ACHARYA NARENDRA DEV COLLEGE

Department of Computer Science

2021

SIMULATION OF MICROWAVE OVEN THROUGH DFA

STAR ELITE PROJECT REPORT

Under the Supervision of

Ms. VANDITA GROVER

SUBMITTED BY

SHIVANSH TYAGI (AB-1273)

[B. Sc. (Hons.) Computer Science]

SARVESH SHARMA (AB-1271)

[B. Sc. (Hons.) Computer Science]

SAMBIT BASU (AB-1264)

[B. Sc. (Hons.) Computer Science]

ACKNOWLEDGEMENT

This Project was jointly undertaken by **Shivansh Tyagi**, **Sarvesh Sharma** and **Sambit Basu** of B. Sc. (H.) Computer Science as part of Project Work of **Star ELITE Project 2021-2022**, under the supervision of **Ms. Vandita Grover**. Our primary thanks go to her, who poured over every inch of our project with painstaking attention and helped us throughout the working of the project. It is our privilege to acknowledge our deepest gratitude to her for inspiring us, which has helped us immensely. We are extremely grateful to her for her unstilted support and encouragement in the preparation of this project.

Shivansh Tyagi **(AB-1273)**

Sarvesh Sharma **(AB-1271)**

Sambit Basu **(AB-1264)**

ACHARYA NARENDRA DEV COLLEGE
UNIVERSITY OF DELHI

CERTIFICATE

This is to certify that the project entitled "**SIMULATION OF MICROWAVE OVEN THROUGH DFA**" has been completed by **Shivansh Tyagi**, **Sarvesh Sharma** and **Sambit Basu** of Bachelor of Sciences in Computer Science (Hons.) as a part of Project Work of **Star ELITE Project 2021-2022** from Acharya Narendra Dev College, University of Delhi under the supervision of **Ms. Vandita Grover**.

Prof. RAVI TOTEJA
Officiating Principal
Acharya Narendra Dev College
University of Delhi

Ms. VANDITA GROVER
(Supervisor)
Assistant Professor
Department of Computer Science
Acharya Narendra Dev College
University of Delhi

CONTENTS

1.PROBLEM STATEMENT

1.1 Microwave Oven Simulation

Working of a microwave oven from the switching on of electricity to the prepared food. The Aim of the Project is to simulate a Deterministic Finite Automata based on a simple Microwave oven using a predefined input string and to check whether the Finite Automata accepts the given input string or not.

1.2 Aim of the Project

The Aim of the Project is to

- Simulate a Deterministic Finite Automata based on a simple Microwave oven using a predefined input string and check whether the Finite Automata accepts the given input string or not.

1.2.1 Development Platform

The requirements for a development platform include:

1. The Platform must be widely popular among the users to ensure the fulfillment of the purpose of the Project.
2. The Development Constraints must be minimum.
3. The Development Procedure must be as easy as possible.
4. The Hardware and Software Dependency of the Platform must be minimum.
5. The Users are familiar with the basic interface of the platform.

Based on general research and guidance obtained from experienced individuals, Visual Studio Code qualifies the requirements for a development platform.

1.3 Process Model for the Project

A Process Model defines the flow of all activities, actions and tasks, the degree of iteration, the work products, and the organization of the work that must be done.[1]

Based on the guidance from experienced individuals in the field, The Process Model selected for the Project is the Spiral Development Model.

1.3.1 Spiral Development Model

It is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.[1]

During early iterations, the release might be a model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced. The spiral model can be adapted to apply throughout the entire life cycle of an application, from concept development to maintenance.[1]

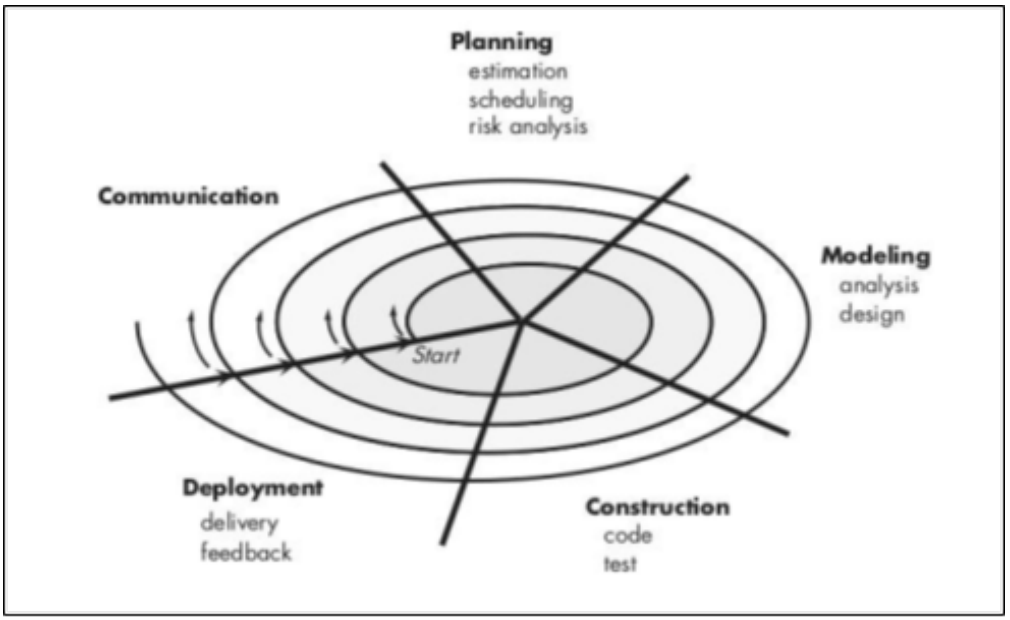


Fig 1.1 Spiral Development Model [1]

Each phase of the Spiral Model is divided into five quadrants as shown in Fig. 1.1. The functions of these five quadrants are:

1. Communication: Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase.
2. Planning: Estimating the cost and time of development, Scheduling, sanctioning and dividing the jobs, and Analyzing the risks involved in the development.
3. Modeling: Alternative solutions possible for the development are proposed in this quadrant. All the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy.
4. Construction: Prototype is built for the best possible solution. The identified features are developed and verified through testing. At the end of the fourth quadrant, the next version of the software is available.
5. Deployment: In the fifth quadrant, the Stakeholders evaluate the so far developed version of the software. In the end, Communication for the next phase is started.

