

# KOLEJ PROFESIONAL MARA INDERA MAHKOTA DIPLOMA IN COMPUTER SCIENCE

|  |  |  |
| --- | --- | --- |
| **COURSE NAME** | : | **OBJECT ORIENTED PROGRAMMING** |
| **COURSE CODE** | : | **CSC2744** |
| **ACADEMIC SESSION** | : | **SESSION 3 2024/2025** |
| **TYPE OF ASSESSMENT** | : | **FINAL ASSIGNMENT** |
| **DURATION** | : | **05/02/2025 - 03/03/2025** |

**CLO 3: Employ third party data in object oriented application development using graphical user interface (GUI) application framework**

**INSTRUCTION TO CANDIDATES:**

1. Late submissions after the given due date will not be accepted.
2. Report should be written using: Font type: Arial

Size: 12 pts

Line Spacing: 1.5

1. Coding format:

Font type: Consolas Size: 10 pts

Line Spacing: Single

|  |  |
| --- | --- |
| **Personal Details** | |
| **Name** | NUR MAISARAH BINTI AHMAD ASRO |
| **I/D Number** | ICS23-11-017 |
| **Class** | DCS4A |
| **Lecturer** | MADAM NUBAILAH |

|  |  |
| --- | --- |
| **Section / Question No.** | **Marks** |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| **Total** | **/ 50** |

I hereby declare that no form of plagiarism will be tolerated in this assessment. Failure to comply will result in a failing grade for the assessment.

Signature: maisarah Date:03/03/2025

**Question**

In JavaScript development, **Electron** is a framework that allows developers to build cross-platform desktop applications using web technologies like HTML, CSS, and JavaScript. It wraps a web application (which would normally run in a browser) inside a desktop environment, enabling it to run as a native application on platforms like Windows, macOS, and Linux.

Many popular desktop applications, like Visual Studio Code, Slack, Discord, and Spotify (desktop version), are built using Electron. As part of your software development course, you have been tasked to develop a desktop application using Electron with web technologies like HTML, CSS, JavaScript and utilizes the third-party data from one of the given API. You also need to incorporate CRUD (Create, Read, Update, Delete) operations to enhance user interaction and usability.

|  |  |
| --- | --- |
| **Name of**  **Application** | **Key Features:** |
| Solah Tracker for Kids | * Help parents track their children’s daily prayers. * Display key information like 5 prayer times, day and hijri date. * User able to search prayer time by prayer zone * CRUD – Personal detail of child like name, age. Able to record daily 5 prayers and also sunnah prayers for each individual. <https://api.waktusolat.app/solat/SGR01> |
| Quran Journal | * Allow users to view list of surah in the Al-Quran, and note their spiritual discovery of every surah. * Display key information like surah name, surah number, English translation, number of ayahs and revelation type. * User able to search surah by surah name * CRUD – number of ayahs completed and notes for self-reflection. <http://api.alquran.cloud/v1/surah> |

|  |  |
| --- | --- |
| **Name of**  **Application** | **Key Features:** |
| Hadith Buddy | * Displays random hadith from Sahih Bukhari. Allows user to track favorite hadiths in list. * Display key information like hadith number, hadith in English, narrator, book source, hadith status, and hadith chapter. * Allows user to search hadith by book source. * CRUD – favorite hadiths by hadith number and personal reflection regarding hadiths. * [https://hadithapi.com/api/sahih-](https://hadithapi.com/api/sahih-bukhari/chapters?apiKey=%242y%2410%24imRVx8ZhnDdBGcKCABQ95ePS8VhQ6gFzUnWyIbR4xeYsGFDrGcnvC) [bukhari/chapters?apiKey=$2y$10$imRVx8ZhnDdBGcKCABQ95e](https://hadithapi.com/api/sahih-bukhari/chapters?apiKey=%242y%2410%24imRVx8ZhnDdBGcKCABQ95ePS8VhQ6gFzUnWyIbR4xeYsGFDrGcnvC) [PS8VhQ6gFzUnWyIbR4xeYsGFDrGcnvC](https://hadithapi.com/api/sahih-bukhari/chapters?apiKey=%242y%2410%24imRVx8ZhnDdBGcKCABQ95ePS8VhQ6gFzUnWyIbR4xeYsGFDrGcnvC) |
| Sunnah Meal | * Suggests menu based on sunnah food as ingredients. * Display key information like recipe title, image, missed ingredients, and used ingredients * CRUD – create meal plan for the week based on searched recipe. * [https://api.spoonacular.com/recipes/findByIngredients?apiKey=15](https://api.spoonacular.com/recipes/findByIngredients?apiKey=153ed267f6fe4cc1978d7d8576103378&ingredients=dates) [3ed267f6fe4cc1978d7d8576103378&ingredients=dates](https://api.spoonacular.com/recipes/findByIngredients?apiKey=153ed267f6fe4cc1978d7d8576103378&ingredients=dates) |

