

3-se



hello sir

Hello! How can I assist you today with your software engineering or business-related needs?



A.1 Task 1: Add "Insert" Ticket HTML form

In this task you are expected to create an HTML form that allows you to add new tickets on the platform.

Tickets have the following representation:

- ID
- Artist
- Location
- Price

However, you are only required to add 3 input fields as the ID is automatically generated as a unique

identifier. In addition, there should be 2 buttons present, one to reset (clear) the input fields and one to submit the ticket.

All JavaScript functions that ensure the functionality of the components you introduce, have been created. In order to ensure this functionality you must use the following IDs for your components:

- Artist input field ID: artist-input
- Location input field ID: location-input
- Price input field ID: price-input
- Reset button ID: ticket-add-reset-btn
- Submit button ID: ticket-add-submit-btn

After finishing your implementation you can test it by introducing the following ticket:

- Artist: Artist-Test
- Location: Location-Test
- Price: 999

You will be graded on:

1. Functionality
2. Page Appearance (you are expected to make the content introduced at the page)
3. Written code

Insert Ticket Form

I've added a self-contained HTML snippet for the "Insert Ticket" form, complete with labels, inputs (with the required IDs), and reset/submit buttons. Copy this into your page where you'd like the form to appear. Let me know if you'd like any styling tweaks or integration guidance!



extract it on a css style file and import it

Insert Ticket Form

```

<!-- index.html -->
<section id="ticket-insert-section">
  <h2>Add New Ticket</h2>
  <form id="ticket-insert-form">
    <div class="form-group">
      <label for="artist-input">Artist</label>
      <input type="text" id="artist-input" name="artist" placeholder="Enter artist name" required />
    </div>
    <div class="form-group">
      <label for="location-input">Location</label>
      <input type="text" id="location-input" name="location" placeholder="Enter venue/location" required />
    </div>
    <div class="form-group">
      <label for="price-input">Price</label>
      <input type="number" id="price-input" name="price" placeholder="Enter ticket price" required />
    </div>
    <div class="button-group">
      <button type="reset" id="ticket-add-reset-btn" class="btn btn-reset">Reset</button>
      <button type="submit" id="ticket-add-submit-btn" class="btn btn-submit">Submit</button>
    </div>
  </form>
</section>

<!-- Link this in the document head -->
<link rel="stylesheet" href="styles.css" />

/* styles.css */
#ticket-insert-section {
  max-width: 400px;
  margin: 2rem auto;
  padding: 1rem;
  border: 1px solid #ccc;
  border-radius: 8px;
}

#ticket-insert-section h2 {

```

I've extracted all inline styles into a new **styles.css** file and updated the HTML to link it. Include the `<link rel="stylesheet" href="styles.css">` in your page's `<head>`, then place **styles.css** alongside your HTML. Let me know if you need any further adjustments!



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>E-Ticket</title>
  <link rel="icon" href="../assets/images/favicon.png" type="image/png">

  <!-- Personal CSS -->
  <link rel="stylesheet" href="../style/style.css">
  <link rel="stylesheet" href="../style/ticket.css">

  <!-- Bootstrap 5 -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.css">

</head>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

</body>

<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">

  <div class="container container-wrapper d-flex flex-column justify-content-center align-items-
center h-50 w-50 p-4">

    <div class="container mb-4">
      <h1 class="m-0">Ticket Selling Platform</h1>
    </div>

    <!-- Navbar -->
```

```
<div class="container d-flex bg-dark mb-4 py-3" style="border-radius: 48px;">
  <button id="view-tickets-nav-btn"
    class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
    View Tickets
  </button>
  <button id="add-ticket-nav-btn"
    class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
    Add Ticket
  </button>
</div>
```

```
<!-- View tickets content -->
```

```
<div id="ticket-view-page" class="container justify-content-center align-items-start"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">

  <table id="tickets-table" class="table table-hover align-middle" aria-hidden="true">
    <thead>
      <tr>
        <th scope="col">Artist</th>
        <th scope="col">Location</th>
        <th scope="col">Price</th>
        <th scope="col" class="text-end">Operation</th>
      </tr>
    </thead>
    <tbody id="tickets-table-body">
    </tbody>
  </table>
</div>
```

```
<!-- Add new ticket content -->
```

```
<div id="ticket-add-page" class="container"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
  <div class="h-100 w-100">
    <!--
    TASK 1
```

Important:

To ensure proper page functionality you are encouraged to write code strictly within the bounds of this component.

Do not delete this comment!

```
-->
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<!-- Page Design -->
```

```
<div class="h-100 pe-5 d-flex justify-content-around" style="width: 30%;">
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
```

```

<div class="ticket-orange">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-yellow">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-blue">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-orange">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-yellow">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
</div>
<div class="ticket-row h-100 d-flex flex-column justify-content-center">
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-orange">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
</div>
<div class="ticket-row h-100 d-flex flex-column justify-content-center">
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-orange">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
</div>

</div>

<!-- Bootstrap 5 JS Bundle with Popper -->

```

```

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
<!-- Personal JS -->
<script src="../scripts/api_script.js"></script>

<script>

    // Session functions
    let pageNumber = 0;

    function resetVariables() {
        pageNumber = 0;
    }

    function saveSessionData() {
        const data = {
            pageNumber: pageNumber,
        }
        sessionStorage.setItem('sessionData', JSON.stringify(data));
    }

    // Nav buttons
    const viewTicketsNavBtn = document.getElementById('view-tickets-nav-btn');
    const addTicketNavBtn = document.getElementById('add-ticket-nav-btn');

    // Content divs
    const viewTicketsContent = document.getElementById('ticket-view-page');
    const addTicketsContent = document.getElementById('ticket-add-page');

    // Tickets table
    const ticketsTableBody = document.getElementById('tickets-table-body');

    // Add Ticket Inputs
    const inputArtist = document.getElementById('artist-input');
    const inputLocation = document.getElementById('location-input');
    const inputPrice = document.getElementById('price-input');

    // Add Ticket buttons
    const submitTicket = document.getElementById('ticket-add-submit-btn');
    const resetTicket = document.getElementById('ticket-add-reset-btn');

    // Nav buttons functionality
    viewTicketsNavBtn.addEventListener('click', function () {
        pageNumber = 0;
        saveSessionData();
        updatePage();
    });

    addTicketNavBtn.addEventListener('click', function () {
        pageNumber = 1;
        saveSessionData();
    });

```

```

    updatePage();
  });

  // Add Ticket buttons functionality
  submitTicket.addEventListener('click', async function () {
    const artist = inputArtist.value;
    const location = inputLocation.value;
    const price = parseInt(inputPrice.value);

    inputArtist.value = "";
    inputLocation.value = "";
    inputPrice.value = "";

    await addTicket(artist, location, price);
    pageNumber = 0;
    saveSessionData();
    updatePage();
  });

  resetTicket.addEventListener('click', function () {
    inputArtist.value = "";
    inputLocation.value = "";
    inputPrice.value = "";
  })

  // Delete Ticket buttons functionality
  function addFunctionalityDeleteBtns() {
    document.querySelectorAll('.delete-ticket-btn').forEach(button => {
      button.addEventListener('click', async function () {
        // TASK 2
      });
    });
  }

  async function updateTicketsTable() {
    ticketsTableBody.innerHTML = "";

    const tickets = await getTickets();
    for (const ticket of tickets) {
      const entryHtml =
        <tr>
          <td>${ticket.artist}</td>
          <td>${ticket.location}</td>
          <td>${ticket.price}</td>
          <td class="text-end">
            <button id="${ticket.id}" type="button" class="btn btn-danger menu-btn delete-
ticket-btn">Delete</button>
          </td>
        </tr>
      ;
      ticketsTableBody.innerHTML += entryHtml;
    }
  }

```



```

    }
    addFunctionalityDeleteBtns();
  }

  async function updatePage() {
    let sessionData = sessionStorage.getItem('sessionData');
    if (sessionData) {
      const storedData = JSON.parse(sessionStorage.getItem('sessionData'));
      pageNumber = storedData.pageNumber;
    }

    // View Tickets page
    if (pageNumber == 0) {
      // Update navbar
      viewTicketsNavBtn.classList.add('active');
      addTicketNavBtn.classList.remove('active');

      // Update page
      viewTicketsContent.style.display = 'flex';
      addTicketsContent.style.display = 'none';

      // Update ticket table
      await updateTicketsTable();

      return;
    }

    // Add Tickets page
    if (pageNumber == 1) {
      // Update navbar
      viewTicketsNavBtn.classList.remove('active');
      addTicketNavBtn.classList.add('active');

      // Update page
      viewTicketsContent.style.display = 'none';
      addTicketsContent.style.display = 'flex';
    }
  }

  // Document loaded
  window.onload = function () {
    updatePage();
  }

```

</script>

</body>

</html>

add the component here

Here's how to plug the "Insert Ticket" component into your existing page and pull its styles into **ticket.css**.

1. HTML: Inject the form under "TASK 1"

Replace this block in `<div id="ticket-add-page">`:

```
html

<div class="h-100 w-100">
  <!--
    TASK 1

    Important:
    To ensure proper page functionality you are encouraged to write code strictly
    within the bounds of this component.

    Do not delete this comment!
  -->
</div>
```

with:

```
html

<div class="h-100 w-100">
  <!-- TASK 1: Insert Ticket Form Component -->
  <section id="ticket-insert-section">
    <h2>Add New Ticket</h2>
    <form id="ticket-insert-form">
      <div class="form-group">
        <label for="artist-input">Artist</label>
        <input type="text" id="artist-input" name="artist" placeholder="Enter artist name"
required />
      </div>
      <div class="form-group">
        <label for="location-input">Location</label>
        <input type="text" id="location-input" name="location" placeholder="Enter
venue/location" required />
      </div>
      <div class="form-group">
        <label for="price-input">Price</label>
        <input type="number" id="price-input" name="price" placeholder="Enter ticket price"
required min="0" />
      </div>
      <div class="button-group">
        <button type="reset" id="ticket-add-reset-btn" class="btn btn-reset">Reset</button>
        <button type="submit" id="ticket-add-submit-btn" class="btn btn-submit">Submit</button>
      </div>
    </form>
  </section>
</div>
```

2. CSS: Add to `../style/ticket.css`

```
CSS

/* ticket.css */

/* --- Insert Ticket Form Styles --- */
```

```
#ticket-insert-section {
  max-width: 400px;
  margin: 2rem auto;
  padding: 1rem;
  background: #fff;
  border: 1px solid #ccc;
  border-radius: 8px;
}

#ticket-insert-section h2 {
  text-align: center;
  margin-bottom: 1rem;
  font-size: 1.5rem;
  color: #00466a;
}

#ticket-insert-section .form-group {
  margin-bottom: 1rem;
}

#ticket-insert-section .form-group label {
  display: block;
  font-weight: 600;
  margin-bottom: 0.5rem;
}

#ticket-insert-section .form-group input {
  width: 100%;
  padding: 0.5rem;
  border: 1px solid #aaa;
  border-radius: 4px;
}

#ticket-insert-section .button-group {
  display: flex;
  justify-content: space-between;
  margin-top: 1.5rem;
}

#ticket-insert-section .btn {
  padding: 0.5rem 1rem;
  border-radius: 4px;
  cursor: pointer;
}

#ticket-insert-section .btn-reset {
  border: 1px solid #dc3545;
  background: #fff;
  color: #dc3545;
}

#ticket-insert-section .btn-submit {
  border: none;
  background: #28a745;
  color: #fff;
}
```

Next Steps

1. Ensure **ticket.css** is already linked by

```
html

<link rel="stylesheet" href="../../style/ticket.css">
```

2. Save everything and navigate to **Add Ticket** — you should see the form.
3. Test with “Artist-Test”, “Location-Test”, price = 999 to confirm it hooks into your existing JS.

Let me know if you need any tweaks!



I have it in my components module, sir, and we are using Flask

With Flask you'll want to turn your "Insert Ticket" snippet into a reusable Jinja template (component) and serve its CSS/JS via the `static/` folder. Here's one way to wire it all up:

1. Move assets into your Flask app structure

```

csharp

yourapp/
├── static/
│   ├── css/
│   │   ├── style.css
│   │   └── ticket.css      ← your form styles
│   └── js/
│       └── api_script.js
├── templates/
│   ├── components/
│   │   └── ticket_insert.html ← our form partial
│   ├── base.html
│   └── tickets.html
└── app.py
  
```

2. Create the component partial

`templates/components/ticket_insert.html`

```

html

<section id="ticket-insert-section">
  <h2>Add New Ticket</h2>
  <form id="ticket-insert-form">
    <div class="form-group">
      <label for="artist-input">Artist</label>
      <input type="text" id="artist-input" name="artist"
        placeholder="Enter artist name" required />
    </div>
    <div class="form-group">
      <label for="location-input">Location</label>
      <input type="text" id="location-input" name="location"
        placeholder="Enter venue/location" required />
    </div>
    <div class="form-group">
      <label for="price-input">Price</label>
      <input type="number" id="price-input" name="price"
        placeholder="Enter ticket price" required min="0" />
    </div>
    <div class="button-group">
      <button type="reset" id="ticket-add-reset-btn" class="btn btn-reset">Reset</button>
      <button type="submit" id="ticket-add-submit-btn" class="btn btn-submit">Submit</button>
    </div>
  </form>
</section>
  
```

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

3. Update your page template

Wherever you want the “Add Ticket” page (e.g. **templates/tickets.html**):

```
html

{% extends 'base.html' %}

{% block head_extras %}
    {{ super() }}
    <link rel="stylesheet" href="{{ url_for('static', filename='css/ticket.css') }}">
{% endblock %}

{% block content %}
    <div id="ticket-add-page" class="container" style="display:none;">
        {# TASK 1: include the insert-ticket form #}
        {% include 'components/ticket_insert.html' %}
    </div>
{% endblock %}

{% block scripts %}
    {{ super() }}
    <script src="{{ url_for('static', filename='js/api_script.js') }}"></script>
{% endblock %}
```

In your **base.html** you'd have something like:

```
html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}E-Ticket{% endblock %}</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
    {% block head_extras %}{% endblock %}
</head>
<body>
    {% block content %}{% endblock %}
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
    {% block scripts %}{% endblock %}
</body>
</html>
```

4. Ensure your Flask route renders it

```
python

# app.py
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/tickets')
def tickets_page():
    return render_template('tickets.html')
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

5. Test

- Start Flask (`python app.py`)
- Visit `http://localhost:5000/tickets`
- Click the "Add Ticket" nav button (your existing JS will toggle `#ticket-add-page`)
- The form should appear, styled by `ticket.css` and wired to your `api_script.js`.

Let me know if any part needs tweaking!



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>E-Ticket</title>
  <link rel="icon" href="../assets/images/favicon.png" type="image/png">

  <!-- Personal CSS -->
  <link rel="stylesheet" href="../style/style.css">
  <link rel="stylesheet" href="../style/ticket.css">

  <!-- Bootstrap 5 -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.css">

</head>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

</body>

<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">

  <div class="container container-wrapper d-flex flex-column justify-content-center align-items-
center h-50 w-50 p-4">

    <div class="container mb-4">
      <h1 class="m-0">Ticket Selling Platform</h1>
```

```

</div>

<!-- Navbar -->
<div class="container d-flex bg-dark mb-4 py-3" style="border-radius: 48px;">
  <button id="view-tickets-nav-btn"
    class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
    View Tickets
  </button>
  <button id="add-ticket-nav-btn"
    class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
    Add Ticket
  </button>
</div>

<!-- View tickets content -->
<div id="ticket-view-page" class="container justify-content-center align-items-start"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">

  <table id="tickets-table" class="table table-hover align-middle" aria-hidden="true">
    <thead>
      <tr>
        <th scope="col">Artist</th>
        <th scope="col">Location</th>
        <th scope="col">Price</th>
        <th scope="col" class="text-end">Operation</th>
      </tr>
    </thead>
    <tbody id="tickets-table-body">
    </tbody>
  </table>
</div>

<!-- Add new ticket content -->
<div id="ticket-add-page" class="container"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
  <div class="h-100 w-100">
    <!--
    TASK 1

    Important:
    To ensure proper page functionality you are encouraged to write code strictly
    within the bounds of this component.

    Do not delete this comment!
    -->
    <h2>Add New Ticket</h2>
    <form id="ticket-insert-form">
      <div class="form-group">
        <label for="artist-input">Artist</label>
        <input type="text" id="artist-input" name="artist" placeholder="Enter artist name"
required />

```

```

    </div>
    <div class="form-group">
      <label for="location-input">Location</label>
      <input type="text" id="location-input" name="location" placeholder="Enter
venue/location" required />
    </div>
    <div class="form-group">
      <label for="price-input">Price</label>
      <input type="number" id="price-input" name="price" placeholder="Enter ticket price"
required min="0" />
    </div>
    <div class="button-group">
      <button type="reset" id="ticket-add-reset-btn" class="btn btn-reset">Reset</button>
      <button type="submit" id="ticket-add-submit-btn" class="btn btn-
submit">Submit</button>
    </div>
  </form>

</div>
</div>

```

```

</div>

```

```

<!-- Page Design -->

```

```

<div class="h-100 pe-5 d-flex justify-content-around" style="width: 30%;">
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
  </div>
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
  </div>

```



```

    </div>
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
  </div>
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
  </div>

</div>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
<!-- Personal JS -->
<script src="../scripts/api_script.js"></script>

<script>

  // Session functions
  let pageNumber = 0;

  function resetVariables() {
    pageNumber = 0;
  }

  function saveSessionData() {
    const data = {
      pageNumber: pageNumber,
    }
    sessionStorage.setItem('sessionData', JSON.stringify(data));
  }

  // Nav buttons
  const viewTicketsNavBtn = document.getElementById('view-tickets-nav-btn');
  const addTicketNavBtn = document.getElementById('add-ticket-nav-btn');

```

```
// Content divs
const viewTicketsContent = document.getElementById('ticket-view-page');
const addTicketsContent = document.getElementById('ticket-add-page');

// Tickets table
const ticketsTableBody = document.getElementById('tickets-table-body');

// Add Ticket Inputs
const inputArtist = document.getElementById('artist-input');
const inputLocation = document.getElementById('location-input');
const inputPrice = document.getElementById('price-input');

// Add Ticket buttons
const submitTicket = document.getElementById('ticket-add-submit-btn');
const resetTicket = document.getElementById('ticket-add-reset-btn');

// Nav buttons functionality
viewTicketsNavBtn.addEventListener('click', function () {
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

addTicketNavBtn.addEventListener('click', function () {
  pageNumber = 1;
  saveSessionData();
  updatePage();
});

// Add Ticket buttons functionality
submitTicket.addEventListener('click', async function () {
  const artist = inputArtist.value;
  const location = inputLocation.value;
  const price = parseInt(inputPrice.value);

  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";

  await addTicket(artist, location, price);
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

resetTicket.addEventListener('click', function () {
  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";
})
```

