriculum and how to use the lab software to create assignments and grade student work.

Lab Technicians: Lab technicians may be responsible for maintaining the virtual electronics lab equipment and ensuring that the software is up-to-date and functioning properly. They would require training on the maintenance and troubleshooting of the virtual lab software and hardware.

Administrators: Administrators may be responsible for managing the virtual electronics lab software and infrastructure, as well as ensuring that the lab meets educational standards and regulations. They would require training on the management and administration of the virtual lab system.

The general characteristics of eventual users of a virtual electronics lab would be that they have a background in physics or are currently studying physics, such as high school or college-level students. They may also have varying degrees of technical proficiency, so the lab software should be designed with user-friendliness in mind, with intuitive interfaces and clear instructions. The virtual lab should also provide clear guidance and feedback to help users navigate through the experiments and understand the physics concepts being taught.

4.1.3 General Constraints

The general constraints of a virtual electronics lab may include:

Availability: The virtual electronics lab should be available to users 24/7 to allow for flexibility in learning.

Usability: The lab software should be user-friendly, with intuitive interfaces and clear instructions, to accommodate users with varying degrees of technical proficiency.

Compatibility: The virtual electronics lab should be compatible with various operating systems and devices, to allow users to access the lab from different locations and devices.

Security: The virtual electronics lab should have appropriate security measures in place to protect user data and prevent unauthorized access to the lab software and equipment.

The constraints of the virtual electronics lab system can be further classified into different categories:

Hardware constraints: These include the physical limitations of the hardware used to run the virtual electronics lab, such as processing power, storage capacity, and availability of input/output devices.

Network constraints: These include the limitations of the network infrastructure used to run the virtual electronics lab, such as bandwidth, latency, and connectivity.

System software constraints: These include the limitations of the system software used to run the virtual electronics lab, such as compatibility with different operating systems, availability of necessary software libraries, and limitations of the software environment.

Software constraints: These include the limitations of the virtual electronics lab software itself, such as the availability of necessary features and functionalities, compatibility with different hardware and software configurations, and scalability.

User constraints: These include the limitations of the users, such as their technical proficiency, access to necessary hardware and software, and availability to use the virtual electronics lab.

Processing constraints: These include the limitations of the system's processing power, such as the ability to handle multiple users simultaneously or to process complex simulations.

Timing constraints: These include the limitations of the system's response time, such as the time it takes to load an experiment or to generate a simulation.

4.1.4 Assumptions and Dependencies

Assumptions and deletions made during the development of a virtual electronics lab can include:

Assumptions made at the beginning of the development effort: These include assumptions about the availability of hardware and software resources, the expertise of the development team, the schedule and budget for the project, and the needs and preferences of the target user group.

Assumptions made during the development of the virtual electronics lab: These may include assumptions about the stability and reliability of third-party software libraries or components, the feasibility of certain design approaches or features, and the scalability and performance of the system.

Factors that can affect the requirements stated in the Software Requirements Specification (SRS) include:

Changes in user requirements: User requirements may change as they gain more experience with the virtual electronics lab or as they encounter new challenges or use cases.

Changes in technology: Advancements in technology can provide new opportunities for the virtual electronics lab, such as the ability to use virtual reality or machine learning algorithms.

Changes in educational standards or regulations: Changes in educational standards or regulations may require the virtual electronics lab to conform to new guidelines or to include additional features or capabilities.

Changes in hardware or software infrastructure: Changes in the hardware or software infrastructure used to run the virtual electronics lab may require modifications to the system's design or implementation.

Changes in budget or schedule: Changes in budget or schedule may require adjustments to the scope or timeline of the virtual electronics lab project.

It is important to note that these factors are not design constraints on the software, but changes to them can affect the requirements stated in the SRS. Therefore, it is essential to periodically review and update the SRS to ensure that it accurately reflects the current needs and constraints of the project.

4.2 External Interface Requirements

The virtual electronics lab should have a user-friendly interface that allows users to easily navigate and interact with the system. This interface may include buttons, menus, and other controls that enable users to perform various actions, such as selecting experiments, inputting data, and viewing results.

4.2.1 User Interface

virtual electronics lab may require an external user interface (UI) to enhance the user experience or provide additional functionality.

For example, the virtual electronics lab require a web-based UI that can be accessed from a web browser such as Internet Explorer, Chrome, or Mozilla Firefox. The UI may be implemented using web technologies such as HTML, CSS, and JavaScript, and may use web frameworks such as Angular or React to enhance the user experience.

Alternatively, the virtual electronics lab may require a desktop UI that can be installed and run locally on a user's computer. The UI may be implemented using a programming language such as Java, and may use graphical user interface (GUI) tools such as JavaFX or Swing to create the interface.

Overall, the specific requirements for the external UI of the virtual electronics lab will depend on the needs of the users and the specific features and functionality of the system.

4.2.2 Hardware Interface

This virtual electronics lab does need to any external hardware like modem or high processor because this is an online web application which can run smoothly irrespective of the user’s system configuration.

However, this software may require some external devices like temperature sensor to get the user’s temperature for some kind of experiment.

Overall, the hardware and external simulator systems requirements depends on the needs of the users and the specific features and functionality of the system.

4.2.3 Software Interface

Yes, the virtual electronics lab website may require external software interactions to provide additional functionality or security features. For example:

The virtual electronics lab may require an external database for storing and querying data related to experiments or user accounts. The database may use standard query languages such as SQL or NoSQL, and may be accessed using APIs such as JDBC or ORM frameworks such as Hibernate.

The virtual electronics lab may require a security system for user authentication and authorization, to ensure that only authorized users can access the system or certain features within it. The security system may use standard protocols such as OAuth or OpenID Connect, and may be integrated with frameworks such as Spring Security.

The virtual electronics lab may require a payment portal to allow users to purchase or access certain features or experiments. The payment portal may use standard payment gateways such as PayPal or Stripe, and may be integrated with frameworks such as Spring Boot to ensure secure and reliable transactions.

The virtual electronics lab may require an application server to deploy and manage the system's components, such as EJBs (Enterprise JavaBeans). The application server may use J2EE (Java 2 Enterprise Edition) or other frameworks such as Spring Framework to ensure high performance and scalability.

Overall, the specific software requirements for the virtual electronics lab will depend on the specific features and functionality of the system, as well as the needs and preferences of the users. It is important to ensure that the software used is reliable, compatible with the system, and able to provide the necessary performance and functionality.

