

Front-End UI/UX Development

Front-End UI/UX Development Mini Project

UI/UX DESIGN FUNDAMENTALS

Submitted by:

SR.NO	NAME	REGISTER NUMBER	COLLEGE EMAIL ID	CLASS
1.	JOANN BINNY	2462088	joann.binny@btech.christuniversity.in	3BTCS AIML C
2.	D PRANAYA	2462059	d.pranaya@btech.christuniversity.in	3BTCS AIML C
3.	DEEKSHA PAUL	2462062	deeksha.paul@btech.christuniversity.in	3BTCS AIML C

Instructor Name: Mr. Narendra

Institution Name: Christ (Deemed to be University),Bangalore

Date of Submission:26/09/2025

RESUME BUILDER WEB APPLICATION

ABSTRACT:

This project, the **Aesthetic Resume Builder**, is a responsive, feature-rich web application designed to empower users in creating modern, professional, and customizable resumes. Developed using a **Front-End UI/UX Development** stack, the application provides a seamless experience by utilizing a **live preview** interface that updates instantly as the user enters data. Key innovations include a dedicated **ATS (Applicant Tracking System) Mode** for maximum parsing compatibility, dynamic theming with custom accent colors, and a built-in **Keyword Analyzer** to optimize content against a specific job description. The application ensures data portability through JSON import/export and robust local storage for version management.

OBJECTIVES:

The primary objectives of this project were to:

1. **Develop a Highly Interactive User Interface:** Create a single-page application where data entry and resume visualization occur simultaneously, maximizing user efficiency and experience.
2. **Ensure Modern Customization:** Implement dynamic styling features, including multiple **templates** (Classic, Compact, Executive), a light/dark **theme toggle**, and a custom accent color selector.
3. **Achieve ATS Compliance:** Design a toggleable **ATS Mode** that strips aesthetic elements (like images and meters) and ensures clean, structured text output preferred by recruitment parsing software.
4. **Implement Data Persistence:** Provide functionality to **save, load, import, and export** resume data (JSON) locally, ensuring users do not lose their work and can transfer data easily.
5. **Add Value-Added Features:** Integrate tools like a **Recruiter Score** and a **Keyword Analyzer** to help users optimize their resume content for specific job roles.

SCOPE OF THE PROJECT:

The project is exclusively a **client-side, front-end application**.

- **In-Scope:** HTML structure, CSS styling, core JavaScript logic for data manipulation, dynamic rendering, user input parsing, local storage management, theme control, and specialized feature implementation (ATS Mode, Analyzer). Integration of external libraries like Bootstrap 5 and jQuery for UI/UX elements.
- **Out-of-Scope:** Server-side database integration, user authentication, or server-rendered dynamic content. The application relies entirely on the client's browser and Local Storage for persistence.

TOOLS & TECHNOLOGIES USED:

Category	Technology	Purpose
Markup	HTML5	Provides the semantic structure for the application and form inputs.
Styling	CSS3 / PostCSS	Handles all visual presentation, including modern design, responsive layout, and variable-based theming.
Frameworks	Bootstrap 5	Used for rapid UI layout, responsiveness, and pre-styled components (buttons, forms, modals).
Scripting	JavaScript (ES6+)	The core logic for data binding, resume rendering, feature implementation, and DOM manipulation.
DOM Helper	jQuery 3.7.1	Used extensively for DOM selection, manipulation, event handling, and simplified AJAX-style interactions (local file loading).
Architecture	Next.js	(Inferred from package.json) Provides the foundational environment for a modern React-based application setup.
Component Lib.	Radix UI Components	(Inferred from package.json) A library of unstyled, accessible components used for sophisticated UI elements.

HTML STRUCTURE OVERVIEW:

The HTML file establishes a two-column, responsive layout:

1. **Header (app-header):** Contains the project title and all global controls:
 - Template Selector (Classic, Compact, Executive).
 - Section Toggles (Summary, Skills, Experience, etc.).
 - Theme/Accent Color controls and the **ATS Mode** switch.
 - Action buttons (Save, Load, Import, Export, Print/PDF).
2. **Main Content (app-main):** Divided into two panels:
 - **Left Panel (Input):** A comprehensive form organized into basic details, links, summary, and specific sections (Skills, Experience, etc.). Also includes the Job Description input for the Keyword Analyzer.
 - **Right Panel (Preview):** The resumePreview container where the dynamic resume output is rendered, including the **Recruiter Score** display.
3. **Modal (saveLoadModal):** A Bootstrap modal for managing saved resume versions.

CSS STYLING STRATEGY:

The CSS employs a modular, variable-driven approach for a highly customizable look:

1. **Theming System:** Uses the :root pseudo-class and the html[data-theme="light/dark"] selector to define CSS variables (--bg, --surface, --text, --accent). This enables instant light/dark theme switching and dynamic accent color updates via JavaScript.
2. **Layout & Components:** Styles for the application chrome (app-header, card-panel) and standard form elements (using Bootstrap overrides).
3. **Resume Preview Styles:** Defines the aesthetics for the generated resume (styled-resume), including header layout, section titles, and item metadata.
4. **Dynamic Elements:** Implements styling for **Animated Skill/Language Meters** (meter, meter-fill) with a CSS transition for smooth visual feedback.
5. **ATS Mode:** Uses the .ats-mode class to globally override visual styles:
 - Forces a **white background and black text** for optimal parsing.
 - Hides aesthetic elements (resume-avatar, .meter).
 - Ensures URLs are explicitly visible by showing the hidden .link-url elements.
6. **Print Media Queries:** Hides all application chrome (.app-header, .card-panel, buttons) and removes borders/backgrounds on the preview for a clean, print-ready output (Save as PDF)

JavaScript Structure Overview (app.js)

The JavaScript file contains the application's core logic, divided into distinct functional blocks:

1. **Data Model:** `getEmptyResumeData()` defines the structured JSON format for the resume data (basics, skills, experience, etc.).
2. **Form Management:**
 - `renderForm()`: Dynamically generates the form fields and initializes event listeners.
 - `readFormIntoModel()`: Parses the raw multiline textarea inputs (e.g., skills, experience) into the structured `currentResumeData` model.
3. **Preview Rendering:** `renderResumePreview(animate)` is the core function. It reads the model, applies the selected template class, filters sections based on user toggles, constructs the HTML for the resume, and finally calls `animateMeters()` if needed.
4. **Feature Logic:**
 - `animateMeters()`: Uses `requestAnimationFrame` and CSS transitions for the visual skill meter animation.
 - `analyzeKeywords()`: Compares job description text against resume content to calculate a keyword match percentage and update the **Recruiter Score**.
 - `autoBulletize()`: A utility function to format sentence-based detail inputs into neat, bulleted lists using pipes and delimiters.
5. **Persistence & Actions:** Contains functions for `saveVersion()`, `loadVersionsList()`, `exportJSON()`, and `importJSONFile()`, all utilizing the browser's **Local Storage** for saving and Blob/URL objects for file operations.

KEY FEATURES:

Feature	Description
Live Dynamic Preview	Instantaneous rendering of resume changes alongside data input.
ATS Mode Toggle	One-click switch to an ATS-friendly, plain-text format that maximizes parsability by recruitment software.
Keyword Analyzer	Compares skills and text from the resume model against a pasted job description, calculating a match percentage.
Animated Skill Meters	Visual, dynamic progress bars for skills and languages that animate upon form application.
Version Control (Save/Load)	Local storage persistence allowing users to save multiple resume versions and load them from a modal interface.
Dynamic Theming	User-controlled light/dark theme toggle and a color picker for selecting the resume's accent color.
Data Portability	Supports JSON export of the entire resume data model and seamless JSON import.
Recruiter Score	A scoring metric calculated based on content completion and keyword match to gauge resume strength.

Challenges Faced and Solutions :

Challenge	Description	Solution Implemented
Smooth Meter Animation	The browser often applies CSS width changes instantaneously, preventing a smooth transition effect.	Used <code>requestAnimationFrame</code> and a forced reflow (<code>void document.body.offsetHeight</code>) to ensure the meters transition smoothly from 0% to their target width.
Parsing Complex Text Input	Users enter multi-line sections (Experience, Skills) using various separators (newlines, pipes, dashes, periods).	Developed dedicated <code>parseSkillLines()</code> , <code>parseLinkPairs()</code> , and <code>readFormIntoModel()</code> functions that use string splitting and <code>map</code> operations to robustly parse text into structured JSON arrays.
ATS Incompatibility	Visual elements like custom fonts, colors, and images can break ATS software parsing, despite looking good visually.	Implemented a dedicated CSS class (<code>.ats-mode</code>) and a JavaScript switch that overrides all aesthetic styles to force a pure black-and-white, unstyled layout, while also ensuring links are fully visible as required by ATS.

Outcomes:

The project successfully delivered a complete, modern, and highly functional resume building tool that addresses both aesthetic appeal and technical compatibility:

- **Successful Feature Delivery:** All key features, including live preview, ATS mode, theming, and the keyword analyzer, were implemented and integrated into a cohesive user interface.
- **Enhanced User Experience:** The two-pane, live update design eliminates the need for manual refreshing, providing immediate feedback on all changes.
- **High Customization:** The combination of multiple templates, theme options, and section toggles allows users to generate a wide range of professional documents from a single data source.
- **Practical Utility:** The keyword analysis and ATS modes provide tangible benefits, helping users create resumes that are not only visually appealing but also strategically optimized for the modern job application process.

Future Enhancements:

Server-Side Persistence: Migrate from Local Storage to a cloud-based database (e.g., Firebase, MongoDB) to allow users to access their resumes across different devices.

Advanced NLP for Keywords: Integrate a true Natural Language Processing (NLP) API to improve the keyword analyzer, enabling it to match synonyms, related terms, and contextual relevance, not just exact token matches.

PDF Generation: Implement server-side rendering (using tools like Puppeteer or an API) to generate high-fidelity, high-resolution PDF files, improving on the current browser-based "Print to PDF" functionality.

More Templates and Layouts: Develop a wider variety of professional template layouts, perhaps leveraging the inferred Next.js/React structure for component-based template management.

Sample Code:

```
1 <!DOCTYPE html>
2 <html lang="en" data-theme="light">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
6   <title>Aesthetic Resume Builder</title>
7
8   <!-- Bootstrap 5 + Icons -->
9   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"/>
10  <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.css" rel="stylesheet"/>
11
12  <!-- App CSS -->
13  <link href="./styles.css" rel="stylesheet"/>
14 </head>
15 <body>
16   <div class="app-container">
17     <!-- Header / Chrome -->
18     <header class="app-header">
19       <div class="container-fluid">
20         <div class="d-flex justify-content-between align-items-center py-3 flex-wrap gap-3">
21           <div class="d-flex align-items-center gap-3">
22             <h1 class="app-title mb-0">Resume Builder</h1>
23           </div>
24
25           <div class="header-controls d-flex align-items-center flex-wrap gap-2">
26             <!-- Template -->
27             <div class="input-group input-group-sm w-auto">
28               <span class="input-group-text">Template</span>
29               <select id="templateSelect" class="form-select">
30                 <option value="classic">Classic</option>
31                 <option value="compact">Compact</option>
32                 <option value="executive">Executive</option>
33
34               <input class="form-check-input" type="checkbox" id="atsMode" />
35               <label class="form-check-label" for="atsMode">ATS Mode</label>
36             </div>
37
38             <!-- Actions -->
39             <div class="btn-group btn-group-sm ms-1">
40               <button id="saveBtn" class="btn btn-success" title="Save version"><i class="bi bi-save"></i></button>
41               <button id="loadBtn" class="btn btn-info" data-bs-toggle="modal" data-bs-target="#saveLoadModal" title="Open saved versions">
42                 <i class="bi bi-filetype-json"></i></button>
43               <input id="importFile" type="file" accept="application/json" class="d-none" />
44               <button id="importBtn" class="btn btn-secondary" title="Import JSON"><i class="bi bi-filetype-json"></i></button>
45               <button id="exportBtn" class="btn btn-warning" title="Export JSON"><i class="bi bi-download"></i></button>
46               <button id="printBtn" class="btn btn-primary" title="Print / Save as PDF"><i class="bi bi-printer"></i></button>
47             </div>
48           </div>
49         </div>
50       </div>
51     </div>
52
53     <!-- Main -->
54     <main class="app-main">
55       <div class="container-fluid">
56         <div class="row g-3">
57           <!-- Left: Data Entry -->
58           <div class="col-lg-5 col-xl-4">
59             <div class="card-panel">
60               <h5 class="panel-title">Your Details</h5>
61             </div>
62           </div>
63         </div>
64       </div>
65     </main>
66   </div>
67 </body>
68 </html>
```



```

        <button class= "btn btn-secondary btn-sm" data-bs-dismiss= "modal" >Close</button>
      </div>
    </div>
  </div>
</div>

<!-- JS -->
<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script src= ". /app.js"></script>
</body>
</html>