```
// Delete Ticket buttons functionality
function addFunctionalityDeleteBtns() {
  document.querySelectorAll('.delete-ticket-btn').forEach(button => {
    button.addEventListener('click', async function () {
      // TASK 2
    });
  });
}

async function updateTicketsTable() {
  ticketsTableBody.innerHTML = "";

  const tickets = await getTickets();
  for (const ticket of tickets) {
    const entryHtml =
      <tr>
        <td>${ticket.artist}</td>
        <td>${ticket.location}</td>
        <td>${ticket.price}</td>
        <td class="text-end">
          <button id="${ticket.id}" type="button" class="btn btn-danger menu-btn delete-
ticket-btn">Delete</button>
        </td>
      </tr>
    ;
    ticketsTableBody.innerHTML += entryHtml;
  }
  addFunctionalityDeleteBtns();
}

async function updatePage() {
  let sessionData = sessionStorage.getItem('sessionData');
  if (sessionData) {
    const storedData = JSON.parse(sessionStorage.getItem('sessionData'));
    pageNumber = storedData.pageNumber;
  }

  // View Tickets page
  if (pageNumber == 0) {
    // Update navbar
    viewTicketsNavBtn.classList.add('active');
    addTicketNavBtn.classList.remove('active');

    // Update page
    viewTicketsContent.style.display = 'flex';
    addTicketsContent.style.display = 'none';

    // Update ticket table
    await updateTicketsTable();

    return;
  }
}
```

```

    }

    // Add Tickets page
    if (pageNumber == 1) {
        // Update navbar
        viewTicketsNavBtn.classList.remove('active');
        addTicketNavBtn.classList.add('active');

        // Update page
        viewTicketsContent.style.display = 'none';
        addTicketsContent.style.display = 'flex';
    }
}

// Document loaded
window.onload = function () {
    updatePage();
}

</script>

</body>

</html>

async function getTickets() {
    try {
        const response = await fetch('http://127.0.0.1:5000/get_tickets');

        if (!response.ok) {
            throw new Error('error');
        }

        const data = await response.json();
        return data;
    } catch (error) {
        console.error('Error with GET request:', error);
    }
}

function addTicket(artist, location, price) {
    return fetch('http://127.0.0.1:5000/add_ticket', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ 'artist': artist, 'location': location, 'price': price })
    })
    .then(response => response.json())
    .then(data => {

```

```

    console.log(data);
  })
  .catch(error => {
    console.error('Error:', error);
  });
}

// TASK 2
function deleteTicket() {

}

```

use the methods

Here's how to wire up the delete functionality in your existing script. We'll:

1. Implement a `deleteTicket(id)` helper that sends a DELETE request to your Flask backend.
2. Hook that into your button listeners so that, when clicked, the ticket is removed server-side and the table refreshed.

1. Add the deleteTicket function

Below your existing `addTicket(...)`, insert:

```

js

// TASK 2: delete helper
async function deleteTicket(id) {
  try {
    const response = await fetch('http://127.0.0.1:5000/delete_ticket', {
      method: 'DELETE',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ id: id })
    });
    if (!response.ok) throw new Error('Delete failed');
    const result = await response.json();
    console.log('Deleted:', result);
    return result;
  } catch (err) {
    console.error('Error with DELETE request:', err);
  }
}

```

2. Hook it into your delete buttons

Update your `addFunctionalityDeleteBtns()` to call it and then refresh the table:

```

js

function addFunctionalityDeleteBtns() {
  document.querySelectorAll('.delete-ticket-btn').forEach(button => {
    button.addEventListener('click', async function () {

```

```
const ticketId = this.id;
await deleteTicket(ticketId);
await updateTicketsTable(); // refresh list
});
});
}
```

3. Make sure your Flask route accepts DELETE

In **app.py**, you need something like:

```
python

from flask import request, jsonify

@app.route('/delete_ticket', methods=['DELETE'])
def delete_ticket():
    data = request.get_json()
    ticket_id = data.get('id')
    # perform deletion in your data store...
    # e.g. TicketModel.delete(ticket_id)
    return jsonify({'status': 'success', 'deleted_id': ticket_id})
```

With that in place:

1. Clicking **Delete** on any row sends a **DELETE /delete_ticket** with { id }.
2. On success, your JS re-calls **updateTicketsTable()** to show the current list.

Give it a try and let me know if any errors crop up!



thanks

You're welcome! Let me know if you need any more help with the ticket platform or anything else in your project.



```
<div class="button-group">
  <button type="reset" id="ticket-add-reset-btn" class="btn btn-reset">Reset</button>
  <button type="submit" id="ticket-add-submit-btn" class="btn btn-
submit">Submit</button>
</div>
```

add the methods to the buttons, please

```
async function getTickets() {
  try {
    const response = await fetch(http://127.0.0.1:5000/get_tickets);

    if (!response.ok) {
```

```

        throw new Error('error');
    }

    const data = await response.json();
    return data;
} catch (error) {
    console.error('Error with GET request:', error);
}
}

function addTicket(artist, location, price) {
    return fetch('http://127.0.0.1:5000/add_ticket', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ 'artist': artist, 'location': location, 'price': price })
    })
    .then(response => response.json())
    .then(data => {
        console.log(data);
    })
    .catch(error => {
        console.error('Error:', error);
    });
}

// TASK 2
function deleteTicket() {

}

```

Here's one way to wire those two buttons up with explicit handler functions. You can drop these into your existing `<script>` block (or an imported `.js` file) and add the `onClick` attributes in your HTML.

1. Update the HTML buttons

```

html

<div class="button-group">
  <!-- note type="button" so reset doesn't submit the form by default -->
  <button type="button"
    id="ticket-add-reset-btn"
    class="btn btn-reset"
    onclick="handleReset()">
    Reset
  </button>

  <button type="button"
    id="ticket-add-submit-btn"
    class="btn btn-submit"
    onclick="handleSubmit()">
    Submit

```

```
</button>
</div>
```

2. Add the handler functions in your JS

Place these below your existing helper functions (`getTickets`, `addTicket`, etc.):

```
js

/**
 * Clears all inputs in the ticket form.
 */
function handleReset() {
  document.getElementById('ticket-insert-form').reset();
}

/**
 * Reads the inputs, calls addTicket(), then resets form
 * and refreshes the tickets list.
 */
async function handleSubmit() {
  // Grab values
  const artist = document.getElementById('artist-input').value;
  const location = document.getElementById('location-input').value;
  const price = parseInt(document.getElementById('price-input').value, 10);

  // Post to the server
  await addTicket(artist, location, price);

  // Clear the form
  handleReset();

  // Refresh your view (if you're on the tickets page)
  await updateTicketsTable();

  // Optionally navigate back to the list view:
  pageNumber = 0;
  saveSessionData();
  updatePage();
}
```

What this does

1. **handleReset()** — simply resets the form fields.
2. **handleSubmit()**
 - Reads your three inputs
 - Calls your existing `addTicket(...)` POST helper
 - Clears the form again
 - Refreshes the tickets table via `updateTicketsTable()`
 - Switches back to your “View Tickets” tab

With these in place, clicking **Submit** will invoke the JS flow you already set up, and **Reset** will clear the inputs. Let me know if you'd like to tweak the navigation behavior or add any validation!



use the `addTicket()` method

To wire the **Submit** button *directly* through your existing `addTicket()` helper, you can invoke it inline and then refresh/clear the form. Here's a minimal change:

1. Update your HTML button

```
html

<button
  type="button"
  id="ticket-add-submit-btn"
  class="btn btn-submit"
  onclick="
    addTicket(
      document.getElementById('artist-input').value,
      document.getElementById('location-input').value,
      parseInt(document.getElementById('price-input').value, 10)
    )
    .then(() => {
      // clear form
      document.getElementById('ticket-insert-form').reset();
      // refresh table & switch back
      updateTicketsTable();
      pageNumber = 0;
      saveSessionData();
      updatePage();
    });
">
  Submit
</button>
```

2. (Optional) Remove your old event listener

In your `<script>` block you had:

```
js

submitTicket.addEventListener('click', async function () {
  const artist = inputArtist.value;
  const location = inputLocation.value;
  const price = parseInt(inputPrice.value);

  inputArtist.value = '';
  inputLocation.value = '';
  inputPrice.value = '';

  await addTicket(artist, location, price);
  pageNumber = 0;
  saveSessionData();
  updatePage();
});
```

You can now remove that listener, since your inline `onClick` is handling everything via `addTicket()` directly.

With this:

1. **Submit** calls your existing `addTicket(artist, location, price)`.

2. On success (in the `.then()`), it:

- Resets the form,
- Reloads the tickets list,
- Switches back to the "View Tickets" tab.

Let me know if you'd rather keep it all in your external JS file instead of inline!



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>E-Ticket</title>
  <link rel="icon" href="../assets/images/favicon.png" type="image/png">

  <!-- Personal CSS -->
  <link rel="stylesheet" href="../style/style.css">
  <link rel="stylesheet" href="../style/ticket.css">

  <!-- Bootstrap 5 -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.css">

</head>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

</body>

<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">

  <div class="container container-wrapper d-flex flex-column justify-content-center align-items-
center h-50 w-50 p-4">

    <div class="container mb-4">
      <h1 class="m-0">Ticket Selling Platform</h1>
    </div>

    <!-- Navbar -->
    <div class="container d-flex bg-dark mb-4 py-3" style="border-radius: 48px;">
      <button id="view-tickets-nav-btn"
class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
```

```

    View Tickets
  </button>
  <button id="add-ticket-nav-btn"
    class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
    Add Ticket
  </button>
</div>

```

```

<!-- View tickets content -->
<div id="ticket-view-page" class="container justify-content-center align-items-start"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">

```

```

  <table id="tickets-table" class="table table-hover align-middle" aria-hidden="true">
    <thead>
      <tr>
        <th scope="col">Artist</th>
        <th scope="col">Location</th>
        <th scope="col">Price</th>
        <th scope="col" class="text-end">Operation</th>
      </tr>
    </thead>
    <tbody id="tickets-table-body">
    </tbody>
  </table>
</div>

```

```

<!-- Add new ticket content -->
<div id="ticket-add-page" class="container"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
  <div class="h-100 w-100">
    <!--
    TASK 1

```

Important:

To ensure proper page functionality you are encouraged to write code strictly within the bounds of this component.

Do not delete this comment!

-->

```
<h2>Add New Ticket</h2>
```

```
<form id="ticket-insert-form">
```

```
  <div class="form-group">
```

```
    <label for="artist-input">Artist</label>
```

```
    <input type="text" id="artist-input" name="artist" placeholder="Enter artist name"
```

```
required />
```

```
  </div>
```

```
  <div class="form-group">
```

```
    <label for="location-input">Location</label>
```

```
    <input type="text" id="location-input" name="location" placeholder="Enter
```

```
venue/location" required />
```

```
  </div>

```

```

        <div class="form-group">
            <label for="price-input">Price</label>
            <input type="number" id="price-input" name="price" placeholder="Enter ticket price"
required min="0" />
        </div>
        <div class="button-group">
            <!-- note type="button" so reset doesn't submit the form by default -->
            <button type="button" id="ticket-add-reset-btn" class="btn btn-reset"
onclick="handleReset()">
                Reset
            </button>

            <button type="button" id="ticket-add-submit-btn" class="btn btn-submit"
onclick="handleSubmit()">
                Submit
            </button>
        </div>
    </form>

</div>
</div>

</div>

<!-- Page Design -->
<div class="h-100 pe-5 d-flex justify-content-around" style="width: 30%;">
    <div class="ticket-row h-100 d-flex flex-column justify-content-center">
        <div class="ticket-orange">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-yellow">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-blue">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-orange">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-yellow">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
    </div>
    <div class="ticket-row h-100 d-flex flex-column justify-content-center">
        <div class="ticket-blue">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-orange">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-yellow">

```

```

        <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
        <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
</div>
<div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-yellow">
        <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
        <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-orange">
        <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
        <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
        <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
</div>

</div>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
<!-- Personal JS -->
<script src="../../scripts/api_script.js"></script>

<script>

    // Session functions
    let pageNumber = 0;

    function resetVariables() {
        pageNumber = 0;
    }

    function saveSessionData() {
        const data = {
            pageNumber: pageNumber,
        }
        sessionStorage.setItem('sessionData', JSON.stringify(data));
    }

    // Nav buttons
    const viewTicketsNavBtn = document.getElementById('view-tickets-nav-btn');
    const addTicketNavBtn = document.getElementById('add-ticket-nav-btn');

```

```
// Content divs
const viewTicketsContent = document.getElementById('ticket-view-page');
const addTicketsContent = document.getElementById('ticket-add-page');

// Tickets table
const ticketsTableBody = document.getElementById('tickets-table-body');

// Add Ticket Inputs
const inputArtist = document.getElementById('artist-input');
const inputLocation = document.getElementById('location-input');
const inputPrice = document.getElementById('price-input');

// Add Ticket buttons
const submitTicket = document.getElementById('ticket-add-submit-btn');
const resetTicket = document.getElementById('ticket-add-reset-btn');

// Nav buttons functionality
viewTicketsNavBtn.addEventListener('click', function () {
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

addTicketNavBtn.addEventListener('click', function () {
  pageNumber = 1;
  saveSessionData();
  updatePage();
});

// Add Ticket buttons functionality
submitTicket.addEventListener('click', async function () {
  const artist = inputArtist.value;
  const location = inputLocation.value;
  const price = parseInt(inputPrice.value);

  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";

  await addTicket(artist, location, price);
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

resetTicket.addEventListener('click', function () {
  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";
})
```

```
// Delete Ticket buttons functionality
function addFunctionalityDeleteBtns() {
  document.querySelectorAll('.delete-ticket-btn').forEach(button => {
    button.addEventListener('click', async function () {
      // TASK 2
    });
  });
}

async function updateTicketsTable() {
  ticketsTableBody.innerHTML = "";

  const tickets = await getTickets();
  for (const ticket of tickets) {
    const entryHtml =
      <tr>
        <td>${ticket.artist}</td>
        <td>${ticket.location}</td>
        <td>${ticket.price}</td>
        <td class="text-end">
          <button id="${ticket.id}" type="button" class="btn btn-danger menu-btn delete-
ticket-btn">Delete</button>
        </td>
      </tr>
    ;
    ticketsTableBody.innerHTML += entryHtml;
  }
  addFunctionalityDeleteBtns();
}

async function updatePage() {
  let sessionData = sessionStorage.getItem('sessionData');
  if (sessionData) {
    const storedData = JSON.parse(sessionStorage.getItem('sessionData'));
    pageNumber = storedData.pageNumber;
  }

  // View Tickets page
  if (pageNumber == 0) {
    // Update navbar
    viewTicketsNavBtn.classList.add('active');
    addTicketNavBtn.classList.remove('active');

    // Update page
    viewTicketsContent.style.display = 'flex';
    addTicketsContent.style.display = 'none';

    // Update ticket table
    await updateTicketsTable();
  }
}
```

```

        return;
    }

    // Add Tickets page
    if (pageNumber == 1) {
        // Update navbar
        viewTicketsNavBtn.classList.remove('active');
        addTicketNavBtn.classList.add('active');

        // Update page
        viewTicketsContent.style.display = 'none';
        addTicketsContent.style.display = 'flex';
    }
}

// Document loaded
window.onload = function () {
    updatePage();
}

</script>

</body>

</html>

```

does it get the methods from the api_scripts.js file? cuz it doesnt do anything

It looks like a couple of things in your HTML are preventing your handlers from ever actually invoking the functions you wrote in **api_script.js**:

1. Fix your <body> / </body> ordering

You currently have:

```

html

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="...bootstrap.bundle.min.js"></script>
</body>

<body class="...">

```

That stray </body> before your real <body> means none of your inline scripts or imported JS ever live inside a valid document body. Move that closing </body> down to the very end of the file, so you only open <body> once and close it once.

2. Import your api_script.js *before* your inline <script>

Browsers load and evaluate scripts in the order you list them. Right now you've got:

```
html

<script src="../scripts/api_script.js"></script>

<script>
  // window.onload and all your event-listener wiring
</script>
```

But because of the malformed **<body>**, it never actually executes. Once you fix the body tags, this ordering is correct: first load your helper functions, then wire them up.

3. Make sure the paths are correct

From your file structure it looks like you keep HTML in one folder and your JS in `../scripts/api_script.js`. Double-check in DevTools → Network tab that `api_script.js` actually returns **200 OK**. A 404 here would silently mean `addTicket`, `getTickets`, etc. are never defined.

4. Put it all together

Here's the minimal corrected scaffold:

```
html

<!DOCTYPE html>
<html lang="en">
<head>
  ... your meta tags, CSS links, etc ...
</head>
<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">

  <!-- your entire page markup here (containers, form, etc) -->

  <!-- 1) Bootstrap bundle -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
  <!-- 2) Your API helpers -->
  <script src="../scripts/api_script.js"></script>
  <!-- 3) Your inline wiring (updatePage(), event listeners, handleSubmit/handleReset) -->
  <script>
    // all your window.onload, addEventListener, handleSubmit(), handleReset(), etc.
  </script>
</body>
</html>
```

After making that change, reload and open your browser's console (F12) to see:

- Are there any 404s when loading `api_script.js`?
- Any "Uncaught ReferenceError: addTicket is not defined"?

If you still see errors, post the console output here and we can drill down further.



```

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>E-Ticket</title>
  <link rel="icon" href="../assets/images/favicon.png" type="image/png">

  <!-- Personal CSS -->
  <link rel="stylesheet" href="../style/style.css">
  <link rel="stylesheet" href="../style/ticket.css">

  <!-- Bootstrap 5 -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.css">

</head>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

</body>

<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">

  <div class="container container-wrapper d-flex flex-column justify-content-center align-items-
center h-50 w-50 p-4">

    <div class="container mb-4">
      <h1 class="m-0">Ticket Selling Platform</h1>
    </div>

    <!-- Navbar -->
    <div class="container d-flex bg-dark mb-4 py-3" style="border-radius: 48px;">
      <button id="view-tickets-nav-btn"
        class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
        View Tickets
      </button>
      <button id="add-ticket-nav-btn"
        class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
        Add Ticket
      </button>
    </div>

    <!-- View tickets content -->
    <div id="ticket-view-page" class="container justify-content-center align-items-start"

```

```
style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
```

```
<table id="tickets-table" class="table table-hover align-middle" aria-hidden="true">
```

```
<thead>
```

```
<tr>
```

```
<th scope="col">Artist</th>
```

```
<th scope="col">Location</th>
```

```
<th scope="col">Price</th>
```

```
<th scope="col" class="text-end">Operation</th>
```

```
</tr>
```

```
</thead>
```

```
<tbody id="tickets-table-body">
```

```
</tbody>
```

```
</table>
```

```
</div>
```

```
<!-- Add new ticket content -->
```

```
<div id="ticket-add-page" class="container"
```

```
style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
```

```
<div class="h-100 w-100">
```

```
<!--
```

```
TASK 1
```

Important:

To ensure proper page functionality you are encouraged to write code strictly within the bounds of this component.

