

# SE: Problem Statement

## Bright Impact: Shine to Mine

### Inhaltsangabe

<b>1. Einführung</b>	<b>2</b>
1.1 Projekt Thema	2
<b>2. Scenarios</b>	<b>2</b>
<b>3. Use-Cases</b>	<b>3</b>
<b>4. Software-Komponenten</b>	<b>8</b>
4.1. Nutzerseitige Applikationen	8
4.1 Donation Box	8
4.2 Client App	8
4.3 Vereins-Verwaltungs Website	8
(4.4 Job Register Overview)	8
(4.5 Public Overview)	8
4.2 Serverseitige Applikation (Mainframe)	8
4.3 Datenbank	9
<b>5. Requirements</b>	<b>9</b>
5.1 Functional Requirements (FR)	9
5.2 Non-functional Requirements (NFRs)	11
<b>6. Target Environment (Vorschläge)</b>	<b>11</b>
<b>7. Deliverables und Abnahmekriterien</b>	<b>12</b>
7.1 Deliverables	12
7.2 Abnahmekriterien	12
<b>8. Appendix</b>	<b>12</b>
8.1 LF Tooling	12
8.2 Glossar	12
8.3 Quellen	14

### Dokument History

Datum	Author(en)	Änderungen
08.11.2024	Team Orange	Erste Iteration über das Problem, Erstellung von Einführung und Scenarios
15.11.2024	Team Orange	Erstellen der ersten Use-Cases und FR/NFRs
19/20.11.2024	Samuel Sacher, Simon Okutan	Zweite Iteration über die Use-Cases, Finalisierung der FRs/NFRs und Definition der Deliverables und Abnahmekriterien

# 1. Einführung

## 1.1 Projekt Thema

Bright Impact will Menschen mit Solaranlagen motivieren, ihren überschüssigen Strom für einen guten Zweck zu spenden und so das Stromnetz zu entlasten. Dies geschieht durch die Bereitstellung von Rechenkapazität über geliehene Hardware, die mit dem überschüssigen Strom betrieben wird. Die generierte Rechenleistung wird anschließend für gemeinnützige Projekte und Organisationen genutzt, z. B. für medizinische Forschung, Klimasimulationen oder Bildungstechnologien.

## 1.2 Das Problem

- **Unterschwelliges Spenden – Engagement ohne emotionale Bindung:** Viele Menschen spenden ungern, da sie die Wirkung ihres Beitrags nicht sehen oder sich emotional nicht verbunden fühlen. Unterschwellige Spendenformen bleiben oft unbemerkt und haben daher wenig motivierende Wirkung.
- **Überlastung des Stromnetzes bei hoher Solarproduktion:** An sonnigen Tagen produzieren Solaranlagen mehr Strom, als Haushalte und das Netz aufnehmen können. Dieser überschüssige Strom bleibt oft ungenutzt, obwohl er sinnvoll eingesetzt werden könnte.
- **Fehlende Verbindung zwischen Energieüberschuss und sozialem Nutzen:** Aktuelle Einspeisemodelle bieten keine sichtbare Verbindung zwischen überschüssigem Strom und dessen gesellschaftlichem Mehrwert. Ohne messbaren Impact fehlt es an Motivation, den Strom aktiv zu spenden.

## 2. Szenarios

Szenarien erfassen das System aus der Außenperspektive, zum Beispiel aus Sicht eines Nutzers anhand spezifischer Beispiele. Sie vermitteln unseren Entwicklern ein Verständnis davon, wie die spezifische Interaktion zwischen Nutzer und System in bestimmten Fällen aussehen sollte.