```

CSS & JavaScript:

```

/* Dark/light + accent system */
:root{
  --bg:#0f1420;
  --surface:#151c2f;
  --border:#293657;
  --text:#e8eefc;
  --muted:#9aa6cb;
  --accent:#6366f1;
  --chip:#1f2a44;
  --paper:#0b1120;
}
html[data-theme="light"]{
  --bg:#ffffff;
  --surface:#f7f8fb;
  --border:#e2e6ef;
  --text:#0a0f1e;
  --muted:#4b5670;
  --paper:#ffffff;
}

body{
  background:var(--bg);
  color:var(--text);
  min-height:100vh;
  font-family: ui-sans-serif, system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial;
}

/* Chrome */
.app-header{ background:var(--surface); border-bottom:1px solid var(--border); }
.app-title{ font-size:1.2rem; font-weight:800; letter-spacing:.2px; }
.header-controls .form-select, header-controls .form-control-color{ min-width: 120px; }

```

```

.card-panel{ background:var(--surface); border:1px solid var(--border); border-radius:16px; padding:18px; }
.panel-title{ margin-bottom:12px; font-weight:700; }

.form-label{ color:var(--muted); }
.form-control, .form-select, .btn{ border-radius:10px; }
.small-muted{ color:var(--muted); }

/* Preview */
.resume-preview{
  background:var(--paper);
  border:1px dashed var(--border);
  border-radius:16px; padding:28px;
  min-height: 320px;
  position: relative;
}

/* super-light sample overlay for empty preview */
.preview-empty-hint{
  position:absolute; inset:0;
  display:flex; align-items:center; justify-content:center;
  color:var(--muted);
  opacity:.25; pointer-events:none; text-align:center;
  padding:10px 16px; font-size:.95rem;
}

.score-chip{ background: var(--chip); border:1px solid var(--border); padding:.15rem .5rem; border-radius:999px; }

/* Resume blocks */
.styled-resume .resume-header{ border-bottom:2px solid var(--accent); padding-bottom:10px; margin-bottom:14px; }
.styled-resume .resume-name{ font-size:1.6rem; font-weight:900; letter-spacing:.2px; margin:0; }
.styled-resume .resume-role{ font-weight:700; color:var(--accent); }
.styled-resume .contact-line{ color:var(--muted); }

```

```

.styled-resume .contact-line{ color:var(--muted); }
.section-title{ font-size:1.05rem; font-weight:800; margin:16px 0 8px; color:var(--accent); }
.section-item{ margin-bottom:10px; }
.section-item .title{ font-weight:700; }
.section-item .meta{ color:var(--muted); font-size:.95rem; }

/* Avatar */
.resume-avatar{ width:90px; height:90px; object-fit:cover; border-radius:999px; border:2px solid var(--accent); }

/* Templates */
.template-compact .styled-resume .resume-name{ font-size:1.35rem; }
.template-compact .section-title{ margin:12px 0 6px; }

.template-executive .styled-resume .resume-header{
  border-bottom:3px solid var(--accent);
  display:flex; align-items:center; gap:16px;
}
.template-executive .styled-resume .resume-role{ color:var(--text); opacity:.85; font-weight:600; }

/* Skill/Lang Meters */
.meter{
  width:100%;
  height:10px;
  background:transparent;
  border:1px solid var(--border);
  border-radius:999px;
  overflow:hidden;
}
.meter-fill{
  height:100%;
  width:0; /* start collapsed */
  background:var(--accent);
  transition: width .9s ease; /* smoother animation */
  will-change: width;
}

/* Respect OS reduced motion */
@media (prefers-reduced-motion: reduce){
  .meter-fill{ transition:none; }
}

/* ATS Mode (text-only) */
.ats-mode .resume-preview{ background:#fff !important; color:#000 !important; border-color:#000 !important; }
.ats-mode .styled-resume .resume-header{ border-color:#000 !important; }
.ats-mode .section-title{ color:#000 !important; }
.ats-mode a{ color:#000 !important; text-decoration: none; }
.ats-mode .resume-avatar{ display:none !important; }
.ats-mode .contact-line .link-url{ display:inline !important; }
.ats-mode .meter{ display:none !important; } /* hide visual meters */
.ats-mode .skill-val, .ats-mode .lang-val{ display:inline !important; } /* show percent text clearly */

/* Photo picker thumbnail */
.avatar-thumb{ width:72px; height:72px; border-radius:999px; object-fit:cover; border:1px solid var(--border); }

/* Print */
@media print{
  .app-header, .card-panel .panel-title, .btn, .input-group, .form-check, .form-control, .form-select { display:none !important; }
  body{ background:#fff; color:#000; }
  .resume-preview{ border:none; padding:0; }
}