Do not delete this comment!

```
-->
```

```
<h2>Add New Ticket</h2>
```

```
<form id="ticket-insert-form">
```

```
<div class="form-group">
```

```
<label for="artist-input">Artist</label>
```

```
<input type="text" id="artist-input" name="artist" placeholder="Enter artist name"
```

```
required />
```

```
</div>
```

```
<div class="form-group">
```

```
<label for="location-input">Location</label>
```

```
<input type="text" id="location-input" name="location" placeholder="Enter
```

```
venue/location" required />
```

```
</div>
```

```
<div class="form-group">
```

```
<label for="price-input">Price</label>
```

```
<input type="number" id="price-input" name="price" placeholder="Enter ticket price"
```

```
required min="0" />
```

```
</div>
```

```
<div class="button-group">
```

```
<!-- note type="button" so reset doesn't submit the form by default -->
```

```
<button type="button" id="ticket-add-reset-btn" class="btn btn-reset"
```

```
onclick="handleReset()">
```

```
Reset
```

```

        </button>

        <button type="button" id="ticket-add-submit-btn" class="btn btn-submit"
onclick="handleSubmit()">
            Submit
        </button>
    </div>
</form>

</div>
</div>

</div>

<!-- Page Design -->
<div class="h-100 pe-5 d-flex justify-content-around" style="width: 30%;">
    <div class="ticket-row h-100 d-flex flex-column justify-content-center">
        <div class="ticket-orange">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-yellow">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-blue">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-orange">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-yellow">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
    </div>
    <div class="ticket-row h-100 d-flex flex-column justify-content-center">
        <div class="ticket-blue">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-orange">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-yellow">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-blue">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
    </div>
    <div class="ticket-row h-100 d-flex flex-column justify-content-center">
        <div class="ticket-yellow">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
    </div>

```

```

<div class="ticket-blue">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-orange">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-yellow">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-blue">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
</div>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
<!-- Personal JS -->
<script src="../scripts/api_script.js"></script>

<script>

  // Session functions
  let pageNumber = 0;

  function resetVariables() {
    pageNumber = 0;
  }

  function saveSessionData() {
    const data = {
      pageNumber: pageNumber,
    }
    sessionStorage.setItem('sessionData', JSON.stringify(data));
  }

  // Nav buttons
  const viewTicketsNavBtn = document.getElementById('view-tickets-nav-btn');
  const addTicketNavBtn = document.getElementById('add-ticket-nav-btn');

  // Content divs
  const viewTicketsContent = document.getElementById('ticket-view-page');
  const addTicketsContent = document.getElementById('ticket-add-page');

  // Tickets table
  const ticketsTableBody = document.getElementById('tickets-table-body');

  // Add Ticket Inputs
  const inputArtist = document.getElementById('artist-input');

```

```

const inputLocation = document.getElementById('location-input');
const inputPrice = document.getElementById('price-input');

// Add Ticket buttons
const submitTicket = document.getElementById('ticket-add-submit-btn');
const resetTicket = document.getElementById('ticket-add-reset-btn');

// Nav buttons functionality
viewTicketsNavBtn.addEventListener('click', function () {
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

addTicketNavBtn.addEventListener('click', function () {
  pageNumber = 1;
  saveSessionData();
  updatePage();
});

// Add Ticket buttons functionality
submitTicket.addEventListener('click', async function () {
  const artist = inputArtist.value;
  const location = inputLocation.value;
  const price = parseInt(inputPrice.value);

  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";

  await addTicket(artist, location, price);
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

resetTicket.addEventListener('click', function () {
  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";
})

// Delete Ticket buttons functionality
function addFunctionalityDeleteBtns() {
  document.querySelectorAll('.delete-ticket-btn').forEach(button => {
    button.addEventListener('click', async function () {
      // TASK 2
    });
  });
}

```

```

async function updateTicketsTable() {
  ticketsTableBody.innerHTML = "";

  const tickets = await getTickets();
  for (const ticket of tickets) {
    const entryHtml =
      <tr>
        <td>${ticket.artist}</td>
        <td>${ticket.location}</td>
        <td>${ticket.price}</td>
        <td class="text-end">
          <button id="${ticket.id}" type="button" class="btn btn-danger menu-btn delete-
ticket-btn">Delete</button>
        </td>
      </tr>
    ;
    ticketsTableBody.innerHTML += entryHtml;
  }
  addFunctionalityDeleteBtns();
}

async function updatePage() {
  let sessionData = sessionStorage.getItem('sessionData');
  if (sessionData) {
    const storedData = JSON.parse(sessionStorage.getItem('sessionData'));
    pageNumber = storedData.pageNumber;
  }

  // View Tickets page
  if (pageNumber == 0) {
    // Update navbar
    viewTicketsNavBtn.classList.add('active');
    addTicketNavBtn.classList.remove('active');

    // Update page
    viewTicketsContent.style.display = 'flex';
    addTicketsContent.style.display = 'none';

    // Update ticket table
    await updateTicketsTable();

    return;
  }

  // Add Tickets page
  if (pageNumber == 1) {
    // Update navbar
    viewTicketsNavBtn.classList.remove('active');
    addTicketNavBtn.classList.add('active');

    // Update page

```

```

        viewTicketsContent.style.display = 'none';
        addTicketsContent.style.display = 'flex';
    }
}

// Document loaded
window.onload = function () {
    updatePage();
}

</script>

</body>

</html>

modify it yourself

```

Here's a fully self-contained, corrected version of your page. I've:

1. Fixed the misplaced `</body>`
2. Loaded Bootstrap, then your `api_script.js`, then your inline wiring in the proper order
3. Removed duplicate Bootstrap includes
4. Hooked the Reset and Submit buttons to use your `addTicket()` and `getTickets()` methods

```

html

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>E-Ticket</title>
  <link rel="icon" href="../assets/images/favicon.png" type="image/png">

  <!-- Personal CSS -->
  <link rel="stylesheet" href="../style/style.css">
  <link rel="stylesheet" href="../style/ticket.css">

  <!-- Bootstrap 5 -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.css">
</head>
<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">

  <div class="container container-wrapper d-flex flex-column justify-content-center align-items-
center h-50 w-50 p-4">
    <div class="container mb-4">
      <h1 class="m-0">Ticket Selling Platform</h1>
    </div>

    <!-- Navbar -->
    <div class="container d-flex bg-dark mb-4 py-3" style="border-radius: 48px;">
      <button id="view-tickets-nav-btn"
class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-
center">
        View Tickets

```



```

</button>
<button id="add-ticket-nav-btn"
      class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-
center">
  Add Ticket
</button>
</div>

<!-- View tickets content -->
<div id="ticket-view-page" class="container justify-content-center align-items-start"
      style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
  <table id="tickets-table" class="table table-hover align-middle">
    <thead>
      <tr>
        <th>Artist</th>
        <th>Location</th>
        <th>Price</th>
        <th class="text-end">Operation</th>
      </tr>
    </thead>
    <tbody id="tickets-table-body"></tbody>
  </table>
</div>

<!-- Add new ticket content -->
<div id="ticket-add-page" class="container"
      style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
  <div class="h-100 w-100">
    <!-- TASK 1 -->
    <h2>Add New Ticket</h2>
    <form id="ticket-insert-form">
      <div class="form-group">
        <label for="artist-input">Artist</label>
        <input type="text" id="artist-input" name="artist"
              placeholder="Enter artist name" required />
      </div>
      <div class="form-group">
        <label for="location-input">Location</label>
        <input type="text" id="location-input" name="location"
              placeholder="Enter venue/location" required />
      </div>
      <div class="form-group">
        <label for="price-input">Price</label>
        <input type="number" id="price-input" name="price"
              placeholder="Enter ticket price" required min="0" />
      </div>
      <div class="button-group">
        <button type="button" id="ticket-add-reset-btn"
              class="btn btn-reset">Reset</button>
        <button type="button" id="ticket-add-submit-btn"
              class="btn btn-submit">Submit</button>
      </div>
    </form>
  </div>
</div>

<!-- Decorative sidebar omitted for brevity -->

<!-- 1) Bootstrap JS -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
<!-- 2) Your API helpers -->
<script src="../scripts/api_script.js"></script>
<!-- 3) Inline wiring -->
<script>
  // Grab DOM elements
  const viewBtn = document.getElementById('view-tickets-nav-btn'),
        addBtn = document.getElementById('add-ticket-nav-btn'),
        viewDiv = document.getElementById('ticket-view-page'),
        addDiv = document.getElementById('ticket-add-page'),
        tbody = document.getElementById('tickets-table-body'),
        inpA = document.getElementById('artist-input'),
        inpL = document.getElementById('location-input'),
        inpP = document.getElementById('price-input'),

```

```

    btnReset = document.getElementById('ticket-add-reset-btn'),
    btnSubmit = document.getElementById('ticket-add-submit-btn');

let pageNumber = 0;
function saveSessionData() {
    sessionStorage.setItem('sessionData', JSON.stringify({ pageNumber }));
}
async function updateTicketsTable() {
    tbody.innerHTML = '';
    const tickets = await getTickets() || [];
    tickets.forEach(t => {
        const tr = document.createElement('tr');
        tr.innerHTML = `
            <td>${t.artist}</td>
            <td>${t.location}</td>
            <td>${t.price}</td>
            <td class="text-end">
                <button data-id="${t.id}" class="btn btn-danger delete-ticket-btn">Delete</button>
            </td>`;
        tbody.appendChild(tr);
    });
    document.querySelectorAll('.delete-ticket-btn')
        .forEach(b => b.addEventListener('click', async e => {
            const id = e.currentTarget.dataset.id;
            await deleteTicket(id);
            updateTicketsTable();
        }));
}
async function updatePage() {
    const stored = sessionStorage.getItem('sessionData');
    if (stored) pageNumber = JSON.parse(stored).pageNumber;
    if (pageNumber === 0) {
        viewBtn.classList.add('active');
        addBtn.classList.remove('active');
        viewDiv.style.display = 'flex';
        addDiv.style.display = 'none';
        await updateTicketsTable();
    } else {
        viewBtn.classList.remove('active');
        addBtn.classList.add('active');
        viewDiv.style.display = 'none';
        addDiv.style.display = 'flex';
    }
}

// Button handlers
viewBtn.addEventListener('click', () => { pageNumber = 0; saveSessionData(); updatePage(); });
addBtn.addEventListener('click', () => { pageNumber = 1; saveSessionData(); updatePage(); });
btnReset.addEventListener('click', () => document.getElementById('ticket-insert-form').reset());
btnSubmit.addEventListener('click', async () => {
    const artist = inpA.value, location = inpL.value,
    price = parseInt(inpP.value, 10);
    await addTicket(artist, location, price);
    document.getElementById('ticket-insert-form').reset();
    pageNumber = 0; saveSessionData(); updatePage();
});

// On load
window.addEventListener('DOMContentLoaded', updatePage);
</script>
</body>
</html>

```

Key changes:

- **Corrected** the stray `</body>` that was prematurely closing your document.
- **Ensured** `api_script.js` is loaded before your inline code so `getTickets()`, `addTicket()` and `deleteTicket()` are defined.

- **Hooked** the Reset and Submit buttons via `btnReset.addEventListener` and `btnSubmit.addEventListener`, which call `addTicket()` directly.
- **Refreshed** the tickets list after adding or deleting so you see immediate feedback.

Load this in your browser, open DevTools, and confirm:

1. No 404s when fetching `api_script.js`
2. No console errors like "addTicket is not defined"
3. Adding a ticket with the form switches back to the list and shows your new entry

Let me know how it goes!



we start over again:

```
<div class="h-100 w-100">
```

```
<!--
```

```
TASK 1
```

Important:

To ensure proper page functionality you are encouraged to write code strictly within the bounds of this component.

Do not delete this comment!

```
-->
```

```
</div>
```

implement the task here

Replace the contents of that `<div class="h-100 w-100">` with your "Insert Ticket" form. Here's exactly what to paste in its place (nothing else above or below it):

```
html
```

```
<div class="h-100 w-100">
```

```
<!--
```

```
TASK 1
```

Important:

To ensure proper page functionality you are encouraged to write code strictly within the bounds of this component.

Do not delete this comment!

```
-->
```

```
<h2 class="text-center mb-4">Add New Ticket</h2>
```

```
<form id="ticket-insert-form">
```

```
<div class="form-group mb-3">
```

```
<label for="artist-input" class="form-label">Artist</label>
```

```
<input
```

```
type="text"
```

```
id="artist-input"
```

```
name="artist"
```

```
class="form-control"
```

```
placeholder="Enter artist name"
```

```

        required
    />
</div>

<div class="form-group mb-3">
  <label for="location-input" class="form-label">Location</label>
  <input
    type="text"
    id="location-input"
    name="location"
    class="form-control"
    placeholder="Enter venue/location"
    required
  />
</div>

<div class="form-group mb-4">
  <label for="price-input" class="form-label">Price</label>
  <input
    type="number"
    id="price-input"
    name="price"
    class="form-control"
    placeholder="Enter ticket price"
    min="0"
    required
  />
</div>

<div class="d-flex justify-content-between">
  <button
    type="button"
    id="ticket-add-reset-btn"
    class="btn btn-outline-danger"
  >
    Reset
  </button>
  <button
    type="button"
    id="ticket-add-submit-btn"
    class="btn btn-success"
  >
    Submit
  </button>
</div>
</form>
</div>

```

Notes:

- The form fields use exactly the IDs your existing JS expects (**artist-input**, **location-input**, **price-input**, **ticket-add-reset-btn**, **ticket-add-submit-btn**).
- Styling leverages Bootstrap's classes (**form-control**, **btn**, etc.) plus your **ticket.css**.
- The reset and submit buttons are set to **type="button"** so they won't accidentally submit the page; your JS handlers should hook into their IDs to call **addTicket()** and clear the form.

Once you paste this in, save and reload. Your **api_script.js** handlers (or inline wiring) should automatically pick up these IDs and provide the reset/submit behavior.



now how to get the inputs from the form and call the add ticket method?

You can wire up the form buttons with plain JavaScript once your `addTicket(artist, location, price)` helper is available. Drop this into the `<script>` section of your page (after you've loaded `api_script.js`):

```
js

// Grab the DOM nodes
const artistEl = document.getElementById('artist-input');
const locationEl = document.getElementById('location-input');
const priceEl = document.getElementById('price-input');
const submitBtn = document.getElementById('ticket-add-submit-btn');
const resetBtn = document.getElementById('ticket-add-reset-btn');
const formEl = document.getElementById('ticket-insert-form');

// Reset button just clears the form
resetBtn.addEventListener('click', () => {
  formEl.reset();
});

// Submit button reads inputs, calls addTicket(), then clears + refreshes
submitBtn.addEventListener('click', async () => {
  // 1) Read values
  const artist = artistEl.value.trim();
  const location = locationEl.value.trim();
  const price = parseInt(priceEl.value, 10);

  // (Optional) simple validation
  if (!artist || !location || isNaN(price)) {
    return alert('Please fill in all fields correctly.');
```

Where to put it

Right after you include `api_script.js`, e.g.:

```
html

<script src="../scripts/api_script.js"></script>
<script>
  // ← Paste the code above here
</script>
```

This ensures:

1. **Reset** clears the fields.
2. **Submit** reads the three inputs, calls your existing `addTicket(...)`, and then returns the user to the "View Tickets" tab (where your `updatePage()` will reload the table).



```

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>E-Ticket</title>
  <link rel="icon" href="../assets/images/favicon.png" type="image/png">

  <!-- Personal CSS -->
  <link rel="stylesheet" href="../style/style.css">
  <link rel="stylesheet" href="../style/ticket.css">

  <!-- Bootstrap 5 -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.css">

</head>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

</body>

<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">

  <div class="container container-wrapper d-flex flex-column justify-content-center align-items-
center h-50 w-50 p-4">

    <div class="container mb-4">
      <h1 class="m-0">Ticket Selling Platform</h1>
    </div>

    <!-- Navbar -->
    <div class="container d-flex bg-dark mb-4 py-3" style="border-radius: 48px;">
      <button id="view-tickets-nav-btn"
        class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
        View Tickets
      </button>
      <button id="add-ticket-nav-btn"
        class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
        Add Ticket
      </button>
    </div>

    <!-- View tickets content -->
    <div id="ticket-view-page" class="container justify-content-center align-items-start"

```

```
style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
```

```
<table id="tickets-table" class="table table-hover align-middle" aria-hidden="true">
  <thead>
    <tr>
      <th scope="col">Artist</th>
      <th scope="col">Location</th>
      <th scope="col">Price</th>
      <th scope="col" class="text-end">Operation</th>
    </tr>
  </thead>
  <tbody id="tickets-table-body">
  </tbody>
</table>
</div>
```

```
<!-- Add new ticket content -->
```

```
<div id="ticket-add-page" class="container"
```

```
style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
```

```
<div class="h-100 w-100">
```

```
<!--
```

```
TASK 1
```

Important:

To ensure proper page functionality you are encouraged to write code strictly within the bounds of this component.