4.3 Functional requirements

Functional requirements specifies what this software should do, or what functions it should perform. The functions should be like user authentication, user input, data processing, user output, user interaction, data storage etc.

4.3.1 FR1 Login Requirement

Here are some login functional requirements that the virtual electronics lab website may have:

User authentication: The login system should require users to provide a unique username and a secure password to verify their identity.

User registration: The login system should allow new users to create an account and provide their personal information, such as name, email address, and contact information.

Password recovery: The login system should provide a way for users to recover their password in case they forget it or need to reset it.

Session management: The login system should maintain a user session throughout the website and log out the user automatically after a certain period of inactivity.

User roles and permissions: The login system should assign different roles and permissions to users based on their level of access and privileges, such as administrator, instructor, or student.

Security features: The login system should include security features, such as encryption, validation, and error handling, to prevent unauthorized access or malicious attacks.

These are just a few examples of the login functional requirements that the virtual electronics lab website may have. The specific requirements will depend on the needs and goals of the website, as well as the preferences and expectations of its users. It is important to design and implement a robust login system that provides a secure and user-friendly experience for all users.

4.3.1 FR1 Login Requirement

Here are some possible functional requirements for the registration form of the virtual electronics lab website:

User information: The registration form should collect basic user information such as name, email address, and contact information.

Password requirements: The registration form should enforce strong password requirements, such as minimum length, complexity, and special characters.

Validation: The registration form should validate user inputs to ensure they are accurate and complete, such as validating email addresses and phone numbers.

Captcha: The registration form should use a captcha to prevent automated spam and fraudulent registrations.

Terms and conditions: The registration form should include a checkbox for users to agree to the terms and conditions of the virtual electronics lab website.

Error handling: The registration form should display appropriate error messages when the user inputs invalid or incomplete information.

Confirmation: The registration form should provide a confirmation message or email to the user after successful registration.

Integration with the database: The registration form should integrate with the database to store user information securely.

User privacy: The registration form should ensure user privacy by implementing appropriate data protection and security measures.

These are just a few examples of the functional requirements for the registration form of the virtual electronics lab website. The actual requirements may vary depending on the specific needs and goals of the website. It is important to design and implement a registration form that is user-friendly, secure, and meets the expectations and needs of the target users.

4.4 Design Constraints

Here are some examples of design constraints that may apply to a virtual electronics lab website:

Programming Language: All coding for the website must be done using HTML, CSS, and JavaScript.

Platform: The website must be designed to run on modern web browsers such as Chrome, Firefox, and Safari.

User Interface: The website's user interface must be intuitive and easy to navigate for both students and teachers.

Accessibility: The website must conform to accessibility standards to accommodate users with disabilities.

Security: The website must have strong security measures in place to protect user data and prevent unauthorized access.

Performance: The website must be designed to load quickly and efficiently, with minimal lag time between actions.

CHAPTER 05 - ESTIMATIONS

Software estimation is the process of predicting the amount of effort, time, and resources required to develop software or complete a project. It involves analyzing and defining the requirements, breaking down tasks, and evaluating various factors that can impact the development process.

5.1 Function Points

Function points are a technique used to measure the size and complexity of a software system based on the user's interactions with the system. Function points are needed because they provide a way to estimate the resources needed to develop, test, and maintain software systems.

Count Justification **:-**

1. User Input
Drag and drop circuit maker:
User selects a component from the available components: Simple
User places the component at a specific location on the circuit board: Simple
User connects the components by drawing wires: Average

Ohm's law calculator:
User enters the value of voltage: Simple
User enters the value of current: Simple
User enters the value of resistance: Simple

Total User Inputs = 6
Average Weighing Factor = Simple

2. User Output
Drag and drop circuit maker:
Displaying the circuit diagram created by the user: Simple

Ohm's law calculator:
Displaying the calculated value of voltage: Simple
Displaying the calculated value of current: Simple
Displaying the calculated value of resistance: Simple

Total User Outputs = 4
Average Weighing Factor = Simple

3. User Inquiries
Login:
Authenticating user credentials and logging the user into the system: Average

Password reset:
Processing user request to reset password: Simple

Support request:
Processing user request for technical support: Complex

Total User Inquiries = 3
Average Weighing Factor = Average

4. Files

Drag and drop circuit maker:
None

Ohm's law calculator:
None

Login:
User account data file: Simple

Password reset:
User account data file: Simple

Support request:
User account data file: Simple

Total Files = 3
Average Weighing Factor = Simple

Total User Inquiries = 3
Average Weighing Factor = Average

5. External Interfaces

Drag and drop circuit maker:
None

Ohm's law calculator:
None

Login:
User account database: Simple

Password reset:
User account database: Simple
Email server: Average

Support request:
User account database: Simple
Email server: Average

Total External Interfaces = 5
Average Weighing Factor = Average

Measurement parameter	**Count**	**simple**	**Average**	**Complex**	**Weighing Factor**	**total**
Number of User Inputs	6	3	4	6	3	18
Number of User Outputs	4	4	5	7	4	16

Number of user inquiries	3	3	4	6	4	12
Number of files	3	7	10	15	10	30
Number of external interfaces	5	5	7	10	7	35
COUNT TOTAL						111

Table 5.1: Measurements Parameters

To compute Function Points (FP), the following formula is used:

FP= Count Total * [0.65 + 0.01 * ∑(Fi)]

Where the count total is the sum of all FP entries obtained from Table 5.1 . ∑(Fi) is the sum of allresponses to the question in Table 5.2. It is called Value Adjustment Factor (VAF).

The response values in the value adjustment table are typically integers between 0 and 5. A response value of 0 means that the particular question does not apply or is not relevant to the software being developed. A response value of 1 means that the question is somewhat applicable or relevant, while a response value of 5 means that the question is highly applicable or relevant. The response values for each question are added up to determine the total response value, which is used to calculate the value adjustment factor (VAF).