```

```

let currentResumeData = null;
const LS_KEY = "resume_versions_cool";

/* ----- Data Model ----- */
function getEmptyResumeData() {
  return {
    basics: { name:"", role:"", email:"", phone:"", location:"", links:[], avatar:"", summary:"" },
    skills: [], // [{name, level}] 0..100
    languages: [], // [{name, level}] 0..100
    experience: [], // [{role, company, period, details}]
    projects: [], // [{name, link, details, image}]
    education: [], // [{degree, school, year}]
    certifications:[],// [{name, org, year}]
    achievements:[] // ["..."]
  };
}

/* ----- Utils ----- */
const esc = (s='') => s.replace(/[/[&<'"/]/g, c => ({'&':'&amp;','<':'&lt;','>':'&gt;','"':'&quot;','\':'&#39;'}[c]));

/* ----- Form Rendering ----- */
function renderForm(){
  const b = currentResumeData.basics;

  const html = `
    <!-- Photo & Basics -->
    <div class="row g-2 align-items-center mb-2">
      <div class="col-auto">
        
      </div>
  
```

```

    </div>
    <div class="col">
      <label class="form-label">Profile Photo</label>
      <div class="d-flex gap-2">
        <button id="pickAvatar" class="btn btn-outline-secondary btn-sm"><i class="bi bi-image"></i> Choose</button>
        <button id="clearAvatar" class="btn btn-outline-danger btn-sm"><i class="bi bi-x"></i> Remove</button>
        <input id="avatarInput" type="file" accept="image/*" class="d-none" />
      </div>
      <div class="small-muted mt-1">Optional; hidden in ATS mode.</div>
    </div>
  </div>

  <div class="mb-2">
    <label class="form-label">Name</label>
    <input id="f_name" type="text" class="form-control" placeholder="Your full name" value="">
  </div>
  <div class="mb-2">
    <label class="form-label">Role / Title</label>
    <input id="f_role" type="text" class="form-control" placeholder="e.g., Frontend Developer" value="">
  </div>

  <div class="row g-2">
    <div class="col-12 col-md-6">
      <label class="form-label">Email</label>
      <input id="f_email" type="email" class="form-control" placeholder="you@example.com" value="">
    </div>
    <div class="col-12 col-md-6">
      <label class="form-label">Phone</label>
      <input id="f_phone" type="text" class="form-control" placeholder="+91 9xxxxxxx" value="">
    </div>
  </div>

```

```

  <div class="mb-3">
    <label class="form-label">Experience (one per line: Role @ Company | Period | Details)</label>
    <textarea id="f_exp" class="form-control" rows="4" placeholder="Frontend Intern @ Acme | Jun 2024-Aug 2024 | Built UI components; i
  </div>

  <!-- Projects (with optional image URL) -->
  <div class="mb-3">
    <label class="form-label">Projects (one per line: Name | Link | Details | ImageURL)</label>
    <textarea id="f_projects" class="form-control" rows="3" placeholder="Recipe App | https://repo | React + Node app with filters... |
  </div>

  <!-- Education -->
  <div class="mb-3">
    <label class="form-label">Education (one per line: Degree | School | Year)</label>
    <textarea id="f_ed" class="form-control" rows="3" placeholder="B.Tech CSE (AI/ML) | Christ University | 2027"></textarea>
  </div>

  <!-- Certifications -->
  <div class="mb-3">
    <label class="form-label">Certifications (one per line: Name | Org | Year)</label>
    <textarea id="f_certs" class="form-control" rows="2" placeholder="AWS Cloud Practitioner | AWS | 2024"></textarea>
  </div>

  <!-- Achievements -->
  <div>
    <label class="form-label">Achievements (one per line)</label>
    <textarea id="f_ach" class="form-control" rows="2" placeholder="Winner, SDG-7 Hackathon 2025"></textarea>
  </div>

  <button id="applyBtn" class="btn btn-outline-primary mt-3 w-100">
    <i class="bi bi-magic"></i> Apply to Preview
  </button>

```



```

    $("#formSections").html(html);
    bindFormEvents();
}

/* ----- Bind form controls ----- */
function bindFormEvents(){
    $("#pickAvatar").on("click", () => $("#avatarInput").trigger("click"));
    $("#clearAvatar").on("click", () => {
        currentResumeData.basics.avatar = "";
        $("#avatarThumb").attr("src", "");
        renderResumePreview();
    });
    $("#avatarInput").on("change", function(){
        const f = this.files?.[0];
        if(!f) return;
        const r = new FileReader();
        r.onload = e => {
            currentResumeData.basics.avatar = String(e.target.result || "");
            $("#avatarThumb").attr("src", currentResumeData.basics.avatar);
            renderResumePreview();
        };
        r.readAsDataURL(f);
        this.value = "";
    });

    $("#f_summary").on("input", function(){
        const len = (this.value||"").trim().length;
        $("#sumCount").text(len ? `· ${len} chars` : "");
    });
}

```

```

    $("#applyBtn").on("click", () => {
        readFormIntoModel();
        renderResumePreview(true); // animate meters on apply
    });
}

/* ----- Parse helpers ----- */
function parseSkillLines(raw){
    if(!raw) return [];
    return raw.split("\n").map(l=>{
        const [name, levelRaw] = l.split("|").map(x => (x||"").trim());
        if(!name) return null;
        let level = parseInt(String(levelRaw||"").replace("%","").trim(),10);
        if (isNaN(level)) level = 60; // default if not provided
        level = Math.max(0, Math.min(100, level));
        return {name, level};
    }).filter(Boolean);
}

function parseLinkPairs(raw){
    return raw
        ? raw.split(",").map(s => s.trim()).filter(Boolean).map(pair => {
            const [label,url] = pair.split("|").map(x => (x||"").trim());
            return (label||url) ? {label,url} : null;
        }).filter(Boolean)
        : [];
}

/* ----- Model ----- */
function readFormIntoModel(){
    const b = currentResumeData.basics;
}