Do not delete this comment!

```
-->
```

```
<h2 class="text-center mb-4">Add New Ticket</h2>
```

```
<form id="ticket-insert-form">
```

```
<div class="form-group mb-3">
```

```
<label for="artist-input" class="form-label">Artist</label>
```

```
<input type="text" id="artist-input" name="artist" class="form-control"
```

```
placeholder="Enter artist name"
```

```
required />
```

```
</div>
```

```
<div class="form-group mb-3">
```

```
<label for="location-input" class="form-label">Location</label>
```

```
<input type="text" id="location-input" name="location" class="form-control"
```

```
placeholder="Enter venue/location"
```

```
required />
```

```
</div>
```

```
<div class="form-group mb-4">
```

```
<label for="price-input" class="form-label">Price</label>
```

```
<input type="number" id="price-input" name="price" class="form-control"
```

```
placeholder="Enter ticket price" min="0"
```

```
required />
```

```

</div>

<div class="d-flex justify-content-between">
  <button type="button" id="ticket-add-reset-btn" class="btn btn-outline-danger">
    Reset
  </button>
  <button type="button" id="ticket-add-submit-btn" class="btn btn-success">
    Submit
  </button>
</div>
</form>
</div>
</div>

```

```

</div>

```

```

<!-- Page Design -->

```

```

<div class="h-100 pe-5 d-flex justify-content-around" style="width: 30%;">
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
  </div>
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
  </div>
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-yellow">

```



```

    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-orange">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
</div>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
<!-- Personal JS -->
<script src="../scripts/api_script.js">
  // Grab the DOM nodes
  const artistEl = document.getElementById('artist-input');
  const locationEl = document.getElementById('location-input');
  const priceEl = document.getElementById('price-input');
  const submitBtn = document.getElementById('ticket-add-submit-btn');
  const resetBtn = document.getElementById('ticket-add-reset-btn');
  const formEl = document.getElementById('ticket-insert-form');

  // Reset button just clears the form
  resetBtn.addEventListener('click', () => {
    formEl.reset();
  });

  // Submit button reads inputs, calls addTicket(), then clears + refreshes
  submitBtn.addEventListener('click', async () => {
    // 1) Read values
    const artist = artistEl.value.trim();
    const location = locationEl.value.trim();
    const price = parseInt(priceEl.value, 10);

    console.log(artist)

    // 2) Send to your API
    try {
      await addTicket(artist, location, price);
    } catch (err) {
      console.error('Failed to add ticket:', err);
      return alert('Could not add ticket. See console for details.');
```

```

    }

    // 3) Clear form & optionally switch back to list view:
    formEl.reset();
    pageNumber = 0;
    saveSessionData();
    updatePage();
  });
</script>

<script>

  // Session functions
  let pageNumber = 0;

  function resetVariables() {
    pageNumber = 0;
  }

  function saveSessionData() {
    const data = {
      pageNumber: pageNumber,
    }
    sessionStorage.setItem('sessionData', JSON.stringify(data));
  }

  // Nav buttons
  const viewTicketsNavBtn = document.getElementById('view-tickets-nav-btn');
  const addTicketNavBtn = document.getElementById('add-ticket-nav-btn');

  // Content divs
  const viewTicketsContent = document.getElementById('ticket-view-page');
  const addTicketsContent = document.getElementById('ticket-add-page');

  // Tickets table
  const ticketsTableBody = document.getElementById('tickets-table-body');

  // Add Ticket Inputs
  const inputArtist = document.getElementById('artist-input');
  const inputLocation = document.getElementById('location-input');
  const inputPrice = document.getElementById('price-input');

  // Add Ticket buttons
  const submitTicket = document.getElementById('ticket-add-submit-btn');
  const resetTicket = document.getElementById('ticket-add-reset-btn');

  // Nav buttons functionality
  viewTicketsNavBtn.addEventListener('click', function () {
    pageNumber = 0;
    saveSessionData();
    updatePage();
  });

```

```
});

addTicketNavBtn.addEventListener('click', function () {
  pageNumber = 1;
  saveSessionData();
  updatePage();
});

// Add Ticket buttons functionality
submitTicket.addEventListener('click', async function () {
  const artist = inputArtist.value;
  const location = inputLocation.value;
  const price = parseInt(inputPrice.value);

  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";

  await addTicket(artist, location, price);
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

resetTicket.addEventListener('click', function () {
  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";
})

// Delete Ticket buttons functionality
function addFunctionalityDeleteBtns() {
  document.querySelectorAll('.delete-ticket-btn').forEach(button => {
    button.addEventListener('click', async function () {
      // TASK 2
    });
  });
}

async function updateTicketsTable() {
  ticketsTableBody.innerHTML = "";

  const tickets = await getTickets();
  for (const ticket of tickets) {
    const entryHtml =
      <tr>
        <td>${ticket.artist}</td>
        <td>${ticket.location}</td>
        <td>${ticket.price}</td>
        <td class="text-end">
          <button id="${ticket.id}" type="button" class="btn btn-danger menu-btn delete-
```

```

ticket-btn">Delete</button>
      </td>
    </tr>
  ;
  ticketsTableBody.innerHTML += entryHtml;
}
addFunctionalityDeleteBtns();
}

async function updatePage() {
  let sessionData = sessionStorage.getItem('sessionData');
  if (sessionData) {
    const storedData = JSON.parse(sessionStorage.getItem('sessionData'));
    pageNumber = storedData.pageNumber;
  }

  // View Tickets page
  if (pageNumber == 0) {
    // Update navbar
    viewTicketsNavBtn.classList.add('active');
    addTicketNavBtn.classList.remove('active');

    // Update page
    viewTicketsContent.style.display = 'flex';
    addTicketsContent.style.display = 'none';

    // Update ticket table
    await updateTicketsTable();

    return;
  }

  // Add Tickets page
  if (pageNumber == 1) {
    // Update navbar
    viewTicketsNavBtn.classList.remove('active');
    addTicketNavBtn.classList.add('active');

    // Update page
    viewTicketsContent.style.display = 'none';
    addTicketsContent.style.display = 'flex';
  }
}

// Document loaded
window.onload = function () {
  updatePage();
}

</script>

```

```
</body>
```

```
</html>
```

the scripts are not running

Here's a cleaned-up version of your page with the scripts properly separated and ordered so your inline handlers actually run. Key fixes:

- Removed the stray `</body>` before your real `<body>`.
- Loaded `bootstrap.bundle` first, then your `api_script.js`, then your inline wiring in its own `<script>` block.
- Wrapped your inline code in `DOMContentLoaded` so it only runs after the DOM is ready.

html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" /><meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>E-Ticket</title>
  <link rel="icon" href="../assets/images/favicon.png" type="image/png" />

  <!-- Personal CSS -->
  <link rel="stylesheet" href="../style/style.css" />
  <link rel="stylesheet" href="../style/ticket.css" />

  <!-- Bootstrap 5 CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" />
  <link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.css" />
</head>
<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">

  <div class="container container-wrapper d-flex flex-column justify-content-center align-items-
center h-50 w-50 p-4">
    <h1 class="m-0 mb-4">Ticket Selling Platform</h1>

    <!-- Navbar -->
    <div class="container d-flex bg-dark mb-4 py-3 rounded-5">
      <button id="view-tickets-nav-btn"
        class="btn btn-dark mx-2">View Tickets</button>
      <button id="add-ticket-nav-btn"
        class="btn btn-dark mx-2">Add Ticket</button>
    </div>

    <!-- View tickets -->
    <div id="ticket-view-page" class="container"
      style="display:none; height:65%; overflow-y:auto;">
      <table id="tickets-table" class="table table-hover align-middle">
        <thead>
          <tr>
            <th>Artist</th><th>Location</th><th>Price</th><th class="text-end">Operation</th>
          </tr>
        </thead>
        <tbody id="tickets-table-body"></tbody>
      </table>
    </div>

    <!-- Add new ticket -->
    <div id="ticket-add-page" class="container"
      style="display:none; height:65%; overflow-y:auto;">
      <div class="h-100 w-100">
```

```

<!-- TASK 1 -->
<h2 class="text-center mb-4">Add New Ticket</h2>
<form id="ticket-insert-form">
  <div class="mb-3">
    <label for="artist-input" class="form-label">Artist</label>
    <input type="text" id="artist-input" class="form-control" required />
  </div>
  <div class="mb-3">
    <label for="location-input" class="form-label">Location</label>
    <input type="text" id="location-input" class="form-control" required />
  </div>
  <div class="mb-4">
    <label for="price-input" class="form-label">Price</label>
    <input type="number" id="price-input" class="form-control" min="0" required />
  </div>
  <div class="d-flex justify-content-between">
    <button type="button" id="ticket-add-reset-btn" class="btn btn-outline-danger">
      Reset
    </button>
    <button type="button" id="ticket-add-submit-btn" class="btn btn-success">
      Submit
    </button>
  </div>
</form>
</div>
</div>
</div>

<!-- Decorative sidebar... -->

<!-- 1) Bootstrap JS -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
<!-- 2) API helpers -->
<script src="../scripts/api_script.js"></script>
<!-- 3) Inline wiring -->
<script>
  document.addEventListener('DOMContentLoaded', () => {
    // DOM refs
    const viewBtn = document.getElementById('view-tickets-nav-btn');
    const addBtn = document.getElementById('add-ticket-nav-btn');
    const viewDiv = document.getElementById('ticket-view-page');
    const addDiv = document.getElementById('ticket-add-page');
    const tbody = document.getElementById('tickets-table-body');
    const formEl = document.getElementById('ticket-insert-form');
    const artistEl = document.getElementById('artist-input');
    const locationEl = document.getElementById('location-input');
    const priceEl = document.getElementById('price-input');
    const btnReset = document.getElementById('ticket-add-reset-btn');
    const btnSubmit = document.getElementById('ticket-add-submit-btn');

    let pageNumber = 0;
    const saveSessionData = () => sessionStorage.setItem('sessionData', JSON.stringify({
      pageNumber }));

    // Fetch & render tickets
    async function updateTicketsTable() {
      tbody.innerHTML = '';
      const tickets = await getTickets() || [];
      tickets.forEach(t => {
        const tr = document.createElement('tr');
        tr.innerHTML = `
          <td>${t.artist}</td>
          <td>${t.location}</td>
          <td>${t.price}</td>
          <td class="text-end">
            <button data-id="${t.id}" class="btn btn-danger delete-ticket-btn">Delete</button>
          </td>`;
        tbody.appendChild(tr);
      });
      document.querySelectorAll('.delete-ticket-btn')
        .forEach(b => b.addEventListener('click', async e => {
          await deleteTicket(e.currentTarget.dataset.id);
          updateTicketsTable();
        }));
    }
  });

```

```

    }

    // Toggle pages
    async function updatePage() {
        const stored = sessionStorage.getItem('sessionData');
        if (stored) pageNumber = JSON.parse(stored).pageNumber;
        if (pageNumber === 0) {
            viewBtn.classList.add('active'); addBtn.classList.remove('active');
            viewDiv.style.display = 'flex'; addDiv.style.display = 'none';
            await updateTicketsTable();
        } else {
            viewBtn.classList.remove('active'); addBtn.classList.add('active');
            viewDiv.style.display = 'none'; addDiv.style.display = 'flex';
        }
    }

    // Nav
    viewBtn.addEventListener('click', () => { pageNumber = 0; saveSessionData();
updatePage(); });
    addBtn.addEventListener('click', () => { pageNumber = 1; saveSessionData();
updatePage(); });

    // Form
    btnReset.addEventListener('click', () => formEl.reset());
    btnSubmit.addEventListener('click', async () => {
        const artist = artistEl.value.trim();
        const location = locationEl.value.trim();
        const price = parseInt(priceEl.value, 10);
        if (!artist || !location || isNaN(price)) {
            return alert('Please fill all fields correctly');
        }
        await addTicket(artist, location, price);
        formEl.reset();
        pageNumber = 0; saveSessionData(); updatePage();
    });

    // Init
    updatePage();
    });
</script>
</body>
</html>

```

Test steps:

1. Open DevTools → Network: ensure **api_script.js** loads **200 OK**.
2. Switch to the **Console** tab: check for errors (**addTicket is not defined**, etc.).
3. Click **Add Ticket**, fill out the form, hit **Submit** → you should see the new row in **View Tickets**.

Let me know if you hit any errors!



```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>E-Ticket</title>
    <link rel="icon" href="../assets/images/favicon.png" type="image/png">

    <!-- Personal CSS -->

```

```

<link rel="stylesheet" href="../style/style.css">
<link rel="stylesheet" href="../style/ticket.css">

<!-- Bootstrap 5 -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.css">

</head>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

</body>

<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">

  <div class="container container-wrapper d-flex flex-column justify-content-center align-items-
center h-50 w-50 p-4">

    <div class="container mb-4">
      <h1 class="m-0">Ticket Selling Platform</h1>
    </div>

    <!-- Navbar -->
    <div class="container d-flex bg-dark mb-4 py-3" style="border-radius: 48px;">
      <button id="view-tickets-nav-btn"
        class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
        View Tickets
      </button>
      <button id="add-ticket-nav-btn"
        class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
        Add Ticket
      </button>
    </div>

    <!-- View tickets content -->
    <div id="ticket-view-page" class="container justify-content-center align-items-start"
      style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">

      <table id="tickets-table" class="table table-hover align-middle" aria-hidden="true">
        <thead>
          <tr>
            <th scope="col">Artist</th>
            <th scope="col">Location</th>
            <th scope="col">Price</th>
            <th scope="col" class="text-end">Operation</th>
          </tr>
        </thead>

```



```

        <tbody id="tickets-table-body">
        </tbody>
    </table>
</div>

```

```

<!-- Add new ticket content -->

```

```

<div id="ticket-add-page" class="container"
    style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
    <div class="h-100 w-100">
        <!--
        TASK 1

```

Important:

To ensure proper page functionality you are encouraged to write code strictly within the bounds of this component.

Do not delete this comment!

```

-->

```

```

<h2 class="text-center mb-4">Add New Ticket</h2>

```

```

<form id="ticket-insert-form">
    <div class="form-group mb-3">
        <label for="artist-input" class="form-label">Artist</label>
        <input type="text" id="artist-input" name="artist" class="form-control"
placeholder="Enter artist name"
            required />
    </div>

```

```

    <div class="form-group mb-3">
        <label for="location-input" class="form-label">Location</label>
        <input type="text" id="location-input" name="location" class="form-control"
placeholder="Enter venue/location"
            required />
    </div>

```

```

    <div class="form-group mb-4">
        <label for="price-input" class="form-label">Price</label>
        <input type="number" id="price-input" name="price" class="form-control"
placeholder="Enter ticket price" min="0"
            required />
    </div>

```

```

<div class="d-flex justify-content-between">
    <button type="button" id="ticket-add-reset-btn" class="btn btn-outline-danger">
        Reset
    </button>
    <button type="button" id="ticket-add-submit-btn" class="btn btn-success">
        Submit
    </button>
</div>
</form>

```

```

    </div>
  </div>

</div>

<!-- Page Design -->
<div class="h-100 pe-5 d-flex justify-content-around" style="width: 30%;">
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
  </div>
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
  </div>
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
  </div>

```

```

<div class="ticket-blue">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
</div>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
<!-- Personal JS -->
<script src="../scripts/api_script.js">
  // Grab the DOM nodes
  const artistEl = document.getElementById('artist-input');
  const locationEl = document.getElementById('location-input');
  const priceEl = document.getElementById('price-input');
  const submitBtn = document.getElementById('ticket-add-submit-btn');
  const resetBtn = document.getElementById('ticket-add-reset-btn');
  const formEl = document.getElementById('ticket-insert-form');

  // Reset button just clears the form
  resetBtn.addEventListener('click', () => {
    formEl.reset();
  });

  // Submit button reads inputs, calls addTicket(), then clears + refreshes
  submitBtn.addEventListener('click', async () => {
    // 1) Read values
    const artist = artistEl.value.trim();
    const location = locationEl.value.trim();
    const price = parseInt(priceEl.value, 10);

    console.log(artist)

    // 2) Send to your API
    try {
      await addTicket(artist, location, price);
    } catch (err) {
      console.error('Failed to add ticket:', err);
      return alert('Could not add ticket. See console for details.');
```

```
// Session functions
let pageNumber = 0;

function resetVariables() {
  pageNumber = 0;
}

function saveSessionData() {
  const data = {
    pageNumber: pageNumber,
  }
  sessionStorage.setItem('sessionData', JSON.stringify(data));
}

// Nav buttons
const viewTicketsNavBtn = document.getElementById('view-tickets-nav-btn');
const addTicketNavBtn = document.getElementById('add-ticket-nav-btn');

// Content divs
const viewTicketsContent = document.getElementById('ticket-view-page');
const addTicketsContent = document.getElementById('ticket-add-page');

// Tickets table
const ticketsTableBody = document.getElementById('tickets-table-body');

// Add Ticket Inputs
const inputArtist = document.getElementById('artist-input');
const inputLocation = document.getElementById('location-input');
const inputPrice = document.getElementById('price-input');

// Add Ticket buttons
const submitTicket = document.getElementById('ticket-add-submit-btn');
const resetTicket = document.getElementById('ticket-add-reset-btn');

// Nav buttons functionality
viewTicketsNavBtn.addEventListener('click', function () {
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

addTicketNavBtn.addEventListener('click', function () {
  pageNumber = 1;
  saveSessionData();
  updatePage();
});

// Add Ticket buttons functionality
submitTicket.addEventListener('click', async function () {
  const artist = inputArtist.value;
```

```

const location = inputLocation.value;
const price = parseInt(inputPrice.value);

inputArtist.value = "";
inputLocation.value = "";
inputPrice.value = "";

await addTicket(artist, location, price);
pageNumber = 0;
saveSessionData();
updatePage();
});

resetTicket.addEventListener('click', function () {
  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";
})

// Delete Ticket buttons functionality
function addFunctionalityDeleteBtns() {
  document.querySelectorAll('.delete-ticket-btn').forEach(button => {
    button.addEventListener('click', async function () {
      // TASK 2
    });
  });
}

async function updateTicketsTable() {
  ticketsTableBody.innerHTML = "";

  const tickets = await getTickets();
  for (const ticket of tickets) {
    const entryHtml =
      <tr>
        <td>${ticket.artist}</td>
        <td>${ticket.location}</td>
        <td>${ticket.price}</td>
        <td class="text-end">
          <button id="${ticket.id}" type="button" class="btn btn-danger menu-btn delete-
ticket-btn">Delete</button>
        </td>
      </tr>
    ;
    ticketsTableBody.innerHTML += entryHtml;
  }
  addFunctionalityDeleteBtns();
}

async function updatePage() {
  let sessionData = sessionStorage.getItem('sessionData');
```

```

if (sessionData) {
  const storedData = JSON.parse(sessionStorage.getItem('sessionData'));
  pageNumber = storedData.pageNumber;
}

// View Tickets page
if (pageNumber == 0) {
  // Update navbar
  viewTicketsNavBtn.classList.add('active');
  addTicketNavBtn.classList.remove('active');

  // Update page
  viewTicketsContent.style.display = 'flex';
  addTicketsContent.style.display = 'none';

  // Update ticket table
  await updateTicketsTable();

  return;
}

// Add Tickets page
if (pageNumber == 1) {
  // Update navbar
  viewTicketsNavBtn.classList.remove('active');
  addTicketNavBtn.classList.add('active');

  // Update page
  viewTicketsContent.style.display = 'none';
  addTicketsContent.style.display = 'flex';
}
}

// Document loaded
window.onload = function () {
  updatePage();
}

```

</script>

</body>

</html>

why are the scripts not running?

Your inline handlers aren't running because of two main issues in how the `<script>` tags are structured:

1. You're putting inline code *inside* a `<script src="...">` tag

When you write:

```
html

<script src="../scripts/api_script.js">
  // inline code here...
</script>
```

the browser completely ignores any content between the opening and closing `<script>`—it only loads the external file. Inline code must go in a **separate** `<script>` tag with no `src` attribute.

2. There's a stray `</body>` above your real `<body>`

That prematurely closes the document, so anything after it is in limbo and won't execute.

How to fix

1. **Remove** the stray `</body>` just before your actual `<body>`.
2. **Split** your scripts into three ordered blocks at the bottom of the real body:

```
html

<!-- 1) Bootstrap bundle -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>

<!-- 2) API helpers -->
<script src="../scripts/api_script.js"></script>

<!-- 3) Your inline wiring -->
<script>
document.addEventListener('DOMContentLoaded', () => {
  // Grab elements
  const artistEl = document.getElementById('artist-input');
  const locationEl = document.getElementById('location-input');
  const priceEl = document.getElementById('price-input');
  const btnReset = document.getElementById('ticket-add-reset-btn');
  const btnSubmit = document.getElementById('ticket-add-submit-btn');

  // Reset
  btnReset.addEventListener('click', () => {
    document.getElementById('ticket-insert-form').reset();
  });

  // Submit
  btnSubmit.addEventListener('click', async () => {
    const artist = artistEl.value.trim();
    const location = locationEl.value.trim();
    const price = parseInt(priceEl.value, 10);
    if (!artist || !location || isNaN(price)) {
      return alert('Please fill in all fields correctly');
    }
    await addTicket(artist, location, price); // comes from api_script.js
    document.getElementById('ticket-insert-form').reset();
    // then switch back & refresh...
    pageNumber = 0; saveSessionData(); updatePage();
  });
});
</script>
```

By separating the external-script tag from your inline code and ensuring all your scripts live *after* the real `<body>` open (and before its closing `</body>`), the browser will load `api_script.js` (defining `addTicket()`), then run your event-listener wiring, and everything will fire correctly.



All JavaScript functions that ensure the functionality of the components you introduce, have been created. In order to ensure this functionality you must use the following IDs for your components:

- Artist input field ID: artist-input
- Location input field ID: location-input
- Price input field ID: price-input
- Reset button ID: ticket-add-reset-btn
- Submit button ID: ticket-add-submit-btn

add those

Here's the "Add New Ticket" form with exactly the IDs your existing JavaScript expects—just drop this inside your `<div class="h-100 w-100">` in your "Add Ticket" pane:

```
html

<!-- TASK 1 -->
<h2 class="text-center mb-4">Add New Ticket</h2>
<form id="ticket-insert-form">
  <div class="form-group mb-3">
    <label for="artist-input" class="form-label">Artist</label>
    <input
      type="text"
      id="artist-input"
      name="artist"
      class="form-control"
      placeholder="Enter artist name"
      required
    />
  </div>

  <div class="form-group mb-3">
    <label for="location-input" class="form-label">Location</label>
    <input
      type="text"
      id="location-input"
      name="location"
      class="form-control"
      placeholder="Enter venue/location"
      required
    />
  </div>

  <div class="form-group mb-4">
    <label for="price-input" class="form-label">Price</label>
    <input
      type="number"
      id="price-input"
      name="price"
      class="form-control"
      placeholder="Enter ticket price"
      min="0"
      required
    />
  </div>

  <div class="d-flex justify-content-between">
    <button
      type="button"
    />
  </div>
```



```

        id="ticket-add-reset-btn"
        class="btn btn-outline-danger"
    >
        Reset
    </button>
    <button
        type="button"
        id="ticket-add-submit-btn"
        class="btn btn-success"
    >
        Submit
    </button>
</div>
</form>

```

- **Artist input field:** `id="artist-input"`
- **Location input field:** `id="location-input"`
- **Price input field:** `id="price-input"`
- **Reset button:** `id="ticket-add-reset-btn"`
- **Submit button:** `id="ticket-add-submit-btn"`

With these IDs in place, your pre-written JS handlers will automatically wire up to read the inputs, clear the form, and call `addTicket()` when you click **Submit**.