1.3.2 Reasons for the Choice

The Spiral Development Model is highly advantageous for the Project in the following ways:

1. Better Risk Handling: The project involves many unknown risks that occur as the development proceeds, The Spiral Model provides sections involving the risk analysis and risk handling at every phase.
2. Flexibility in Requirements: Change requests in the Requirements at a later phase can be incorporated accurately.
3. Stakeholder Satisfaction: Stakeholders can see the development of the product at an early phase of the software development and thus, they get habituated with the system by using it before completion of the total product.
4. Project Scope: Considering the Length of the Project and its future scope, the spiral model provides a better insight into the stepwise development of different versions of the software.

2.REQUIREMENT ANALYSIS & MODELLING

2.1 Introduction

Analysis of requirements starts with the gathering of requirements (Requirements Elicitation). The Requirements are analysed to identify inconsistencies, defects, omissions, ambiguous information. It is described in terms of relationships and also resolves conflicts if any.[1]

The Requirements of the project are modeled using Diagrammatic Representation of a Deterministic Finite Automata, Data Dictionary.

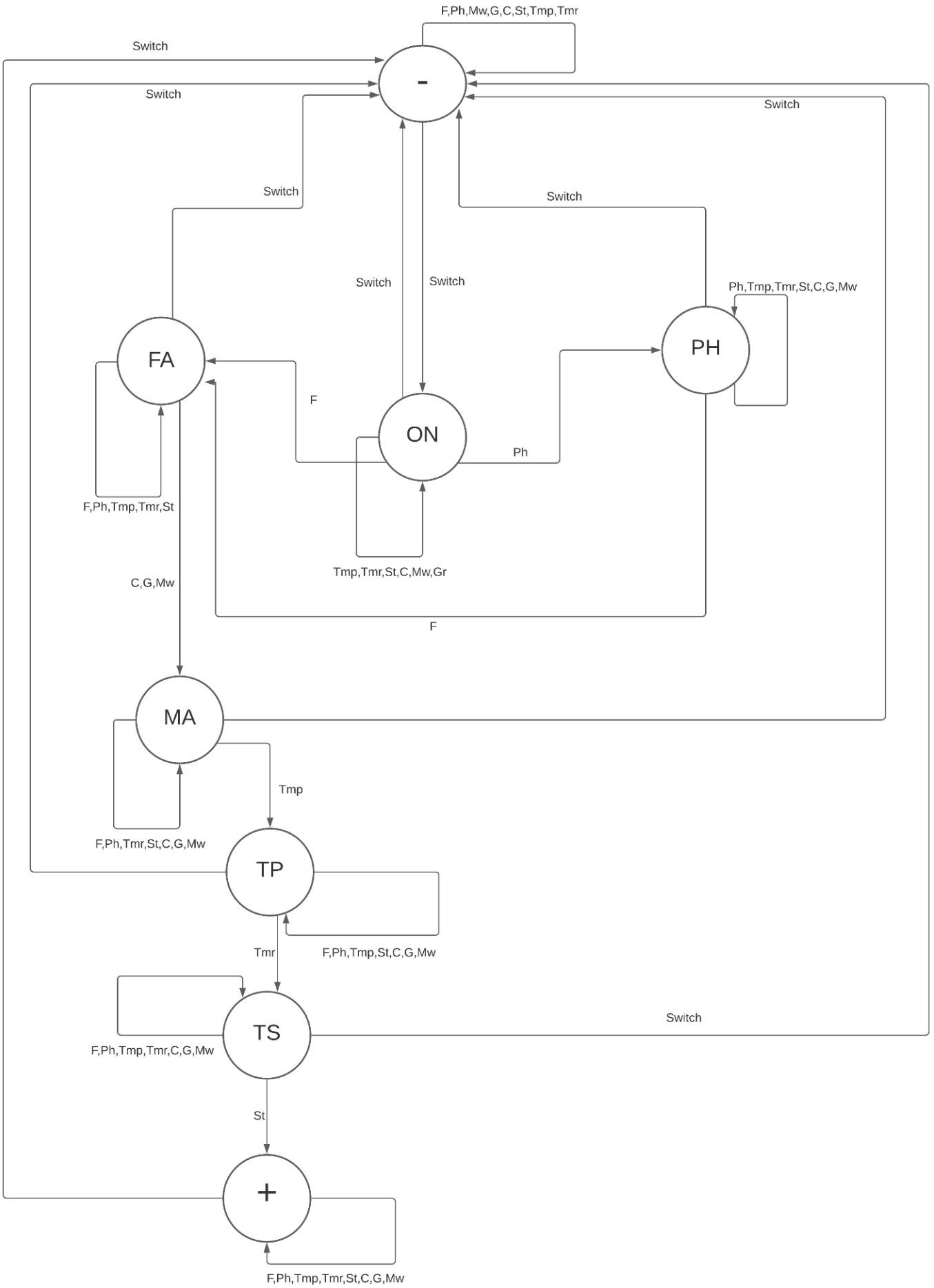
2.2 Deterministic Finite Automata

Deterministic finite automata (or DFA) are finite state machines that accept or reject strings of characters by parsing them through a sequence that is uniquely determined by each string.[5]

Table 2.1 Legend for the DFA states in Fig. 2.1

Symbol	Description
-	Initial State
ON	On State
PH	Preheating done
FA	Food Accepted
MA	Mode Accepted
TP	Temperature Set
TS	Timer Set
+	Final State

The DFA for the project is :


F,Ph,Mw,G,C,St,Tmp,Tmr
Switch
-
Switch
Switch
Switch
Switch
Switch
Switch
Switch
Ph,Tmp,Tmr,St,C,G,Mw
FA
PH
F
ON
Ph
F,Ph,Tmp,Tmr,St
Tmp,Tmr,St,C,Mw,Gr
C,G,Mw
F
MA
Tmp
F,Ph,Tmr,St,C,G,Mw
TP
Tmr
F,Ph,Tmp,St,C,G,Mw
TS
Switch
F,Ph,Tmp,Tmr,C,G,Mw
St
+
F,Ph,Tmp,Tmr,St,C,G,Mw

2.3 Data Dictionary

The data dictionary is an organized listing of all data elements that are pertinent to the system, with precise, rigorous definitions so that both user and system analyst will have a common understanding of inputs, outputs, components of stores and intermediate calculations.[2]

The data dictionary of the project is :

Table 2.2 Data Dictionary for SOMOTD - FSM

Field	Data type	Data format	Description
str	List	[a-zA-Z0-9,]*	Stores the Input String.
currState	String	[a-zA-Z]*	Stores the name of the current state
visibleState	Integer	[0-9]*	How many states are visible in the output
visibleInput	Integer	[0-9]*	How many input have been calculated till now
runClicked	Boolean	[true|false]	Whether run button is clicked or not

3.SOFTWARE REQUIREMENT SPECIFICATION

A Software Requirement Specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform.[2]

3.1 Overall Description

The purpose of the Software Requirements Specification document is to clearly define the system under development, namely 'SIMULATION OF MICROWAVE through DFA'. The intended audience of this document includes the development team such as the requirements team, requirements analyst, design team, and other members of the developing organisation.