# Tasks:

1. Create desktop application using electron framework with the integration of the given API. Your application needs to output the information required above. It should have at least 2 pages, and you may add extra functionality or features of your choice to the application.
2. Implement CRUD (create, read, update, delete) process to the application to make your application more meaningful.
3. Apply HTML and CSS for user interface and provide evidence for application.
4. GUI Elements:
   1. Apply GUI elements that assist users in using applications.
   2. The application’s ‘look and feel’ is attractive and informative.
5. Produce a report on your application. Include the following:
   1. Overview of the application and its purpose. Include explanation of the information retrieved from the given API and CRUD process.
   2. Description of the functionalities and features of the developed application with print screen of the pages and explanation.
   3. Program codes of your system
6. Submit files in GitHub and include the link in the report.

**Table of Contents**

**Topic Page Number**

Table of contents 5

1.0 Overview 6

2.0 CRUD Process Explanation 6

3.0 Description of Functionalities and Features 6 - 7

4.0 Screenshot and overview 7 - 9

5.0 Coding 10 - 28

Rubric                           29-35

1. Overview

The "Solah Tracker for Kids" application is designed to help parents or guardians track and manage their children's prayer habits, including both daily and Sunnah prayers. The purpose of the app is to create an easy-to-use platform for recording prayer times, monitoring their consistency, and saving the information in a text file for future reference. The application allows users to add, update, read, and delete child information.

The app retrieves prayer times from an external API (e.g., the waktusolat API) that provides prayer schedules based on the selected prayer zone and date. This API gives prayer times for various locations and allows users to view when prayers are expected for the day. The information from this API helps users keep track of prayer schedules, especially for children, and ensures they follow the correct times for performing their prayers.

1. CRUD Process Explanation:
   1. The CRUD (Create, Read, Update, Delete) process is implemented for managing child information:
   2. Create: Users can enter new child information (name, age, Sunnah prayers, and daily prayers) and save it to a text file.
   3. Read: Users can retrieve previously saved child information by entering the file name, displaying the data saved earlier.
   4. Update: Users can update any of the child’s details and save the updated information, overwriting the previous file.
   5. Delete: Users can delete a child's information from the app, removing the saved file after confirmation.
2. Description of Functionalities and Features
   1. The "Solah Tracker for Kids" application includes several key features:
   2. Homepage: The homepage introduces the app with a brief explanation and provides links to other pages, such as "Prayer Zone" and "Solah Tracker."
   3. Prayer Zone Page: This page allows users to select their prayer zone and the date. It retrieves prayer times from the API based on the selected zone and date, providing users with prayer schedules.
   4. Feature Highlight: Users can select different prayer zones (such as Tapah, Ipoh, etc.) and choose the date to get the prayer times.
   5. Solah Tracker Page: On this page, users can input their child’s details, including name, age, daily prayers, and Sunnah prayers. This page allows users to save the child’s information to a text file.
   6. Feature Highlight: Users can tick checkboxes for daily prayers, and the app will save which prayers have been performed.
   7. Manage Child’s Information: This section provides three buttons to:
      1. Read Child Info: Allows users to input a file name to read the saved child’s information.
      2. Update Child Info: Users can update the saved information by modifying fields and saving the updated file.
      3. Delete Child Info: Users can delete a child’s information after confirming the file name.
   8. Feature Highlight: Each button provides easy access to manage, update, or delete child records.
3. Screenshots and Explanation:
   1. Homepage:

Contains basic information about the app and links to the "Solah Tracker" and "Prayer Zone" pages.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. Prayer Zone Page:
* Allows users to select the prayer zone and date, then fetches the prayer times from the API.
* Displays prayer times for the selected zone and date.

A screenshot of a computer

AI-generated content may be incorrect.

4.3 Solah Tracker Page:

* The page where parents enter their child’s information, including name, age, and prayer details.
* Provides options for daily and Sunnah prayers, and users can save the information.

A screenshot of a computer

AI-generated content may be incorrect.

4.4 Manage Child’s Information:

* Three buttons for managing saved child information: Read, Update, and Delete.
* Each button performs a different action for managing the child’s records.

A screenshot of a computer

AI-generated content may be incorrect.

