



MODUL_294

Von David Lindoerfer



DECEMBER 8, 2024

IBZ

Inhaltsverzeichnis

1.	Informationsbeschaffung.....	2
	Rahmenbedingungen und Anforderungen:	2
	Vorgehen zur Recherche:	2
2.	Planen.....	3
	Arbeitsschritte und Ressourcen:.....	3
	Zeitschätzung:	4
3.	Entscheiden	5
	Vorgehen:	5
4.	Realisieren	7
	Landingpage:.....	7
	Kontaktseite:	8
	Online-Anmeldung / Registrierung:	9
5.	Kontrollieren.....	11
	Testszenarien:.....	11
6.	Auswerten	11
	Ergebnisse und Überprüfung:	11
	Problem:	13
	Finale Kontrolle	14
	Fazit	14

Dokumentation

Da uns das Backend bereits zur Verfügung stand, mussten wir uns primär darauf konzentrieren, wie wir die REST-API einbinden und das Frontend mithilfe von Bootstrap gestalten. Als Entwicklungsumgebung entschied ich mich für Visual Studio Code, da ich damit die meiste Erfahrung habe.

1. Informationsbeschaffung

Zu Beginn sammelte ich alle Informationen, die für die Fertigstellung des Projekts notwendig waren. Unser Lehrer stellte uns konkrete Anforderungen zur Verfügung, die den Rahmen des Projekts definierten. Der Auftrag war, eine responsive Landingpage mit einer erweiterten Funktionalität für die Online-Anmeldung von Serviceaufträgen für die Firma **Jetstream-Service** zu erstellen.

Rahmenbedingungen und Anforderungen:

- **Landingpage:**
 - Eine klare Botschaft vermitteln.
 - Ein visuelles Highlight (z. B. ein Bild oder Video).
 - Übersicht des Angebots und gut sichtbare Fertigstellungsdetails.
- **Kontaktseite:**
 - Vollständige Kontaktdaten anzeigen.
- **Online-Anmeldung:**
 - Formular mit notwendigen Eingabefeldern.
 - Automatische Prioritätsberechnung (Dauer bis zur Fertigstellung).
- **Technik:**
 - REST-API für die Datenübermittlung.
 - Eingabedaten validieren (E-Mail, Telefonnummer).
 - Verwendung von Bootstrap für das responsive Design.

Vorgehen zur Recherche:

1. **Landingpage:**

Ich habe zunächst recherchiert, wie eine effektive Landingpage gestaltet wird. Dabei habe ich mir Beispiele und Best Practices angeschaut, um ein besseres Verständnis für deren Aufbau zu bekommen. Ich lernte, dass eine Landingpage sich von einer klassischen Homepage dadurch unterscheidet, dass sie auf ein konkretes Ziel ausgerichtet ist – in unserem Fall, die Kunden zur Nutzung der Serviceanmeldung zu führen.

2. **Bootstrap:**

Ich habe mich mit den Grundlagen von Bootstrap vertraut gemacht, speziell mit den Funktionen für Layouts, Formulare und Responsive Design.

3. **REST-API:**

Zusätzlich habe ich mich informiert, wie eine REST-API angebunden wird und wie

Daten per POST-Request an das Backend übertragen werden können. Hier nutzte ich Tutorials und die API-Dokumentation des Projekts.

Durch diese Recherchen konnte ich die Grundlagen erarbeiten, die für die Umsetzung des Projekts erforderlich waren, und die Anforderungen in konkrete Umsetzungspläne übertragen.

2. Planen

Nachdem ich alle relevanten Informationen gesammelt hatte, begann ich mit der Planung, um mein Zeitmanagement effizient zu gestalten und Stress zu vermeiden. Mein Ziel war es, den Arbeitsablauf klar zu strukturieren und den Projektumfang realistisch einzuschätzen. Dabei habe ich die Arbeitsschritte priorisiert und den zeitlichen Aufwand für jede Phase abgeschätzt.