3.1.1 Product Functions

Product Functions provides a summary of major functions that the software will perform.[2]

SIMULATION OF MICROWAVE is a GUI based web application created using Visual Studio Code. The application aims to :

- Simulate a Deterministic Finite Automata based on a simple Microwave using predefined input alphabets entered using buttons.
- To check whether the Finite Automata accepts the given input string or not.

3.1.2 User Characteristics

User Characteristics for the Project are :

- A user is anyone who wishes to simulate of a microwave.
- The amount of training needed by the user includes running HTML files and clicking buttons using a mouse.

3.1.3 General Constraints

The user interface will be intuitive enough so that no training is required by the users.

3.1.4 Assumptions and Dependencies

The Assumptions include :

- The User has basic knowledge of the functioning of a microwave.
- A steady supply of electricity.
- The User will run the HTML file on a PC.
- The HTML, CSS, JS files should be in the same folder.
- Only one food item can be prepared at one time.
- The leftover string is read until completely exhausted, once the system reaches either of the final states.
- Once the system reaches the final state, the input string is deemed to be ACCEPTED no matter what comes next except if the electricity is switched off.

The Dependencies include :

- A web browser must be installed in the system of the User.

3.2 External Interface Requirements

The user requires a computer system with basic input-output devices available such as a Monitor, mouse/touchpad and a Keyboard.

3.2.1 User Interface

A user interface is a method by which the user and the device exchange information and instructions.

- The user can input data through their Mouse to click the buttons.
- The user can use the monitor to analyse the displayed output.

3.2.2 Hardware Interface

Hardware Interface refers to the hardware required to access the application like processor, memory requirements, graphics processing unit, etc.

- At least 2GB RAM.
- At least 1MB Internal Storage Required.
- Any standard CPU and GPU capable of running the above specification.

3.2.3 Software Interface

A Browser is required to run the HTML file.

3.3 Performance Requirements

No such Performance requirements as such.

3.4 Functional Requirements

The simulated finite state machine software must include the following functions :

- Check if an input string is accepted by the machine or not.
- Allows the User to Preheat the oven.
- Allows the User to prepare food using the Grilling mechanism.
- Allows the User to prepare food using the Convection mechanism.
- Allows the User to prepare food using Microwaves.

4.ESTIMATIONS

4.1 Introduction

By its nature, engineering is a quantitative discipline. Product metrics help software engineers gain insight into the design and construction of the software they build by focusing on specific, measurable attributes of software engineering work products.[1]

4.2 Function Points

Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and qualitative assessments of software complexity.[1]

- External Inputs : Each external input originates from a user or is transmitted from another application and provides distinct application-oriented data or control information.

  Inputs are often used to update internal logical files (ILFs). Input should be distinguished from inquiries, which are counted separately.

- External Outputs : Each external output is derived data within an application that provides information to the user. In this context, external output refers to reports, screens, error messages, etc. Individual data items within the report are not counted separately.

- External Inquiries : An external inquiry is defined as an online input that results in the generation of some immediate software response in the form of an online output (often retrieved from an ILF).

- Internal Logical Files : Each internal logical file is a logical grouping of data that resides within the application's boundary and is maintained via external inputs.

- External Interface Files : Each external interface file is a logical grouping of data that resides external to the application but provides information that may be of use to the application.

Table 4.1 Computing Function Points - FSM

Information Domain Value	**Count**	Weighing factor			Net Count
		Optimal	Average	Complex	
External Inputs (EIs)	12	3	4	6	36
External Outputs (EOs)	1	4	6	8	4
External Inquiries (EQs)	3	3	4	6	12
Internal Logical Files (ILFs)	3	7	10	15	30
External Interface Files (EIFs)	0	5	7	10	0
				Count Total=	82

Table 4.2 The VAF for SOMOTD - FSM

The Value Adjustment Factors (VAF) based on responses to the following questions are:

S. No.	Question	Value Points
1.	Does the system require reliable backup and recovery?	0
2.	Are specialized data communications required to transfer information to or from the application?	0
3.	Are there distributed processing functions?	1
4.	Is performance critical?	2
5.	Will the system run in an existing, heavily utilized operational environment?	3
6.	Does the system require online data entry?	0
7.	Does the online data entry require the input transaction to be built over multiple screens or operations?	0
8.	Are the ILFs updated online?	0
9.	Are the inputs, outputs, files, or inquiries complex?	0
10.	Is the internal processing complex?	4
11.	Is the code designed to be reusable?	4
12.	Are conversion and installation included in the design?	1
13.	Is the system designed for multiple installations in different organizations?	5
14.	Is the application designed to facilitate change and ease of use by the user?	5

Therefore, the $\Sigma (F_i) = F_1 + F_2 + F_3 + F_4 + F_5 + F_6 + F_7 + F_8 + F_9 + F_{10} + F_{11} + F_{12} + F_{13} + F_{14}$

$= 0 + 0 + 1 + 2 + 3 + 0 + 0 + 0 + 0 + 4 + 4 + 1 + 5 + 5$

$= 25$

Hence, Function Points (FP) = Count Total $\times [0.65 + 0.01 \times \Sigma (F_i)]$

$= 82 \times [0.65 + 0.01 \times 25] = 82 \times 0.9 = 73.8$

4.3 Efforts

Efforts (E) = $\frac{FP}{P}$ person-months

where,

E = effort in person-months or person-years [1]

P = “productivity parameter” that reflects: overall process maturity and management practices, the extent to which good software engineering practices are used, the level of programming languages used, the state of the software environment, the skills and experience of the software team, and the complexity of the application.[1]

After careful estimation of the project, the value of the productivity parameter is designated as:

P = 16

Therefore, E = $\frac{FP}{P}$ person-months

= 73.8 / 16 person-months = 4.61 person-months

5.DESIGN

5.1 Input Alphabet

An alphabet is a finite, non-empty set of symbols called characters. Typically represented using the symbol Σ. A string over an alphabet Σ is a finite sequence of characters drawn from Σ.[3]

The set of input alphabet of this project is :

Σ = { Food (F), Preheat (Ph), Microwave (Mw), Grilling (G), Convection (C), St(Start), Tmp (Temperature), Tmr (Timer), Switch }

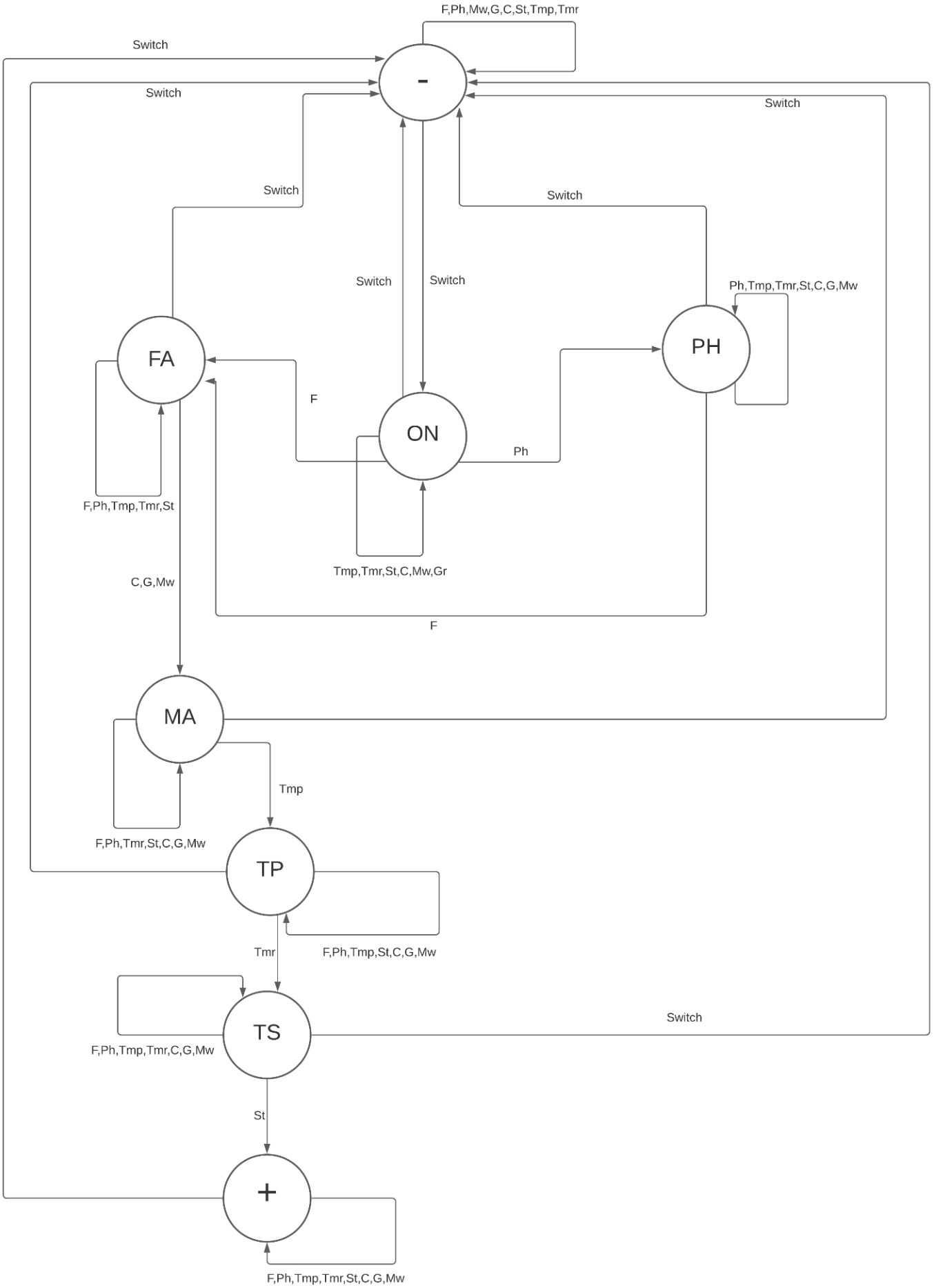
5.2 States and Transitions

The FSM states are as follows :

Table 5.1 State of SOMOTD- FSM

Symbol	Description
-	Initial State
ON	On State
PH	Preheating done
FA	Food Accepted
MA	Mode Accepted
TP	Temperature Set
TS	Timer Set
+	Final State

The DFA of the project is :


F,Ph,Mw,G,C,St,Tmp,Tmr
Switch
-
Switch
Switch
Switch
Switch
Switch
Switch
Ph,Tmp,Tmr,St,C,G,Mw
FA
PH
F
ON
Ph
F,Ph,Tmp,Tmr,St
Tmp,Tmr,St,C,Mw,Gr
C,G,Mw
F
MA
Tmp
F,Ph,Tmr,St,C,G,Mw
TP
Tmr
F,Ph,Tmp,St,C,G,Mw
TS
Switch
F,Ph,Tmp,Tmr,C,G,Mw
St
+
F,Ph,Tmp,Tmr,St,C,G,Mw

The Transition Table is as follows :

	Switch	F	Ph	Mw	G	C	Tmp	Tmr	St
-	ON	-	-	-	-	-	-	-	-
ON	-	FA	PH	ON	ON	ON	ON	ON	ON
PH	-	FA	PH	PH	PH	PH	PH	PH	PH
FA	-	FA	FA	MA	MA	MA	FA	FA	FA
MA	-	MA	MA	MA	MA	MA	TP	MA	MA
TP	-	TP	TP	TP	TP	TP	TP	TS	TP
TS	-	TS	TS	TS	TS	TS	TS	TS	+
+	-	+	+	+	+	+	+	+	+

6.SOURCE CODE

6.1 Prerequisites

The Prerequisites include :