1. Coding

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>About Solah Tracker for Kids</title>

    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;700&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <nav class="navbar navbar-expand-lg navbar-light bg-white sticky-top">

        <div class="container-fluid px-5">

            <span class="navbar-brand fw-bold fs-2" style="color: #165621; font-weight: bold;">

                Solah Tracker for Kids

            </span>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav">

                    <li><a href="prayerzone.html">Prayer Zone</a></li>

                    <li><a href="readchildinfo.html">View Info</a></li>

                </ul>

            </div>

        </div>

    </nav>

    <section class="home">

        <div class="flex-container">

            <div class="image">

                <img id="hover-image" src="homepage.png" alt="Image" />

            </div>

            <div class="typing-effect">

                <h2>Solah Tracker<br /><span class="c-orange">for Kids</span></h2>

                <a href="prayerzone.html">

                    <button class="c-btn h-btn mt-3">Get Started</button>

                </a>

            </div>

        </div>

    </section>

    <footer>

        <p>&copy; 2025 Solah Tracker | All Rights Reserved.</p>

    </footer>

    <script src="renderer.js"></script>

</body>

</html>

Index.html

const { app, BrowserWindow } = require(‘electron’);

const path = require(‘path’);

const createWindow = () => {

  const mainWindow = new BrowserWindow({

    width: 800,

    height: 600,

    webPreferences: {

      nodeIntegration: true,

      contextIsolation: false,

    }

  });

  mainWindow.loadFile(path.join(\_\_dirname, ‘index.html’));

};

app.whenReady().then(createWindow);

app.on(‘window-all-closed’, () => {

  if (process.platform !== ‘darwin’) {

    app.quit();

  }

});

app.on(‘activate’, () => {

  if (BrowserWindow.getAllWindows().length === 0) {

    createWindow();

  }

});

Index.js

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Solah Tracker for Kids</title>

    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;700&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <nav class="navbar navbar-expand-lg navbar-light bg-white sticky-top">

        <div class="container-fluid px-5">

            <span class="navbar-brand fw-bold fs-2" style="color: #165621; font-weight: bold;">

                Solah Tracker for Kids

            </span>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav">

                    <li><a href="index.html">Homepage</a></li>

                    <li><a href="prayerzone.html">Prayer Zone</a></li>

                    <li><a href="readchildinfo.html">View Info</a></li>

                </ul>

            </div>

        </div>

    </nav>

*<!-- Prayer Zone Section -->*

    <div class="container px-5">

        <div class="box-container">

            <h3>Select a Prayer Zone</h3>

            <select id="solatZone">

                <option value="">-- Select Zone --</option>

                <option value="PRK01">Tapah, Slim River, Tanjung Malim</option>

                <option value="PRK02">Kuala Kangsar, Sg. Siput, Ipoh, Batu Gajah, Kampar</option>

                <option value="PRK03">Lenggong, Pengkalan Hulu, Grik</option>

                <option value="PRK04">Temengor, Belum</option>

                <option value="PRK05">Kg Gajah, Teluk Intan, Bagan Datuk, Seri Iskandar, Beruas, Parit, Lumut, Sitiawan, Pulau Pangkor</option>

                <option value="PRK06">Selama, Taiping, Bagan Serai, Parit Buntar</option>

                <option value="PRK07">Bukit Larut</option>

            </select>

            <label>Select Date:</label>

            <input type="date" id="solatDate">

            <button id="btnFetchSolat">Get Prayer Times</button>

            <p id="hijriDate"></p>

            <p id="currentDay"></p>

            <ul id="solatList"></ul>

        </div>

    </div>

*<!-- Child Information Section -->*

    <div class="container px-5">

        <div class="box-container">

            <h3>Enter Child's Information</h3>

            <label>Child's Name:</label>

            <input type="text" id="childName">

            <label>Child's Age:</label>

            <input type="number" id="childAge">

            <h4>Record Daily Prayers</h4>

            <label><input type="checkbox" id="subuh"> Subuh</label><br>

            <label><input type="checkbox" id="zohor"> Zohor</label><br>

            <label><input type="checkbox" id="asar"> Asar</label><br>

            <label><input type="checkbox" id="maghrib"> Maghrib</label><br>

            <label><input type="checkbox" id="isyak"> Isyak</label><br>

            <h4>Sunnah Prayers</h4>

            <label id="sunnahPrayers">Sunnah Prayers:</label>

            <div class="button-group">

                <button id="btnSaveChild">Save Data</button>

                <button id="btnReadChild">Read Data</button>

                <button id="btnDeleteChild">Delete Data</button>

            </div>

            <label for="editChildDropdown"><strong>Select a Child to Edit:</strong></label>

            <select id="editChildDropdown">

                <option disabled selected>Select a child</option>

            </select>

        </div>

    </div>

    </div>

    <footer>

        <p>&copy; 2025 Solah Tracker | All Rights Reserved.</p>

    </footer>

    <script src="renderer.js"></script>

</body>

</html>

Prayerzone.html

const { contextBridge, ipcRenderer } = require('electron');