```

```

/* ----- Preview Rendering ----- */
function renderResumePreview(animate=false){
  const mount = $("#resumePreview").empty();

  // Template
  const template = $("#templateSelect").val() || "classic";
  mount.removeClass("template-compact template-executive");
  if (template === "compact") mount.addClass("template-compact");
  if (template === "executive") mount.addClass("template-executive");

  // Section visibility
  const selected = Array.from($("#sectionToggles option")).filter(o => o.selected).map(o => o.value);
  const show = key => selected.includes(key);

  const hasAny =
    (currentResumeData.basics.name || currentResumeData.basics.role || currentResumeData.basics.summary ||
     (currentResumeData.skills[[]].length || (currentResumeData.experience[[]].length ||
      (currentResumeData.projects[[]].length || (currentResumeData.education[[]].length ||
       (currentResumeData.certifications[[]].length || (currentResumeData.achievements[[]].length ||
        (currentResumeData.languages[[]].length));

  if(!hasAny){
    mount.append('<div class="preview-empty-hint">
      (very light sample) Add your details on the left, then click "Apply to Preview".<br/>
      Tip: try "Skills → React || 85%" to see animated meters.
    </div>');
  }

  const root = $('<div class="styled-resume"></div>');

```

```

b.name = $("#f_name").val().trim();
b.role = $("#f_role").val().trim();
b.email = $("#f_email").val().trim();
b.phone = $("#f_phone").val().trim();
b.location = $("#f_location").val().trim();
b.summary = $("#f_summary").val().trim();
b.links = parseLinkPairs($("#f_links").val().trim());

currentResumeData.skills = parseSkillLines($("#f_skills").val().trim());
currentResumeData.languages = parseSkillLines($("#f_langs").val().trim());

const expRaw = $("#f_exp").val().trim();
currentResumeData.experience = expRaw ? expRaw.split("\n").map(l => {
  const [roleCompany, period, details] = l.split("|").map(x => (x||"").trim());
  const [role, company] = (roleCompany||"").split("@").map(x => x.trim());
  return (role||company||period||details) ? { role:(role||"").trim(), company:(company||"").trim(), period:period||"", details:details||""} : null;
}).filter(Boolean) : [];

const projRaw = $("#f_projects").val().trim();
currentResumeData.projects = projRaw ? projRaw.split("\n").map(l => {
  const [name, link, details, img] = l.split("|").map(x => (x||"").trim());
  return (name||link||details||img) ? { name, link, details, image: img||"" } : null;
}).filter(Boolean) : [];

const edRaw = $("#f_ed").val().trim();
currentResumeData.education = edRaw ? edRaw.split("\n").map(l => {
  const [degree, school, year] = l.split("|").map(x => (x||"").trim());
  return (degree||school||year) ? { degree, school, year } : null;
}).filter(Boolean) : [];

const certRaw = $("#f_certs").val().trim();
currentResumeData.certifications = certRaw ? certRaw.split("\n").map(l => {
  const [name, org, year] = l.split("|").map(x => (x||"").trim());
  return (name||org||year) ? { name, org, year } : null;
}).filter(Boolean) : [];

const achRaw = $("#f_ach").val().trim();
currentResumeData.achievements = achRaw ? achRaw.split("\n").map(s => s.trim()).filter(Boolean) : [];
}

/* ----- Animate helpers ----- */
function animateMeters() {
  // Set all fills to 0% first
  $(".skill-row .meter-fill, .lang-row .meter-fill").each(function(){
    this.style.width = "0%";
  });

  // Force reflow so the browser registers width:0 before we set the target
  // eslint-disable-next-line no-unused-expressions
  void document.body.offsetHeight;

  // Next animation frame: set target widths (guarantees transition)
  requestAnimationFrame(() => {
    $(".skill-row .meter-fill").each(function(i){
      const pct = currentResumeData.skills?.[i]?.level || 0;
      this.style.width = pct + "%";
    });
    $(".lang-row .meter-fill").each(function(i){
      const pct = currentResumeData.languages?.[i]?.level || 0;
      this.style.width = pct + "%";
    });
  });
}

```

Output:

File

C:\Users\Pranay\Downloads\index%20(1).html

☆

🔍

🔖

🔖

Resume Builder

Template: Executive

Sections: Summary


ATS Mode

📄

🔗

🔗

Your Details



Profile Photo

Choose Remove

Optional: hidden in ATS mode.

Name

RICHARD

Role / Title

AIML ENGINEER

Email

Ricard.klen@gmail.com

Phone

8766335212

Location

Bengaluru

Links (label,url, comma separated)

#=random+photo&frame=tee_+26869_3pc

c&type=E210IN885G919538imgurl=https%3A%2F%2Fdata.pic.ai%2Fimages%2FYbKH-QrQY2nVXX_1712121324.png?id=28&url


Summary

2-3 line professional summary

Skills (one per line: Name | Level%)

Live Preview

Recruiter Score 65/100



RICHARD

AIML ENGINEER

Ricard.klen@gmail.com • 8766335212 • Bengaluru

https://in.imagesearch.yahoo.com/search/images?to=random+photo&frame=tee_+26869_3pc&type=E210IN885G919538imgurl=https%3A%2F%2Fdata.pic.ai%2Fimages%2FYbKH-QrQY2nVXX_1712121324.png?id=28&url

Achievements

• Winner, SDG Hackathon

Export JSON

Skills (one per line: Name | Level%)

java
python

Use 0–100. % is optional. Bars animate.

Languages (one per line: Language | Proficiency%)

English
Hindi

Experience (one per line: Role @ Company | Period | Details)

frontend developer at ITC

Projects (one per line: Name | Link | Details | ImageURL?)