1. Any latest web browser.

6.2 Code

6.2.1 HTML Code

```
<html>
	<head>
		<title>Microwave DFA</title>
		<link rel="stylesheet" href="styles.css" />
	</head>

	<body>
		<h2 class="Heading">
			<br /><br />
			<b> MICROWAVE DFA </b> <br /><br />
		</h2>
		<img src="DFA Microwave.png" alt="DFA-img" class="dfa-img" />
		<h3><b> LEGEND </b></h3>
		<br />
		<p class="fullform">
			<b> Alphabet Abbreviations : </b> <br />
			F - Food <br />
			Ph - Pre Heat <br />
			Mw - Microwave <br />
			G - Grilling <br />
			C - Convection <br />
			Tmp - Temperature <br />
			Tmr - Timer <br />
			St - Start
		</p>
		<br />
		<br />
		<p class="fullform">
			<b> State Abbreviations : </b> <br />
			ON - On State <br />
			FA - Food Accepted <br />
			PH - Pre-Heating Done <br />
			MA - Mode Accepted <br />
			TP - Temperature Set <br />
			TS - Timer Set
		</p>
		<br />
		<br />
		<br />
		<h2>Alphabets</h2>
```

```
		<div class="btn-grp">
			<button class="btn">Switch</button>
			<button class="btn">Ph</button>
			<button class="btn">F</button>
			<button class="btn">Mw</button>
			<button class="btn">G</button>
			<button class="btn">C</button>
			<button class="btn">Tmp</button>
			<button class="btn">Tmr</button>
			<button class="btn">St</button>
		</div>

		<h1>String :</h1>

		<a class="btn-run btn-main">RUN</a>
<a class="btn-clearone btn-main">CLEAR</a>
		<a class="btn-clear btn-main">CLEAR ALL</a>

		<div class="container">
			<div class="outer-div">
				<p class="self-arrow-p self-arrow-p-1"></p>
				<p class="self-arrow self-arrow-1 white-color">&#x21b7;</p>
				<div class="state state-1 white-color">
					<p class="state-p state-p-1"></p>
				</div>
			</div>
			<div class="input-div">
				<p class="input input-1"></p>
				<p class="arrow arrow-1 white-color">&#8594;</p>
			</div>
			<div class="outer-div">
				<p class="self-arrow-p self-arrow-p-2"></p>
				<p class="self-arrow self-arrow-2 white-color">&#x21b7;</p>
				<div class="state state-2 white-color">
					<p class="state-p state-p-2"></p>
				</div>
			</div>
			<div class="input-div">
				<p class="input input-2"></p>
				<p class="arrow arrow-2 white-color">&#8594;</p>
			</div>
			<div class="outer-div">
				<p class="self-arrow-p self-arrow-p-3"></p>
				<p class="self-arrow self-arrow-3 white-color">&#x21b7;</p>
				<div class="state state-3 white-color">
					<p class="state-p state-p-3"></p>
				</div>
			</div>
			<div class="input-div">
				<p class="input input-3"></p>
				<p class="arrow arrow-3 white-color">&#8594;</p>
			</div>
			<div class="outer-div">
				<p class="self-arrow-p self-arrow-p-4"></p>
				<p class="self-arrow self-arrow-4 white-color">&#x21b7;</p>
				<div class="state state-4 white-color">
					<p class="state-p state-p-4"></p>
				</div>
			</div>
```

```
<div class="input-div">
        <p class="input input-4"></p>
        <p class="arrow arrow-4 white-color">&#8594;</p>
</div>
<div class="outer-div">
        <p class="self-arrow-p self-arrow-p-5"></p>
        <p class="self-arrow self-arrow-5 white-color">&#x21b7;</p>
        <div class="state state-5 white-color">
                <p class="state-p state-p-5"></p>
        </div>
</div>
<div class="input-div">
        <p class="input input-5"></p>
        <p class="arrow arrow-5 white-color">&#8594;</p>
</div>
<div class="outer-div">
        <p class="self-arrow-p self-arrow-p-6"></p>
        <p class="self-arrow self-arrow-6 white-color">&#x21b7;</p>
        <div class="state state-6 white-color">
                <p class="state-p state-p-6"></p>
        </div>
</div>
<div class="input-div">
        <p class="input input-6"></p>
        <p class="arrow arrow-6 white-color">&#8594;</p>
</div>
<div class="outer-div">
        <p class="self-arrow-p self-arrow-p-7"></p>
        <p class="self-arrow self-arrow-7 white-color">&#x21b7;</p>
        <div class="state state-7 white-color">
                <p class="state-p state-p-7"></p>
        </div>
</div>
<div class="input-div">
        <p class="input input-7"></p>
        <p class="arrow arrow-7 white-color">&#8594;</p>
</div>
<div class="outer-div">
        <p class="self-arrow-p self-arrow-p-8"></p>
        <p class="self-arrow self-arrow-8 white-color">&#x21b7;</p>
        <div class="state state-8 white-color">
                <p class="state-p state-p-8"></p>
        </div>
</div>
<div class="input-div">
        <p class="input input-8"></p>
        <p class="arrow arrow-8 white-color">&#8594;</p>
</div>
<div class="outer-div">
        <p class="self-arrow-p self-arrow-p-9"></p>
        <p class="self-arrow self-arrow-9 white-color">&#x21b7;</p>
        <div class="state state-9 white-color">
                <p class="state-p state-p-9"></p>
        </div>
</div>
<div class="input-div">
        <p class="input input-8"></p>
        <p class="arrow arrow-9 white-color">&#8594;</p>
</div>
```

```
                        <div class="outer-div">
                                <p class="self-arrow-p self-arrow-p-10"></p>
                                <p class="self-arrow self-arrow-10 white-color">&#x21b7;</p>
                                <div class="state state-10 white-color">
                                        <p class="state-p state-p-10"></p>
                                </div>
                        </div>
                        <p class="foodstate"></p>
                </div>
                <footer>© Shivansh Tyagi, Sambit Basu, Sarvesh Sharma</footer>

                <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
                <script src="index.js"></script>
        </body>
</html>
```