contextBridge.exposeInMainWorld('api', {

    name: "Solah Tracker",

    createNote: (data) => ipcRenderer.invoke('create-file', data),

    readFile: (name) => ipcRenderer.invoke('read-file', name),

    deleteFile: (name) => ipcRenderer.invoke('delete-file', name)

});

Preload.js

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Solah Tracker for Kids</title>

    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;700&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <nav class="navbar navbar-expand-lg navbar-light bg-white sticky-top">

        <div class="container-fluid px-5">

            <span class="navbar-brand fw-bold fs-2" style="color: #165621; font-weight: bold;">

                Solah Tracker for Kids

            </span>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav">

                    <li><a href="index.html">Homepage</a></li>

                    <li><a href="prayerzone.html">Prayer Zone</a></li>

                    <li><a href="readchildinfo.html">View Info</a></li>

                </ul>

            </div>

        </div>

    </nav>

<body>

  <div class="mainWrapper">

    <h3>Saved Records</h3>

    <label>Select Child:</label>

    <select id="childDropdown"></select>

    <button onclick="showChildInfo()">Show Information</button>

    <div id="childInfo">

      <h3>Child Details</h3>

      <p id="childName"></p>

      <p id="childAge"></p>

      <p id="dateSaved"></p>

      <h3>Daily Prayer Tracking</h3>

      <ul id="prayerList"></ul>

      <button id="btnDeleteChild">Delete This Record</button>

    </div>

  </div>

  <script>

    const fs = require('fs');

    const path = require('path');

    const dataFile = path.join(\_\_dirname, 'data.txt');

    function loadDropdown() {

      const dropdown = document.getElementById("childDropdown");

      dropdown.innerHTML = "";

      if (fs.existsSync(dataFile)) {

        let records = JSON.parse(fs.readFileSync(dataFile));

        records.forEach((record, index) => {

          let option = document.createElement("option");

          option.value = index;

          option.textContent = record.name;

          dropdown.appendChild(option);

        });

      } else {

        dropdown.innerHTML = "<option>No records found</option>";

      }

    }

    function showChildInfo() {

      const index = document.getElementById("childDropdown").value;

      if (index === "") return;

      const records = JSON.parse(fs.readFileSync(dataFile));

      const child = records[index];

      document.getElementById("childInfo").*style*.display = "block";

      document.getElementById("childName").textContent = `Name: ${child.name}`;

      document.getElementById("childAge").textContent = `Age: ${child.age}`;

      document.getElementById("dateSaved").textContent = `Saved on: ${child.dateSaved}`;

      const prayerList = document.getElementById("prayerList");

      prayerList.innerHTML = `

        <li>Subuh: ${child.prayers.subuh ? '✔' : '✘'}</li>

        <li>Zohor: ${child.prayers.zohor ? '✔' : '✘'}</li>

        <li>Asar: ${child.prayers.asar ? '✔' : '✘'}</li>

        <li>Maghrib: ${child.prayers.maghrib ? '✔' : '✘'}</li>

        <li>Isyak: ${child.prayers.isyak ? '✔' : '✘'}</li>

        <li><strong>Sunnah Prayers:</strong> ${child.sunnahPrayers || "None"}</li>

      `;

      document.getElementById("btnDeleteChild").onclick = function() {

        deleteChild(index);

    };

    }

    function deleteChild(index) {

      let records = JSON.parse(fs.readFileSync(dataFile));

      if (confirm(`Are you sure you want to delete ${records[index].name}'s record?`)) {

        records.splice(index, 1);

        fs.writeFileSync(dataFile, JSON.stringify(records, null, 2));

        alert("Record deleted successfully!");

        document.getElementById("childInfo").*style*.display = "none";

        loadDropdown();

      }

    }

    function goBack() {

      window.location.href = "index.html";

    }

    window.onload = loadDropdown;