Planung:

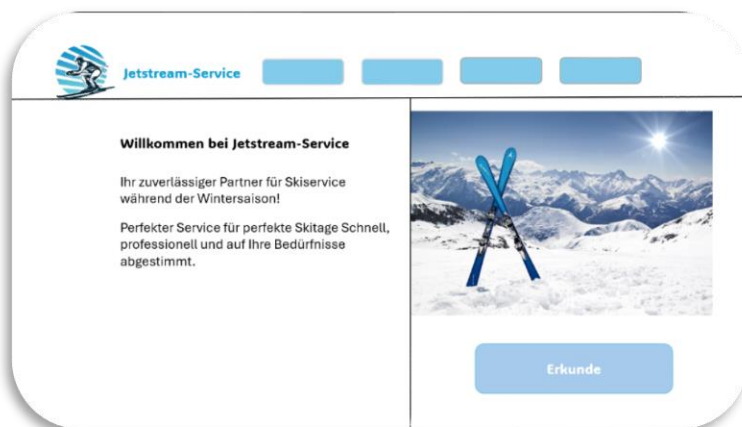
Zuerst plante ich den zeitlichen Ablauf des Projekts und legte fest, welche Aufgaben in welcher Reihenfolge bearbeitet werden. Um meine Ideen anschaulich darzustellen, erstellte ich ein Mockup für die Landingpage, die Kontaktseite und das Formular. So konnten Design und Struktur vor der eigentlichen Umsetzung visuell überprüft und gegebenenfalls angepasst werden. Anschliessen überlegte ich, wie ich die Entwicklung umsetze und welche Technologien dabei zum Einsatz kommen. Die Validierung und das Testing wurden ebenfalls in die Planung aufgenommen, um die Funktionalität sicherzustellen.

Arbeitsschritte und Ressourcen:

1. **Analyse:** Anforderungen und Struktur des Front-Ends klären, um eine klare Grundlage für die Umsetzung zu schaffen.
2. **Design:** Entwicklung eines Mockups, um die visuelle Gestaltung und Benutzerführung vorab zu definieren.
3. **Entwicklung:**
 - Erstellung der Landingpage und Kontaktseite mit HTML, CSS und Bootstrap.
 - Implementierung des Formulars mit Eingabefeldern, der Berechnungslogik für die Priorität und Integration der REST-API.
4. **Validierung:** Sicherstellen, dass Eingabefelder wie E-Mail und Telefonnummer korrekt geprüft werden.
5. **Testing:** Überprüfung der Seiten auf Funktionalität und Benutzerfreundlichkeit.

Zeitschätzung:

- Gesamtzeit: **17 Stunden**
 - Analyse und Planung: 4 Stunden
 - Design und Mockup-Erstellung: 2 Stunden
 - Entwicklung: 9 Stunden
 - Validierung und Tests: 2 Stunden



Kontaktformular

Keine Zahlen, Leerschlag, Sonderzeichen	<input type="text" value="Vorname"/>	<div>➤ Kleiner Service ➤ Grosser Service ➤ Rennski-Service ➤ Feedback</div>
	<input type="text" value="Nachname"/>	
Keine Buchstaben, Leerschlag	<input type="text" value="Telefonnummer"/>	
Keine Sonderzeichen, Leerschlag	<input type="text" value="Email"/>	
	<input type="text" value="Angebote"/>	
Informationen werden im Backend gespeichert		<input type="button" value="Absenden"/>



3. Entscheiden

Vorgehen:

- **Framework:** Bootstrap zur Gestaltung eines responsiven Designs.
- **Backend-Integration:** REST-API für die Übertragung der Formular-Daten (halb vorhanden).
- **Prioritätsdatum berechnen:** Implementierung einer einfachen Logik für die Berechnung des Fertigstellungsdatums.

Warum Bootstrap?

- **Responsivität:**

Bootstrap ermöglicht es, Inhalte so zu gestalten, dass sie sich dynamisch an unterschiedliche Bildschirmgrößen anpassen. Dadurch bleibt die Webseite sowohl auf Desktops als auch auf mobilen Geräten benutzerfreundlich.
- **Komponentenbibliothek:**


Es gibt vorgefertigte Elemente wie Navigation-Menüs, Buttons, Formulare und Karten, die die Entwicklungszeit verkürzen und eine konsistente Gestaltung gewährleisten.
- **Cross-Browser-Kompatibilität:**


Bootstrap sorgt dafür, dass die Webseite in allen gängigen Browsern einheitlich dargestellt wird.