6.2.2 CSS Code

```
* {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
}

body {
        font-size: 1rem;
        text-align: center;
        background-color: white;
}

.Heading {
        font-size: 35px;
        background-color: #000;
        color: #fff;
}
.dfa-img {
        width: 60%;
        height: auto;
        margin: 50px 20%;
}

h3 {
        font-size: 26px;
}
```

```
.fullform {
	font-size: 22px;
}

.btn-grp {
	width: 80%;
	margin: 40px auto;
}

.btn {
	margin: 0 15px;
	color: #1f1f1f;
	border: 2px solid #1f1f1f;
	padding: 18px 36px;
	display: inline-block;
	font-family: "Lucida Console", Monaco, monospace;
	font-size: 14px;
	letter-spacing: 1px;
	cursor: pointer;
	box-shadow: inset 0 0 0 0 white;
	transition: ease-out 0.4s;
}

.btn:hover {
	color: white;
	box-shadow: inset 0 100px 0 0 #1f1f1f;
}

h1 {
	font-size: 2.5rem;
	margin: 70px 0;
}

.btn-main {
	cursor: pointer;
	text-decoration: none;
	padding: 12px 18px;
   margin: 0 15px;
	font-size: 20px;
	font-family: "Lucida Console", Monaco, monospace;
	letter-spacing: 3px;
	transition: all 0.4s ease-in-out;
	border: solid 2px white;
	background: #1f1f1f;
	color: white;
}

.btn-main:hover {
	transition: all 0.4s ease-in-out;
	border: solid 2px #1f1f1f;
	background: white;
	color: #1f1f1f;
}

.container {
	width: 80%;
	padding: 50px 0 200px;
	margin: 50px auto;
}
```

```
.outer-div {
	display: inline-block;
	position: relative;
}

.self-arrow-p {
	position: absolute;
	top: -30px;
	left: 8px;
}

.self-arrow {
	font-size: 3rem;
	position: absolute;
	top: -30px;
	left: 7px;
}

.state {
	border: 2px solid black;
	border-radius: 40px;
	display: inline-block;
	height: 50px;
	width: 50px;
	position: relative;
	text-align: center;
}

.state-p {
	font-size: 3rem;
	height: 80%;
	margin-top: 10%;
	display: inline-block;
	font-weight: 100;
	position: absolute;
	top: -12px;
	left: 12px;
}

.named-state {
	font-size: 1rem;
	top: 7px;
}

.final-state {
	position: absolute;
	top: -10px;
	left: 10px;
}

.input-div {
	display: inline-block;
	height: 50px;
	width: 50px;
	position: relative;
}

.input {
```

```
        display: inline-block;
        position: absolute;
        top: 5px;
        left: 7px;
}

.arrow {
        font-size: 3rem;
        position: absolute;
        top: 1px;
}

.white-color {
        color: white;
        border-color: white;
}

footer {
        padding: 50px;
        background-color: #eee;
}

.foodstate {
   font-size: 30px;
   color: #000;
   transform: translateY(40px);
}
```

6.2.3 JavaScript Code

```
var str = [];
var currState = "-";
var visibleState = 1;
var visibleInput = 0;
var runClicked = false;

$("button").click(function () {
	if (!runClicked) {
		if (str.length < 9) {
			$("h1").html($("h1").html() + " " + this.innerHTML);
			str.push(this.innerHTML);
		}
	}
});

$(".btn-run").click(function () {
	$(".state-" + 1).removeClass("white-color");
	$(".state-p-" + 1).html("&#x2013;");
	if (!runClicked) {
		runClicked = true;
		for (var i = 0; i < str.length; i++) {
			if (currState === "-" && str[i] === "Switch") {
				currState = "ON";
				visibleState++;
				visibleInput++;
				$(".state-" + visibleState).removeClass("white-color");
				$(".state-p-" + visibleState).html("ON");
				$(".state-p-" + visibleState).addClass("named-state");
				$(".arrow-" + visibleInput).removeClass("white-color");
				$(".input-" + visibleInput).html("Switch");
			} else if (currState === "-" && str[i] != "Switch") {
				$(".self-arrow-" + visibleState).removeClass("white-color");
				$(".self-arrow-p-" + visibleState).html(
					$(".self-arrow-p-" + visibleState).html() + str[i]
				);
			} else if (currState === "ON" && str[i] === "Ph") {
				visibleState++;
				visibleInput++;
				currState = "PH";
				$(".state-" + visibleState).removeClass("white-color");
				$(".state-p-" + visibleState).html("PH");
				$(".state-p-" + visibleState).addClass("named-state");
				$(".arrow-" + visibleInput).removeClass("white-color");
				$(".input-" + visibleInput).html("Ph");
			} else if (currState === "ON" && str[i] === "F") {
				visibleState++;
				visibleInput++;
				currState = "FA";
				$(".state-" + visibleState).removeClass("white-color");
				$(".state-p-" + visibleState).html("FA");
				$(".state-p-" + visibleState).addClass("named-state");
				$(".arrow-" + visibleInput).removeClass("white-color");
				$(".input-" + visibleInput).html("F");
			} else if (
				currState === "ON" &&
```

```
                str[i] != "F" &&
                str[i] != "Switch" &&
                str[i] != "Ph"
        ) {
                var visibleSelfArrow = visibleInput + 1;
                $(".self-arrow-" + visibleSelfArrow).removeClass("white-color");
                $(".self-arrow-p-" + visibleSelfArrow).html(
                        $(".self-arrow-p-" + visibleSelfArrow).html() + str[i]
                );
        } else if (currState === "PH" && str[i] === "F") {
                visibleState++;
                visibleInput++;
                currState = "FA";
                $(".state-" + visibleState).removeClass("white-color");
                $(".state-p-" + visibleState).html("FA");
                $(".state-p-" + visibleState).addClass("named-state");
                $(".arrow-" + visibleInput).removeClass("white-color");
                $(".input-" + visibleInput).html("F");
        } else if (currState === "PH" && str[i] != "F" && str[i] != "Switch") {
                var visibleSelfArrow = visibleInput + 1;
                $(".self-arrow-" + visibleSelfArrow).removeClass("white-color");
                $(".self-arrow-p-" + visibleSelfArrow).html(
                        $(".self-arrow-p-" + visibleSelfArrow).html() + str[i]
                );
        } else if (
                currState === "FA" &&
                (str[i] === "C" || str[i] === "G" || str[i] === "Mw")
        ) {
                visibleState++;
                visibleInput++;
                currState = "MA";
                $(".state-" + visibleState).removeClass("white-color");
                $(".state-p-" + visibleState).html("MA");
                $(".state-p-" + visibleState).addClass("named-state");
                $(".arrow-" + visibleInput).removeClass("white-color");
                $(".input-" + visibleInput).html(str[i]);
        } else if (
                currState === "FA" &&
                str[i] != "C" &&
                str[i] != "Switch" &&
                str[i] != "Mw" &&
                str[i] != "G"
        ) {
                var visibleSelfArrow = visibleInput + 1;
                $(".self-arrow-" + visibleSelfArrow).removeClass("white-color");
                $(".self-arrow-p-" + visibleSelfArrow).html(
                        $(".self-arrow-p-" + visibleSelfArrow).html() + str[i]
                );
        } else if (currState === "MA" && str[i] === "Tmp") {
                visibleState++;
                visibleInput++;
                currState = "TP";
                $(".state-" + visibleState).removeClass("white-color");
                $(".state-p-" + visibleState).html("TP");
                $(".state-p-" + visibleState).addClass("named-state");
                $(".arrow-" + visibleInput).removeClass("white-color");
                $(".input-" + visibleInput).html("Tmp");
        } else if (currState === "MA" && str[i] != "Tmp" && str[i] != "Switch") {
                var visibleSelfArrow = visibleInput + 1;
```

```
                                        $(".self-arrow-" + visibleSelfArrow).removeClass("white-color");
                                        $(".self-arrow-p-" + visibleSelfArrow).html(
                                                $(".self-arrow-p-" + visibleSelfArrow).html() + str[i]
                                        );
                                } else if (currState === "TP" && str[i] === "Tmr") {
                                        visibleState++;
                                        visibleInput++;
                                        currState = "TS";
                                        $(".state-" + visibleState).removeClass("white-color");
                                        $(".state-p-" + visibleState).html("TS");
                                        $(".state-p-" + visibleState).addClass("named-state");
                                        $(".arrow-" + visibleInput).removeClass("white-color");
                                        $(".input-" + visibleInput).html("Tmr");
                                } else if (currState === "TP" && str[i] != "Tmr" && str[i] != "Switch") {
                                        var visibleSelfArrow = visibleInput + 1;
                                        $(".self-arrow-" + visibleSelfArrow).removeClass("white-color");
                                        $(".self-arrow-p-" + visibleSelfArrow).html(
                                                $(".self-arrow-p-" + visibleSelfArrow).html() + str[i]
                                        );
                                } else if (currState === "TS" && str[i] === "St") {
                                        visibleState++;
                                        visibleInput++;
                                        currState = "+";
                                        $(".state-" + visibleState).removeClass("white-color");
                                        $(".state-p-" + visibleState).html("&#43;");
                                        $(".arrow-" + visibleInput).removeClass("white-color");
                                        $(".input-" + visibleInput).html("St");
                                } else if (currState === "TS" && str[i] != "St" && str[i] != "Switch") {
                                        var visibleSelfArrow = visibleInput + 1;
                                        $(".self-arrow-" + visibleSelfArrow).removeClass("white-color");
                                        $(".self-arrow-p-" + visibleSelfArrow).html(
                                                $(".self-arrow-p-" + visibleSelfArrow).html() + str[i]
                                        );
                                } else if (str[i] === "Switch") {
                                        visibleState++;
                                        visibleInput++;
                                        currState = "-";
                                        $(".state-" + visibleState).removeClass("white-color");
                                        $(".state-p-" + visibleState).html("&#x2013;");
                                        $(".arrow-" + visibleInput).removeClass("white-color");
                                        $(".input-" + visibleInput).html("Switch");
                                } else if (currState === "+" && str[i] != "Switch") {
                                        var visibleSelfArrow = visibleInput + 1;
                                        $(".self-arrow-" + visibleSelfArrow).removeClass("white-color");
                                        $(".self-arrow-p-" + visibleSelfArrow).html(
                                                $(".self-arrow-p-" + visibleSelfArrow).html() + str[i]
                                        );
                                }
                        }
                }
});

$(".btn-clear").click(function () {
        str = [];
        $("h1").html("String : ");
        currState = "-";
        visibleState = 1;
        visibleInput = 0;
        runClicked = false;
```