  </script>

<footer>

  <p>&copy; 2025 Solah Tracker | All Rights Reserved.</p>

</footer>

<script src="renderer.js"></script>

</body>

</html>

Readchildinfo.html

const { ipcRenderer } = require('electron');

const fs = require('fs');

const path = require('path');

const dataFile = path.join(\_\_dirname, 'data.txt');

document.addEventListener("DOMContentLoaded", function () {

    document.getElementById("btnSaveChild").addEventListener("click", saveChildDetails);

    document.getElementById("btnFetchSolat").addEventListener("click", displayPrayerTimes);

    document.getElementById("btnReadChild").addEventListener("click", readChildDetails);

    document.getElementById("btnDeleteChild").addEventListener("click", deleteChildDetails);

    document.getElementById("btnEditChild").addEventListener("click", editChildDetails);

    loadEditDropdown();

});

async function displayPrayerTimes() {

    const solatList = document.getElementById("solatList");

    solatList.innerHTML = "";

    const selectedDate = document.getElementById("solatDate").value;

    const selectedZone = document.getElementById("solatZone").value;

    if (!selectedDate) {

        alert("Please select a date.");

        return;

    }

    try {

        const response = await fetch(`https://api.waktusolat.app/solat/${selectedZone}`);

        const data = await response.json();

        if (!data || !data.prayerTime) {

            throw new Error("Invalid API response.");

        }

        const [year, month, day] = selectedDate.split("-");

        const monthNames = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"];

        const formattedDate = `${day}-${monthNames[parseInt(month, 10) - 1]}-${year}`;

        const prayerTimes = data.prayerTime.find(entry => entry.date === formattedDate);

        if (!prayerTimes) {

            solatList.innerHTML = `<li style="color: red;">No data available for selected date.</li>`;

            return;

        }

        document.getElementById("hijriDate").textContent = `Hijri Date: ${prayerTimes.hijri}`;

        document.getElementById("currentDay").textContent = `Day: ${prayerTimes.day}`;

        const solatKeys = { fajr: "Subuh", dhuhr: "Zohor", asr: "Asar", maghrib: "Maghrib", isha: "Isyak" };

        for (const key in solatKeys) {

            const li = document.createElement("li");

            li.textContent = `${solatKeys[key]}: ${prayerTimes[key]}`;

            solatList.appendChild(li);

        }

    } catch (error) {

        solatList.innerHTML = `<li style="color: red;">Error fetching data.</li>`;

    }

}

function saveChildDetails() {

    const name = document.getElementById("childName").value;

    const age = document.getElementById("childAge").value;

    const sunnahPrayers = document.getElementById("sunnahPrayers").value.trim();

    const prayers = {

        subuh: document.getElementById("subuh").checked,

        zohor: document.getElementById("zohor").checked,

        asar: document.getElementById("asar").checked,

        maghrib: document.getElementById("maghrib").checked,

        isyak: document.getElementById("isyak").checked

    };

    if (!name || !age) {

        alert("Please enter child's name and age.");

        return;

    }

    const today = new Date();

    const formattedDate = `${today.getDate().toString().padStart(2, '0')}-${(today.getMonth() + 1).toString().padStart(2, '0')}-${today.getFullYear()}`;

    let data = [];

    if (fs.existsSync(dataFile)) {

        data = JSON.parse(fs.readFileSync(dataFile));

    }

    data.push({ name, age, prayers,sunnahPrayers: sunnahPrayers || "None",  dateSaved: formattedDate});

    fs.writeFileSync(dataFile, JSON.stringify(data, null, 2));

    alert("Child data saved successfully on " + formattedDate);

    loadEditDropdown();

}

function readChildDetails() {

    if (!fs.existsSync(dataFile)) {

        alert("No saved child data found.");

        return;

    }

    const records = JSON.parse(fs.readFileSync(dataFile));

    let displayText = records.map(record => `Name: ${record.name}\nAge: ${record.age}\nSubuh: ${record.prayers.subuh}\nZohor: ${record.prayers.zohor}\nAsar: ${record.prayers.asar}\nMaghrib: ${record.prayers.maghrib}\nIsyak: ${record.prayers.isyak}\nSunnah Prayers: ${record.sunnahPrayers}`).join("\n\n");

    alert(displayText);

}

function loadEditDropdown() {

    const dropdown = document.getElementById("editChildDropdown");

    dropdown.innerHTML = "";

    if (!fs.existsSync(dataFile)) {

        console.warn("data.txt not found!");

        dropdown.innerHTML = `<option disabled selected>No children available</option>`;

        return;

    }

    try {

        const text = fs.readFileSync(dataFile, "utf8");

        const childData = JSON.parse(text);

        if (!Array.isArray(childData) || childData.length === 0) {

            dropdown.innerHTML = `<option disabled selected>No children available</option>`;

            return;