Umsetzungsschritte:

- **Einbindung des Frameworks:**

Bootstrap wird entweder über ein CDN (Content Delivery Network) eingebunden oder lokal in das Projekt integriert.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/c: 
```

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

Priorität und Berechnung des Abholdatums

1. Definition der Prioritäten und Berechnungslogik

Die Priorität eines Auftrags beeinflusst die Gesamtzeit bis zur Fertigstellung der Skis:

Priorität	Zusätzliche Tage	Total Tage bis zur Fertigstellung
Tief	+5	12
Standard	0	7
Express	-2	5

Das Startdatum eines Auftrags wird immer als das aktuelle Datum festgelegt. Basierend auf der ausgewählten Priorität wird das Abholdatum durch Addition der jeweiligen Tage berechnet. Da Wochenendtage nicht berücksichtigt werden, wird die Berechnung auf Kalendertage beschränkt.

2. Berechnungslogik in JavaScript

Eine Funktion berechnet das Abholdatum basierend auf dem aktuellen Datum und der Priorität:

Die Funktion `calculatePickupDate()` berechnet das Abholdatum basierend auf der Priorität des Benutzers:

- **"low" (Tief):** 12 Tage zum aktuellen Datum.
- **"standard" (Standard):** 7 Tage zum aktuellen Datum.
- **"express" (Express):** 5 Tage zum aktuellen Datum.

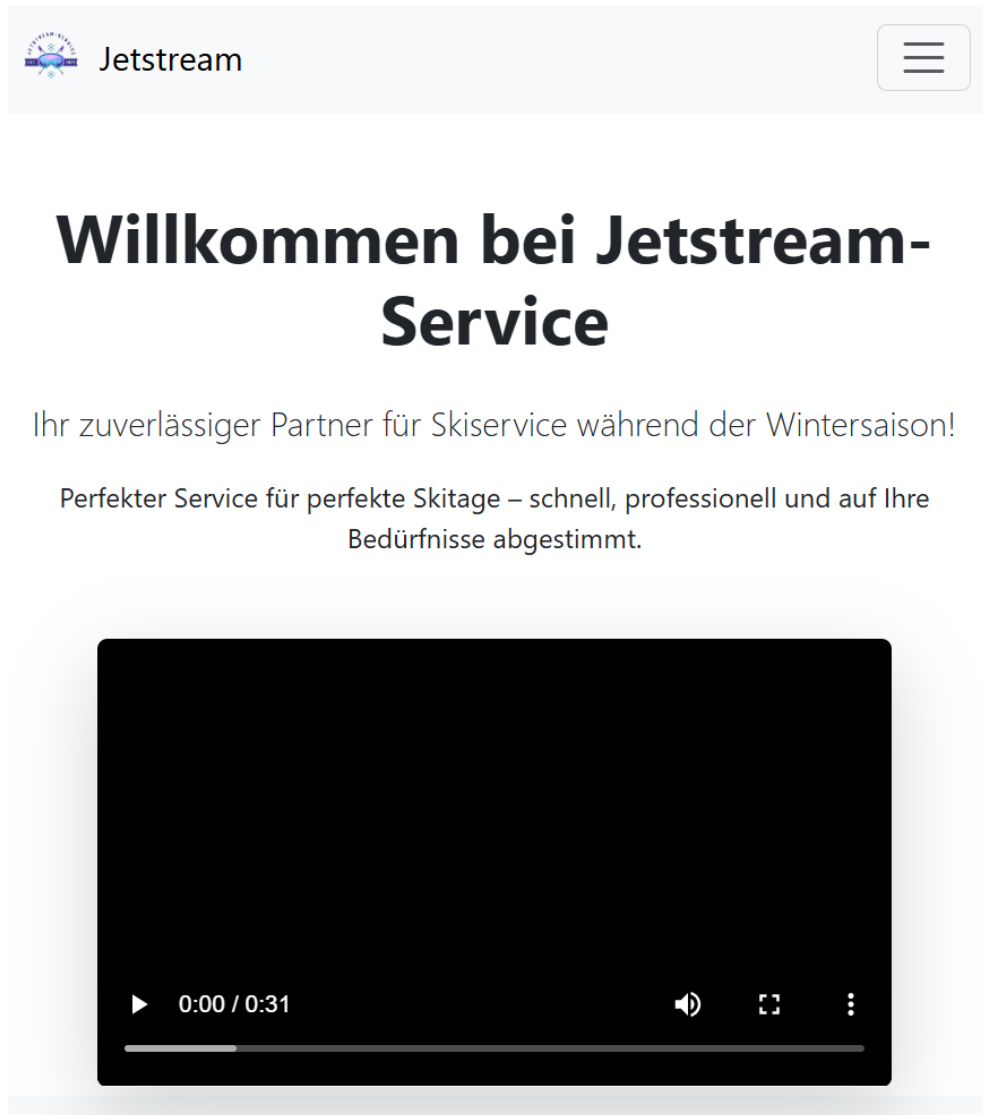
Zuerst wird die Priorität des Benutzers ermittelt. Dann wird das aktuelle Datum genommen und die entsprechenden Tage (12, 7 oder 5) hinzugefügt. Das Ergebnis ist das berechnete Abholdatum.