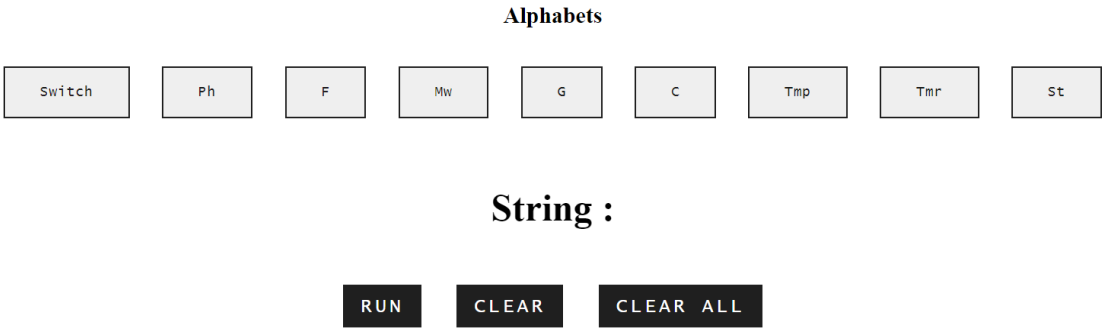
```
        $(".state").addClass("white-color");
        $(".self-arrow").addClass("white-color");
        $(".arrow").addClass("white-color");
        $(".self-arrow-p").html("");
        $(".input").html("");
});

$(".btn-clearone").click(function() {
        $("h1").html( $("h1").html().substring(0, $("h1").html().length - 1 - str[str.length - 1].length));
        str.pop();
})

setInterval(function() {
        if(currState === "+")
                $(".foodstate").html("Food Prepared");
        else
        $(".foodstate").html("Food Not Prepared");
}, 1000);
```