S No.	**Question**	**Response**
01	Are there any hardware or platform dependencies that significantly affect the design or implementation of the system?	0
02	Does the software require high performance or real-time processing?	4
03	Does the software require a good documentation?	3
04	Is the software expected to be used by a large number of users?	4
05	Is the software developed using new technologies or frameworks?	5
06	Does the software require extensive data privacy measures?	5
07	Does the software require a exceptional UI/UX design?	3
08	Are there any third party libraries that has been used to develop this software?	2
09	Are there any hardware or software platform dependencies to run this software?	0
10	Is the software being developed for a highly competitive or rapidly changing market, requiring extensive feature development and iteration?	2
11	Does the software require maintenance and support over time?	4
12	Is the software expected to have large number of features for users?	3
13	Does the system requires a payment gateway?	1

Based on the above results following calculations are done:

Count Total = 111

∑(Fi)=36

FP = Count Total * [0.65 + 0.01 * ∑(Fi)]

FP = 111 * [0.65 +0.01 * 36]

$\boxed{\text{FP= 112.11}}$

5.1 Effort Estimation

Effort describes the estimated amount of effort required to develop the software. Table 5.3 specifies productivity on the basis of developer's experience and capability.

Developer's experience/capability	Very Low	Low	Normal	High	Very High
Environment maturity/capability	Very Low	Low	Normal	High	Very High
PROD	4	7	13	25	50

Table 5.3 Productivity Chart

PROD=13

Effort=FP/PROD

Effort=112.11/13

$\boxed{\text{Effort=8.62pm}}$

5.2 Cost

Cost refers to the approximated cost of the project. It is calculated based on the efforts that the project requires.

Cost=Effort(pm) * Labour Rate(INR/pm)

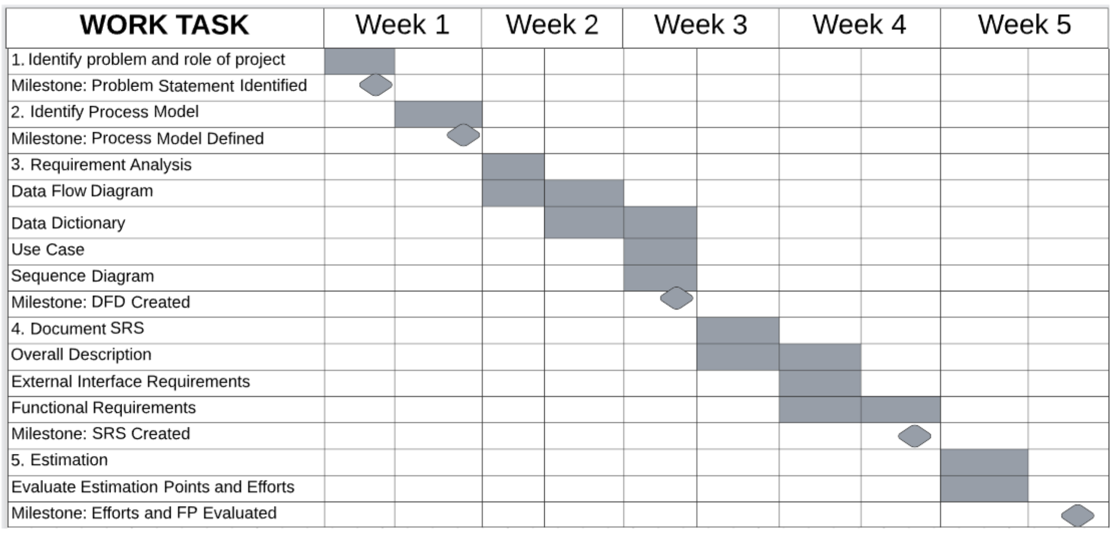
Cost=8.62 * 25,000

$\boxed{\text{Cost=215,000}}$

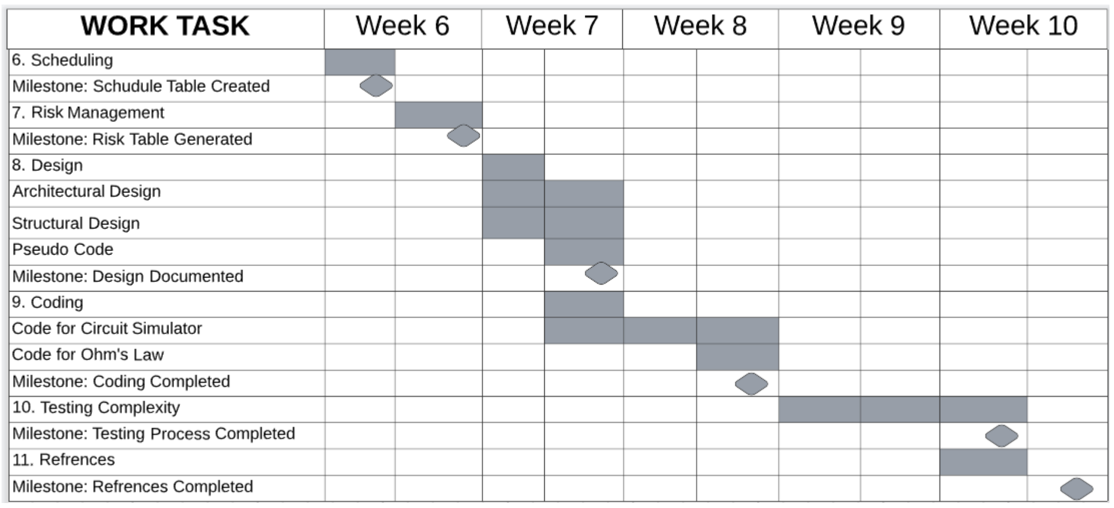
CHAPTER 06 - SCHEDULING

A Gantt chart is a visual representation of a project schedule that shows the start and end dates of individual tasks and their dependencies. The chart typically includes columns for task names, start and end dates, and progress status. The rows of the chart represent individual tasks or groups of related tasks.

