Yuno SDK FULL — Demo Project (Generic Checkout)

This canvas contains a ready-to-run demo project implementing the requirements from the **Yuno Technical Support Analyst - Assessment**. It includes a simple frontend (generic checkout page) and a minimal backend (Node/Express). The code uses **placeholders** for Yuno credentials and the SDK script URL — replace with the real values from Yuno Dashboard / Documentation when you have them.

Project structure

```
yuno-demo/
├─ frontend/
│  ├─ index.html
│  └─ README.md
├─ backend/
│  ├─ server.js
│  ├─ package.json
│  └─ .env.example
└─ README.md
```

Top-level README (quick)

- Frontend displays an embedded checkout using **Yuno SDK FULL (JavaScript)**.
- Backend exposes an endpoint to create a payment session / order and communicates with **Yuno Testing Gateway**.
- Replace placeholders `YUNO_API_KEY` and `YUNO_MERCHANT_ID` in `.env` with the values provided by Yuno Dashboard invitation.

---

frontend/index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Yuno SDK FULL — Demo Checkout</title>
  <style>
    body { font-family: system-ui, Arial; padding: 24px; }
    .card { max-width: 720px; margin: 0 auto; border: 1px solid #e6e6e6;
padding: 18px; border-radius: 8px; }
    label { display:block; margin-top:10px }
```

```
input[type=text], input[type=number] { width:100%; padding:8px; margin-
top:6px } button { margin-top:12px; padding:10px 14px; border-radius:6px }
#checkout-frame { margin-top:18px; border: none; width:100%; height:
520px }
</style>
</head>
<body>
<div class="card">
<h2>Yuno Demo — Embedded Checkout</h2>
    <p>This page demonstrates a generic checkout and uses a backend endpoint
to create a payment session. It uses the <strong>SDK FULL (JavaScript)</
strong> — replace the SDK script reference with the official one from Yuno
docs.</p>

<label>Customer Name
<input id="customerName" type="text" value="John Doe">
</label>

<label>Amount (minor units, e.g. 1000 = 10.00)
<input id="amount" type="number" value="1000">
</label>

<label>Currency
<input id="currency" type="text" value="USD">
</label>

<button id="startCheckout">Start Embedded Checkout</button>

<div id="status" style="margin-top:12px;color:#333"></div>

    <!-- Placeholder iframe for embedded checkout (if using an iframe
integration) -->
<iframe id="checkout-frame" style="display:none"></iframe>
</div>

<!--
    IMPORTANT: Replace the placeholder below with the official Yuno SDK FULL
JavaScript include     from the documentation. Example (fictional):
    <script        src="https://cdn.y.uno/sdk-full/v1/yuno-sdk-full.min.js"></
script>   -->
<script>
    // NOTE: This demo uses a generic flow where the backend returns a
`checkout_url` or a `sessionId`.
    // The exact usage of the Yuno SDK FULL might differ (init method names,
fields). Check Yuno docs
// and adapt the `initYunoCheckout` function to match the SDK API.

async function startEmbeddedCheckout() {
```

```
const name = document.getElementById('customerName').value; const amount =
Number(document.getElementById('amount').value); const currency =
document.getElementById('currency').value || 'USD'; setStatus('Creating
payment session...');

try {
const resp = await fetch('/api/create-session', { method: 'POST',
headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ name,
amount, currency })
});
const data = await resp.json();

if (!resp.ok) throw new Error(data.message || 'Failed to create
session');

        // Data from backend should include either a `checkout_url` (for
iframe embed) or a `sessionId`
// Adapt below to match Yuno SDK FULL usage

if (data.checkout_url) { // Show iframe embedded checkout
const frame = document.getElementById('checkout-frame'); frame.src =
data.checkout_url; frame.style.display = 'block';

setStatus('Embedded checkout loaded. Follow the checkout inside the page.');
} else if (data.sessionId) {
          // Example of calling the SDK init. Replace with actual SDK method
names.
          // window.Yuno && window.Yuno.init && window.Yuno.init({sessionId:
data.sessionId, container: '#checkout-frame'})

setStatus('SDK session created (sessionId). Initialize the Yuno SDK using the
provided sessionId.');
// For demo, open a new tab to simulate
window.open(data.sessionUrl || '/', '_blank');
} else {
          setStatus('Payment session created — but response format is
unrecognized. Check backend output.');
}

} catch (err) {
console.error(err);
  setStatus('Error: ' + (err.message || err)); }
}

    function setStatus(msg) { document.getElementById('status').textContent
= msg }
```