```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>E-Ticket</title>
    <link rel="icon" href="../assets/images/favicon.png" type="image/png">

    <!-- Personal CSS -->
    <link rel="stylesheet" href="../style/style.css">
    <link rel="stylesheet" href="../style/ticket.css">

    <!-- Bootstrap 5 -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.css">

</head>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

</body>

<body class="d-flex justify-content-between align-items-center" style="background-color:

```

```
#00466a;">
```

```
<div class="container container-wrapper d-flex flex-column justify-content-center align-items-center h-50 w-50 p-4">
```

```
  <div class="container mb-4">
    <h1 class="m-0">Ticket Selling Platform</h1>
  </div>
```

```
<!-- Navbar -->
```

```
<div class="container d-flex bg-dark mb-4 py-3" style="border-radius: 48px;">
  <button id="view-tickets-nav-btn"
    class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
    View Tickets
  </button>
  <button id="add-ticket-nav-btn"
    class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
    Add Ticket
  </button>
</div>
```

```
<!-- View tickets content -->
```

```
<div id="ticket-view-page" class="container justify-content-center align-items-start"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">

  <table id="tickets-table" class="table table-hover align-middle" aria-hidden="true">
    <thead>
      <tr>
        <th scope="col">Artist</th>
        <th scope="col">Location</th>
        <th scope="col">Price</th>
        <th scope="col" class="text-end">Operation</th>
      </tr>
    </thead>
    <tbody id="tickets-table-body">
    </tbody>
  </table>
</div>
```

```
<!-- Add new ticket content -->
```

```
<div id="ticket-add-page" class="container"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
  <div class="h-100 w-100">
    <!--
    TASK 1
```

Important:

To ensure proper page functionality you are encouraged to write code strictly within the bounds of this component.

Do not delete this comment!

```
-->
<h2 class="text-center mb-4">Add New Ticket</h2>

<form id="ticket-insert-form">
  <div class="form-group mb-3">
    <label for="artist-input" class="form-label">Artist</label>
    <input type="text" id="artist-input" name="artist" class="form-control"
placeholder="Enter artist name"
      required />
    </div>

    <div class="form-group mb-3">
      <label for="location-input" class="form-label">Location</label>
      <input type="text" id="location-input" name="location" class="form-control"
placeholder="Enter venue/location"
        required />
      </div>

      <div class="form-group mb-4">
        <label for="price-input" class="form-label">Price</label>
        <input type="number" id="price-input" name="price" class="form-control"
placeholder="Enter ticket price" min="0"
          required />
        </div>

        <div class="d-flex justify-content-between">
          <button type="button" id="ticket-add-reset-btn" class="btn btn-outline-danger">
            Reset
          </button>
          <button type="button" id="ticket-add-submit-btn" class="btn btn-success">
            Submit
          </button>
        </div>
      </form>
    </div>
  </div>

</div>

<!-- Page Design -->
<div class="h-100 pe-5 d-flex justify-content-around" style="width: 30%;">
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
  </div>
</div>
```

```

<div class="ticket-orange">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-yellow">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
</div>
<div class="ticket-row h-100 d-flex flex-column justify-content-center">
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-orange">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
</div>
<div class="ticket-row h-100 d-flex flex-column justify-content-center">
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-orange">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
</div>

```

</div>

<!-- Bootstrap 5 JS Bundle with Popper -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">

</script>

<!-- Personal JS -->

<script src="../scripts/api_script.js"></script>

<script>

// Session functions

let pageNumber = 0;

```
function resetVariables() {
  pageNumber = 0;
}

function saveSessionData() {
  const data = {
    pageNumber: pageNumber,
  }
  sessionStorage.setItem('sessionData', JSON.stringify(data));
}

// Nav buttons
const viewTicketsNavBtn = document.getElementById('view-tickets-nav-btn');
const addTicketNavBtn = document.getElementById('add-ticket-nav-btn');

// Content divs
const viewTicketsContent = document.getElementById('ticket-view-page');
const addTicketsContent = document.getElementById('ticket-add-page');

// Tickets table
const ticketsTableBody = document.getElementById('tickets-table-body');

// Add Ticket Inputs
const inputArtist = document.getElementById('artist-input');
const inputLocation = document.getElementById('location-input');
const inputPrice = document.getElementById('price-input');

// Add Ticket buttons
const submitTicket = document.getElementById('ticket-add-submit-btn');
const resetTicket = document.getElementById('ticket-add-reset-btn');

// Nav buttons functionality
viewTicketsNavBtn.addEventListener('click', function () {
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

addTicketNavBtn.addEventListener('click', function () {
  pageNumber = 1;
  saveSessionData();
  updatePage();
});

// Add Ticket buttons functionality
submitTicket.addEventListener('click', async function () {
  const artist = inputArtist.value;
  const location = inputLocation.value;
  const price = parseInt(inputPrice.value);
```

```

inputArtist.value = "";
inputLocation.value = "";
inputPrice.value = "";

await addTicket(artist, location, price);
pageNumber = 0;
saveSessionData();
updatePage();
});

resetTicket.addEventListener('click', function () {
  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";
})

// Delete Ticket buttons functionality
function addFunctionalityDeleteBtns() {
  document.querySelectorAll('.delete-ticket-btn').forEach(button => {
    button.addEventListener('click', async function () {
      // TASK 2
    });
  });
}

async function updateTicketsTable() {
  ticketsTableBody.innerHTML = "";

  const tickets = await getTickets();
  for (const ticket of tickets) {
    const entryHtml =
      <tr>
        <td>${ticket.artist}</td>
        <td>${ticket.location}</td>
        <td>${ticket.price}</td>
        <td class="text-end">
          <button id="${ticket.id}" type="button" class="btn btn-danger menu-btn delete-
ticket-btn">Delete</button>
        </td>
      </tr>
    ;
    ticketsTableBody.innerHTML += entryHtml;
  }
  addFunctionalityDeleteBtns();
}

async function updatePage() {
  let sessionData = sessionStorage.getItem('sessionData');
  if (sessionData) {
    const storedData = JSON.parse(sessionStorage.getItem('sessionData'));
    pageNumber = storedData.pageNumber;
  }
}

```

```

    }

    // View Tickets page
    if (pageNumber == 0) {
        // Update navbar
        viewTicketsNavBtn.classList.add('active');
        addTicketNavBtn.classList.remove('active');

        // Update page
        viewTicketsContent.style.display = 'flex';
        addTicketsContent.style.display = 'none';

        // Update ticket table
        await updateTicketsTable();

        return;
    }

    // Add Tickets page
    if (pageNumber == 1) {
        // Update navbar
        viewTicketsNavBtn.classList.remove('active');
        addTicketNavBtn.classList.add('active');

        // Update page
        viewTicketsContent.style.display = 'none';
        addTicketsContent.style.display = 'flex';
    }
}

// Document loaded
window.onload = function () {
    updatePage();
}

```

</script>

</body>

</html>

nothign gets submitted, why?

Your form looks perfect—but none of your JavaScript is ever firing because of a few structural hiccups in the page. Let's fix them step by step:

1. Remove the stray </body>

You have this near the top:

```
html

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="...bootstrap.bundle.min.js"></script>
</body>

<body ...>
```

That `</body>` closes your document before most of your markup ever loads. Delete that line so you only ever open `<body>` once and close it at the very end.

2. Load your scripts in the correct order (after the real `<body>`)

At the bottom of your page, **after** all of your HTML and **before** the final `</body>`, you want exactly:

```
html

<!-- 1) Bootstrap bundle -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>

<!-- 2) Your API helpers (defines addTicket, getTickets, deleteTicket...) -->
<script src="../scripts/api_script.js"></script>

<!-- 3) Your inline wiring -->
<script>
  document.addEventListener('DOMContentLoaded', () => {
    // NAVIGATION
    const viewBtn = document.getElementById('view-tickets-nav-btn');
    const addBtn = document.getElementById('add-ticket-nav-btn');
    viewBtn.addEventListener('click', () => switchPage(0));
    addBtn.addEventListener('click', () => switchPage(1));

    // FORM
    const formEl = document.getElementById('ticket-insert-form');
    const artistEl = document.getElementById('artist-input');
    const locationEl = document.getElementById('location-input');
    const priceEl = document.getElementById('price-input');
    const btnReset = document.getElementById('ticket-add-reset-btn');
    const btnSubmit = document.getElementById('ticket-add-submit-btn');

    btnReset.addEventListener('click', () => formEl.reset());

    btnSubmit.addEventListener('click', async () => {
      const artist = artistEl.value.trim();
      const location = locationEl.value.trim();
      const price = parseInt(priceEl.value, 10);

      if (!artist || !location || isNaN(price)) {
        return alert('All fields are required');
      }
      // ← This comes from api_script.js
      await addTicket(artist, location, price);

      formEl.reset();
      switchPage(0); // go back to view
      await updatePage(); // refresh list
    });

    // INITIALIZE
    switchPage(0);
  });

  // Helper to toggle pages
```



```

async function switchPage(pageNum) {
  sessionStorage.setItem('pageNumber', pageNum);
  document.getElementById('ticket-view-page').style.display = pageNum === 0 ? 'flex' :
'none';
  document.getElementById('ticket-add-page').style.display = pageNum === 1 ? 'flex' :
'none';
  document.getElementById('view-tickets-nav-btn').classList.toggle('active', pageNum === 0);
  document.getElementById('add-ticket-nav-btn').classList.toggle('active', pageNum === 1);
}

// Load and render tickets
async function updatePage() {
  const tickets = await getTickets(); // also from api_script.js
  const tbody = document.getElementById('tickets-table-body');
  tbody.innerHTML = '';
  tickets.forEach(t => {
    const tr = document.createElement('tr');
    tr.innerHTML = `
      <td>${t.artist}</td>
      <td>${t.location}</td>
      <td>${t.price}</td>
      <td class="text-end">
        <button data-id="${t.id}" class="btn btn-danger delete-ticket-btn">Delete</button>
      </td>`;
    tbody.appendChild(tr);
  });
  document.querySelectorAll('.delete-ticket-btn').forEach(btn =>
    btn.addEventListener('click', async () => {
      await deleteTicket(btn.dataset.id);
      updatePage();
    })
  );
}
</script>
</body>
</html>

```

Why this fixes it:

1. **No stray </body>** means your actual HTML and scripts live in the document.
2. **Three script tags** in order ensure:
 - Bootstrap loads first (for any components you use),
 - `api_script.js` loads next (defining `addTicket()`, `getTickets()`, etc.),
 - Your inline code runs last, inside a `DOMContentLoaded` handler so the DOM is fully parsed.
3. **All your IDs** (`artist-input`, `location-input`, `price-input`, `ticket-add-reset-btn`, `ticket-add-submit-btn`) are hooked up with event listeners that call the functions in `api_script.js`.

Give that structure a try. Open DevTools → **Console** and **Network** to verify:

- No 404 on `api_script.js`.
- No “`addTicket` is not defined” errors.
- Clicking **Submit** actually calls your endpoints and refreshes the ticket list.

That should get your form submitting!



what line is the stray body?

The stray `</body>` is the one immediately following your Bootstrap `<script>` tag—right before your real `<body>` tag. In your snippet it appears here:

```
html

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>

</body> ← REMOVE THIS

<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">
```

Just delete that lone `</body>` so you only have one opening `<body>` and one closing `</body>` at the very end. That will let all of your markup and scripts actually live inside the page.



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>E-Ticket</title>
  <link rel="icon" href="../assets/images/favicon.png" type="image/png">

  <!-- Personal CSS -->
  <link rel="stylesheet" href="../style/style.css">
  <link rel="stylesheet" href="../style/ticket.css">

  <!-- Bootstrap 5 -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.css">

</head>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">

  <div class="container container-wrapper d-flex flex-column justify-content-center align-items-
center h-50 w-50 p-4">

    <div class="container mb-4">
      <h1 class="m-0">Ticket Selling Platform</h1>
    </div>
```

```

<!-- Navbar -->
<div class="container d-flex bg-dark mb-4 py-3" style="border-radius: 48px;">
  <button id="view-tickets-nav-btn"
    class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
    View Tickets
  </button>
  <button id="add-ticket-nav-btn"
    class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
    Add Ticket
  </button>
</div>

```

```

<!-- View tickets content -->
<div id="ticket-view-page" class="container justify-content-center align-items-start"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">

  <table id="tickets-table" class="table table-hover align-middle" aria-hidden="true">
    <thead>
      <tr>
        <th scope="col">Artist</th>
        <th scope="col">Location</th>
        <th scope="col">Price</th>
        <th scope="col" class="text-end">Operation</th>
      </tr>
    </thead>
    <tbody id="tickets-table-body">
    </tbody>
  </table>
</div>

```

```

<!-- Add new ticket content -->
<div id="ticket-add-page" class="container"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
  <div class="h-100 w-100">
    <!--
    TASK 1

```

Important:

To ensure proper page functionality you are encouraged to write code strictly within the bounds of this component.

Do not delete this comment!

-->

```
<h2 class="text-center mb-4">Add New Ticket</h2>
```

```

<form id="ticket-insert-form">
  <div class="form-group mb-3">
    <label for="artist-input" class="form-label">Artist</label>
    <input type="text" id="artist-input" name="artist" class="form-control"
placeholder="Enter artist name"

```

```

        required />
    </div>

    <div class="form-group mb-3">
        <label for="location-input" class="form-label">Location</label>
        <input type="text" id="location-input" name="location" class="form-control"
placeholder="Enter venue/location"
        required />
    </div>

    <div class="form-group mb-4">
        <label for="price-input" class="form-label">Price</label>
        <input type="number" id="price-input" name="price" class="form-control"
placeholder="Enter ticket price" min="0"
        required />
    </div>

    <div class="d-flex justify-content-between">
        <button type="button" id="ticket-add-reset-btn" class="btn btn-outline-danger">
            Reset
        </button>
        <button type="button" id="ticket-add-submit-btn" class="btn btn-success">
            Submit
        </button>
    </div>
</form>
</div>
</div>

</div>

<!-- Page Design -->
<div class="h-100 pe-5 d-flex justify-content-around" style="width: 30%;">
    <div class="ticket-row h-100 d-flex flex-column justify-content-center">
        <div class="ticket-orange">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-yellow">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-blue">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-orange">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-yellow">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
    </div>
</div>
<div class="ticket-row h-100 d-flex flex-column justify-content-center">

```

```

<div class="ticket-blue">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-orange">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-yellow">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-blue">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
</div>
<div class="ticket-row h-100 d-flex flex-column justify-content-center">
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-orange">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
</div>

</div>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
<!-- Personal JS -->
<script src="../scripts/api_script.js"></script>

<script>

  // Session functions
  let pageNumber = 0;

  function resetVariables() {
    pageNumber = 0;
  }

  function saveSessionData() {
    const data = {
      pageNumber: pageNumber,

```

```

    }
    sessionStorage.setItem('sessionData', JSON.stringify(data));
  }

  // Nav buttons
  const viewTicketsNavBtn = document.getElementById('view-tickets-nav-btn');
  const addTicketNavBtn = document.getElementById('add-ticket-nav-btn');

  // Content divs
  const viewTicketsContent = document.getElementById('ticket-view-page');
  const addTicketsContent = document.getElementById('ticket-add-page');

  // Tickets table
  const ticketsTableBody = document.getElementById('tickets-table-body');

  // Add Ticket Inputs
  const inputArtist = document.getElementById('artist-input');
  const inputLocation = document.getElementById('location-input');
  const inputPrice = document.getElementById('price-input');

  // Add Ticket buttons
  const submitTicket = document.getElementById('ticket-add-submit-btn');
  const resetTicket = document.getElementById('ticket-add-reset-btn');

  // Nav buttons functionality
  viewTicketsNavBtn.addEventListener('click', function () {
    pageNumber = 0;
    saveSessionData();
    updatePage();
  });

  addTicketNavBtn.addEventListener('click', function () {
    pageNumber = 1;
    saveSessionData();
    updatePage();
  });

  // Add Ticket buttons functionality
  submitTicket.addEventListener('click', async function () {
    const artist = inputArtist.value;
    const location = inputLocation.value;
    const price = parseInt(inputPrice.value);

    inputArtist.value = "";
    inputLocation.value = "";
    inputPrice.value = "";

    await addTicket(artist, location, price);
    pageNumber = 0;
    saveSessionData();
    updatePage();
  });

```

```

});

resetTicket.addEventListener('click', function () {
  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";
})

// Delete Ticket buttons functionality
function addFunctionalityDeleteBtns() {
  document.querySelectorAll('.delete-ticket-btn').forEach(button => {
    button.addEventListener('click', async function () {
      // TASK 2
    });
  });
}

async function updateTicketsTable() {
  ticketsTableBody.innerHTML = "";

  const tickets = await getTickets();
  for (const ticket of tickets) {
    const entryHtml =
      <tr>
        <td>${ticket.artist}</td>
        <td>${ticket.location}</td>
        <td>${ticket.price}</td>
        <td class="text-end">
          <button id="${ticket.id}" type="button" class="btn btn-danger menu-btn delete-
ticket-btn">Delete</button>
        </td>
      </tr>
    ;
    ticketsTableBody.innerHTML += entryHtml;
  }
  addFunctionalityDeleteBtns();
}

async function updatePage() {
  let sessionData = sessionStorage.getItem('sessionData');
  if (sessionData) {
    const storedData = JSON.parse(sessionStorage.getItem('sessionData'));
    pageNumber = storedData.pageNumber;
  }

  // View Tickets page
  if (pageNumber == 0) {
    // Update navbar
    viewTicketsNavBtn.classList.add('active');
    addTicketNavBtn.classList.remove('active');
  }
}

```

```
// Update page
viewTicketsContent.style.display = 'flex';
addTicketsContent.style.display = 'none';

// Update ticket table
await updateTicketsTable();

return;
}

// Add Tickets page
if (pageNumber == 1) {
  // Update navbar
  viewTicketsNavBtn.classList.remove('active');
  addTicketNavBtn.classList.add('active');

  // Update page
  viewTicketsContent.style.display = 'none';
  addTicketsContent.style.display = 'flex';
}
}
```

```
// Document loaded
window.onload = function () {
  updatePage();
}
```

</script>

</body>

</html>

is it ok now?