Gantt Chart:



WORK TASK	Week 1		Week 2		Week 3		Week 4		Week 5	
1. Identify problem and role of project	▨									
Milestone: Problem Statement Identified	◆									
2. Identify Process Model		▨								
Milestone: Process Model Defined		◆								
3. Requirement Analysis			▨							
Data Flow Diagram			▨	▨						
Data Dictionary				▨	▨					
Use Case					▨					
Sequence Diagram					▨					
Milestone: DFD Created					◆					
4. Document SRS						▨				
Overall Description						▨	▨			
External Interface Requirements							▨			
Functional Requirements							▨	▨		
Milestone: SRS Created								◆		
5. Estimation									▨	
Evaluate Estimation Points and Efforts									▨	
Milestone: Efforts and FP Evaluated										◆



WORK TASK	Week 6		Week 7		Week 8		Week 9		Week 10	
6. Scheduling	▨									
Milestone: Schudule Table Created	◆									
7. Risk Management		▨								
Milestone: Risk Table Generated		◆								
8. Design			▨							
Architectural Design			▨	▨						
Structural Design			▨	▨						
Pseudo Code				▨						
Milestone: Design Documented				◆						
9. Coding				▨						
Code for Circuit Simulator				▨	▨	▨				
Code for Ohm's Law						▨				
Milestone: Coding Completed						◆				
10. Testing Complexity							▨	▨	▨	
Milestone: Testing Process Completed									◆	
11. Refrences									▨	
Milestone: Refrences Completed										◆

1. **Project Planning :**

   a. Define project goals and requirements
   b. Identify user needs and expectations
   c. Create project plan and deadline

2. **Designing:**

   a. Design a landing page
   b. Design beautiful user interface

3. **Coding:**

   a. Develop code for landing page
   b. Develop code for drag and drop functionality
   c. Develop code for circuit simulation
   d. Develop code for Ohm’s law calculator

4. **Debugging:**

   a. Debugging any errors or issues

5. **Testing:**

   a. Conduct unit testing for drag and drop functionality
   b. Conduct unit testing for circuit simulation
   c. Conduct unit testing for Ohm’s law calculator
   d. Conduct integration testing

6. **Deployment:**

   a. Prepare website for deployment
   b. Deploy website to servers like netlify.

CHAPTER 07 - RISK MANAGEMENT

Risk management in software engineering is the process of identifying, assessing, and controlling potential risks that could impact the success of a software project. It involves identifying possible problems or issues that may arise during the development process, assessing the likelihood and potential impact of those risks, and developing strategies to mitigate or control those risks.

7.1 Risk Table

A risk table is a table that lists all identified risks for a project, along with their likelihood of occurring, potential impact, and proposed mitigation strategies. The risk table is an important tool for risk management in software engineering as it allows project managers and team members to track and monitor potential risks throughout the project life cycle.

Risk Summary	Risk Category	Probability	Impact	RMMM
Server Downtime	Technical	Medium	Critical	Section 4.2 of the RMMM plan
Data Security Breach	Security	Low	Catastrophic	Section 4.5 of the RMMM plan
Inaccurate Circuit Simulation	Technical	High	Marginal	Section 4.1 of the RMMM plan
Compatibility Issues with Web Browsers	Technical	Medium	Negligible	Section 4.3 of the RMMM plan
User Error in Inputting Data	Human	High	Marginal	Section 4.4 of the RMMM plan

Impact:

1. Catastrophic
2. Critical
3. Marginal
4. Negligible

RMMM Plan:

Section 4.1: Risk mitigation strategies for inaccurate circuit simulation
Section 4.2: Risk mitigation strategies for server downtime
Section 4.3: Risk mitigation strategies for compatibility issues with web browsers
Section 4.4: Risk mitigation strategies for user errors in inputting data
Section 4.5: Risk mitigation strategies for data security breaches

CHAPTER 08 - DESIGN

Design description is a document that describes the design of a software application or system. It typically includes information such as the purpose and scope of the software, the system architecture, the design principles and patterns used, and the detailed design of each component or module of the software.

8.1 Structured Chart

A structured chart is a graphical representation of a program's control flow. It uses a specific set of symbols to represent the various elements of the program's logic and control structure.

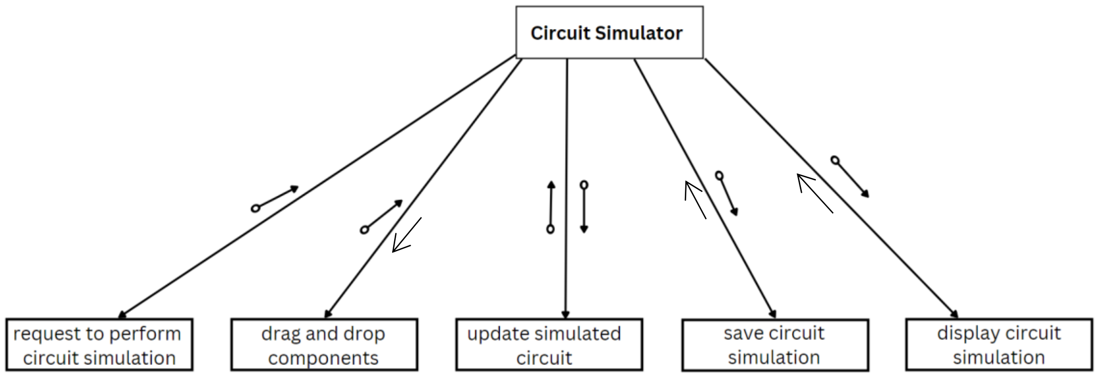


Fig 8.1

Start | |-- Load Homepage | | | |-- Display Homepage | | | |-- If Circuit Simulator Button is Pressed | | | |-- Load Circuit Simulator | | | |-- Display Circuit Simulator | | | |-- If Circuit Components are Dragged and Dropped | | | |-- Update Circuit Diagram | | | |-- If Calculate Ohm's Law Button is Pressed | | | |-- Calculate Ohm's Law | | | |-- Display Results | |-- If About Us Button is Pressed | | | |-- Load About Us Page | | | |-- Display About Us Page | |-- If Contact Us Button is Pressed | |-- Load Contact Us Page | |-- Display Contact Us Page End

8.2 Pseudo Code

```
// Function to create an empty circuit board
function createCircuitBoard() {
  // Display empty circuit board
}

// Function to add a component to the circuit board
function addComponent(component) {
  // Add component to the circuit board
}

// Function to remove a component from the circuit board
function removeComponent(component) {
  // Remove component from the circuit board
}

// Function to calculate Ohm's law based on user inputs
function calculateOhmsLaw(voltage, current, resistance) {
  // If voltage is missing, calculate voltage
  if (!voltage) {
    voltage = current * resistance;
  }
  // If current is missing, calculate current
  else if (!current) {
    current = voltage / resistance;
  }
  // If resistance is missing, calculate resistance
  else if (!resistance) {
    resistance = voltage / current;
  }
  // Display the results to the user
  displayResults(voltage, current, resistance);
}

// Function to display the results of Ohm's law calculation
function displayResults(voltage, current, resistance) {
  // Display the results to the user
}

// Main program
createCircuitBoard();  // Display empty circuit board

// User adds components to the circuit board
addComponent(resistor);
addComponent(battery);
addComponent(wire);

// User removes a component from the circuit board
removeComponent(battery);

// User calculates Ohm's law based on user inputs
calculateOhmsLaw(12, null, 4);  // Calculate current

// User calculates Ohm's law based on user inputs
calculateOhmsLaw(null, 2, 4);   // Calculate voltage
```

