lEstablished – 1961 Subject: **WEB PROGRAM**

SEVA SADAN'S

R. K. TALREJA COLLEGE

OF

ARTS, SCIENCE & COMMERCE

ULHASNAGAR – 421 003

CERTIFICATE

This is to certify that Mr./Ms. LEELAVATI SAW of F.Y. Information Technology (FYIT) Roll No. 2541040 has satisfactorily completed the Web Designing Mini Project entitled PORTFOLIO WITH INTERACTIVE DASHBOARD STYLE HOMEPAGE during the academic year 2025 – 2026, as a part of the practical requirement. The project work is found to be satisfactory and is approved for submission.

PROF. INCHARGE **HEAD OF DEPT**

SAHIL SHUKLA **LAXMI JESWANI**

INDEX

INTRODUCTION:

"**Initialize System:** Accessing the digital identity of **Leela Saw**. Currently operating within the **BSCFYIT** framework at RKT College, I am a dedicated Information Technology student specialized in mastering the architecture of modern software systems. My technical evolution is built on a rigorous foundation of **C Programming** and **Object-Oriented Programming (C++)**, where I bridge the gap between abstract logic and high-performance deployment. Beyond core coding, my focus extends to algorithm optimization, data structure efficiency, and the principles of scalable system design. This dashboard serves as a real-time uplink to my technical deployments, representing a relentless commitment to continuous learning, problem-solving, and the pursuit of engineering excellence within the global tech ecosystem. System status: **Optimized and evolving.**"

REQUIREMENT SPECIFICATION :

1. Functional Requirements (What the system does)
These are the specific behaviors of the software. For your dashboard, these would be:

- **Navigation:** The system must allow users to jump between the Main Frame, Identity, and Deployments.
- **Real-time Clock:** The system must display a live 24-hour clock updated every second.
- **Project Linking:** The system must provide external uplinks to GitHub repositories.

2. Non-Functional Requirements (How the system performs)
These define the quality attributes of the software:

- **Performance:** The dashboard must load within 2 seconds.
- **Usability:** The interface must follow a "Cyberpunk/Alpha" aesthetic with a custom cursor for user immersion.
- **Portability:** The code must be responsive and viewable on both mobile and desktop browsers.

3. Hardware & Software Requirements
The environment needed to run or develop the project:

- **Software:** VS Code, Git, Chrome/Edge Browser.
- **Languages:** HTML5, CSS3 (including Flexbox/Grid), and Vanilla JavaScript.

SYSTEM DESIGN:

1. High-Level Design (HLD)

This focuses on the overall architecture and how different components interact.

- **Architecture Patterns:** Choosing between Monolithic (all in one), Microservices (modular), or Client-Server (like your portfolio).
- **Data Flow:** Mapping how information moves from a user input to the database and back.
- **System Components:** Identifying external APIs, databases, and servers.

2. Low-Level Design (LLD) / Detailed Design

This is where your **OOPs** skills shine. It involves designing the actual code structure.

- **Class Diagrams:** Defining classes, their attributes, and their relationships (Inheritance, Association, Aggregation).
- **Logic Design:** Writing pseudocode or flowcharts for complex algorithms.
- **Database Schema:** Designing tables, primary keys, and foreign key relationships.

🔑 System Design Principles (SOLID)

As a developer, you use these principles to ensure your system is easy to maintain:

1. **Single Responsibility:** Each class/module should do one thing only.

2. **Open/Closed:** Software entities should be open for extension but closed for modification.
3. **Liskov Substitution:** Derived classes must be substitutable for their base classes.
4. **Interface Segregation:** Don't force a class to implement methods it doesn't use.
5. **Dependency Inversion:** Depend on abstractions, not concretions.

SYSTEM IMPLEMENTATION:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<title>Leela Saw | BSCFYIT Alpha Dashboard</title>

<link
href="https://fonts.googleapis.com/css2?family=Orbitron
:wght@400;700&family=Plus+Jakarta+Sans:wght@300;
400;600&display=swap" rel="stylesheet">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css">

<style>
:root{
--glow:#00f2ff;
--primary:#6366f1;
--dark-bg:#030712;
--card-glass:rgba(15,23,42,0.8);
--border:rgba(255,255,255,0.1);
}
*{margin:0;padding:0;box-sizing:border-
box;cursor:none;}
body{
background:var(--dark-bg);
```

