WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Guntakala Lavanya

Email: mailto:lavanyag1@udayton.edu

Short-bio: full stack developer interested in creating web applications.

Figure 1: lavanya's headshot

Overview

This project involved creating a professional portfolio website using front-end web development skills and deploying it on GitHub Pages. The project required integrating JavaScript functionalities, public APIs, and cookie-based visit tracking, along with presenting my resume, skills, and experience for potential employers. I used a Bootstrap-based template to style the website and incorporated dynamic content using JavaScript.

Website URL: https://yourusername.github.io/
GitHub Project Folder: https://github.com/yourusername/your-repo

Through this project, I learned how to combine different technologies—HTML, CSS, JavaScript, Bootstrap, jQuery, and third-party APIs—to build an interactive and responsive website hosted in the cloud.

General Requirements

Personal Website and Resume (25 pts)

I used a professional Bootstrap resume template and customized it with my name, education, skills, work experience, projects, and a downloadable resume button.

Bootstrap Template used : https://github.com/startbootstrap/startbootstrap-resume

!General Requriements

Course Page with Link (5 pts)

I added a separate HTML page `waph.html` that introduces the Web Application Programming and Hacking course and includes brief descriptions of relevant hands-on projects.

View WAPH Page

Non-Technical Requirements

Bootstrap Template and Employer-Focused Content (20 pts)

The portfolio uses a Bootstrap-based responsive template. I designed the content and styling to match what potential employers would expect in a professional online profile.

Page Tracker

I integrated FlagCounter to show visitor count statistics:

```
<!-- Flag Counter -->
<a href="https://info.flagcounter.com/XXXX">
  <img src="https://s11.flagcounter.com/count2/XXXX/bg_FFFFFF/txt_000000/border_CCCCCC/colum
</a>
```

Technical Requirements (50 pts)

This section summarizes the JavaScript features and API integrations implemented in my personal portfolio website.

JavaScript Features (20 pts)

1. Digital Clock using jQuery (5 pts) A live digital clock updates every second using `setInterval()` and jQuery to display current local time dynamically.

```
$(document).ready(function () {
  setInterval(() => {
    const now = new Date();
    $('#digit-clock').text("Current time: " + now.toLocaleTimeString());
  }, 1000);
});
```

Analog Clock using Course-Provided JS (5 pts)

The analog clock is rendered using the canvas API and functions from the course's provided script clock.js.

```
<canvas id="analog-clock" width="150" height="150" style="background-color: #999;"></canvas
<script src="https://waph-phung.github.io/clock.js"></script>
<script>
  var canvas = document.getElementById("analog-clock");
  var ctx = canvas.getContext("2d");
  var radius = canvas.height / 2;
  ctx.translate(radius, radius);
  radius = radius * 0.90;
  setInterval(() => {
    drawFace(ctx, radius);
    drawNumbers(ctx, radius);
    drawTime(ctx, radius);
  }, 1000);
</script>
```

Show/Hide Email (5 pts)

This function toggles the visibility of the email link when the text "Show my email" is clicked.

```
<a id="email" onclick="showhideEmail()">Show my email</a>
<a id="email-link" href="mailto:lavanyag1@udayton.edu" style="display:none;">lavanyag1@udayt
<script>
function showhideEmail() {
  $('#email-link').toggle();
}
</script>
```

Age Calculator (5 pts)

A form that calculates and displays the user's age based on a selected date of birth.

```
<input type="date" id="dob">
<button onclick="calculateAge()">Calculate Age</button>
<p id="ageOutput"></p>
<script>
function calculateAge() {
  const dob = new Date(document.getElementById("dob").value);
  const diff = Date.now() - dob.getTime();
  const ageDate = new Date(diff);
  const age = Math.abs(ageDate.getUTCFullYear() - 1970);
  document.getElementById("ageOutput").innerText = `You are ${age} years old.`;
}
</script>
```

Public API Integrations (20 pts)

1. JokeAPI Integration (10 pts) Fetches a random joke from JokeAPI every 60 seconds using fetch().
2. XKCD Comic API Integration (10 pts) Fetches and displays the latest XKCD comic using the xkcd.now.sh proxy API. Disclaimer: The joke and comic content above are fetched from third-party APIs (JokeAPI and XKCD). I do not control or take responsibility for the content shown.

Cookie-Based Visit Tracker (10 pts)

JavaScript cookies are used to detect new vs. returning visitors. The last visit time is stored and updated automatically.

```
function setCookie(name, value, days) {
  const date = new Date();
  date.setTime(date.getTime() + (days*24*60*60*1000));
  const expires = "expires=" + date.toUTCString();
  document.cookie = name + "=" + value + ";" + expires + ";path=/";
}

function getCookie(name) {
  const cname = name + "=";
  const decodedCookie = decodeURIComponent(document.cookie);
  const ca = decodedCookie.split(';');
  for(let i = 0; i < ca.length; i++) {
    let c = ca[i].trim();
    if (c.indexOf(cname) === 0) {
      return c.substring(cname.length, c.length);
    }
  }
  return "";
}

function showVisitPopup() {
  const lastVisit = getCookie("lastVisit");
  const now = new Date().toLocaleString();

  if (!lastVisit) {
    alert("Welcome to my homepage for the first time!");
  } else {
    alert(" Welcome back! Your last visit was on: " + lastVisit);
  }

  setCookie("lastVisit", now, 30);
}

window.onload = showVisitPopup;
```

All code is integrated within the #Additional Tasks section of my deployed GitHub Pages website.