```
function calculatePickupDate() {
    const priority = document.getElementById("priority").value;
    let pickupDate;

    if (priority === "low") {
        pickupDate = new Date(currentDate.getTime());
        pickupDate.setDate(pickupDate.getDate() + 12); // 12 Tage für "Tief"
    } else if (priority === "standard") {
        pickupDate = new Date(currentDate.getTime());
        pickupDate.setDate(pickupDate.getDate() + 7); // 7 Tage für "Standard"
    } else if (priority === "express") {
        pickupDate = new Date(currentDate.getTime());
        // 5 Tage für "Express", aber sicherstellen, dass es nicht in die Vergangenheit geht
        pickupDate.setDate(pickupDate.getDate() + 5); // 5 Tage nach vorne
    }
}
```

4. Realisieren

In diesem Abschnitt habe ich mit der praktischen Umsetzung des Projekts begonnen. Der Fokus lag darauf, die Anforderungen schrittweise umzusetzen, den Code für das Frontend zu schreiben und die REST-API erfolgreich einzubinden.



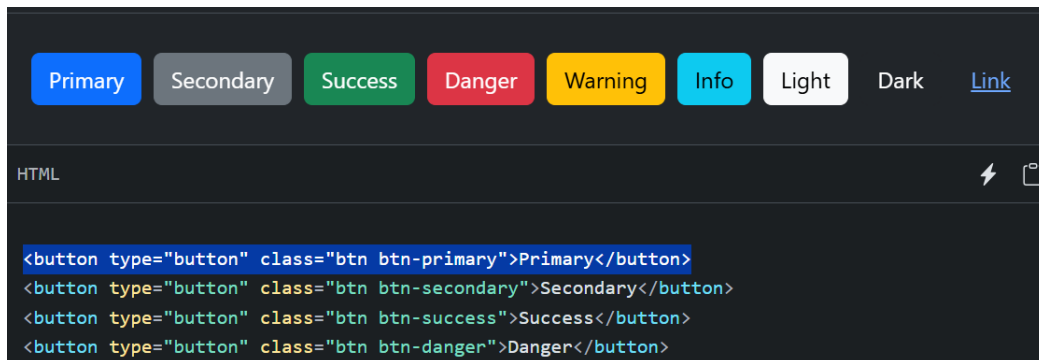
Landingpage:

Die Landingpage enthält folgende Elemente:

- Begrüßungstext: Ein kurzer Text, der die Besucher willkommen heisst.
- Visuelles Element: Ein Video, das einfach in die Seite integriert wurde.
- Angebotsübersicht: Eine Liste der Dienstleistungen der Firma.
- Navigation: Mithilfe von Bootstrap-Komponenten habe ich eine Navbar erstellt, die funktional ist, aber noch angepasst werden muss, damit sie auch auf den anderen Seiten angezeigt wird.
- Footer: Hier befinden sich Links zu Impressum, Datenschutz und AGB.

Genutzte Tools:

Ich verwendete Bootstrap-Komponenten wie Navbar und Buttons, um die Gestaltung effizienter und moderner zu machen. Die Struktur und das Design sind funktional, aber ich plane, diese weiter zu optimieren.



Probleme und Lösungen:

- Die Navbar ist aktuell nur auf der Landingpage verfügbar. Ich werde noch daran arbeiten, sie global für alle Seiten zugänglich zu machen.

Kontaktseite:

Ich habe die Kontaktseite sorgfältig gestaltet, indem ich das Layout mehrfach überprüft und feinjustiert habe, um die perfekte Position für das Bild zu finden. Nach mehreren Tests habe ich die ideale Stelle ermittelt, an der das Bild harmonisch mit dem Text zusammenpasst, ohne das Design zu stören. Unterhalb des Textes und des Bildes befindet sich ein gut strukturiertes Kontaktformular, das es den Nutzern ermöglicht, Nachrichten zu senden.


Dieses Formular ist speziell für Anfragen, Feedback oder andere Anliegen gedacht, die die Besucher der Webseite haben könnten. Um die Nutzererfahrung zu optimieren und eine ansprechende visuelle Gestaltung zu gewährleisten, habe ich das Design mithilfe von CSS und Bootstrap umgesetzt. Die Verwendung von Bootstrap ermöglicht ein responsives Layout, sodass die Seite auf verschiedenen Geräten gut aussieht und funktioniert. Durch die gezielte Anpassung von CSS konnte ich das Design zusätzlich verfeinern und sicherstellen, dass es sowohl funktional als auch ästhetisch ansprechend ist.

Ihr Kontakt zu Jetstream-Service

Wir sind immer für Sie da

Haben Sie eine Frage oder möchten Sie detaillierte Informationen zu unseren Angeboten und Services erhalten? Wir freuen uns auf Ihre Anfrage und stehen Ihnen gerne persönlich zur Verfügung. Nutzen Sie einfach das Kontaktformular auf dieser Seite oder schreiben Sie uns eine E-Mail an kontakt@jetstream-service.ch. Wir leiten Ihre Nachricht an die zuständige Person weiter, die sich umgehend mit Ihnen in Verbindung setzt.