Recipe App | <https://repo> | React + Node app with filters... | <https://.../thumb.png>

Education (one per line: Degree | School | Year)

B.Tech in CSE(AI/ML)

Certifications (one per line: Name | Org | Year)

AWS cloud computing
<https://.../thumb.png>

Education (one per line: Degree | School | Year)

B.Tech in CSE(AI/ML)

Certifications (one per line: Name | Org | Year)

AWS cloud computing

Achievements (one per line)

Winner, SDG Hackathon

Apply to Preview

Job Description

Paste the JD here for keyword analysis

Analyze Keywords

Auto-bulitize

Match: 0%

Found: —

Conclusion:

The **Aesthetic Resume Builder** project successfully utilized core web technologies (HTML, CSS, JavaScript, and jQuery) to deliver a powerful, interactive, and highly valuable utility. By focusing on user experience and strategic features like ATS compatibility and keyword optimization, the application meets the evolving needs of job seekers in the digital age. The modular design, clean code structure, and use of CSS variables ensure the project is maintainable and easily extensible for future development.

References:

L&T LMS : <https://learn.intedutech.com/Landing/MyCourse>

HOLIDAY PLANNER WEB APPLICATION

ABSTRACT:

This project, the **Holiday Planner Web Application**, is a fully responsive, client-side tool designed to help users organize and budget their travel plans. Built using **HTML5, CSS3, and Vanilla JavaScript** with **Bootstrap 5**, it offers travel suggestions, a detailed trip planning form, and persistence via **Local Storage**. The application calculates the budget balance for each saved trip and features an aesthetically pleasing design with modern CSS animations.

OBJECTIVES:

The primary goals for developing this application were:

1. **Develop an Intuitive User Interface:** To create a visually clean and responsive layout using Bootstrap 5 that ensures a smooth planning experience on any device.
2. **Implement Financial Tracking:** To establish a mechanism where users can input both a target Planned Budget and Amount Spent to calculate and display the crucial Remaining Balance for each saved trip.
3. **Ensure Client-Side Data Persistence:** To use Vanilla JavaScript and the browser's Local Storage to save and retrieve all trip records, allowing users to access their plans across sessions without requiring a server or login.
4. **Provide Engagement Features:** To include an initial Travel Suggestions section, giving users visual inspiration and quick starting points for their own planning

SCOPE OF THE PROJECT:

The Holiday Planner is defined as a **purely client-side, single-page application (SPA)**.

- **In-Scope:** The design and implementation of the four main UI sections (Hero, Suggestions, Planner Form, and Saved Trips List). All data management, including form collection, budget calculation, and read/write operations for **Local Storage**. Responsive design using Bootstrap 5 and custom CSS.
- **Out-of-Scope:** Any form of server-side data persistence (e.g., database integration), user authentication, or integration with external APIs for live data (e.g., currency conversion or flight prices).

TOOLS & TECHNOLOGIES USED:

Category	Technology	Purpose
Markup/Structure	HTML5	Provides the semantic foundation for the application's distinct sections, including the main form (<code>#tripForm</code>) and display areas (<code>#tripsList</code>).
Styling	CSS3	Handles all visual aesthetics, including the custom Poppins font import, card styling, and the use of <code>@keyframes</code> for UI entry animations.
Frameworks	Bootstrap 5 (CDN)	Utilized for a robust, mobile-first responsive grid system , pre-styled input fields, buttons, and the main container layout.
Scripting	JavaScript (Vanilla)	Contains the core logic for event handling, budget arithmetic, dynamic DOM manipulation, and the persistence layer using the browser's native Local Storage .

HTML STRUCTURE OVERVIEW:

The HTML file establishes the foundational content and links the necessary resources (Bootstrap and custom files). It is organized into distinct, easy-to-manage sections:

1. **Header and Links:** Includes the document metadata, title, and links to the Bootstrap CDN and the custom style.css file.
2. **Hero Section (div.hero):** The introductory, full-width banner containing the main title and a catchy tagline, designed to be visually prominent upon loading.
3. **Travel Suggestions Section (#suggestions):** A content block that displays hardcoded visual ideas to inspire the user. The suggestionsList div serves as the render target for these suggestion cards.
4. **Plan a Trip Section (#planner):** Hosts the central user input area, featuring the main `#tripForm` with fields for the destination, two financial inputs (Planned Budget and Amount Spent), members, and notes.
5. **Saved Trips Section (#savedTrips):** This area is initially empty and contains the `#tripsList` container, which is dynamically populated by JavaScript with the saved trip cards.

CSS STYLING STRATEGY:

The stylesheet is responsible for the application's clean, modern aesthetic:

1. **Typography and Base:** Imports the **Poppins** font from Google Fonts and defines the primary color palette and body background (#f4f9f9).
2. **Hero Styling:** Implements a dramatic background using a URL image combined with a **linear-gradient overlay** for text contrast. Crucially, it defines and applies **@keyframes animations** (fadeIn and slideDown) to introduce the title and tagline smoothly.
3. **Component Styling:** Applies a consistent style to cards and forms with large **border-radius (15px)** and subtle box-shadow for a lifted, material design feel. A transition and transform on the **.card:hover** state provide interactive feedback.

JavaScript Structure Overview (app.js)

The Vanilla JavaScript file governs all interactive functionality and data management:

1. **Data Persistence Layer:** Defines STORAGE_KEY and two primary utility functions:
 - **getTrips():** Retrieves the JSON string from Local Storage, parses it, and returns the array of trip objects.
 - **saveTrips(trips):** Takes the current array of trips, serializes it to a JSON string, and stores it in Local Storage.
2. **Form Submission Handler:** An event listener attached to #tripForm executes on submission. It **prevents the default browser action** (e.preventDefault()), collects all field values, performs the critical budget arithmetic to calculate the **balance** property, and packages the data into a new trip object before calling addTrip().
3. **Rendering Engine:** renderTrips() is the core display function. It fetches the latest trip array, clears the existing HTML in #tripsList, and iterates through the array to construct and inject new HTML div.card elements for each trip.
4. **Initialization:** The script uses document.addEventListener("DOMContentLoaded", renderTrips) to ensure that any previously saved trips are loaded and displayed as soon as the page structure is ready.