CHAPTER 09 - CODING

9.1 Code Snippet 1 (Circuit Simulator)

```
<html>

<head>
  <title>Electric Circuit Simulation</title>
  <meta charset="UTF-8" />
  <link type="text/css" rel="stylesheet" href="../css/aristo/jquery-ui-1.8.16.custom.css" />
  <link type="text/css" rel="stylesheet" href="../css/jquery.layout.css" />
  <link type="text/css" rel="stylesheet" href="../css/application.css" />
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
    integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous" />
  <link rel="icon" href="../css/icons/microchip.png" type="image/icon type" />
  <link rel="preconnect" href="https://fonts.gstatic.com" />
  <link href="https://fonts.googleapis.com/css2?family=PT+Sans:wght@700&display=swap"
rel="stylesheet" />
  <script src="https://cdnjs.cloudflare.com/ajax/libs/mathjs/7.3.0/math.min.js"></script>
  <script src="../src/gui/dragdrop.js"></script>
  <script src="../src/lib/jquery.js"></script>
  <script src="../src/lib/jquery-ui.js"></script>
  <script src="../src/lib/jquery.browser.js"></script>
  <script src="../src/lib/jquery.layout.js"></script>
  <script src="../src/draw2d.js"></script>
  <script src="../src/components/VoltageSource.js"></script>
  <script src="../src/components/Resistor.js"></script>
  <script src="../src/components/CurrentSource.js"></script>
  <script src="../src/components/CCVS_Gen.js"></script>
  <script src="../src/components/CCCS_Gen.js"></script>
  <script src="../src/components/VCCS.js"></script>
  <script src="../src/components/VCVS.js"></script>
  <script src="../src/gui/Application.js"></script>
  <script src="../src/gui/View.js"></script>
  <script src="../src/gui/Toolbar.js"></script>
  <style>
    .component_div {
      overflow-y: hidden;
      overflow-x: hidden;
    }
  </style>
</head>
<body id="container">
  <div id="navbar">
    <nav class="navbar navbar-expand-lg navbar-dark navbar-expand">
      <a class="navbar-brand" href="start.html">
        <h3>Electric Circuit Simulator</h3>
      </a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarSupportedContent">
```

```html
        <ul class="navbar-nav ml-auto">
          <li class="nav-item">
            <a class="nav-link" href="../circuit-nav.html">
              <h5>Home</h5>
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="start.html">
              <h5>Editor</h5>
            </a>
          </li>
        </ul>
      </div>
    </nav>
  </div>
  <div id="content">
    <div id="toolbar" class="navbar sticky-top navbar-expand-lg navbar-dark"></div>
    <div id="canvas" draggable="true" style="width: 3000px; height: 3000px"></div>
    <div id="navigation" class="component_div">
      <h4 style="padding-top: 10%">COMPONENTS</h4>
      <div data-shape="Resistor" draggable="true" class="palette_node_element
draw2d_droppable">
        Resistor
      </div>
      <div data-shape="CurrentSource" draggable="true" class="palette_node_element
draw2d_droppable">
        Current Src
      </div>
      <div data-shape="VoltageSource" draggable="true" class="palette_node_element
draw2d_droppable">
        Voltage Src
      </div>
      <div data-shape="VCCS" draggable="true" class="palette_node_element draw2d_droppable">
        VCCS
      </div>
      <div data-shape="VCVS" draggable="true" class="palette_node_element draw2d_droppable">
        VCVS
      </div>
      <div data-shape="CCCS_Gen" draggable="true" class="palette_node_element
draw2d_droppable">
        CCCS
      </div>
      <div data-shape="CCVS_Gen" draggable="true" class="palette_node_element
draw2d_droppable">
        CCVS
      </div>
    </div>
  </div>
  <div class="mask" role="dialog"></div>
  <div class="modal" id="svg" role="alert">
    <button class="close" role="button">
      <img src="https://img.icons8.com/ios-glyphs/30/000000/macos-close.png" />
    </button>
    <div style="
          overflow-y: auto !important;
          display: flex !important;
          justify-content: center;
          align-items: center;
          height: 550;
          padding-top: 5%;
        ">
      <img class="shadow" id="preview" style="
            border-radius: 5px;
            background: white;
            width: 600px;
            height: 500px;
            border: 3px solid gray;
            margin-left: 15px;
            overflow: hidden;
          " />
      <div id="output"></div>
    </div>
    <button class="btn btn-primary again btn-modal btn-lg px-5" style="left: 15%">
      Edit
    </button>
    <button class="btn btn-primary btn-modal btn-lg" onclick="window.location.reload()"
style="left: 35%">
      Simulate Another
    </button>
  </div>
  <div class="mask-docs" role="dialog"></div>
  <div class="modal-docs" id="svg" role="alert">
    <button class="close-docs" role="button">
      <img src="https://img.icons8.com/ios-glyphs/30/000000/macos-close.png" />
    </button>
    <h1>Instructions</h1>
    <ol>
      <li>
        Drag and drop the elements from the components section to the playground section.
      </li>
      <li>
        Double click on the elements to rotate it 90 degrees in clockwise direction.
      </li>
      <li>
        Click on the element and then drag it to change its position.
      </li>
      <li>
        Click on the port of first element and then click on the port to second element to
connect these to ports.
      </li>
      <li>
        Double click on value to change the value.
      </li>
      <li>
        To undo/redo an element click on undo/redo button.
      </li>
      <li>
        Click on an element and then click on delete button to delete the element.
      </li>
      <li>
        Click on Editor on navbar to get a new playground.
      </li>
    </ol>
  </div>
  <pre id="json" style="
        overflow: auto;
        position: absolute;
        top: 100px;
        right: 10px;
        width: 350;
        height: 500;
        background: white;
        border: 1px solid gray;
      "></pre>
  <script src="../src/index.js"></script>
</body>
</html>
```

This is only html code for the circuit simulator. Javascript code cannot be attached here due to number of lines of code. You can check the full code at https://github.com/AstroDeveloper1010/Virtual-Physics-Lab