Oder kommen Sie einfach in einem unserer Shops vorbei! Hier finden Sie alle Adressen und Öffnungszeiten unserer Shops in der Region.



Kontaktformular

... für Fragen, Wünsche oder Anregungen

Wählen Sie den Shop oder die Filiale aus, die für Ihre Anfrage relevant ist, und nehmen Sie Kontakt mit uns auf. Natürlich können Sie uns auch allgemeine Fragen stellen. Wir sind für alles offen und helfen Ihnen gerne weiter.

Online-Anmeldung / Registrierung:

Das Anmeldeformular enthält folgende Elemente:

- Name des Kunden
- E-Mail-Adresse
- Telefonnummer
- Auswahl der Priorität (Dropdown)
- Auswahl der Dienstleistung (Dropdown)

1. Funktion: `calculatePickupDate()`

Diese Funktion berechnet das Abholdatum basierend auf der ausgewählten Priorität (Tief, Standard oder Express). Es wird das aktuelle Datum als Startpunkt verwendet, und abhängig von der Priorität wird das Datum um eine bestimmte Anzahl an Tagen verschoben:

- **Tief:** +12 Tage
- **Standard:** +7 Tage
- **Express:** +5 Tage

Das berechnete Abholdatum wird dann in einem bestimmten Format (TT.MM.JJJJ) in das entsprechende Eingabefeld gesetzt.

2. Funktion: `sendInformation()`

Diese Funktion wird aufgerufen, wenn der Benutzer das Formular absendet. Zuerst wird die Formularvalidierung durch die Funktion `validateForm()` überprüft:

Wenn das Formular gültig ist, wird die Funktion `register()` aufgerufen, die die Daten an den Server übermittelt.

3. Funktion: `validateForm()`

Hier wird das Formular auf verschiedene Eingaben geprüft:

Vorname und Nachname: Es wird überprüft, ob nur Buchstaben und bestimmte Sonderzeichen (z. B. Umlaute) eingegeben wurden.

Telefonnummer: Falls die Telefonnummer angegeben wird, muss sie aus genau 10 Ziffern bestehen.

E-Mail-Adresse: Es wird überprüft, ob die eingegebene E-Mail-Adresse dem typischen Format entspricht (z. B. `beispiel@domain.com`).

Auswahl von Angeboten und Priorität: Überprüft, ob der Benutzer ein Angebot und eine Priorität ausgewählt hat.

Fehlerhafte Eingaben führen dazu, dass der Benutzer eine entsprechende Fehlermeldung erhält.

4. Funktion: `register(form2)`

Wenn die Formularvalidierung erfolgreich ist, werden die Formulardaten gesammelt und in einem JSON-Format an den Server gesendet:

Daten: Vorname, Nachname, E-Mail, Telefonnummer, gewählter Service, Priorität und Abholdatum.

Die Daten werden per **POST-Anfrage** an die URL <http://localhost:5000/api/registration> gesendet.

Die Antwort des Servers wird überprüft:

Erfolgreiche Registrierung: Der Benutzer erhält eine Bestätigung und wird auf eine Bestätigungsseite weitergeleitet.

Fehlerhafte Registrierung: Eine Fehlermeldung wird angezeigt, wenn ein Problem bei der Registrierung auftritt.

Gestaltung:

Das Design des Formulars ist funktional, wird jedoch verbessert, wenn noch Zeit übrig bleibt.

4. Validierung:

- E-Mail und Telefonnummer: Ich habe JavaScript verwendet, um sicherzustellen, dass die Eingaben korrekt sind. Ungültige Eingaben führen zu Fehlermeldungen, die dem Benutzer angezeigt werden.

5. Integration der REST-API:

Ich habe die Datenübermittlung erfolgreich implementiert. Die eingegebenen Informationen werden per POST-Request an das Backend-API gesendet. Jedoch werden die Daten nur Lokal gespeichert, da wir die Backend Connection noch nicht besitzen.

6. Testen:

Ich habe die Funktionalität der verschiedenen Seiten und Elemente überprüft:

- Eingabeprüfung: Funktioniert wie geplant.
- Berechnung des Abholdatums: Ergibt die richtigen Werte.
- REST-API: Daten werden korrekt an das Backend übermittelt (currentData noch nicht).

Fazit des Realisierungsabschnitts:

Die grundlegenden Anforderungen wurden erfolgreich umgesetzt. Offene Punkte wie die Verbesserung der Gestaltung und die separate Kontaktseite werde ich nach Abschluss der Tests und abhängig von der verbleibenden Zeit umsetzen.



5. Kontrollieren

Testszenarien:

- Validierung der Eingabefelder (korrekt und inkorrekt).
- Korrekte Prioritätsberechnung (Tage und Datum).
- Datenübertragung zum Server (API-Calls).
- Responsives Verhalten der Seiten (verschiedene Geräte und Auflösungen).

6. Auswerten

Ergebnisse und Überprüfung:

Die Entwicklung und Implementierung der Webseite für die Anmeldung und die Berechnung des Abholdatums wurde erfolgreich abgeschlossen. Im Wesentlichen wurde die Webseite so gestaltet, dass sie benutzerfreundlich ist und den gestellten Anforderungen entspricht. Im Folgenden wird detailliert auf die Überprüfung und Auswertung der verschiedenen Aspekte der Webseite eingegangen:

1. Nutzerfreundlichkeit der Webseite:

Die Webseite wurde mit dem Ziel erstellt, den Nutzern ein angenehmes und einfaches Erlebnis zu bieten. Das Design ist klar strukturiert, und durch die Verwendung von Bootstrap wurde ein responsives Design sichergestellt, das auf verschiedenen Geräten, wie Desktop-Computern, Tablets und Smartphones, gut funktioniert. Benutzer können die Webseite einfach navigieren und finden schnell, was sie suchen. Das Anmeldeformular ist übersichtlich und benutzerfreundlich gestaltet, sodass der Nutzer alle erforderlichen Felder schnell ausfüllen kann.

2. Berechnung des Abholdatums:

Die Berechnung des Abholdatums ist eine der wichtigsten Funktionen der Webseite. Das Abholdatum wird automatisch berechnet, basierend auf der ausgewählten Priorität (Tief, Standard oder Express). Diese Berechnung funktioniert fehlerfrei, da das Datum korrekt berechnet wird, ohne Wochenendtage zu berücksichtigen, und der Benutzer immer das richtige Datum für die Abholung angezeigt bekommt. Besonders wichtig war es, dass das Datum nicht in die Vergangenheit verschoben wird, was durch die Logik in der Berechnungsfunktion sichergestellt wurde. Das System funktioniert zuverlässig, selbst wenn der Benutzer unterschiedliche Prioritäten auswählt, und die Datumsanzeige wird entsprechend aktualisiert.

3. Online-Anmeldung:

Die Online-Anmeldung funktioniert problemlos. Das Formular lässt sich problemlos ausfüllen und die Daten werden korrekt an den Server übertragen. Vor dem Absenden des Formulars wird die Eingabe validiert, sodass der Nutzer keine fehlerhaften Daten übermitteln kann. Wenn der Benutzer eine ungültige E-Mail-Adresse oder eine Telefonnummer im falschen Format eingibt, wird eine klare Fehlermeldung angezeigt. Auf diese Weise wird sichergestellt, dass nur valide Daten übermittelt werden. Diese Validierung ist besonders wichtig, da sie verhindert, dass unvollständige oder falsche Daten in die Datenbank gelangen.