You've correctly removed that stray </body>, so structurally your page is valid now. A quick sanity-check:

1. **One <body>** — opened after <head> and closed at the very end.
2. **No orphan </body>** halfway through.
3. **Three script tags** at the bottom (Bootstrap, then `api_script.js`, then your inline wiring).

A few final tweaks to make sure your JS actually runs:

1. **Wrap your inline code in DOMContentLoaded** so it waits for the DOM:

```
html

<script>
document.addEventListener('DOMContentLoaded', () => {
```



```
// All your document.getElementById and addEventListener calls go here
// e.g.
const submitTicket = document.getElementById('ticket-add-submit-btn');
submitTicket.addEventListener('click', async () => {
  /* ... your addTicket() logic ... */
});
// etc...
});
</script>
```

2. **Move your Bootstrap <script> into the footer** (below your HTML) rather than right before <body>. That way all your markup is parsed before any scripts run.

3. Check DevTools

- Open **Network** → verify `api_script.js` is 200 OK.
- Open **Console** → look for errors like `addTicket is not defined` or `Cannot read property 'addEventListener' of null`.

If you make those two small changes—moving the bootstrap bundle down and wrapping your inline wiring in `DOMContentLoaded`—your form buttons (`ticket-add-reset-btn` and `ticket-add-submit-btn`) will reliably hook up to your prewritten JS and start submitting tickets. Let me know if you still see any console errors after that!



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>E-Ticket</title>
  <link rel="icon" href="../assets/images/favicon.png" type="image/png">

  <!-- Personal CSS -->
  <link rel="stylesheet" href="../style/style.css">
  <link rel="stylesheet" href="../style/ticket.css">

  <!-- Bootstrap 5 -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.css">

</head>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

</body>

<body class="d-flex justify-content-between align-items-center" style="background-color:
```

```
#00466a;">
```

```
<div class="container container-wrapper d-flex flex-column justify-content-center align-items-center h-50 w-50 p-4">
```

```
  <div class="container mb-4">
    <h1 class="m-0">Ticket Selling Platform</h1>
  </div>
```

```
<!-- Navbar -->
```

```
<div class="container d-flex bg-dark mb-4 py-3" style="border-radius: 48px;">
  <button id="view-tickets-nav-btn"
    class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
    View Tickets
  </button>
  <button id="add-ticket-nav-btn"
    class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
    Add Ticket
  </button>
</div>
```

```
<!-- View tickets content -->
```

```
<div id="ticket-view-page" class="container justify-content-center align-items-start"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">

  <table id="tickets-table" class="table table-hover align-middle" aria-hidden="true">
    <thead>
      <tr>
        <th scope="col">Artist</th>
        <th scope="col">Location</th>
        <th scope="col">Price</th>
        <th scope="col" class="text-end">Operation</th>
      </tr>
    </thead>
    <tbody id="tickets-table-body">
    </tbody>
  </table>
</div>
```

```
<!-- Add new ticket content -->
```

```
<div id="ticket-add-page" class="container"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">
  <div class="h-100 w-100">
    <!--
    TASK 1
```

Important:

To ensure proper page functionality you are encouraged to write code strictly within the bounds of this component.

Do not delete this comment!

```
-->
<h2 class="text-center mb-4">Add New Ticket</h2>

<form id="ticket-insert-form">
  <div class="form-group mb-3">
    <label for="artist-input" class="form-label">Artist</label>
    <input type="text" id="artist-input" name="artist" class="form-control"
placeholder="Enter artist name"
      required />
    </div>

    <div class="form-group mb-3">
      <label for="location-input" class="form-label">Location</label>
      <input type="text" id="location-input" name="location" class="form-control"
placeholder="Enter venue/location"
        required />
      </div>

      <div class="form-group mb-4">
        <label for="price-input" class="form-label">Price</label>
        <input type="number" id="price-input" name="price" class="form-control"
placeholder="Enter ticket price" min="0"
          required />
        </div>

        <div class="d-flex justify-content-between">
          <button type="button" id="ticket-add-reset-btn" class="btn btn-outline-danger">
            Reset
          </button>
          <button type="button" id="ticket-add-submit-btn" class="btn btn-success">
            Submit
          </button>
        </div>
      </form>

</div>
</div>

</div>

<!-- Page Design -->
<div class="h-100 pe-5 d-flex justify-content-around" style="width: 30%;">
  <div class="ticket-row h-100 d-flex flex-column justify-content-center">
    <div class="ticket-orange">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-yellow">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
    <div class="ticket-blue">
      <h3 style="transform: rotate(270deg);">Ticket</h3>
    </div>
  </div>
</div>
```

```

</div>
<div class="ticket-orange">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-yellow">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
</div>
<div class="ticket-row h-100 d-flex flex-column justify-content-center">
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-orange">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
</div>
<div class="ticket-row h-100 d-flex flex-column justify-content-center">
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-orange">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
</div>
</div>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>
<!-- Personal JS -->
<script src="../scripts/api_script.js"></script>

<script>

// Session functions

```

```

let pageNumber = 0;

function resetVariables() {
  pageNumber = 0;
}

function saveSessionData() {
  const data = {
    pageNumber: pageNumber,
  }
  sessionStorage.setItem('sessionData', JSON.stringify(data));
}

// Nav buttons
const viewTicketsNavBtn = document.getElementById('view-tickets-nav-btn');
const addTicketNavBtn = document.getElementById('add-ticket-nav-btn');

// Content divs
const viewTicketsContent = document.getElementById('ticket-view-page');
const addTicketsContent = document.getElementById('ticket-add-page');

// Tickets table
const ticketsTableBody = document.getElementById('tickets-table-body');

// Add Ticket Inputs
const inputArtist = document.getElementById('artist-input');
const inputLocation = document.getElementById('location-input');
const inputPrice = document.getElementById('price-input');

// Add Ticket buttons
const submitTicket = document.getElementById('ticket-add-submit-btn');
const resetTicket = document.getElementById('ticket-add-reset-btn');

// Nav buttons functionality
viewTicketsNavBtn.addEventListener('click', function () {
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

addTicketNavBtn.addEventListener('click', function () {
  pageNumber = 1;
  saveSessionData();
  updatePage();
});

// Add Ticket buttons functionality
submitTicket.addEventListener('click', async function () {
  const artist = inputArtist.value;
  const location = inputLocation.value;
  const price = parseInt(inputPrice.value);

```

```

inputArtist.value = "";
inputLocation.value = "";
inputPrice.value = "";

await addTicket(artist, location, price);
pageNumber = 0;
saveSessionData();
updatePage();
});

resetTicket.addEventListener('click', function () {
  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";
})

// Delete Ticket buttons functionality
function addFunctionalityDeleteBtns() {
  document.querySelectorAll('.delete-ticket-btn').forEach(button => {
    button.addEventListener('click', async function () {
      // TASK 2
    });
  });
}

async function updateTicketsTable() {
  ticketsTableBody.innerHTML = "";

  const tickets = await getTickets();
  for (const ticket of tickets) {
    const entryHtml =
      <tr>
        <td>${ticket.artist}</td>
        <td>${ticket.location}</td>
        <td>${ticket.price}</td>
        <td class="text-end">
          <button id="${ticket.id}" type="button" class="btn btn-danger menu-btn delete-
ticket-btn">Delete</button>
        </td>
      </tr>
    ;
    ticketsTableBody.innerHTML += entryHtml;
  }
  addFunctionalityDeleteBtns();
}

async function updatePage() {
  let sessionData = sessionStorage.getItem('sessionData');
  if (sessionData) {
    const storedData = JSON.parse(sessionStorage.getItem('sessionData'));

```

```

    pageNumber = storedData.pageNumber;
  }

  // View Tickets page
  if (pageNumber == 0) {
    // Update navbar
    viewTicketsNavBtn.classList.add('active');
    addTicketNavBtn.classList.remove('active');

    // Update page
    viewTicketsContent.style.display = 'flex';
    addTicketsContent.style.display = 'none';

    // Update ticket table
    await updateTicketsTable();

    return;
  }

```

```

  // Add Tickets page
  if (pageNumber == 1) {
    // Update navbar
    viewTicketsNavBtn.classList.remove('active');
    addTicketNavBtn.classList.add('active');

    // Update page
    viewTicketsContent.style.display = 'none';
    addTicketsContent.style.display = 'flex';
  }
}

```

```

// Document loaded
window.onload = function () {
  updatePage();
}

```

</script>

</body>

</html>

why is this working for reset, but not for submit?

Your **Reset** button works because it only calls native DOM methods on elements that already exist and doesn't rely on any external code. The **Submit** flow, however, calls your `addTicket()` helper from `api_script.js`, and that isn't actually getting loaded or bound the way you expect—so when you click Submit nothing happens (and you probably see a “`addTicket is not defined`” or silent promise rejection in your console).

The root causes are:

1. Stray `</body>` before your real `<body>`

You still have this line right before your `<body>` tag:

```
html

<script src="../../bootstrap.bundle.min.js"></script>
</body> ← (this kills the document before your markup)
<body ...>
```

Anything between that early `</body>` and the final `</body>` lives in limbo. Your `<script src="../../scripts/api_script.js">` and your inline wiring end up outside the real document, so `addTicket()` never makes it into the page's JavaScript context.

2. Double-loading Bootstrap

You're including the Bootstrap bundle **twice**, once before the stray `</body>` and once after your page. That's not directly breaking submit, but it's a symptom of the same ordering issue.

How to fix it

1. **Remove that stray `</body>`** so your `<body>` only opens once and closes once, at the very end.

2. **Load scripts after your real `<body>`** (and before its closing tag) in this order:

```
html

<!-- YOUR PAGE CONTENT HERE -->

<!-- 1) Bootstrap bundle -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
</script>

<!-- 2) Your API helper functions (defines addTicket/getTickets/...) -->
<script src="../../scripts/api_script.js"></script>

<!-- 3) Your inline wiring -->
<script>
  document.addEventListener('DOMContentLoaded', () => {
    const artistEl = document.getElementById('artist-input');
    const locationEl = document.getElementById('location-input');
    const priceEl = document.getElementById('price-input');
    const btnReset = document.getElementById('ticket-add-reset-btn');
    const btnSubmit = document.getElementById('ticket-add-submit-btn');

    btnReset.addEventListener('click', () => {
      document.getElementById('ticket-insert-form').reset();
    });

    btnSubmit.addEventListener('click', async () => {
      const artist = artistEl.value.trim();
      const location = locationEl.value.trim();
      const price = parseInt(priceEl.value, 10);
      if (!artist || !location || isNaN(price)) {
        return alert('Please fill in all fields correctly');
      }

      // ← now addTicket is defined
      await addTicket(artist, location, price);

      document.getElementById('ticket-insert-form').reset();
      // then switch back to the list view...
    });
  });
</script>
```



```
</script>
</body>
</html>
```

Once you remove the stray `</body>` and consolidate your script tags at the bottom in that order, `addTicket()` will actually be present on the page when the Submit handler runs, and clicking **Submit** will fire your API call as expected. Don't forget to open DevTools → **Console** to confirm there are no `"addTicket is not defined"` errors after you make these changes.



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>E-Ticket</title>
  <link rel="icon" href="../assets/images/favicon.png" type="image/png">

  <!-- Personal CSS -->
  <link rel="stylesheet" href="../style/style.css">
  <link rel="stylesheet" href="../style/ticket.css">

  <!-- Bootstrap 5 -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.css">

</head>

<!-- Bootstrap 5 JS Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

<body class="d-flex justify-content-between align-items-center" style="background-color:
#00466a;">

  <div class="container container-wrapper d-flex flex-column justify-content-center align-items-
center h-50 w-50 p-4">

    <div class="container mb-4">
      <h1 class="m-0">Ticket Selling Platform</h1>
    </div>

    <!-- Navbar -->
    <div class="container d-flex bg-dark mb-4 py-3" style="border-radius: 48px;">
      <button id="view-tickets-nav-btn"
class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
        View Tickets
```

```

</button>
<button id="add-ticket-nav-btn"
  class="btn btn-dark menu-btn mx-2 d-flex align-items-center justify-content-center">
  Add Ticket
</button>
</div>

```

<!-- View tickets content -->

```

<div id="ticket-view-page" class="container justify-content-center align-items-start"
  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">

```

```

<table id="tickets-table" class="table table-hover align-middle" aria-hidden="true">

```

```

  <thead>

```

```

    <tr>

```

```

      <th scope="col">Artist</th>

```

```

      <th scope="col">Location</th>

```

```

      <th scope="col">Price</th>

```

```

      <th scope="col" class="text-end">Operation</th>

```

```

    </tr>

```

```

  </thead>

```

```

  <tbody id="tickets-table-body">

```

```

  </tbody>

```

```

</table>

```

```

</div>

```

<!-- Add new ticket content -->

```

<div id="ticket-add-page" class="container"

```

```

  style="display: none; height: 65%; overflow-y: auto; scrollbar-width: none;">

```

```

  <div class="h-100 w-100">

```

```

    <!--

```

```

    TASK 1

```

Important:

To ensure proper page functionality you are encouraged to write code strictly within the bounds of this component.

Do not delete this comment!

```

-->

```

```

<h2 class="text-center mb-4">Add New Ticket</h2>

```

```

<form id="ticket-insert-form">

```

```

  <div class="form-group mb-3">

```

```

    <label for="artist-input" class="form-label">Artist</label>

```

```

    <input type="text" id="artist-input" name="artist" class="form-control"

```

```

    placeholder="Enter artist name"

```

```

    required />

```

```

  </div>

```

```

  <div class="form-group mb-3">

```

```

    <label for="location-input" class="form-label">Location</label>

```

```

    <input type="text" id="location-input" name="location" class="form-control"

```

```

placeholder="Enter venue/location"
    required />
</div>

<div class="form-group mb-4">
    <label for="price-input" class="form-label">Price</label>
    <input type="number" id="price-input" name="price" class="form-control"
placeholder="Enter ticket price" min="0"
    required />
</div>

<div class="d-flex justify-content-between">
    <button type="button" id="ticket-add-reset-btn" class="btn btn-outline-danger">
        Reset
    </button>
    <button type="button" id="ticket-add-submit-btn" class="btn btn-success">
        Submit
    </button>
</div>
</form>

</div>
</div>

</div>

<!-- Page Design -->
<div class="h-100 pe-5 d-flex justify-content-around" style="width: 30%;">
    <div class="ticket-row h-100 d-flex flex-column justify-content-center">
        <div class="ticket-orange">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-yellow">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-blue">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-orange">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-yellow">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
    </div>
    <div class="ticket-row h-100 d-flex flex-column justify-content-center">
        <div class="ticket-blue">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
        <div class="ticket-orange">
            <h3 style="transform: rotate(270deg);">Ticket</h3>
        </div>
    </div>
</div>

```

```

</div>
<div class="ticket-yellow">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
<div class="ticket-blue">
  <h3 style="transform: rotate(270deg);">Ticket</h3>
</div>
</div>
<div class="ticket-row h-100 d-flex flex-column justify-content-center">
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-orange">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-yellow">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
  <div class="ticket-blue">
    <h3 style="transform: rotate(270deg);">Ticket</h3>
  </div>
</div>

```

```
</div>
```

```
<!-- Bootstrap 5 JS Bundle with Popper -->
```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js">
```

```
</script>
```

```
<!-- Personal JS -->
```

```
<script src="../scripts/api_script.js"></script>
```

```
<script>
```

```
// Session functions
```

```
let pageNumber = 0;
```

```
function resetVariables() {
```

```
  pageNumber = 0;
```

```
}
```

```
function saveSessionData() {
```

```
  const data = {
```

```
    pageNumber: pageNumber,
```

```
  }
```

```
  sessionStorage.setItem('sessionData', JSON.stringify(data));
```

```
}
```

```
// Nav buttons
```

```

const viewTicketsNavBtn = document.getElementById('view-tickets-nav-btn');
const addTicketNavBtn = document.getElementById('add-ticket-nav-btn');

// Content divs
const viewTicketsContent = document.getElementById('ticket-view-page');
const addTicketsContent = document.getElementById('ticket-add-page');

// Tickets table
const ticketsTableBody = document.getElementById('tickets-table-body');

// Add Ticket Inputs
const inputArtist = document.getElementById('artist-input');
const inputLocation = document.getElementById('location-input');
const inputPrice = document.getElementById('price-input');

// Add Ticket buttons
const submitTicket = document.getElementById('ticket-add-submit-btn');
const resetTicket = document.getElementById('ticket-add-reset-btn');

// Nav buttons functionality
viewTicketsNavBtn.addEventListener('click', function () {
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

addTicketNavBtn.addEventListener('click', function () {
  pageNumber = 1;
  saveSessionData();
  updatePage();
});

// Add Ticket buttons functionality
submitTicket.addEventListener('click', async function () {
  const artist = inputArtist.value;
  const location = inputLocation.value;
  const price = parseInt(inputPrice.value);

  inputArtist.value = "";
  inputLocation.value = "";
  inputPrice.value = "";

  await addTicket(artist, location, price);
  pageNumber = 0;
  saveSessionData();
  updatePage();
});

resetTicket.addEventListener('click', function () {
  inputArtist.value = "";
  inputLocation.value = "";

```

```

    inputPrice.value = "";
  })

  // Delete Ticket buttons functionality
  function addFunctionalityDeleteBtns() {
    document.querySelectorAll('.delete-ticket-btn').forEach(button => {
      button.addEventListener('click', async function () {
        // TASK 2
      });
    });
  }

  async function updateTicketsTable() {
    ticketsTableBody.innerHTML = "";

    const tickets = await getTickets();
    for (const ticket of tickets) {
      const entryHtml =
        <tr>
          <td>${ticket.artist}</td>
          <td>${ticket.location}</td>
          <td>${ticket.price}</td>
          <td class="text-end">
            <button id="${ticket.id}" type="button" class="btn btn-danger menu-btn delete-
ticket-btn">Delete</button>
          </td>
        </tr>
      ;
      ticketsTableBody.innerHTML += entryHtml;
    }
    addFunctionalityDeleteBtns();
  }

  async function updatePage() {
    let sessionData = sessionStorage.getItem('sessionData');
    if (sessionData) {
      const storedData = JSON.parse(sessionStorage.getItem('sessionData'));
      pageNumber = storedData.pageNumber;
    }