        }

        dropdown.innerHTML = `<option disabled selected>Select a child</option>`;

        childData.forEach((child) => {

            const option = document.createElement("option");

            option.value = child.name;

            option.textContent = `${child.name} (Age: ${child.age})`;

            dropdown.appendChild(option);

        });

        dropdown.addEventListener("change", populateChildDetails);

    } catch (error) {

        console.error("Error parsing JSON:", error);

    }

}

function deleteChildDetails() {

    if (fs.existsSync(dataFile)) {

        fs.unlinkSync(dataFile);

        alert("Child data deleted successfully.");

    } else {

        alert("No data found to delete.");

    }

}

function populateChildDetails() {

    const selectedChild = document.getElementById("editChildDropdown").value;

    const childInfoContainer = document.getElementById("childInfoContainer");

    if (!fs.existsSync(dataFile)) return;

    const childData = JSON.parse(fs.readFileSync(dataFile, "utf8"));

    const child = childData.find(c => c.name === selectedChild);

    if (!child) {

        childInfoContainer.style.display = "none";

        return;

    }

    document.getElementById("childNameEdit").value = child.name;

    document.getElementById("childAgeEdit").value = child.age;

    document.getElementById("sunnahPrayersEdit").value = child.sunnahPrayers;

    document.getElementById("subuhEdit").checked = child.prayers.subuh;

    document.getElementById("zohorEdit").checked = child.prayers.zohor;

    document.getElementById("asarEdit").checked = child.prayers.asar;

    document.getElementById("maghribEdit").checked = child.prayers.maghrib;

    document.getElementById("isyakEdit").checked = child.prayers.isyak;

    childInfoContainer.style.display = "block";

}

function editChildDetails() {

    const selectedChild = document.getElementById("editChildDropdown").value;

    if (!fs.existsSync(dataFile)) return;

    let childData = JSON.parse(fs.readFileSync(dataFile, "utf8"));

    const childIndex = childData.findIndex(c => c.name === selectedChild);

    if (childIndex === -1) return;

    const updatedChild = {

        name: document.getElementById("childNameEdit").value,

        age: document.getElementById("childAgeEdit").value,

        sunnahPrayers: document.getElementById("sunnahPrayersEdit").value.trim() || "None",

        prayers: {

            subuh: document.getElementById("subuhEdit").checked,

            zohor: document.getElementById("zohorEdit").checked,

            asar: document.getElementById("asarEdit").checked,

            maghrib: document.getElementById("maghribEdit").checked,

            isyak: document.getElementById("isyakEdit").checked

        }

    };

    childData[childIndex] = updatedChild;

    fs.writeFileSync(dataFile, JSON.stringify(childData, null, 2));

    alert("Child data updated successfully!");

    loadEditDropdown();

}

Renderer.js

body {

  font-family: 'Poppins', *sans-serif*;

  margin: 0;

  padding: 0;

  background-color: #f4f4f4;

  min-height: 100vh;

}

main {

  flex: 1; */\* Fills available space and pushes footer down \*/*

}

nav {

  background-color: #bfd3ca;

  padding: 10px;

  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

  width: 100%;

}

nav .navbar-brand {

  font-size: 2rem;

  color: #1f602a;

}

nav ul {

  list-style: *none*;

  display: *flex*;

  justify-content: *center*;

  padding: 0;

  margin: 0;

}

section {

  padding: 50px 0;

  text-align: *center*;

}

section h2 {

  color: #0c1a0e;

  font-family: 'Poppins', *sans-serif*;

}

.home img {

  max-width: 40%;

  height: *auto*;

}

nav ul li a {

  text-decoration: *none*;

  font-size: 1.1rem;

  color: *black*;

  padding: 10px;

  width: 100%;

  box-sizing: *border-box*;

  transition: *color* 0.3s *ease*, background-color 0.3s *ease*;

}

nav ul li a:hover {

  color: #1f602a;

  background-color: rgba(255, 153, 0, 0.1);

  border-radius: 80px;

}

.c-btn {

  background-color: #3b7645;

  font-family: 'Poppins', *sans-serif*;

  color: *white*;

  border: *none*;

  padding: 10px 20px;

  cursor: *pointer*;

  border-radius: 80px;

  text-decoration: *none*;

  transition: background-color 0.3s *ease*, transform 0.2s *ease*;

}

.c-btn:hover {

  background-color: #1f602a;

  transform: scale(1.05);