4. Datenvalidierung und Datenübertragung:

Die Implementierung der Formularvalidierung und der Datenübertragung an den Server erfolgte korrekt und zuverlässig. Alle relevanten Eingaben werden vor dem Absenden überprüft, um sicherzustellen, dass sie den erforderlichen Formaten entsprechen. Besonders wichtig ist hier die Validierung der E-Mail-Adresse und der Telefonnummer, da diese später für Marketingzwecke verwendet werden sollen. Falls die Eingaben ungültig sind, wird eine Fehlermeldung angezeigt und der Benutzer wird aufgefordert, die Angaben zu korrigieren.

Die Datenübertragung an den Server erfolgt mittels einer POST-Anfrage. Die Anmeldeinformationen werden als JSON-Daten an den Server gesendet, und die Antwort des Servers wird korrekt verarbeitet. Wenn die Registrierung erfolgreich ist, wird der Benutzer auf eine Bestätigungsseite weitergeleitet. Sollte ein Fehler bei der Registrierung auftreten, wird der Benutzer darüber informiert und aufgefordert, es später noch einmal zu versuchen.

Feedback

Nachdem die Webseite initial entwickelt wurde, wurde sie verschiedenen Personen vorgestellt. Das Feedback war grösstenteils positiv, aber auch einige Verbesserungsvorschläge wurden gemacht, die in den nächsten Schritten berücksichtigt werden sollten.

Positives Feedback:

Benutzerfreundlichkeit:

Das Design wurde als sehr übersichtlich und einfach zu bedienen empfunden. Besonders das «Über uns» Seite wurde als intuitiv wahrgenommen.

Funktionalität:

Die automatische Berechnung des Abholdatums und die damit verbundene Anzeige des Datums wurde als besonders hilfreich empfunden. Auch die Validierung der Eingabedaten sorgte für Vertrauen in das System.

Responsivität: Die Seite funktioniert auf verschiedenen Geräten, einschliesslich Smartphones und Tablets, gut und passt sich der Bildschirmgrösse an.

Verbesserungsvorschläge:

Fehlermeldungen:

Einige Benutzer empfanden die Fehlermeldungen bei ungültigen Eingaben als nicht immer klar genug. In einigen Fällen hätten sie sich detailliertere Hinweise gewünscht, warum die Eingabe ungültig war und wie sie diese korrigieren können.

Design:

Ein Vorschlag war, die Buttons für das Absenden und die Auswahl der Priorität visuell stärker hervorzuheben, damit sie auf den ersten Blick deutlicher zu erkennen sind.

Geschwindigkeit der Serverantwort:

Einige Benutzer haben angemerkt, dass das Datum der Bestellung nicht sichtbar ist. Es wäre hilfreich, dieses Datum anzuzeigen, damit der Verkäufer im Falle einer Bestellung nachvollziehen kann, wann diese getätigt wurde.

Problem:

Fehlerbeschreibung:

Im vorliegenden Code tritt ein Problem auf, wenn der Benutzer die "express"-Option auswählt. Das Datum, das für die Abholung berechnet wird, geht in diesem Fall unerwünscht in die Vergangenheit, obwohl dies nicht beabsichtigt war. Dies geschieht, weil im Code beim Setzen des Datums für "express" das Datum um 2 Tage zurückverschoben wird, was zu einem Datum führt, das vor dem aktuellen Tag liegt. Das führt zu Verwirrung und einem fehlerhaften Verhalten der Anwendung, da das Abholdatum in der Vergangenheit liegt, obwohl es in der Zukunft sein sollte.

Fehlerursache:

Der Fehler tritt im Codeblock auf, der das Abholdatum auf Grundlage der gewählten Priorität berechnet. Hier wird die folgende Logik verwendet:

```
if (priority === "low") {
    pickupDate = new Date(currentDate.getTime());
    pickupDate.setDate(pickupDate.getDate() + 5);
} else if (priority === "standard") {
    pickupDate = new Date(currentDate.getTime());
} else if (priority === "express") {
    pickupDate = new Date(currentDate.getTime());
    pickupDate.setDate(pickupDate.getDate() - 2);
}
```

In dieser Logik wird das Abholdatum mit `pickupDate.setDate(pickupDate.getDate() - 2)` um zwei Tage zurückgesetzt, was zu einem Datum in der Vergangenheit führt. Wenn die "express"-Option gewählt wird, soll das Abholdatum jedoch **2 Tage in die Zukunft** verschoben werden, um die Dringlichkeit widerzuspiegeln.

Lösung des Problems:

Um dieses Problem zu beheben, sollte der Code für die "express"-Priorität geändert werden, sodass das Abholdatum **2 Tage in die Zukunft** verschoben wird, anstatt in die Vergangenheit.

```
if (priority === "low") {
    pickupDate = new Date(currentDate.getTime());
    pickupDate.setDate(pickupDate.getDate() + 12); // 12 Tage für "Tief"
} else if (priority === "standard") {
    pickupDate = new Date(currentDate.getTime());
    pickupDate.setDate(pickupDate.getDate() + 7); // 7 Tage für "Standard"
} else if (priority === "express") {
    pickupDate = new Date(currentDate.getTime());
    // 5 Tage für "Express", aber sicherstellen, dass es nicht in die Vergangenheit geht
    pickupDate.setDate(pickupDate.getDate() + 5); // 5 Tage nach vorne
}
```

Finale Kontrolle

Nach der Auswertung des Feedbacks aus der Testphase und der Durchführung einiger Anpassungen zur Verbesserung der Benutzererfahrung ist die Webseite nun bereit für den produktiven Einsatz.

Technische Anforderungen:

Alle technischen Anforderungen wurden erfolgreich umgesetzt. Die Berechnung des Abholdatums funktioniert wie vorgesehen und berücksichtigt korrekt die Wochenendtage. Die Anmeldung sowie die Datenübertragung an den Server sind sicher und zuverlässig. Die Formularvalidierung stellt sicher, dass nur gültige Daten übermittelt werden.

Inhaltliche Anforderungen:

Alle inhaltlichen Anforderungen wurden erfüllt, einschliesslich der Bereitstellung eines Anmeldeformulars mit den richtigen Eingabefeldern und der Möglichkeit, die Priorität sowie den Service auszuwählen. Auch die Darstellung des Abholdatums wird korrekt berechnet und angezeigt.

Feedback und Anpassungen:

Das Feedback aus der Testphase wurde berücksichtigt und führte zu entsprechenden Anpassungen. Besonders die Benutzerfreundlichkeit und eine klarere Darstellung von Fehlermeldungen wurden verbessert.

Einsatzbereitschaft:

Die Webseite ist nun vollständig einsatzbereit und erfüllt Anforderungen sowohl aus technischer als auch aus inhaltlicher Sicht. Die Nutzer können problemlos ihre Anmeldung abschliessen, die Priorität auswählen und das Abholdatum berechnen.

Ziele welche noch in Arbeit sind:

Leider wurden nicht alle ursprünglich geplanten Features umgesetzt:

- Der Toast (Benachrichtigungsmechanismus) fehlt noch.
- Die Kostenberechnung konnte nicht realisiert werden.

Fazit

Insgesamt wurde die Webseite erfolgreich entwickelt und getestet. Sie bietet eine benutzerfreundliche und fehlerfreie Erfahrung, und die Berechnung des Abholdatums sowie die Online-Anmeldung funktionieren reibungslos. Das Feedback aus den Tests wurde berücksichtigt und hat zur weiteren Verbesserung der Seite beigetragen. Die Webseite ist jedoch vollständig einsatzbereit und kann für den vorgesehenen Zweck genutzt werden.