Table of Contents

Hier ist ein detaillierter Schritt-für-Schritt-Plan, um dein Projekt – eine Webseite, bei der der Benutzer Text eingibt und diesen als PDF herunterladen kann – mithilfe von serverseitiger PDF-Erstellung mit PDFKit umzusetzen:

---

1. Projektvorbereitung

- **Anforderungen definieren:**
  Lege fest, welche Funktionalitäten du bieten möchtest (z. B. einfacher Texteingabe-Bereich, PDF-Download, Fehlerbehandlung).
- **Verzeichnisstruktur erstellen:**
  Richte einen Projektordner ein und erstelle Unterordner für den Server-Code (z. B. `/src`), für statische Dateien wie HTML, CSS und clientseitige JavaScript-Dateien (z. B. `/public`).

- **Versionskontrolle:**
  Initialisiere ein Git-Repository, um deinen Code zu verwalten.

---

2. Einrichtung der Entwicklungsumgebung

- **Node.js-Projekt initialisieren:**
  Führe `npm init` aus, um eine `package.json` zu erstellen.
- **Abhängigkeiten installieren:**

  ```bash
  npm install express pdfkit body-parser
  ```

  Hinweis: Body-parser hilft beim Parsen von POST-Daten (ab Express 4.16+ kannst du auch `express.json()` nutzen).

---

3. Aufbau des Express-Servers

- **Server-Datei erstellen (z. B. `app.js`):**
  - Importiere Express und andere benötigte Module.
  - Richte Middleware ein (z. B. `express.json()` oder Body-parser).
  - Setze einen statischen Ordner (z. B. `/public`), damit HTML, CSS und JS vom Server ausgeliefert werden.
  - Erstelle eine Basisroute (z. B. `/`), die die Startseite liefert.
  - Starte den Server (z. B. auf Port 3000).

---

4. Frontend-Entwicklung

- **HTML-Seite erstellen:**
  Erstelle eine `index.html` in deinem `/public`-Ordner mit:
  - Einem **Textarea-Element** für die Texteingabe.
  - Einem **Button** oder **Formular**, das den Text an den Server sendet.

- **Clientseitiges JavaScript (optional):**
  Füge Skripte hinzu, um das Formular via AJAX (z. B. mit `fetch`) abzuschicken, sodass die Seite nicht neu geladen werden muss.
- **CSS:**
  Style die Seite nach Bedarf, damit die Benutzeroberfläche ansprechend und benutzerfreundlich ist.

---

5. Route zur PDF-Erstellung implementieren

- **POST-Route erstellen:**
  Richte in deiner Express-Anwendung eine Route (z. B. `/generate-pdf`) ein, die den vom Benutzer eingegebenen Text empfängt.
- **PDF-Erstellung mit PDFKit:**
  - Erstelle in der Route ein neues PDF-Dokument:

```js
const PDFDocument = require('pdfkit');
const doc = new PDFDocument();
```

  - Setze die HTTP-Header so, dass der Browser das PDF als Download erkennt:

```js
res.setHeader('Content-Disposition', 'attachment; filename="download.pdf"');
res.setHeader('Content-Type', 'application/pdf');
```

  - Pipe das PDF-Dokument in den Response-Stream:

```js
doc.pipe(res);
```

  - Füge den vom Benutzer übermittelten Text in das PDF ein:

```js
doc.fontSize(12).text(req.body.text, { width: 410, align: 'left' });
```

  - Schließe das Dokument ab:

```js

```

```
doc.end();
```

- **Beispiel einer kompletten Route:**

```js
app.post('/generate-pdf', (req, res) => {
  const { text } = req.body; // Text aus der Anfrage

  const doc = new PDFDocument();
  res.setHeader('Content-Disposition', 'attachment; filename="download.pdf"');
  res.setHeader('Content-Type', 'application/pdf');

  doc.pipe(res);
  doc.fontSize(12).text(text, 50, 50);
  doc.end();
});
```

---

6. Verbindung zwischen Frontend und Backend

- **Formularabsendung:**
  Sorge dafür, dass das HTML-Formular den Text an deine `/generate-pdf`-Route sendet. Das kann über einen klassischen Form-POST oder über eine AJAX-Anfrage erfolgen.
- **Fetch-Beispiel (AJAX):**

```js
document.getElementById('pdfForm').addEventListener('submit', function(e) {
  e.preventDefault();
  const text = document.getElementById('textInput').value;
  fetch('/generate-pdf', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ text })
  })
  .then(response => response.blob())
  .then(blob => {
    const url = window.URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = 'download.pdf';
    document.body.appendChild(a);
    a.click();
    a.remove();
  })
```

```
  .catch(err => console.error('Error:', err));
});
```

---

7. Testen und Debuggen

- **Lokal testen:**
  Starte deinen Server und teste alle Funktionalitäten im Browser.
- **Fehlerbehandlung:**
  Füge sowohl im Frontend als auch im Backend geeignete Fehlerbehandlungen hinzu, um etwaige Probleme zu identifizieren und abzufangen.

---

8. Deployment und Produktion

- **Deployment-Vorbereitung:**
  - Überprüfe, ob alle Umgebungsvariablen korrekt gesetzt sind.
  - Wähle einen Hosting-Dienst (z. B. Heroku, DigitalOcean, AWS) aus und konfiguriere den Server.
- **Sicherheit:**
  Stelle sicher, dass deine Anwendung gegen häufige Sicherheitsrisiken abgesichert ist (z. B. Cross-Site Scripting, unsichere Header).
- **Monitoring:**
  Implementiere Logging und überwache deine Anwendung, um im Produktionsbetrieb auf Fehler reagieren zu können.

Mit diesen Schritten solltest du in der Lage sein, dein Projekt von der Konzeptphase bis zum fertigen, deployten System umzusetzen. Viel Erfolg bei der Umsetzung!