    // View Tickets page
    if (pageNumber == 0) {
      // Update navbar
      viewTicketsNavBtn.classList.add('active');
      addTicketNavBtn.classList.remove('active');

      // Update page
      viewTicketsContent.style.display = 'flex';
      addTicketsContent.style.display = 'none';

      // Update ticket table

```

```

        await updateTicketsTable();

        return;
    }

    // Add Tickets page
    if (pageNumber == 1) {
        // Update navbar
        viewTicketsNavBtn.classList.remove('active');
        addTicketNavBtn.classList.add('active');

        // Update page
        viewTicketsContent.style.display = 'none';
        addTicketsContent.style.display = 'flex';
    }
}

// Document loaded
window.onload = function () {
    updatePage();
}

</script>

</body>

</html>

```

when I submit the form dissapears but the values doesnt appear to be pushed

When you click **Submit**, you immediately switch back to the “View Tickets” tab (`pageNumber = 0; updatePage()`) before waiting for your list to refresh with the new data. As a result, the form disappears but your table never gets the new row.

Let’s tweak the flow:

1. **Don’t switch views until after you refresh the table.**
2. **Make sure `updateTicketsTable()` is actually re-populating the `<tbody>`.**
3. **Add some logging** so you can see exactly what’s happening.

Here’s a revised submit handler (inside your existing `<script>` block, after `api_script.js` is loaded):

```

js

submitTicket.addEventListener('click', async function () {
    // 1) Read & validate
    const artist    = inputArtist.value.trim();
    const location  = inputLocation.value.trim();
    const price     = parseInt(inputPrice.value, 10);

    if (!artist || !location || isNaN(price)) {
        return alert('Please fill in all fields correctly.');
```

```

console.log('Submitting ticket:', {artist, location, price});

// 2) Send to API
try {
  const result = await addTicket(artist, location, price);
  console.log('addTicket result:', result);
} catch (err) {
  console.error('addTicket() failed:', err);
  return alert('Failed to add ticket. Check console.');
```

```

}

// 3) Clear form
inputArtist.value = '';
inputLocation.value = '';
inputPrice.value = '';

// 4) Reload the table
await updateTicketsTable();
console.log('Table after submit:', ticketsTableBody.innerHTML);

// 5) Now switch back to the list view
pageNumber = 0;
saveSessionData();
updatePage();
});
```

Why this helps

- **Step 4** now explicitly reloads the table *before* you hide the “Add” form.
- The `console.log` calls will show in your DevTools console:
 - The payload you’re sending (**Submitting ticket:...**)
 - The response from your server (**addTicket result:...**)
 - The new table HTML (**Table after submit:...**)

Next steps

1. **Open DevTools** → **Console** and watch those logs when you hit **Submit**.
2. If `addTicket()` is returning an error or `updateTicketsTable()` prints an empty `<tbody>`, you’ll know whether:
 - The POST never reached your Flask backend, or
 - Your GET endpoint `/get_tickets` isn’t returning the new ticket.

Once you confirm the API calls are succeeding and the table is repopulated, you should immediately see your new row appear. Let me know what the console logs show!



```

Uncaught (in promise) ReferenceError: can't access lexical declaration 'artist' before initialization
<anonymous> http://127.0.0.1:5000/:222
127.0.0.1:5000:222:49
<anonymous> http://127.0.0.1:5000/:222
```

The error message —


```
rust
```

```
Uncaught (in promise) ReferenceError: can't access lexical declaration 'artist' before initialization
```

— means that JavaScript hit the name `artist` before your `const artist = ...` line actually ran. This typically happens when your code is referencing that variable too early, or you've inadvertently shadowed or conflicted with a global (for instance, `location` is already a global referring to `window.location`).

Let's fix this by:

1. **Renaming** the field variables to avoid clashing with globals.
2. Ensuring **all reads and logs** happen after those `const` declarations.
3. Wrapping in `DOMContentLoaded` so nothing runs before the DOM is ready.

Here's a bullet-proof snippet to replace your submit handler:

```
html
```

```
<script>
document.addEventListener('DOMContentLoaded', () => {
  const artistInputEl = document.getElementById('artist-input');
  const venueInputEl = document.getElementById('location-input'); // renamed
  const priceInputEl = document.getElementById('price-input');
  const btnSubmit = document.getElementById('ticket-add-submit-btn');
  const btnReset = document.getElementById('ticket-add-reset-btn');
  const tbody = document.getElementById('tickets-table-body');

  btnReset.addEventListener('click', () => {
    document.getElementById('ticket-insert-form').reset();
  });

  btnSubmit.addEventListener('click', async () => {
    // 1) Read values into new names
    const artistName = artistInputEl.value.trim();
    const venueName = venueInputEl.value.trim();
    const ticketPrice = parseInt(priceInputEl.value, 10);

    // 2) Validate
    if (!artistName || !venueName || isNaN(ticketPrice)) {
      return alert('Please fill in all fields correctly.');
    }

    console.log('Submitting:', { artistName, venueName, ticketPrice });

    // 3) Call your existing helper
    try {
      const response = await addTicket(artistName, venueName, ticketPrice);
      console.log('addTicket response:', response);
    } catch (err) {
      console.error('addTicket() failed:', err);
      return alert('Failed to add ticket – see console.');
    }

    // 4) Clear and reload table before switching view
    document.getElementById('ticket-insert-form').reset();
    await updateTicketsTable();
    console.log('Table HTML after reload:', tbody.innerHTML);

    // 5) Switch back to list
    pageNumber = 0;
    saveSessionData();
    updatePage();
  });
});
```

```
});
</script>
```

Key changes:

- **Renamed** `location` → `venueName` (and similarly renamed the input element variable) to avoid shadowing the built-in `window.location`.
- **All const declarations** come before any use (the logs now reference `artistName` / `venueName` only after they're declared).
- Wrapped everything in `DOMContentLoaded` so the elements definitely exist before you grab them.

Drop that in place of your old submit handler, reload, and watch your console logs — you should no longer get the “can't access... before initialization” error, and the new ticket should appear once `updateTicketsTable()` runs.



```
async function getTickets() {
  try {
    const response = await fetch('http://127.0.0.1:5000/get_tickets');

    if (!response.ok) {
      throw new Error('error');
    }

    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Error with GET request:', error);
  }
}

function addTicket(artist, location, price) {
  return fetch('http://127.0.0.1:5000/add_ticket', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ 'artist': artist, 'location': location, 'price': price })
  })
  .then(response => response.json())
  .then(data => {
    console.log(data);
  })
  .catch(error => {
    console.error('Error:', error);
  });
}
```

// TASK 2

```
function deleteTicket() {

}

function handleReset() {
  document.getElementById('ticket-insert-form').reset();
}

/**
 * Reads the inputs, calls addTicket(), then resets form
 * and refreshes the tickets list.
 */
async function handleSubmit() {
  // Grab values
  const artist = document.getElementById('artist-input').value;
  const location = document.getElementById('location-input').value;
  const price = parseInt(document.getElementById('price-input').value, 10);

  // Post to the server
  await addTicket(artist, location, price);

  // Clear the form
  handleReset();

  // Refresh your view (if you're on the tickets page)
  await updateTicketsTable();

  // Optionally navigate back to the list view:
  pageNumber = 0;
  saveSessionData();
  updatePage();
}
```

client side

```
from model.ticket import Ticket
from data.sql_data_service import SqlDataService
```

```
class TicketManager:
```

```
    @staticmethod
    def retrieve_tickets():
        data_service = SqlDataService()
        tickets = data_service.retrieve_tickets()
        return [{'id': t.ticket_id, 'artist': t.artist, 'location': t.location, 'price': t.price} for t in tickets]

    @staticmethod
    def insert_ticket(data):
        artist = data.get('artist')
        location = data.get('location')
        price = data.get('price')
```

```
data_service = SqlDataService()
return data_service.insert_ticket(Ticket(artist, location, price))
```

```
# TASK 2
@staticmethod
def delete_ticket(data):
    pass
```

server side

In this task you are expected to implement the logic behind the "delete" ticket operation. Once the delete button has been clicked, the platform needs to send a request to the server to delete the ticket

from the database. Once the request has been "nalized the table needs to refresh. You are required to implement both client and server side logic.

To test your implementation you can introduce more false tickets using the form implemented in the

irst task. Once you have "nished your implementation please delete the following ticket (and all other

tickets created for testing):

- Artist: Artist-Delete
- Location: Location-Delete
- Price: -1

task

Below is a complete, end-to-end implementation of the **Delete** flow.

1. Client-side JavaScript

a) Hook up each Delete button

In your existing `addFunctionalityDeleteBtns()`:

```
js

function addFunctionalityDeleteBtns() {
    document.querySelectorAll('.delete-ticket-btn').forEach(button => {
        button.addEventListener('click', async () => {
            const id = button.dataset.id; // see step (b)
            await deleteTicket(id);
            await updateTicketsTable(); // refresh the table
        });
    });
}
```

Make sure your table-row template uses **data-id** rather than **id**:

```
js

// inside updateTicketsTable()
tr.innerHTML = `
    <td>${t.artist}</td>
```

```
<td>${t.location}</td>
<td>${t.price}</td>
<td class="text-end">
  <button
    type="button"
    data-id="${t.id}"
    class="btn btn-danger delete-ticket-btn">
    Delete
  </button>
</td>`;
```

b) Implement deleteTicket(id)

Add this alongside your other API helpers in `api_script.js`:

```
js

/**
 * Sends DELETE /delete_ticket { id }
 * Returns the parsed JSON response.
 */
async function deleteTicket(id) {
  try {
    const resp = await fetch('http://127.0.0.1:5000/delete_ticket', {
      method: 'DELETE',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ id })
    });
    if (!resp.ok) throw new Error(`HTTP ${resp.status}`);
    const data = await resp.json();
    console.log('deleteTicket result:', data);
    return data;
  } catch (err) {
    console.error('Error in deleteTicket():', err);
    alert('Failed to delete ticket. See console.');
```

2. Server-side Python

a) Extend TicketManager.delete_ticket

In `model/ticket_manager.py` (or wherever you keep it):

```
python

from data.sql_data_service import SqlDataService

class TicketManager:

    # ... existing methods ...

    @staticmethod
    def delete_ticket(data):
        """
        data: dict containing {'id': <ticket_id>}
        Returns True on success, False or raises on failure.
        """
        ticket_id = data.get('id')
        if ticket_id is None:
            raise ValueError("Missing 'id' in delete_ticket payload")
        ds = SqlDataService()
        return ds.delete_ticket(ticket_id)
```

b) Add a Flask route

In your `app.py` (or wherever you define routes):

```
python

from flask import Flask, request, jsonify
from model.ticket_manager import TicketManager

app = Flask(__name__)

# ... existing routes ...

@app.route('/delete_ticket', methods=['DELETE'])
def delete_ticket_route():
    payload = request.get_json()
    try:
        success = TicketManager.delete_ticket(payload)
        return jsonify({'status': 'success', 'deleted_id': payload.get('id')}), 200
    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)}), 400
```

c) Implement `SqlDataService.delete_ticket`

In `data/sql_data_service.py`:

```
python

import sqlite3
from model.ticket import Ticket

class SqlDataService:
    DB_PATH = 'your_database.db'

    # ... existing methods ...

    def delete_ticket(self, ticket_id: int) -> bool:
        """Delete the ticket row with the given ID; return True if a row was deleted."""
        conn = sqlite3.connect(self.DB_PATH)
        cur = conn.cursor()
        cur.execute("DELETE FROM tickets WHERE ticket_id = ?", (ticket_id,))
        deleted = cur.rowcount
        conn.commit()
        conn.close()
        return deleted > 0
```

3. Testing

1. **Add** a few tickets via your form (including the special "Artist-Delete" etc.).
2. **Open** DevTools -> Network to watch the **DELETE** `/delete_ticket` calls.
3. **Click** a row's **Delete** button. You should see in Console:

```
css

deleteTicket result: {status: "success", deleted_id: 42}
```

4. The table will auto-refresh, and the deleted row should vanish.

With these pieces in place, the client calls your new DELETE endpoint, the Flask route delegates to `TicketManager.delete_ticket()`, which in turn deletes the row via your SQL service—and your UI immediately reflects the change.



```
tr.innerHTML =
  <td>${t.artist}</td>
  <td>${t.location}</td>
  <td>${t.price}$</td>
  <td class="text-end">
    <button
      type="button"
      data-id="${t.id}"
      class="btn btn-danger delete-ticket-btn">
      Delete
    </button>
  </td>;
```

where is this?

That snippet belongs inside your `updateTicketsTable()` function—where you build each row of the tickets table. Here's the full context, showing exactly where to replace your existing string concatenation:

```
js

async function updateTicketsTable() {
  ticketsTableBody.innerHTML = "";

  const tickets = await getTickets();
  for (const t of tickets) {
    // Create a new <tr> for each ticket
    const tr = document.createElement('tr');

    // Populate the row's HTML, including our data-id attr on the Delete button
    tr.innerHTML = `
      <td>${t.artist}</td>
      <td>${t.location}</td>
      <td>${t.price}$</td>
      <td class="text-end">
        <button
          type="button"
          data-id="${t.id}"
          class="btn btn-danger delete-ticket-btn">
          Delete
        </button>
      </td>`;

    // Append the row to the table body
    ticketsTableBody.appendChild(tr);
  }

  // Wire up all delete buttons after they exist
  addFunctionalityDeleteBtns();
}
```

Steps to integrate:

1. **Open** the script block (or external JS file) where `updateTicketsTable()` is defined.

2. **Locate** the loop where you currently do something like `ticketsTableBody.innerHTML += entryHtml;`
3. **Replace** that block with the code above, creating a `<tr>`, setting its `innerHTML` to the template, and appending it.
4. **Ensure** `addFunctionalityDeleteBtns()` then selects on `.delete-ticket-btn` (which now have `data-id` attributes).

With that in place, each Delete button carries its ticket's ID in `button.dataset.id`, and your `deleteTicket(id)` helper will know which ticket to remove.



```
const entryHtml =
  <tr>
    <td>${ticket.artist}</td>
    <td>${ticket.location}</td>
    <td>${ticket.price}$</td>
    <td class="text-end">
      <button id="${ticket.id}" type="button" class="btn btn-danger menu-btn delete-
ticket-btn">Delete</button>
    </td>
  </tr>
;
```

so shall I change it here?

Yes—exactly in that spot. You just need to swap out the `id="..."` on the button for a `data-id="..."`, so your delete handler can read it without colliding with DOM IDs.

Change this:

```
js

const entryHtml = `
  <tr>
    <td>${ticket.artist}</td>
    <td>${ticket.location}</td>
    <td>${ticket.price}$</td>
    <td class="text-end">
      <button id="${ticket.id}"
        type="button"
        class="btn btn-danger menu-btn delete-ticket-btn">
        Delete
      </button>
    </td>
  </tr>
`;
```

to this:

```
js

const entryHtml = `
  <tr>
    <td>${ticket.artist}</td>
    <td>${ticket.location}</td>
    <td>${ticket.price}$</td>
```



```
<td class="text-end">
  <button data-id="${ticket.id}"
    type="button"
    class="btn btn-danger menu-btn delete-ticket-btn">
    Delete
  </button>
</td>
</tr>
`;
```

Why this helps

- **data-id** attributes don't collide with the element's **id** namespace.
- In your **addFunctionalityDeleteBtns()**, you'll now do:

```
js

button.addEventListener('click', async () => {
  const id = button.dataset.id; // grabs the ticket ID
  await deleteTicket(id);
  await updateTicketsTable();
});
```

That small change will let your delete handler know exactly which ticket to remove.



I'm getting 400 response, so there are ISSUES WITH ID

A 400 almost always means your Flask route didn't like the payload it received. Let's debug and fix it step by step:

1. Inspect exactly what you're sending

Open your browser's DevTools → **Network** tab. Click **Delete**, then select the DELETE request. Check:

- **Request URL:** should be **http://127.0.0.1:5000/delete_ticket**
- **Request Method:** DELETE
- **Request Payload** (under the "Headers" or "Payload" pane): should look like

```
json

{ "id": "42" }
```

or

```
json

{ "id": 42 }
```

If instead you see something like `{}` or `{"undefined": ...}` or no JSON at all, that's your problem.

Client fix

Make sure your client-side helper is sending the right body:

```
js

async function deleteTicket(id) {
  console.log('Deleting ticket id=', id);
  const resp = await fetch('http://127.0.0.1:5000/delete_ticket', {
    method: 'DELETE',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ id: Number(id) }) // ensure it's a number
  });
  console.log('Status:', resp.status);
  const data = await resp.json().catch(() => null);
  console.log('Response JSON:', data);
  if (!resp.ok) throw new Error(`Server returned ${resp.status}`);
  return data;
}
```

- We `Number(id)` to guarantee we're sending a numeric `id` not a string.
- We `console.log` both the outgoing payload and the response status + JSON.

2. Harden your Flask route

On the server, tweak `delete_ticket_route` to log what it actually received and cast to an integer:

```
python

@app.route('/delete_ticket', methods=['DELETE'])
def delete_ticket_route():
    payload = request.get_json(force=True) # even if Content-Type is off
    app.logger.debug('DELETE payload: %s', payload)
    ticket_id = payload.get('id')
    try:
        ticket_id = int(ticket_id)
    except (TypeError, ValueError):
        return jsonify({'status': 'error', 'message': f'Invalid id: {ticket_id}'}), 400

    try:
        success = TicketManager.delete_ticket({'id': ticket_id})
        if not success:
            return jsonify({'status': 'error', 'message': 'No such ticket'}), 404
        return jsonify({'status': 'success', 'deleted_id': ticket_id}), 200
    except Exception as e:
        app.logger.exception('Error deleting ticket')
        return jsonify({'status': 'error', 'message': str(e)}), 500
```

- `force=True` ensures you get a dict even if headers are off.
- We log the incoming payload so you can see in Flask's console exactly what arrived.
- We cast `id` to `int`, returning a 400 if it's not numeric.
- We return 404 if nothing was deleted, and 500 on unexpected errors.

3. Test again

1. **Restart Flask** so you pick up the new logging.

2. In your browser, delete a ticket.

3. Watch **both** consoles:

- **Browser console / Network:** see "Deleting ticket id=" log and the payload/status.
- **Flask console:** see **DELETE** **payload:** {...} and whether you hit the **Invalid id** branch or proceed.

Once the payload matches what the server expects (an integer **id**), you should get a 200 response and the row will disappear when you **await** **updateTicketsTable()** again.

Let me know what you see in the logs if it still fails!