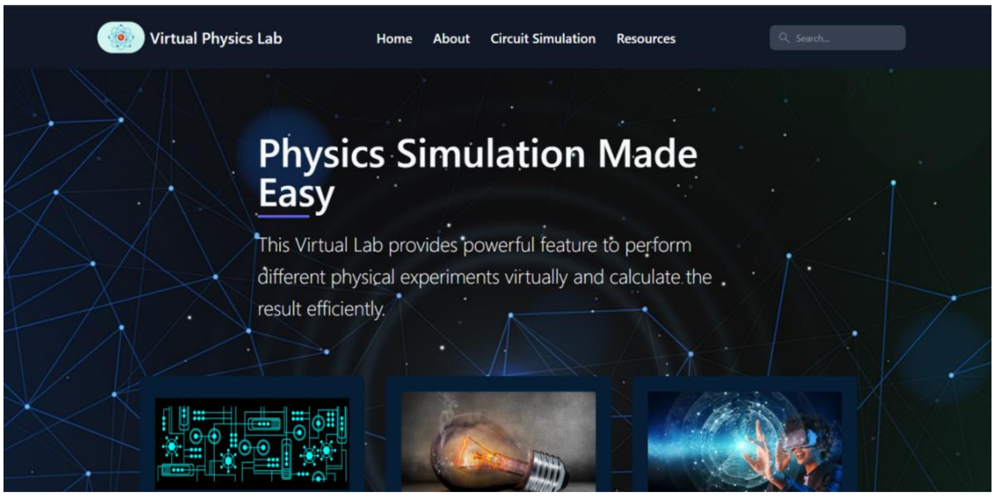
9.2 Code Snippet 2 (Ohm's Law Calculator)

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <meta name="description" content="Ohm Law APP">
    <meta name="viewport" content="width=device-width , initial-scale=1, maximum-scale=1,
user-scalable=no">
    <link rel="shortcut icon" href="images/ohm_logo.png" type="icon">
    <link rel="stylesheet" href="css/ohms-law.css" type="text/css">
    <!-- <script src="https://cdn.tailwindcss.com"></script> -->
<body class="w-full">
    <header class="w-full flex">
        <button class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded
home-btn">Home</button>
        <div class="container-ohms w-full">
            <div class="page_title w-full">
                <img class="image1" src="images/ohm_logo.png" width="50px">
                <div class="flex flex-col w-90">
                    <h1 class="flex justify-center">Ohm's Law Calculator</h1>
                    <p class="">Calculate the relationship b/w current, voltage and
resistance</p>
                </div>
                <img class="image2" src="images/ohm_logo.png" width="50px">
            </div>
        </div>
    </header>

    <article class="flex flex-col items-center w-full">
        <section class="container-ohms-section flex w-full items-center">
            <div class="app_container flex flex-col items-center w-full">
                <div class="selection">
                    <label>Get </label>
                    <select id="get_select">
                        <option selected hidden></option>
                        <option>Resistance</option>
                        <option>Voltage</option>
                        <option>Current</option>
                        <option>Power</option>
                    </select>
                    <label>Calculte by </label>
                    <select id="type_select">
                    </select>
                </div>
```