```
color:white;
font-family:'Plus Jakarta Sans',sans-serif;
height:100vh;
display:flex;
overflow:hidden;
}
#cursor{
width:20px;height:20px;
border:2px solid var(--glow);
border-radius:50%;
position:fixed;
pointer-events:none;
z-index:9999;
}
.bg-animate{
position:fixed;top:0;left:0;width:100%;height:100%;
z-index:-1;
background:radial-gradient(circle at 50% 50%, #1e1b4b
0%, #030712 100%);
}
nav{
width:280px;
background:rgba(15,23,42,0.95);
backdrop-filter:blur(10px);
border-right:1px solid var(--border);
display:flex;
flex-direction:column;
padding:2rem;
justify-content:space-between;
}
.nav-brand{
```

```
font-family:'Orbitron',sans-serif;
font-size:1.5rem;
color:var(--glow);
text-shadow:0 0 10px var(--glow);
}
.nav-item{
padding:15px;margin:10px 0;border-radius:12px;
display:flex;align-items:center;gap:15px;
color:#94a3b8;transition:0.4s;list-style:none;
}
.nav-item:hover,.nav-item.active{
background:rgba(99,102,241,0.2);
color:white;border-left:4px solid var(--glow);
}
main{flex:1;padding:40px;overflow-y:auto;}
.section{display:none;}
.section.active{display:block;}
.top-bar{
display:flex;justify-content:space-between;
align-items:center;margin-bottom:30px;
border-bottom:1px solid var(--border);
padding-bottom:20px;
}
#clock{
font-family:'Orbitron',sans-serif;
color:var(--glow);
}
.grid{
display:grid;
grid-template-columns:repeat(3,1fr);
gap:20px;
```

```
}
.card{
background:var(--card-glass);
border:1px solid var(--border);
padding:25px;border-radius:20px;
transition:0.5s;
}
.card:hover{
border-color:var(--glow);
transform:translateY(-8px);
}
.profile-container{
display:flex;align-items:center;gap:40px;
margin-bottom:40px;
}
.profile-hex{
width:200px;height:200px;
background:url('https://images.unsplash.com/photo-
1633356122544-
f134324a6cee?q=80&w=400&auto=format&fit=crop')
center/cover;
clip-path:polygon(25% 0%,75% 0%,100% 50%,75%
100%,25% 100%,0% 50%);
border:4px solid var(--glow);
}
.skill-row{margin-top:15px;}
.bar-bg{background:#111827;height:10px;border-
radius:5px;margin-top:5px;}
.bar-fill{
height:100%;border-radius:5px;
background:linear-gradient(90deg,var(--primary),var(--
```

```
glow));
transition:2s width ease-in-out;
}
.social-btn{
display:flex;align-items:center;gap:10px;
padding:15px;background:rgba(255,255,255,0.05);
border-radius:10px;text-decoration:none;color:white;
margin-top:10px;border:1px solid transparent;
}
.social-btn:hover{border-color:var(--glow);}
</style>
</head>

<body>

<div id="cursor"></div>
<div class="bg-animate"></div>

<nav>
<div>
<div class="nav-brand">LEELA_OS</div>
<p style="font-size:0.7rem;color:var(--glow);letter-
spacing:2px;">V.2.0.6 RELOADED</p>
<ul style="margin-top:40px;">
<li class="nav-item active" onclick="jump('home')"><i
class="fas fa-chart-line"></i> Main Frame</li>
<li class="nav-item" onclick="jump('about')"><i
class="fas fa-biohazard"></i> Identity</li>
<li class="nav-item" onclick="jump('projects')"><i
class="fas fa-layer-group"></i> Deployments</li>
<li class="nav-item" onclick="jump('contact')"><i
```

```
class="fas fa-satellite-dish"></i> Uplink</li>
</ul>
</div>
<div class="card" style="padding:15px;text-
align:center;font-size:0.8rem;">
System Status: <span
style="color:#4ade80;">ONLINE</span>
</div>
</nav>

<main>
<div class="top-bar">
<h2>DASHBOARD // <span id="section-title">MAIN
FRAME</span></h2>
<div id="clock"></div>
</div>

<!-- HOME -->
<section id="home" class="section active">
<div class="profile-container">
<div class="profile-hex"></div>
<div>
<h1 style="font-size:4rem;font-
family:'Orbitron';">LEELA SAW</h1>
<p style="color:var(--glow);font-
size:1.2rem;">BSCFYIT</p>
</div>
</div>

<div class="grid">
```

```
<div class="card">
<h3>Skills</h3>
<div class="skill-row">
<p>C Programming <span
style="float:right">85%</span></p>
<div class="bar-bg"><div class="bar-fill"
style="width:85%"></div></div>
</div>
<div class="skill-row">
<p>OOPS C++ Programming <span
style="float:right">80%</span></p>
<div class="bar-bg"><div class="bar-fill"
style="width:80%"></div></div>
</div>
</div>