}

footer p {

  color: #a3a2a2;

  font-size: 14px;

  margin-top: 10px;

}

footer {

  background-color: #165621;

  padding: 10px;

  text-align: *center*;

  position: *relative*;

  width: 100%;

  bottom: 0;

  color: *white*;

}

nav .container-fluid {

  display: *flex*;

  justify-content: *space-between*;

  padding: 20px;

  align-items: *center*;

}

.navbar-nav {

  list-style: *none*;

  display: *flex*;

  gap: 20px;

  margin: 0 *auto*;

  justify-content: *center*;

}

ul .c-btn {

  margin-left: *auto*;

  font-family: 'Poppins', *sans-serif*;

}

.flex-container {

  display: *flex*;

  align-items: *center*;

  justify-content: *space-around*;

  flex-wrap: *wrap*;

}

.image img {

  max-width: 40%;

}

@media (max-width: 768px) {

  .flex-container {

      flex-direction: *column*;

      text-align: *center*;

  }

}

.image {

  overflow: *hidden*;

}

@keyframes typing {

  from { width: 0; }

  to { width: 100%; }

}

.typing-effect h2 {

  font-family: 'Poppins', *sans-serif*;

  white-space: *nowrap*;

  overflow: *hidden*;

  border-right: 3px *solid*;

  width: 100%;

  animation: typing 3s steps(20, *end*), *blink* 0.75s *step-end* *infinite*;

}

.container {

  display: *flex*;

  justify-content: *center*;

  align-items: *center*;

  height: 100vh; */\* Full viewport height \*/*

  flex-direction: *column*;

  padding: 1px;

}

.box-container {

  background-color: *white*;

  padding: 20px;

  border-radius: 8px;

  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

  width: 100%;

  max-width: 600px;

  margin-bottom: 20px;

}

.crud-operations {

  margin-top: 20px;

  text-align: *center*;