```html
                <div class="app_body">
                    <label id="input_a_label" place>- - - - -</label>
                    <input id="input_a" type="text" disabled>
                    <label id="input_b_label">- - - - -</label>
                    <input id="input_b" type="text" disabled>
                </div>
                <div class="logo">
                    <img id="electronic_logo" width=80>
                </div>
                <div class="result">
                    <p id="res">Result = .... </p>
                </div>
                <button id="clear_data"
                    class="clear_button bg-blue-500 hover:bg-blue-700 text-white font-bold py-
2 px-4 rounded-full reset-btn"
                    title="Clear inputs data ">Reset</button>
            </div>
        </section>
        <section class="information_container">
            <hr>
            <p class="ohms-law-para justify-content-center">Relationship between electric
current, voltage and
                resistance</p>
            <img src="images/ohm_law.png" width="600px">
        </section>
    </article>
    <footer>
    </footer>
    <script src="js/script.js"></script>
    <script src="js/ohm_law_class.js"></script>
    <script src="js/app.js"></script>
    <script src="js/others.js"></script>
</body>
</html>
```

This is only html code for the circuit simulator. Javascript code cannot be attached here due to number of lines of code. You can check the full code at https://github.com/AstroDeveloper1010/Virtual-Physics-Lab

Output Screenshots :-



Physics Simulation Made Easy

This Virtual Lab provides powerful feature to perform different physical experiments virtually and calculate the result efficiently.

Fig 9.1 Landing Page

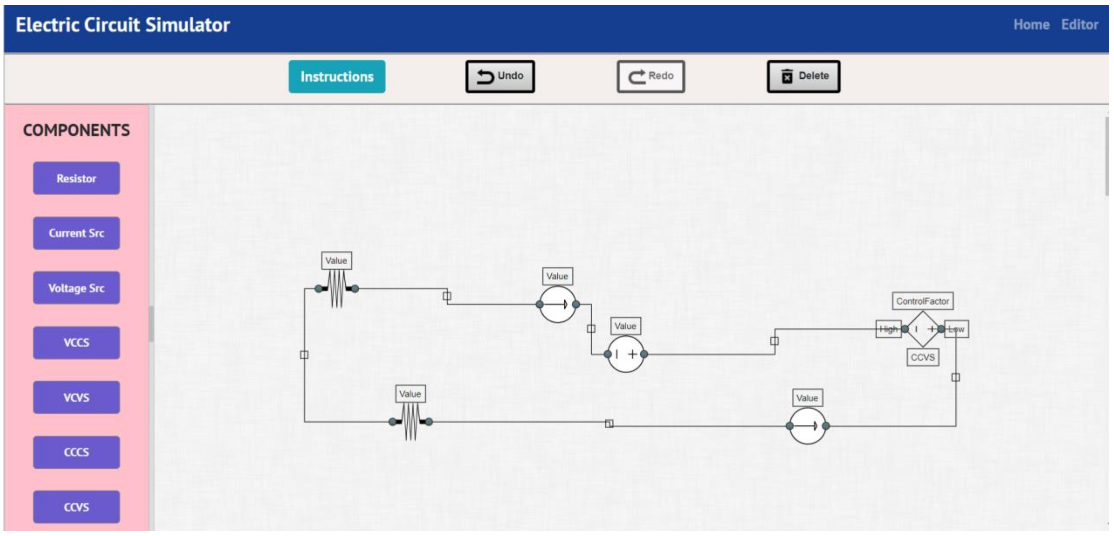
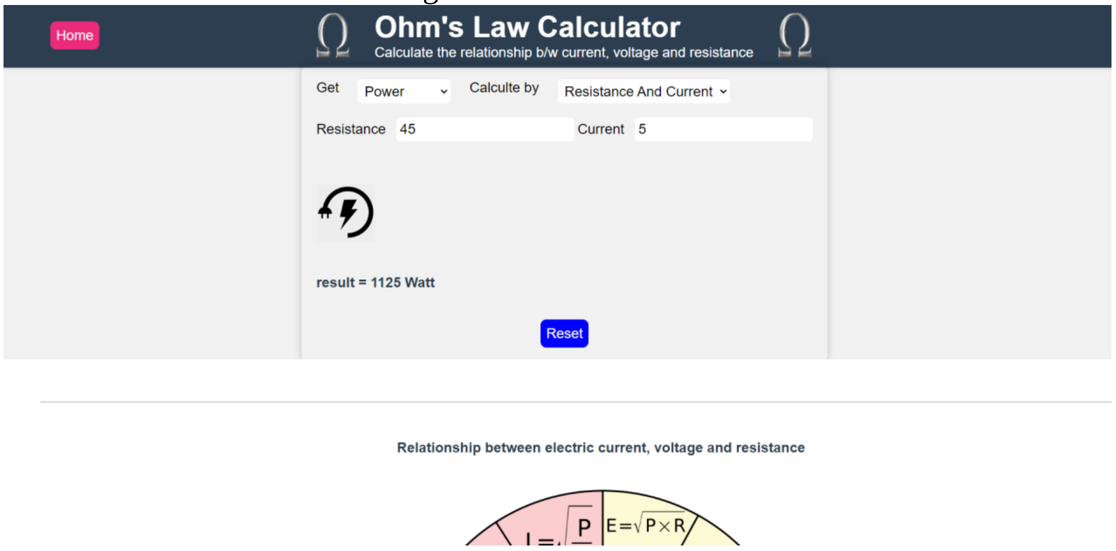


Fig 9.2 Circuit Simulator


Home
Ohm's Law Calculator
Calculate the relationship b/w current, voltage and resistance
Reset
Relationship between electric current, voltage and resistance

Get: Power
Calculte by: Resistance And Current
Resistance: 45
Current: 5

result = 1125 Watt

Fig 9.3 Ohm’s Law Calculator

CHAPTER 10 - TESTING

White Box Testing:

It is a software testing technique in software engineering where the tester has knowledge of the internal workings of the software being tested. In other words, the tester has access to the source code and is able to test the individual components, functions, and algorithms of the software.

The goal of white box testing is to ensure that the software is working as intended by testing all possible paths and outcomes within the code.

Black Box Testing:

It is a software testing technique where the tester does not have knowledge of the internal workings of the software being tested. Instead, the tester only has access to the inputs and outputs of the software, and tests the functionality of the software based on the expected behavior of the inputs and outputs.

The goal of black box testing is to ensure that the software is working correctly from a user's perspective, without being concerned with how it is implemented internally.

10.1 Test Case Design

Test Case ID	Test Case Description	Test Steps	Expected Results	Actual Results	Pass/Fail
TC001	Verify that the circuit simulator loads correctly	1. Open the virtual electronics lab website 2. Click on the Circuit Simulator feature	The Circuit Simulator interface is displayed	The Circuit Simulator Interface loaded successfully	Pass
TC002	Verify that components can be dragged and dropped onto the circuit board	1. Open the Circuit Simulator interface 2. Drag a component from the Component Library 3. Drop the component onto the circuit board	The component is successfully added to the circuit board	The component added to the circuit board successfully	Pass
TC003	Verify that components can be deleted from the circuit board	1. Open the Circuit Simulator interface 2. Add a component to the circuit board 3. Select the component 4. Press the delete key	The component is successfully removed from the circuit board	The component removed from the circuit board	Pass

Test Case ID	Test Case Description	Test Steps	Expected Results	Actual Results	Pass/Fail
TC004	Verify that the Ohm's Law calculator loads correctly	1. Open the virtual electronics lab website 2. Click on the Ohm's Law Calculator feature	The Ohm's Law Calculator interface is displayed	Interface displayed successfully	Pass
TC005	Verify that the Ohm's Law calculator correctly calculates voltage	1. Open the Ohm's Law Calculator interface 2. Enter a value for Current and Resistance 3. Click the "Calculate Voltage" button	The calculated voltage value is correct according to the Ohm's Law equation V=IR	Correct value got displayed	Pass