```
@staticmethod
def delete_ticket(data):
    """
    data: dict containing {'id': <ticket_id>}
    Returns True on success, False or raises on failure.
    """
    print(data)
    ticket_id = data.get('id')
    if ticket_id is None:
        raise ValueError("Missing 'id' in delete_ticket payload")
    ds = SqlDataService()
    return ds.delete_ticket(ticket_id)
```

the id gets here, yet I get 400

Since your **TicketManager.delete_ticket(data)** is clearly seeing the **id**, the **400** is being generated by your **Flask route** before or after calling that method. Let's make sure your route is accepting the payload correctly and returning a 200 when **delete_ticket** succeeds.

A. Add / Fix the Flask route

In your **app.py**, define the route like this:

```
python

from flask import Flask, request, jsonify
from model.ticket_manager import TicketManager

app = Flask(__name__)

@app.route('/delete_ticket', methods=['DELETE'])
def delete_ticket_route():
    # 1) Force-parse JSON so we always get a dict
    payload = request.get_json(force=True)
    app.logger.debug(f"DELETE /delete_ticket payload: {payload}")

    # 2) Extract and validate the ID
    ticket_id = payload.get('id')
    try:
        ticket_id = int(ticket_id)
```

```
except (TypeError, ValueError):
    return (
        jsonify({
            'status': 'error',
            'message': f"Invalid id: {ticket_id!r}"
        }),
        400
    )

# 3) Call your manager
try:
    deleted = TicketManager.delete_ticket({'id': ticket_id})
    if not deleted:
        # No row was deleted
        return jsonify({
            'status': 'error',
            'message': 'Ticket not found'
        }), 404

    # Success
    return jsonify({
        'status': 'success',
        'deleted_id': ticket_id
    }), 200

except Exception as e:
    app.logger.exception("Error in delete_ticket_route")
    return jsonify({
        'status': 'error',
        'message': str(e)
    }), 500
```

Why this helps:

1. **force=True** makes sure you always get a JSON payload, even if the **Content-Type** header wasn't set correctly.
2. We **cast** `ticket_id` to `int`, returning a clear 400 if it's not numeric.
3. We log with `app.logger.debug` so you can watch the Flask console for exactly what arrived.
4. We return **404** if nothing was deleted, **500** for other errors, and **200** on success.

B. Verify the client request

In your browser's DevTools **Network** tab, find the **DELETE** `/delete_ticket` request and confirm:

- **Request URL:** `http://127.0.0.1:5000/delete_ticket`
- **Request Method:** DELETE
- **Request Headers:**

```
pgsql
```

```
Content-Type: application/json
```

- **Request Payload:**

```
json
```

```
{ "id": 42 }
```

If you see an empty payload or a missing header, update your client helper to explicitly set it:

```
js

async function deleteTicket(id) {
  console.log('Deleting ticket id=', id);
  const resp = await fetch('http://127.0.0.1:5000/delete_ticket', {
    method: 'DELETE',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ id: Number(id) })
  });
  console.log('Response status:', resp.status);
  const data = await resp.json().catch(() => null);
  console.log('Response JSON:', data);
  if (!resp.ok) throw new Error(`HTTP ${resp.status}`);
  return data;
}
```

C. Watch the logs

1. **Restart** your Flask app so you see the new debug log.
2. In Flask's console you should see something like:

```
pgsql

[DEBUG] DELETE /delete_ticket payload: {'id': 42}
```

3. In your browser console you'll see:

```
yaml

Deleting ticket id= 42
Response status: 200
Response JSON: {status: "success", deleted_id: 42}
```

If you do still get a 400, the Flask log will tell you whether it was the “Invalid id” branch or something else. From there we can iterate—but these changes will make the failure mode explicit and ensure that a valid `id` produces a 200.



```
deleteTicket http://127.0.0.1:5000/scripts/api_script.js:41
addFunctionalityDeleteBtns http://127.0.0.1:5000/:247
addFunctionalityDeleteBtns http://127.0.0.1:5000/:245
addFunctionalityDeleteBtns http://127.0.0.1:5000/:244
updateTicketsTable http://127.0.0.1:5000/:274
updatePage http://127.0.0.1:5000/:295
onload http://127.0.0.1:5000/:314
EventHandlerNonNull* http://127.0.0.1:5000/:313
api_script.js:46:15
deleteTicket http://127.0.0.1:5000/scripts/api_script.js:46
addFunctionalityDeleteBtns http://127.0.0.1:5000/:247
(Async: EventListener.handleEvent)
```

```
addFunctionalityDeleteBtns http://127.0.0.1:5000/:245
addFunctionalityDeleteBtns http://127.0.0.1:5000/:244
updateTicketsTable http://127.0.0.1:5000/:274
updatePage http://127.0.0.1:5000/:295
onload http://127.0.0.1:5000/:314
(Async: EventHandlerNonNull)
<anonymous> http://127.0.0.1:5000/:313
```

Let's drill into exactly what your client is sending and what your Flask endpoint is seeing. Right now you know the handler is being called (your stack trace lands in `deleteTicket`), but the response is 400. Here's how to pinpoint—and fix—it:

1. Log the outgoing request in `deleteTicket`

Open `api_script.js`, find your `deleteTicket` helper, and replace it with this instrumented version:

```
js

async function deleteTicket(id) {
  console.log('> deleteTicket() called with id=', id);

  const payload = { id: Number(id) };
  console.log('> Sending DELETE with body:', payload);

  const resp = await fetch('http://127.0.0.1:5000/delete_ticket', {
    method: 'DELETE',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(payload),
  });

  console.log('> Response status:', resp.status);
  let text;
  try {
    text = await resp.text();
    console.log('> Raw response text:', text);
  } catch (e) {
    console.warn('> Couldn't read response text', e);
  }

  let data;
  try {
    data = JSON.parse(text);
    console.log('> Parsed JSON:', data);
  } catch (e) {
    console.warn('> Response wasn't JSON', e);
  }

  if (!resp.ok) {
    throw new Error(`DELETE /delete_ticket returned ${resp.status}`);
  }
  return data;
}
```

1. **Logs the ID** you're deleting.
2. **Prints the exact JSON body** you're sending.
3. **Shows the raw response text** and its parsed JSON.
4. **Throws** if the status isn't 2xx so you'll see the error in your console.

Reload the page, open DevTools → Console, click **Delete**, and observe:

- **What payload** was sent?
- **Status code** (400) and the **raw response** body from Flask.

2. Inspect the Flask logs

Meanwhile, in your Flask app add a debug log at the top of the delete route:

```
python

@app.route('/delete_ticket', methods=['DELETE'])
def delete_ticket_route():
    payload = request.get_json(force=True)
    app.logger.debug(f"[DELETE /delete_ticket] received payload: {payload!r}")
    # ... rest of your code ...
```

Restart Flask, and then in its console you should see exactly what it received. If you see:

- `{'id': 42}` → that's correct
- `{'id': '42'}` → still okay after `int()` cast
- `{}` or `None` → something's wrong on the client side

3. Common culprits

- **Missing or wrong Content-Type header:** without `application/json`, Flask may not parse the body
- **Not calling `JSON.stringify`:** an object literal in `body` gets turned into `[object Object]`
- **Button ID vs. data attribute mix-up:** if you used `id="${ticket.id}"` instead of `data-id`, you may have passed `undefined`

Once you see the exact payload and the server log, you'll know:

- If it's client-side, fix the fetch call (often missing `JSON.stringify` or header)
- If it's server-side, adjust the route to accept the form you're sending

Go through these logs, let me know what the request body and Flask debug line show, and we'll correct that mismatch so you get a 200 and the row disappears as intended!



the output is `{'id': '6'}` in the flask route, yet it fails?

Since you're seeing `{'id': '6'}` in Flask but still getting a 400, the failure must be coming from your route's validation or the delete logic returning a "not found" error. Let's pinpoint it.

1. Confirm your Flask route

Make sure it looks like this exactly:

```
python

@app.route('/delete_ticket', methods=['DELETE'])
def delete_ticket_route():
    payload = request.get_json(force=True)
    app.logger.debug(f"[DELETE] payload: {payload!r}")

    # 1) Validate the ID
    ticket_id = payload.get('id')
    try:
        ticket_id = int(ticket_id)
    except (TypeError, ValueError):
        return jsonify({
            'status': 'error',
            'message': f"Invalid id: {ticket_id!r}"
        }), 400

    # 2) Attempt deletion
    try:
        deleted = TicketManager.delete_ticket({'id': ticket_id})
        app.logger.debug(f"[DELETE] delete_ticket returned: {deleted}")
    except Exception as e:
        app.logger.exception("error in delete_ticket")
        return jsonify({'status': 'error', 'message': str(e)}), 500

    # 3) Respond based on result
    if not deleted:
        # No such row in the database
        return jsonify({
            'status': 'error',
            'message': f"Ticket with id {ticket_id} not found"
        }), 404

    return jsonify({'status': 'success', 'deleted_id': ticket_id}), 200
```

Key points:

- `force=True` ensures you always get a JSON body.
- We cast the string '6' to integer 6.
- We log (`app.logger.debug`) both the incoming payload and the Boolean returned by `delete_ticket`.
- If `deleted` is `False`, we return **404**, not 400.

2. Verify your `SqlDataService.delete_ticket`

In `sql_data_service.py`, double-check the column name and table name:

```
python

def delete_ticket(self, ticket_id: int) -> bool:
    conn = sqlite3.connect(self.DB_PATH)
    cur = conn.cursor()
    # Make sure this matches how you created the table:
    cur.execute("DELETE FROM tickets WHERE ticket_id = ?", (ticket_id,))
    deleted_count = cur.rowcount
    conn.commit()
```



```
conn.close()
return deleted_count > 0
```

- If your table's primary key column isn't named `ticket_id` (for example, it's `id`), change the SQL accordingly:

```
sql

DELETE FROM tickets WHERE id = ?
```

- **Add a debug print** before `conn.commit()`:

```
python

print(f"Deleting ticket {ticket_id}, rows affected: {deleted_count}")
```

That will tell you if the SQL deletion is actually finding a row.

3. Test and read the logs

1. **Restart** Flask so your debug prints appear.
2. In the Flask console you should see:

```
pgsql

[DELETE] payload: {'id': '6'}
[DELETE] delete_ticket returned: True    # <-- or False
```

And in the SQL service prints:

```
sql

Deleting ticket 6, rows affected: 1
```

3. If you see `deleted_count: 0`, it means there is **no ticket** with `ticket_id = 6` in your database. That would explain why the route returns a 404 (or a 400 if that's still how your code is currently written).

4. Fix the “400” if it’s coming from your route

If you'd prefer to treat “no row found” as a 400 instead of 404, change:

```
python

if not deleted:
    return jsonify({ 'status': 'error', 'message': 'Ticket not found' }), 400
```

—but 404 is more semantically correct for “resource doesn't exist.”

Summary

- **Payload is fine** ('6' → 6 after `int()`).
- **Likely culprit:** your SQL deletion didn't find a row, so `deleted` is `False` and your route returns an error.
- **Action items:**
 1. Add debug prints in both the Flask route and SQL service.
 2. Verify your table's primary-key column name.
 3. Ensure there really is a ticket with that ID to delete.

Once `deleted` comes back `True`, your route will return `200` and your client will remove the row and refresh the table.



```
# TASK 2
def delete_ticket(self, ticket_id: int) -> bool:
    """Delete the ticket row with the given ID; return True if a row was deleted."""
    conn = sqlite3.connect(self.DB_PATH)
    cur = conn.cursor()
    cur.execute("DELETE FROM tickets WHERE ticket_id = ?", (ticket_id,))
    deleted = cur.rowcount
    conn.commit()
    conn.close()
    return deleted > 0
```

I think the error is here?

Your `delete_ticket` method looks syntactically correct—but if no rows are ever deleted (i.e. `deleted == 0`), it usually means the **WHERE** clause didn't match any primary-key values. Here's how to diagnose and fix it:

1. Verify your table's primary-key column name

In your SQLite database, open a REPL or admin tool and run:

```
sql

PRAGMA table_info(tickets);
```

That will list every column in the `tickets` table. Look for the column that actually holds your ticket ID. Common names are:

- `ticket_id`
- `id`
- `pk`

If the column is **not** named `ticket_id`, update your DELETE query accordingly. For example, if it's just `id`:

```
python

cur.execute("DELETE FROM tickets WHERE id = ?", (ticket_id,))
```

2. Add debug logging around the delete

Right now you return `deleted > 0`, but you don't see how many rows matched. Let's print that out:

```
python

def delete_ticket(self, ticket_id: int) -> bool:
    conn = sqlite3.connect(self.DB_PATH)
    cur = conn.cursor()
    cur.execute("DELETE FROM tickets WHERE ticket_id = ?", (ticket_id,))
    deleted = cur.rowcount
    print(f"[SQL] delete_ticket({ticket_id}) affected rows:", deleted)
    conn.commit()
    conn.close()
    return deleted > 0
```

- After a DELETE, `cur.rowcount` tells you how many rows were removed.
- If it prints `0`, you know the SQL didn't find a matching row.

3. Ensure you're calling the right method

In your Flask route you do:

```
python

deleted = TicketManager.delete_ticket({'id': ticket_id})
```

But your `delete_ticket` signature expects just the integer, not a dict. Make sure you're calling it correctly:

```
python

# Wrong:
deleted = TicketManager.delete_ticket({'id': ticket_id})

# Right:
deleted = TicketManager.delete_ticket(ticket_id)
```

— or if you want to keep the dict wrapper, adjust the manager method:

```
python

@staticmethod
def delete_ticket(data):
    ticket_id = data.get('id')
    return SqlDataService().delete_ticket(ticket_id)
```

Your Flask route then does:

```
python

deleted = TicketManager.delete_ticket({'id': ticket_id})
```

Either approach works, but the signatures must match.

4. Test again

1. Restart your Flask app.
2. Trigger a deletion from the UI.
3. Watch the Flask console:

```
pgsql

[DEBUG] DELETE payload: {'id': '6'}
[SQL] delete_ticket(6) affected rows: 1
[DEBUG] delete_ticket returned: True
```

4. If **affected rows** is **0**, adjust your SQL to the correct column name. If it's **1**, but you still get a 400, verify that your route returns 200 on **deleted == True**.

Once the SQL deletion actually matches a row, you'll see **deleted > 0**, your route will return a 200, and the client will successfully remove the row from the table.



```
# Ticket Functions
def retrieve_tickets(self):
    query = f'SELECT * FROM {self.table_tickets} ORDER BY {self.field_tickets_id} DESC'
    print(query)

    entries = self.execute_query(query)
    results = [Ticket(e[1], e[2], e[3], e[0]) for e in entries]
    return results

def insert_ticket(self, new_ticket):
    query = f'INSERT INTO {self.table_tickets} ({self.field_tickets_artist}, {self.field_tickets_location}, {self.field_tickets_price}) VALUES (?, ?, ?)'
    values = (new_ticket.artist, new_ticket.location, new_ticket.price)
    print(query)

    result = self.execute_query(query, values)
    return result

# TASK 2
def delete_ticket(self, ticket_id: int) -> bool:
    """Delete the ticket row with the given ID; return True if a row was deleted."""
    conn = sqlite3.connect(self.DB_PATH)
    cur = conn.cursor()
```

```
cur.execute("DELETE FROM tickets WHERE ticket_id = ?", (ticket_id,))
deleted = cur.rowcount
conn.commit()
conn.close()
return deleted == 1
```

rewrite the delete with query

Here's a version of `delete_ticket` that uses your existing `execute_query` helper and your class's table/field constants:

python

```
def delete_ticket(self, ticket_id: int) -> bool:
    """
    Delete the ticket row with the given ID; return True if exactly one row was deleted.
    """
    # Build a parameterized DELETE statement using your table and ID field names
    query = f"DELETE FROM {self.table_tickets} WHERE {self.field_tickets_id} = ?"
    values = (ticket_id,)

    print(f"[SQL] {query} values={values}")

    # execute_query should run the DELETE and return the number of affected rows
    rows_affected = self.execute_query(query, values)

    # Depending on your implementation, execute_query might return None or
    # some cursor/result – adjust if necessary. If it returns cursor, you can do:
    # rows_affected = cursor.rowcount

    return rows_affected == 1
```

Notes:

- This assumes `execute_query` accepts (`query`, `values`) and returns the number of affected rows for DML statements.
- If `execute_query` instead returns a cursor, you can extract `cursor.rowcount` and return `cursor.rowcount == 1`.
- We print the generated SQL and values so you can verify it in your logs.


With this in place, deleting ticket #6 will run:

sql

```
DELETE FROM tickets WHERE ticket_id = ? -- [6]
```

and return `True` only if one row was removed.



 Uploaded image
hey, its me again,

Let's compute each ticket's new price according to:

$$P_{\text{new}} = P_0 \times \underbrace{\left(1 + c_{\text{time}} \left(1 - \frac{d_{\text{left}}}{30}\right)\right)}_{\text{time-markup}} + \underbrace{c_{\text{avail}} \left(1 - \frac{v_t}{v_0}\right)}_{\text{availability-markup}} \times \text{seat_mult}$$

with

- $c_{\text{time}} = 0.30$, 30 % maximum for time
- $c_{\text{avail}} = 0.20$, 20 % maximum for availability
- $\text{seat_mult} = 1.0$ for both VIP and Standard

Ticket	Days left d_{left}	v_t/v_0	time-markup	avail-markup	total markup	Multiplier	P_{new}/P_0
A (VIP)	3	$100/500 = 0.20$	$0.30 \times (1 - 3/30) = 0.27$	$0.20 \times (1 - 0.20) = 0.16$	0.43	1.0	1.43
B (Std)	10	$400/1200 = 0.333...$	$0.30 \times (1 - 10/30) = 0.20$	$0.20 \times (1 - 0.333...) = 0.1333$	0.3333	1.0	1.3333
C (Std)	15	$200/1800 = 0.111...$	$0.30 \times (1 - 15/30) = 0.15$	$0.20 \times (1 - 0.111...) = 0.1778$	0.3278	1.0	1.3278

SQL to apply these updates

Assuming your table is named **tickets** with columns **id**, **price** and you know each ticket's **id** (or can match by **artist**), you can issue:

```
sql

BEGIN TRANSACTION;

-- Ticket A (id = 1 for example)
UPDATE tickets
SET price = ROUND(price * 1.43, 2)
WHERE id = 1;

-- Ticket B (id = 2)
UPDATE tickets
SET price = ROUND(price * 1.3333, 2)
WHERE id = 2;

-- Ticket C (id = 3)
UPDATE tickets
SET price = ROUND(price * 1.3278, 2)
WHERE id = 3;

COMMIT;
```

If you don't know the IDs but can identify by artist name:

```
sql

BEGIN TRANSACTION;

UPDATE tickets
SET price = ROUND(price * 1.43, 2)
WHERE artist = 'Artist A';
```

```
UPDATE tickets
SET price = ROUND(price * 1.3333, 2)
WHERE artist = 'Artist B';

UPDATE tickets
SET price = ROUND(price * 1.3278, 2)
WHERE artist = 'Artist C';

COMMIT;
```

- I've used **ROUND(..., 2)** to keep two-decimal precision.
- Adjust the multipliers if you want more exact decimals.

Run these in DB Browser's **Execute SQL** tab, hit the green ► button, then **Write Changes** to commit. That completes Task 2.