KEY FEATURES:

Feature	Description
Financial Budget Tracking	The application automatically computes the financial Balance (<code>Planned Budget - Amount Spent</code>) when a trip is saved, providing instant accountability.
Local Storage Persistence	All trip data is saved directly to the user's browser, meaning plans are retained even after the user closes and reopens the browser.
Dynamic Content Rendering	Saved trip details, including the calculated balance, are converted into visually distinct, dynamically generated cards and appended to the <code>#tripsList</code> container.
Interactive UI	The use of CSS animations in the hero section and hover effects on the suggestion cards creates a modern, engaging user experience.
Pre-Populated Suggestions	A set of six hardcoded travel ideas (e.g., Beach Paradise, Mountain Adventure) provides immediate, inspiring content for the user.

Challenges Faced and Solutions :

Challenge	Description	Solution Implemented
Maintaining Data State	Ensuring that the list of saved trips persisted after a page refresh or navigation.	Implemented a dedicated persistence layer using Local Storage (<code>localStorage.setItem/getItem</code>) to read the data on initialization and save it after every successful trip submission.
Robust Budget Calculation	Preventing errors when users left the "Amount Spent" field empty or entered invalid numbers, which would break the balance calculation.	During form submission, the <code>amountSpent</code> value is explicitly cast using <code>parseFloat()</code> and a logical OR (<code>'</code>
Dynamic HTML Injection	Efficiently displaying the list of saved trips and ensuring the display updates immediately upon adding a new trip.	The <code>renderTrips()</code> function was designed to clear the container (<code>innerHTML = ""</code>) and then re-render the entire list from the current data model, guaranteeing the UI perfectly reflects the stored data state.

Outcomes:

The project successfully delivered a complete, functional, and aesthetically pleasing travel planning utility. The combination of **Vanilla JavaScript's power** for data management and **Bootstrap's efficiency** for responsive design resulted in a fast, stable application. The financial tracking feature provides clear utility, making the application a practical tool for everyday use. The project validates the ability to build sophisticated, data-driven applications purely on the client-side.

Future Enhancements:

- **Advanced Expense Management:** Implement a detailed breakdown of expenses per trip (e.g., logging individual costs for food, lodging, transport) rather than just a total "Amount Spent."
- **Interactive Card Features:** Add functionality to the saved trip cards, such as a "Delete" button to remove a trip or an "Edit" button to modify existing data.
- **Search and Filter:** Introduce a search input and category filters to allow users to quickly find specific saved trips based on destination or budget.
- **Date/Timeline Integration:** Add date fields (Start Date, End Date) and visualize the trips on a simple timeline interface.

Code:

Html:

```
</form>
</div>
</section>

<section id="savedtrips">
  <div class="container">
    <h2 class="text-center mb-4">Saved Trips</h2>
    <div id="tripslist" class="row"></div>
  </div>
</section>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
<script src="app.js"></script>
<script>
  const suggestions = [
    {name:"Beach Paradise", mood:"Relaxation", img:"https://source.unsplash.com/300x200/?beach"},
    {name:"Mountain Adventure", mood:"Adventure", img:"https://source.unsplash.com/300x200/?mountain"},
    {name:"City Lights", mood:"Urban", img:"https://source.unsplash.com/300x200/?city"},
    {name:"Forest Retreat", mood:"Nature", img:"https://source.unsplash.com/300x200/?forest"},
    {name:"Desert Safari", mood:"Adventure", img:"https://source.unsplash.com/300x200/?desert"},
    {name:"Island Escape", mood:"Relaxation", img:"https://source.unsplash.com/300x200/?island"}
  ];

  const container = document.getElementById("suggestionsList");
  suggestions.forEach(s => {
    const card = document.createElement("div");
    card.className = "col-md-4 mb-3";
    card.innerHTML = `<div class="card h-100">
      
      <div class="card-body">
        <h3 class="card-title">${s.name}</h3>
        <p class="card-text">Mood: ${s.mood}</p>
      </div>
    </div>`;
    container.appendChild(card);
  });
</script>
</body>
</html>
```

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Holiday Planner</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <div class="hero">
    <h1>🌍 Plan Your Perfect Holidays</h1>
    <p>Discover, plan, and save your dream trips with ease.</p>
  </div>

  <section id="suggestions">
    <div class="container">
      <h2 class="text-center mb-4">Travel Suggestions</h2>
      <div class="row" id="suggestionsList"></div>
    </div>
  </section>

  <section id="planner">
    <div class="container">
      <h2 class="text-center mb-4">Plan a Trip</h2>
      <form id="tripForm" class="mb-4">
        <div class="mb-3"><input type="text" class="form-control" id="destination" placeholder="Destination" required></div>
        <div class="mb-3"><input type="number" class="form-control" id="plannedBudget" placeholder="Planned Budget" required></div>
        <div class="mb-3"><input type="number" class="form-control" id="amountSpent" placeholder="Amount Spent"></div>
        <div class="mb-3"><input type="text" class="form-control" id="members" placeholder="Members (comma-separated)"></div>
        <div class="mb-3"><textarea class="form-control" id="note" placeholder="Add a note"></div>
        <button type="submit" class="btn btn-primary">Save Trip</button>
      </form>
    </div>
  </section>

  <section id="savedTrips">
    <div class="container">
      <h2 class="text-center mb-4">Saved Trips</h2>

```

```

> Users > deeks > AppData > Local > Microsoft > Windows > INetCache > IE > LI3V8X3O > # style[1].css > ...
1
2  @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap');
3
4  body {
5    background-color: #f4f9f9;
6    font-family: 'Poppins', sans-serif;
7    color: #2c3e50;
8    margin: 0;
9    padding: 0;
10 }
11
12 .hero {
13   background: linear-gradient(rgba(0,0,0,0.5), rgba(0,0,0,0.5)), url('https://source.unsplash.com/1600x900/?travel');
14   color: white;
15   text-align: center;
16   padding: 120px 20px;
17   animation: fadeIn 2s ease-in;
18 }
19
20 .hero h1 {
21   font-size: 3rem;
22   font-weight: 600;
23   animation: slideDown 1.5s ease-out;
24 }
25
26 .hero p {
27   font-size: 1.25rem;
28   margin-top: 15px;
29   animation: fadeIn 2.5s ease-in;

```