<div class="card">
<h3>Academic Trace</h3>
<p style="margin-top:15px;">Course: BSCFYIT</p>
<p>Year: 1st Year (FY)</p>
<p>Focus: Information Technology</p>
</div>

<div class="card">
<h3>Portfolio</h3>
<p style="margin-top:15px;">Personal Portfolio
Dashboard</p>
<p style="font-size:0.8rem;text-align:center;">Active
Projects: 12</p>
</div>
```

```
</div>
</section>

<!-- IDENTITY -->
<section id="about" class="section">
<div class="card">
<h2 style="color:var(--glow);">BSCFYIT
PROFILE</h2>
<p style="margin-top:20px;line-height:1.8;font-
size:1.05rem;">
I am currently pursuing <b>Bachelor of Science in
Information Technology (BSCFYIT)</b>.
This program focuses on programming, database
management, networking,
web development, software engineering, and system
design.
As a FYIT student, I am building strong fundamentals in
C programming,
Object Oriented Programming (C++), and modern
development practices.
My goal is to become a skilled software developer
capable of building
efficient and scalable applications.
</p>
</div>
</section>

<!-- PROJECTS -->
<section id="projects" class="section">
<div class="grid">
```

```
<div class="card">
<h4>01. C Programming</h4>
<p style="font-size:0.85rem;color:#94a3b8;margin-
top:10px;">
GitHub Deployment:
</p>
<a href="https://github.com/leela321/C-programing-
project-.git" target="_blank" style="color:var(--glow);">
View Repository
</a>
</div>

<div class="card">
<h4>02. C++ Project</h4>
<p style="font-size:0.85rem;color:#94a3b8;margin-
top:10px;">
GitHub Deployment:
</p>
<a href="https://github.com/leela321/Oops-C-project-
.git" target="_blank" style="color:var(--glow);">
View Repository
</a>
</div>

</div>
</section>

<!-- CONTACT -->
<section id="contact" class="section">
<div class="card" style="max-width:600px;">
<h2>ESTABLISH CONNECTION</h2>
```

```
<a href="https://instagram.com/officialleela10"
target="_blank" class="social-btn">
<i class="fab fa-instagram" style="color:#e1306c;"></i>
IG // @officialleela10
</a>

<a href="mailto:sawleelavati@gmail.com" class="social-
btn">
<i class="fas fa-envelope" style="color:var(--
glow);"></i> EMAIL // sawleelavati@gmail.com
</a>

<a href="https://github.com/leela321" target="_blank"
class="social-btn">
<i class="fab fa-github"></i> GITHUB // leela321
</a>

</div>
</section>

</main>

<script>
const cursor=document.getElementById('cursor');
document.addEventListener('mousemove',(e)=>{
cursor.style.left=e.clientX+'px';
cursor.style.top=e.clientY+'px';
});
function jump(id){
document.querySelectorAll('.section').forEach(s=>s.class
```

```
List.remove('active'));
document.querySelectorAll('.nav-
item').forEach(n=>n.classList.remove('active'));
document.getElementById(id).classList.add('active');
event.currentTarget.classList.add('active');
document.getElementById('section-
title').innerText=id.toUpperCase().replace('HOME','MAI
N FRAME');
}
function updateClock(){
const now=new Date();
document.getElementById('clock').innerText=
now.getHours().toString().padStart(2,'0')+":"
+now.getMinutes().toString().padStart(2,'0')+":"
+now.getSeconds().toString().padStart(2,'0');
}
setInterval(updateClock,1000);
updateClock();
</script>