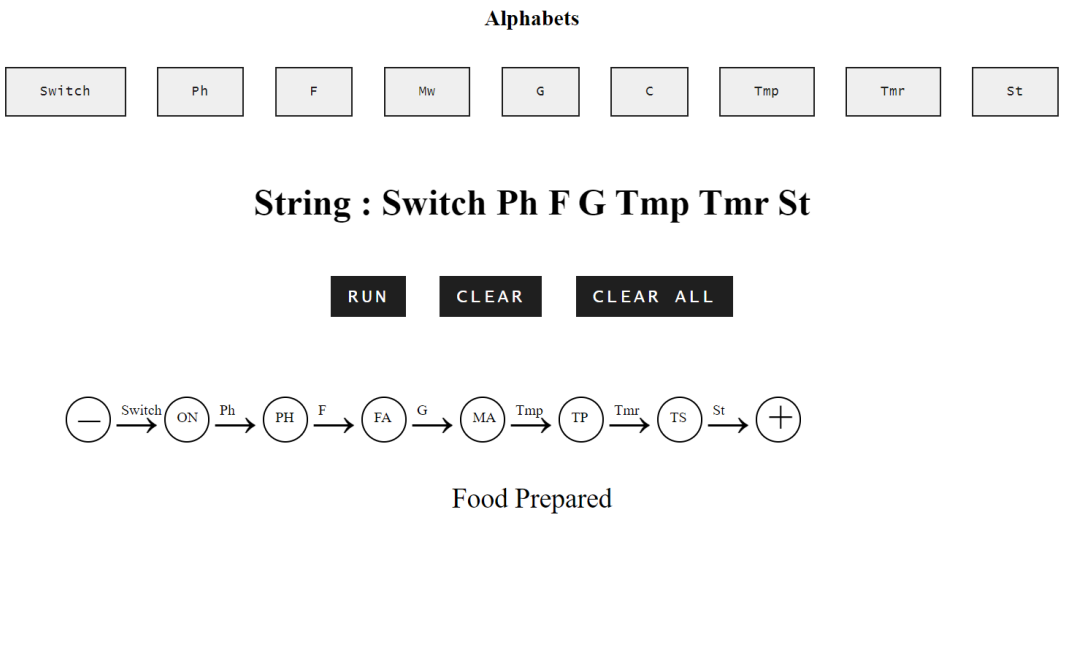
7.OUTPUT

Output 1:



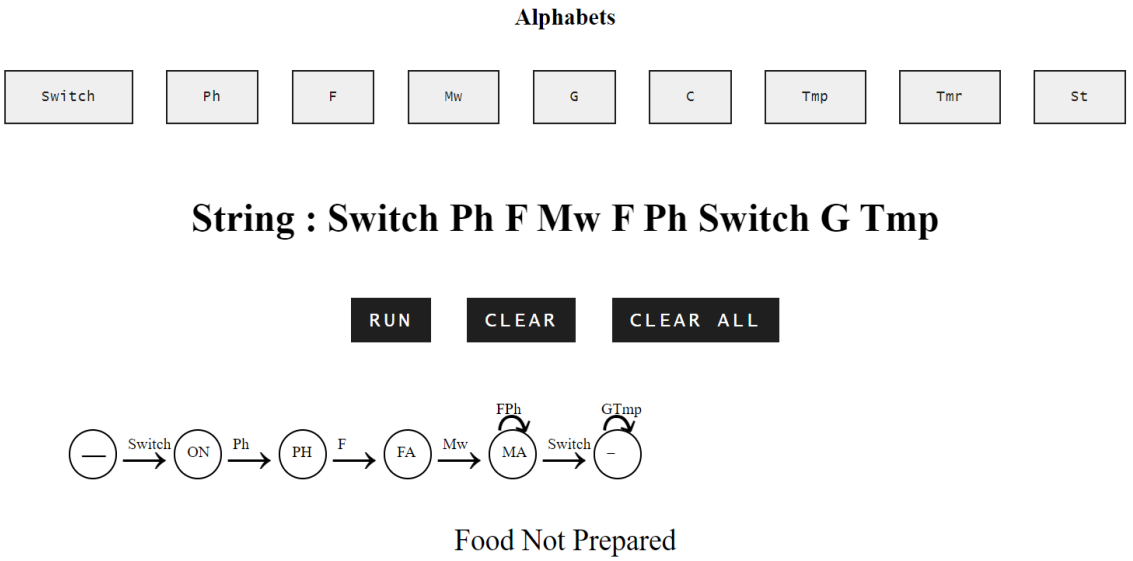
Food Not Prepared

Output 2:

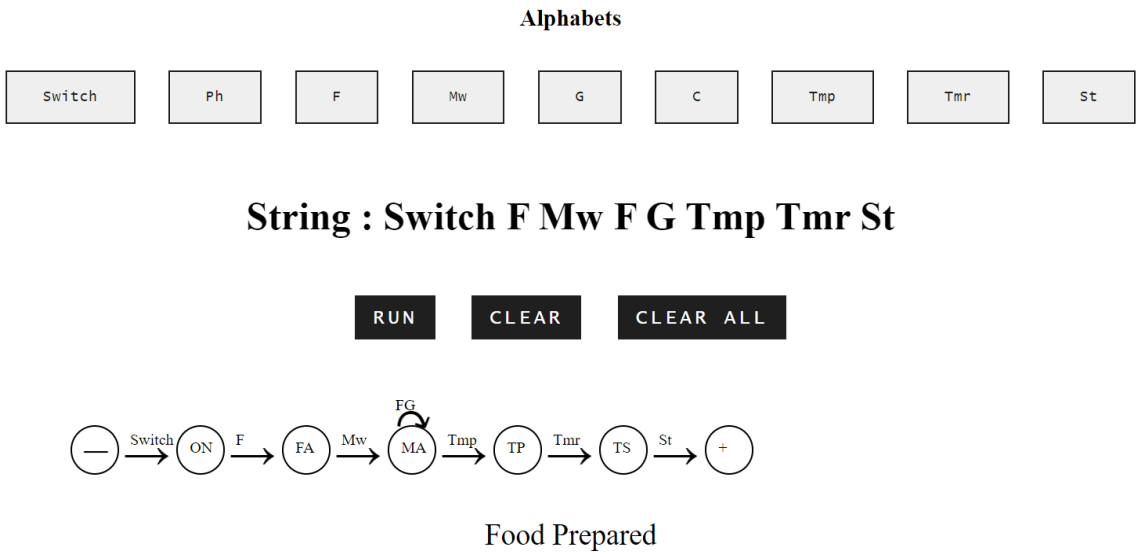


Food Prepared

Output 3:



Output 4:



A. REFERENCES

References

1. Software Engineering – A PRACTITIONER'S APPROACH EIGHTH EDITION by Roger S. Pressman, Ph.D., Bruce R. Maxim, Ph.D. - McGraw Hill Education
2. Software Engineering 5th Edition by K. K. Aggarwal & Yogesh Singh – New Age International Publishers
3. Introduction to Computer Theory Second Edition by Daniel I. A. Cohen
4. Helping websites - https://www.w3schools.com
5. What is Deterministic Finite Automata? - https://www.sisense.com/glossary/deterministic-finiteautomata/
6. Finite State Machine Designer - https://www.lucidchart.com/