```
    document.getElementById('startCheckout').addEventListener('click',
startEmbeddedCheckout);
  </script>
</body>
</html>
```

---

frontend/README.md

```
# Frontend (index.html)

- Simple page that requests a backend session and then embeds the checkout.
- Replace any SDK placeholder with the real JavaScript SDK from Yuno docs.
- The backend endpoint expected: POST /api/create-session -> returns JSON {
  checkout_url | sessionId | sessionUrl }
```

---

backend/server.js (Node + Express)

```
require('dotenv').config(); const
express = require('express');
const fetch = require('node-fetch'); // or use axios
const app = express();
app.use(express.json());

const PORT = process.env.PORT || 3000;
const YUNO_API_KEY = process.env.YUNO_API_KEY || 'REPLACE_ME'; const
YUNO_MERCHANT_ID = process.env.YUNO_MERCHANT_ID || 'REPLACE_ME';
// The base URL for Yuno Testing Gateway (replace with actual testing gateway
URL from docs)
const YUNO_TEST_BASE = process.env.YUNO_TEST_BASE || 'https://apitest.y.uno';

app.post('/api/create-session', async (req, res) => {
  try {
    const { name, amount, currency } = req.body;
    if (!amount || !currency) return res.status(400).json({ message: 'amount
and currency required' });

    // Create a payment/order object in Yuno Testing Gateway
    // IMPORTANT: adapt the payload to the Yuno API contract from their
    docs const payload = { merchant_id: YUNO_MERCHANT_ID, amount: amount,
    currency: currency,
      customer: { name: name || 'Demo Customer' },
```

```
      // callback urls used by the merchant to receive async updates
      return_url: process.env.RETURN_URL || 'https://example.com/return',
      // other routing or payment options can be added here
    };

    const r = await fetch(`${YUNO_TEST_BASE}/v1/payments`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${YUNO_API_KEY}`
      },
      body: JSON.stringify(payload)
    });

    const data = await r.json();
    if (!r.ok) {
      console.error('Yuno create payment failed', data);
      return res.status(502).json({ message: 'Yuno API error', details:
data });
    }

    // Assume response includes a checkout URL or session id
    // Example shape: { id: 'pay_123', checkout_url: 'https://.../checkout/
pay_123' }
    return res.json(data);

  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'internal error', error: String(err) });
  }
});

app.listen(PORT, () => console.log(`Server running on http://localhost:$
{PORT}`));
```

---

backend/package.json

```
{
  "name": "yuno-demo-backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
  },
  "dependencies": {
    "dotenv": "^16.0.0",
    "express": "^4.18.2",

    "node-fetch": "^2.6.7"
  }
}
```

---

backend/.env.example

```
YUNO_API_KEY=sk_test_replace_me
YUNO_MERCHANT_ID=merchant_abc
YUNO_TEST_BASE=https://api-test.y.uno
RETURN_URL=https://example.com/return
PORT=3000
```

How to run locally

1. Clone the canvas content into a folder `yuno-demo .`
2. Start the backend:
3. `cd backend`
4. copy `.env.example` to `.env` and populate values from your Yuno Dashboard invitation
5. `npm install`
6. `npm start`
7. Serve the frontend (open `frontend/index.html` in browser or serve via a simple static server).
8. On the frontend click **Start Embedded Checkout**.

Notes & Next steps

- The exact field names and API endpoints used by Yuno SDK FULL and the Testing Gateway must be adapted following the official Yuno docs. Replace the placeholder SDK script, adjust the `server.js` payload shape to match the Yuno payments endpoint contract, and map response fields (`checkout_url` , `sessionId` , etc.) accordingly.

- Implement webhook handling on backend to listen for asynchronous payment events from Yuno (e.g., `payment.succeeded`, `payment.failed`). Verify signature with the secret the dashboard provides.

- To support **adding new payment methods** without re-integration, use the Dashboard Checkout Builder to enable multiple methods, and make the frontend initialize the SDK session without hardcoding a single method.

- For the role-play demo: prepare screenshots of the Dashboard configuration (payment methods enabled, routing settings) and a recorded test transaction using the Testing Gateway.

Troubleshooting tips (for the assessment role-play)

- If embedded checkout doesn't load, check browser console for CORS or mixed-content problems.
- Verify the `checkout_url` returned by Yuno is an HTTPS URL and allowed to embed (some providers disallow iframe embedding — Yuno SDK FULL should provide an embeddable widget or JS-based method).
- If payment creation returns a 401/403, verify `YUNO_API_KEY` and merchant id, and ensure you're using the **Testing Gateway** base URL for test credentials.