</body>
</html>
```

SYSTEM TESTING AND RESULT:

1. Requirement Specification (The "What")
This is the "Source of Truth." Without a clear SRS (Software Requirement Specification), projects suffer from "scope creep" (where the project grows out of control).

- **User Requirements:** Natural language descriptions of what the user expects (e.g., "The user must be able to calculate interest on their savings").
- **System Requirements:** Technical specifications for the developer.
    - **External Interface Requirements:** How your program interacts with hardware, other programs, or users (e.g., "The C++ program must interface with a .txt file for data persistence").
- **Constraints:** Limits placed on the design (e.g., "The system must run on Linux-based environments with less than 256MB of RAM").

---

2. System Design (The "How")
System Design is where you make decisions that are hard to change later. It is divided into two major "blueprints."
High-Level Design (HLD)

- **Data Storage:** Deciding between a Relational Database (SQL) for structured data like bank accounts, or NoSQL for flexible data.
- **Modules:** Breaking the system into independent pieces. For your dashboard, the "Navigation Logic"

is one module, and the "Clock Engine" is another.

Low-Level Design (LLD)

This is where your **OOPs (Object-Oriented Programming)** skills are used:

- **Entity-Relationship (ER) Diagrams:** Modeling how data entities (like Student and Course) relate to each other.
- **Access Modifiers:** Deciding which data is private (Encapsulation) to prevent unauthorized corruption of variables.
- **Logic Flow:** Using Flowcharts to map out "If-Then-Else" logic before writing a single line of C code.

---

3. System Testing (The "Validation")

Testing proves that your design actually satisfies your requirements.

- **Alpha Testing:** Conducted by the developer (you) in the development environment.
- **Beta Testing:** Conducted by a small group of external users before the final "Uplink" (Deployment).
- **Stress Testing:** Pushing the system beyond its limits. (e.g., What happens to your C program if the user enters a string when an integer is expected? This tests your **Exception Handling**).
- **User Acceptance Testing (UAT):** The final stage where the "client" verifies that the system solves the problem it was intended to solve.

RESULT:

https://leela-2210.netlify.app

FUTURE SCOPE AND CONCLUSION:

1. Technical Scalability (The Next Stack)

The "Future Scope" of your current skill set involves moving from local console applications to distributed, intelligent systems.

- **From C++ to Cloud Architecture:** Integrating C++ backend logic with Cloud platforms (AWS/Azure) for high-performance computing.
- **Transition to Frameworks:** Moving from Vanilla JS to **React.js** or **Next.js** for more complex dashboard states, and from C++ to **Python/Java** for enterprise-level backend services.
- **Database Integration:** Shifting from flat-file storage in your C programs to **Real-time Databases** (Firebase/MongoDB) or **Relational Databases** (PostgreSQL) for large-scale data management.

---

2. Advanced Integration (AI & Automation)

Future iterations of your projects won't just follow static logic; they will predict and adapt.

- **AI Implementation:** Integrating Machine Learning models into your C++ architectures for predictive data analysis.
- **Automated Testing (CI/CD):** Moving from manual testing to **Automated Pipelines**. Every time you push code to GitHub, a system will automatically run your "Requirement Specifications" tests.

- **IoT (Internet of Things):** Using your C/C++ knowledge to program hardware and sensors, connecting physical devices to your LEELA_OS dashboard.

---

3. Professional & Industry Impact

The future scope for a **BSCFYIT** graduate involves aligning with the **Global Tech Ecosystem**.

- **Microservices Architecture:** Designing software as a collection of small, independent services rather than one giant program.
- **Cybersecurity focus:** Implementing advanced encryption protocols and "Zero Trust" architectures within your design specifications.
- **Open Source Contribution:** Transitioning from personal repositories to contributing to global projects, influencing the tools used by millions of developers.

Conclusion

The **LEELA_OS** dashboard represents a holistic integration of the core pillars of software engineering. By strictly adhering to the **Software Development Life Cycle (SDLC)**, the system successfully transitions from abstract **Requirement Specifications** to a tangible, high-performance **System Design** anchored in C and C++ methodologies. The rigorous **System Testing** phase ensures that every functional module—from real-time logic to external GitHub uplinks—operates with

maximum reliability and security. While this deployment achieves its current technical objectives, the defined **Future Scope** ensures the architecture remains forward-compatible with emerging technologies like Cloud computing and AI-driven automation. This dashboard is not merely a terminal of past achievements but a stable, verified foundation for a career dedicated to engineering excellence and continuous innovation within the global Information Technology ecosystem. **System Status: Mission Complete. Terminal: Standby.**

REFERENCE:

Visual Studio Code – Used for writing, editing, and managing the website source code.

Gemini AI – Used for idea generation, concept understanding, and guidance during development.

Google Chrome – Used for running, testing, and debugging the portfolio website.

GLOSSARY:

- **Abstraction:** The process of hiding complex implementation details and showing only the necessary features of an object.

- **Black-Box Testing:** A method of software testing that examines the functionality of an application without peering into its internal structures or workings.

- **Class:** A blueprint or prototype from which objects are created in Object-Oriented Programming (OOPs).

- **Encapsulation:** The bundling of data and the methods that operate on that data into a single unit (Class), restricting direct access to some of the object's components.

- **Inheritance**: A mechanism where a new class (derived) acquires the properties and behaviors of an existing class (base).

- **Polymorphism**: The ability of a function or object to take on multiple forms, typically through method overloading or overriding.

- **Regression Testing:** Re-running functional and non-functional tests to ensure that previously developed and tested software still performs after a change.

 SDLC (Software Development Life Cycle): A structured process used by the IT industry to design, develop, and test high-quality software.

 SRS (Software Requirements Specification): A formal document that describes the features and behavior of a software application.