```

> Users > deeks > AppData > Local > Microsoft > Windows > INetCache > IE > LI3V8X3O > # style[1].css > ...
55
56 @keyframes slideDown {
57   from { transform: translateY(-50px); opacity: 0; }
58   to { transform: translateY(0); opacity: 1; }
59 }
60
61 @keyframes fadeInUp {
62   from { transform: translateY(30px); opacity: 0; }
63   to { transform: translateY(0); opacity: 1; }
64 }
65

```



```

.hero p {
}

section {
  padding: 60px 20px;
}

.card {
  border-radius: 15px;
  box-shadow: 0 4px 8px rgba(0,0,0,0.1);
  transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.card:hover {
  transform: translateY(-8px) scale(1.02);
  box-shadow: 0 10px 20px rgba(0,0,0,0.15);
}

.saved-card {
  animation: fadeInUp 1s ease;
}

@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}

@keyframes slideDown {
  from { transform: translateY(-50px); opacity: 0; }

```

Ln 1, Col 1 Spaces: 2 UTF-8 LF {} CSS

```

    <p class= "text-muted">${t.note || ""}</p>
  );
  container.appendChild(card);
});
}

document.addEventListener("DOMContentLoaded", renderTrips);

document.getElementById("tripForm").addEventListener("submit", e => {
  e.preventDefault();
  const trip = {
    destination: document.getElementById("destination").value,
    plannedBudget: parseFloat(document.getElementById("plannedBudget").value),
    amountSpent: parseFloat(document.getElementById("amountSpent").value) || 0,
    balance: parseFloat(document.getElementById("plannedBudget").value) - (parseFloat(document.getEle
    members: document.getElementById("members").value.split(",").map(m=>m.trim()),
    note: document.getElementById("note").value
  };
  addTrip(trip);
  e.target.reset();
});

```

```

Users > deeks > AppData > Local > Microsoft > Windows > INetCache > IE > U3V8X3O > JS Untitled-1.js > ...
const STORAGE_KEY = "holidayPlanner.trips";

function getTrips() {
  return JSON.parse(localStorage.getItem(STORAGE_KEY)) || [];
}

function saveTrips(trips) {
  localStorage.setItem(STORAGE_KEY, JSON.stringify(trips));
}

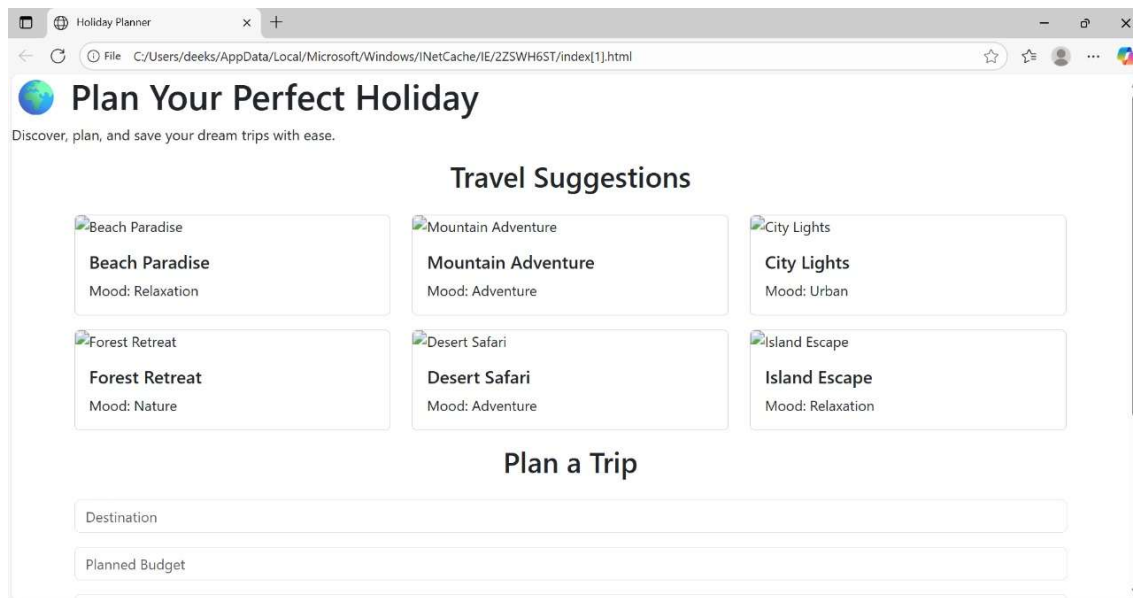
function addTrip(trip) {
  const trips = getTrips();
  trips.push(trip);
  saveTrips(trips);
  renderTrips();
}

function renderTrips() {
  const trips = getTrips();
  const container = document.getElementById("tripsList");
  container.innerHTML = "";
  trips.forEach((t,i) => {
    const card = document.createElement("div");
    card.className = "card p-3 mb-3 saved-card";
    card.innerHTML = `
      <h5>${t.destination}</h5>
      <p>Planned Budget: ${t.plannedBudget}, Spent: ${t.amountSpent}, Balance: ${t.balance}</p>
      <p>Members: ${t.members.join(", ")}</p>
      <p class="text-muted">${t.note || ""}</p>
    `;
  });
}

```

Ln 49, Col 3 Spaces: 4 UTF-8 CRLF {} JavaScript

Output:



Plan a Trip

Destination

Planned Budget

Amount Spent

Members (comma-separated)

Add a note

Save Trip

Saved Trips

Conclusion:

The Holiday Planner Web Application is a successful, practical demonstration of client-side development, built with HTML5, CSS3, and Vanilla JavaScript with Bootstrap 5.

The project effectively delivers a responsive UI, ensures data permanence through Local Storage, and provides a core utility with its automatic budget balance calculation. This makes the application a valuable tool that is both functional and aesthetically pleasing, fulfilling the goal of simplified, data-aware travel planning without needing a server.

References:

L&T LMS : <https://learn.intedutech.com/Landing/MyCourse>