TC006	Verify that the Ohm's Law calculator correctly calculates current	1. Open the Ohm's Law Calculator interface 2. Enter a value for Voltage and Resistance 3. Click the "Calculate Current" button	The calculated current value is correct according to the Ohm's Law equation I=V/R	Correct value got displayed	Pass
TC007	Verify that the Ohm's Law calculator correctly calculates resistance	1. Open the Ohm's Law Calculator interface 2. Enter a value for Voltage and Current 3. Click the "Calculate Resistance" button	The calculated resistance value is correct according to the Ohm's Law equation R=V/I	Correct value got displayed	Pass

10.2 Flow Graph

A flow graph is a visual representation of the control flow or order of execution of instructions or statements in a program. It is a directed graph where each node represents a statement or instruction and each edge represents the possible flow of control from one statement to another. The flow graph can be used to analyze the program's behavior and identify potential issues, such as unreachable code or infinite loops.

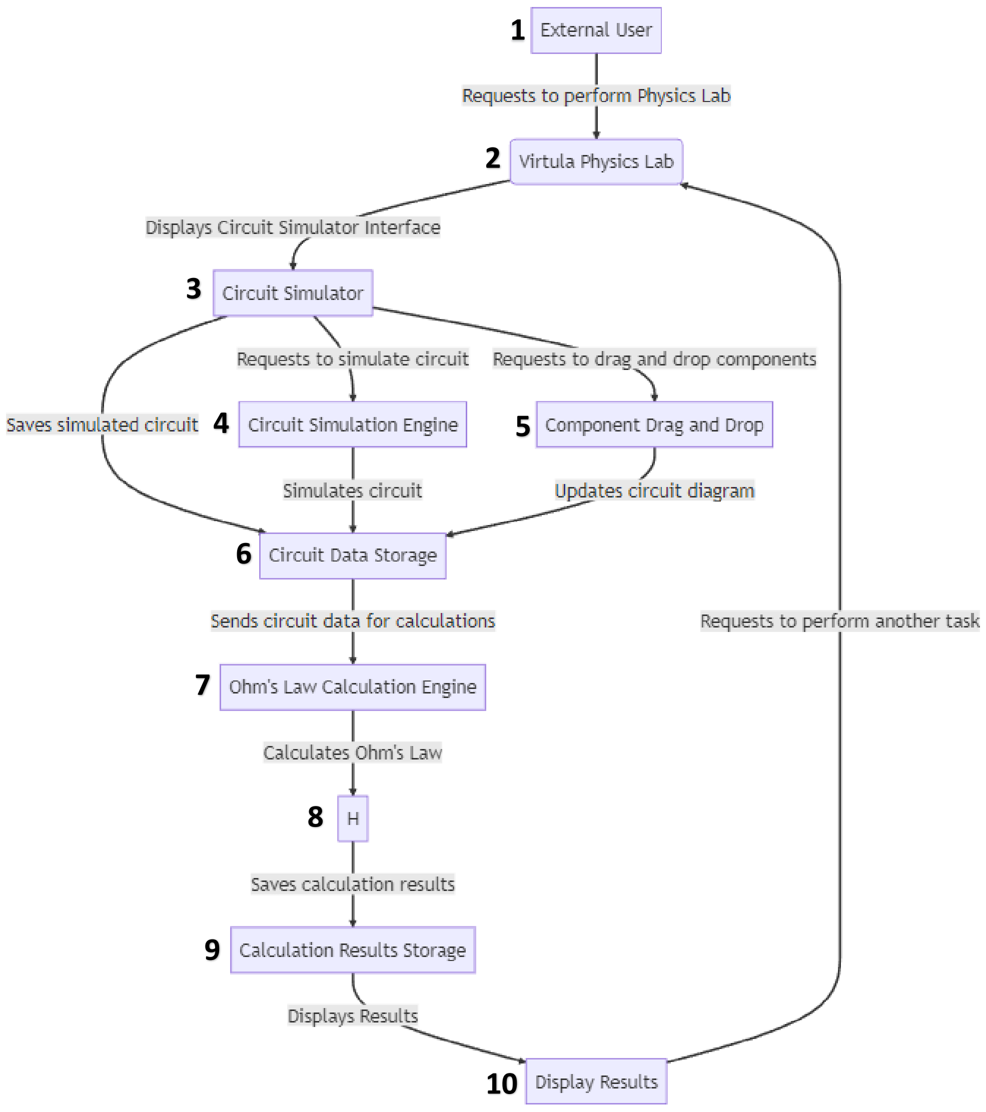

1 External User
Requests to perform Physics Lab
2 Virtula Physics Lab
Displays Circuit Simulator Interface
3 Circuit Simulator
Requests to simulate circuit
Requests to drag and drop components
Saves simulated circuit
4 Circuit Simulation Engine
5 Component Drag and Drop
Simulates circuit
Updates circuit diagram
6 Circuit Data Storage
Sends circuit data for calculations
Requests to perform another task
7 Ohm's Law Calculation Engine
Calculates Ohm's Law
8 H
Saves calculation results
9 Calculation Results Storage
Displays Results
10 Display Results

Fig 10.1

10.3 Basis Path Set

Basis path testing is a white-box testing technique that involves identifying all possible independent paths through the code. The basis path set is a set of test cases derived from the basis path testing technique that ensures that every path through the code is tested at least once.

For the virtual physics lab website, here are the basis path sets for its modules:

Circuit Simulator Module:

Basis Path 1: 1 → 2 → 3 → 4 → 5 → 6 → 9 → 10
Basis Path 2: 1 → 2 → 3 → 4 → 5 → 6 → 9
Basis Path 3: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10
Basis Path 4: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9
Basis Path 5: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8

Ohm's Law Calculator Module:

Basis Path 1: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9
Basis Path 2: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 10

Circuit Component Library Module:

Basis Path 1: 1 → 2 → 3 → 4 → 5 → 6
Basis Path 2: 1 → 2 → 3 → 4 → 5 → 7

10.4 Cyclomatic Complexity

Cyclomatic complexity is a software metric used to indicate the complexity of a program. It is a measure of the number of linearly independent paths through a program's source code.

To calculate the cyclomatic complexity of a module, we can use the formula:
M = E - N + 2P
Where M is the cyclomatic complexity, E is the number of edges in the flow graph, N is the number of nodes in the flow graph, and P is the number of connected components.

Physics Lab System module:
Nodes (N) = 9
Edges (E) = 11
Connected Components (P) = 1
Cyclomatic Complexity (M) = E - N + 2P = 11 - 9 + 2(1) = 4

Circuit Simulator module:
Nodes (N) = 5
Edges (E) = 5
Connected Components (P) = 1
Cyclomatic Complexity (M) = E - N + 2P = 5 - 5 + 2(1) = 2

Ohm's Law Calculator module:
Nodes (N) = 3
Edges (E) = 3
Connected Components (P) = 1
Cyclomatic Complexity (M) = E - N + 2P = 3 - 3 + 2(1) = 2

Circuit Simulation Engine module:
Nodes (N) = 2
Edges (E) = 1
Connected Components (P) = 1
Cyclomatic Complexity (M) = E - N + 2P = 1 - 2 + 2(1) = 1

Component Drag and Drop module:
Nodes (N) = 2

Edges (E) = 1
Connected Components (P) = 1
Cyclomatic Complexity (M) = E - N + 2P = 1 - 2 + 2(1) = 1

The cyclomatic complexity of the entire system can be obtained by summing up the cyclomatic complexities of all the individual modules. In this case, the total cyclomatic complexity is:

M = 4 + 2 + 2 + 1 + 1 = 10

CHAPTER 10 - REFERENCES

Github Repository: https://github.com/AstroDeveloper1010/Virtual-Physics-Lab

Netlify Hosting: https://www.netlify.com/

Coding IDE: https://code.visualstudio.com/

Designing Credits: https://tailwindcss.com/

Designing Credits: https://getbootstrap.com/docs/5.3/getting-started/introduction/

Circuit Simulation Reference: https://github.com/circuit-solver/circuit-solver.github.io

Royalty Free Images Credits: https://pixabay.com/