}

Styles.css

**Assessment Rubric**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ATTRIBUTES** | **CRITERIA** | **POOR**  **(1 mark)** | **FAIR**  **(2 marks)** | **GOOD**  **(3 marks)** | **VERY GOOD**  **(4 marks)** | **EXCELLENT**  **(5 marks)** | **Weightage** | **Mark Obtained** |
| **Reproduce and Process Information** | 1. Create desktop application using electron framework with the integration of the given API. Your application needs to output the information required above. | * The application is an extensive collection and rehash of other people's ideas, products, and images. There is no evidence of new thought. | * The application is somewhat a collection and rehash of other people's ideas, products, and images. There is little evidence of new thought or inventiveness. | * The application is a minimal collection or rehash of other people's ideas, products, and images. There is a few evidence of new thought or inventiveness   . | * The application shows a lot of evidence of originality and inventiveness. | * The application shows significant evidence of originality and inventivenes s. * Most of the content and many of the ideas are fresh,original, and   inventive. | **1** |  |
| * Unable to display all the required data from the API and does not fulfill the requirements. | * Able to display all the required data from the API that meet with the requirements. | * Able to display extra data from the API beyond the application requirements. | * Able to display extra data from the API beyond the application requirements with the   description in the report |  | **1** |  |
| * The data from the API does not reflect the whole purpose of the application developed. | * The data from the API is sufficient but does not reflect the whole purpose of the application developed. | * The data from the API is meaningful but does not reflect the purpose of the application developed. | * Able to utilize the data fetched from the API to come up with meaningful functionalities that reflect the purpose of the application   developed. | * Able to utilize the data fetched from the API to come up with meaningful functionalitie s that reflect the purpose   of the |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | application developed.   * Come up with extra idea for the application developed that enhances the user experience or adds   value to the application. |  |  |
| 2. Implement CRUD (create, read, update, delete) process to the application to make your application more meaningful. | * Able to create only 2 of the CRUD   processes.   * No feedback for the CRUD process. * The design for data input is poor | * Able to create only 3 of the CRUD   processes.   * No feedback for the CRUD process. * The design for data input is good with some room for improvement | * Able to perform all the CRUD   processes.   * No feedback for the CRUD process. * Well-designed data input for CRUD   process. | * Able to perform all the CRUD processes. * No feedback for the CRUD process. * Well-designed data input for CRUD process. | * Able to perform all the CRUD   processes.   * Appropriate feedback for the CRUD process. * Well-designed and user- friendly data input for CRUD   process. | **1** |  |
|  | 3. Apply HTML and CSS for user interface and provide evidence for application. | * **Text -** All text used is too small to view or the font type is wrongly chosen. * **Graphics** - Graphics seem randomly chosen, are of low quality, OR | * **Text** – Some of the text used is too small to view or the font type is wrongly chosen. * **Graphics -** Graphics seem randomly chosen, are of   low quality, | * **Text** - Most text used is clear but does not describe the content well. * **Graphics** - Graphics are related to the theme/purpos e of the application and are of | * **Text** - All text used is clear but does not describe the content well. * **Graphics -** Graphics are related to the theme/purpose of the application, are of | * **Text** - All text used is clear and able to describe the content well. * **Graphics** - Graphics are related to the theme/purpos e of the application, | **1** |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | distract the reader. | OR distract the reader. | excellent quality. | excellent quality and enhance reader interest or understanding | are thoughtfully cropped, are of high quality and enhance reader interest  or understanding. |  |  |
| **Curate** | 1. GUI Elements:    1. Apply GUI elements that assist users in using application. | * Not able to curate for required content. * **Layout** - The HTML   elements in the application are cluttered looking or confusing.   * **Navigation** Links do not take the reader to the sites/ pages described. User typically feels lost. | * Limited curation for required content. * **Layout** - The HTML   elements in the application is messy, may appear busy or boring.   * **Navigation** Links seem to be missing and don’t allow the user to easily navigate. | * Satisfactory curation for required content. * **Layout** - The HTML   elements are suitable.   * **Navigation** Links allow the reader to move from page to page, but some links seem to be missing. | * Good curation for required content. * **Layout** - The HTML   elements are suitable and usable.   * **Navigation** Links are labelled and allow the user to easily move from page to page. | * Excellent curation for required content. * **Layout** - The HTML   elements are well structured, attractive, and usable layout.   * **Navigation** Links are clearly labelled, consistently placed, and allow the user to easily move from page to page. | **1** |  |
|  | ii. The application’s ‘look and feel’ is attractive and informative. | * The application needs polish in its visual design and is not appropriate for the target audience. * **Color** | * The application needs polish in its visual design, but it is still appropriate for the target audience. * **Color**   Choice of colors and | * The application mostly follows good visual design principles (e.g.: alignment, contrast, easily read   text) and is | * The application demonstrate s good visual design principles (e.g.: alignment, contrast, easily read   text) and is | * The application clearly demonstrat es good visual design principles (e.g.: alignment,   contrast, | **1** |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Choice of colors and combinations are not suitable. | combinations do not match the concept of the application. | appropriate for the target audience.   * **Color** Choice of colors and combination s match the concept of the application. | appropriate for the target audience.   * **Color** Appropriate colors   used to produce an atmosphere that expresses the concept of the application. | easily read text) and is appropriate for the target audience.   * **Color** Appropriate colors   used to produce an atmosphere that expresses the concept  of the application. |  |  |
| **Convey** | 1. Produce a report on your application. Include the following:    1. Overview of the application and its purpose. Include explanation of the information retrieved from the given API and CRUD process. | * The overview of the application and is vague | * The overview of the application and is very brief. | * The overview of the application and its purposes is clearly described. | * Include explanation of the information retrieved from the API and CRUD process with its purposes. |  | **1** |  |
|  | ii. Description of the functionalities and features of the developed application with print screen of the pages and explanation. | * The description of the functionalitie s and features is poor and unorganized. * Incomplete print screen of the   application. | * The description of the functionalities and features is very brief. * Incomplete print screen of the application. | * The description of the functionalitie s and features is complete. * Complete screenshot of the pages. | * The description of the functionalities and features is complete with extensive information. * Complete screenshot of the pages. | * The description of the functionalitie s and features is complete with extensive information. * Complete screenshot | **1** |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | of the pages and the labelling of the features  in the application. |  |  |
|  | iii. Program codes of your system | * HTML, CSS,   and JavaScript codes attached are not complete.   * The codes are hardly read. | * HTML, CSS,   and JavaScript codes attached are complete.   * The codes are hardly read. | * HTML, CSS,   and JavaScript codes attached are complete.   * The codes are readable but not organized. | * HTML, CSS,   and JavaScript codes attached are complete.   * The codes are readable and organize. | * HTML, CSS,   and JavaScript codes attached are complete and include comments for the important parts of the codes.   * The codes are readable and   organized. | **1** |  |
|  | 6. Submit files in GitHub and include the link in the report. | * Does not submit complete electron files in GitHub | * Completely submit all the electron file in GitHub and include the link in the   report. |  |  |  | **1** |  |
| **Total Marks Earned** | | | | | | |  | **/50** |
| **Total Percentage (40%)** | | | | | | |  | **/40%** |