### **Szenario: Privathaushalt mit Photovoltaik Anlage ohne Speicher**

*Familie Müller möchte ihren nicht-genutzten Strom an eine ausgewählte Hilfsorganisation spenden, statt ihn ins Netz einzuspeisen. Dies soll möglichst unkompliziert geschehen, es ist ihnen jedoch wichtig nachvollziehen zu können, was mit dem gespendeten Geld passiert.*

- Die App soll die Auswahl von verschiedenen Projekten zum Spenden erlauben
- Eine Übersicht bisher generierter Spenden ist erforderlich
- Eine transparente Aufschlüsselung für bereits gespendete Summen und deren Verwendung wird benötigt

### **Szenario: Firma will ihren überflüssigen Solarstrom für gute Zwecke spenden**

*Wegen der Photovoltaikpflicht musste die Firma X Photovoltaik auf ihrem neuen Gebäude bauen. Da ihre Firma viel analog arbeitet, kann sie den Solarstrom nicht vollständig aufbrauchen. Sie will den Strom nicht nur ins Netz einspeisen, sondern für einen guten Zweck spenden. Dies soll auch für Marketingzwecke nutzbar sein.*

- Der Miner soll nur laufen, wenn man überschüssigen Solarstrom hat
- Es gibt eine API Schnittstelle für die Daten

### **Szenario: Das Rentner-Ehepaar Meier hat Bedenken, dass das alles zu kompliziert ist.**

*Frau Meier freut sich, ihren überschüssigen Strom spenden zu können. Sie sind jedoch nicht sehr technisch versiert. Sie haben jedoch Bedenken, dass Ihnen das alles zu kompliziert ist. Sie wollen nichts selbst einstellen, außer bei der Bestellung anzugeben wo das Geld hingespendet wird. Außerdem sind sie zurückhaltend, wenn sie Begriffe wie "Krypto" und "Mining" hören.*

- Das Produkt soll so einfach wie möglich sein. Es muss möglich sein, dass das Ehepaar Meier **nichts** tun muss, als eine Box in eine Steckdose zu stecken.
- Es wäre gut, am Ende des Jahres einen Spendenbericht per Post/E-Mail zu erhalten.
- Die Kommunikation über das Produkt muss seriös und transparent sein, dass den Prozess auch für Laien verständlich erklärt.

### **Szenario: Staat muss hohe EEG Kosten tragen**

*Der Staat möchte die Einspeisung von Strom reduzieren, da die Stromkosten unter den im EEG festgelegten Vergütungssätzen liegen. Da der bürokratische Aufwand für eine zentrale Steuerung hoch wäre, soll die Reduzierung vorzugsweise direkt in den Haushalten erfolgen. Gleichzeitig sollen die Gesamtergebnisse öffentlich und transparent einsehbar sein, ohne dass einzelne Haushalte detailliert kontrolliert werden müssen.*

- Das System muss alle Komponenten überwachen und Ergebnisse aggregieren können
- Das System sollte dahingehend öffentlich zugänglich sein

#### **Szenario: Student Ferdinand hat ein Balkonkraftwerke**

*Ferdinand hat sich vor einiger Zeit ein Balkonkraftwerk zugelegt, um nachhaltigen Strom zu beziehen. Da er tagsüber meist nicht daheim ist, kann der Strom von der Solarzelle nicht verwendet werden. Einspeisen geht bei dem Balkonkraftwerk auch nicht. Der Strom verfällt einfach. Er hört von einem Produkt, mit dem man überschüssigen Öko-Strom für einen guten Zweck spenden kann. Da er ein sehr sozialer Mensch ist, legt er sich direkt dieses neue Produkt zu.*

### **3. Use-Cases**

Ein Use Case (Anwendungsfall) ist eine Beschreibung, wie Anwender ein System nutzen, um eine Aufgabe zu erledigen, die ein definiertes Ergebnis (das Ziel) hat. Ein wesentlicher Punkt ist die Formulierung aus Anwendersicht.

#### **1-Use-Case: Ein potentieller Nutzer möchte sich informieren**

**Use Case Name:** Ein potentieller Nutzer möchte sich informieren

**Level:** Primärer use case

**Primärer Akteur:** Potentieller Nutzer (Job-Käufer, Verein, Spender)

**Stakeholders:** Bright Impact: Möchte, dass die Plattform verwendet wird.

**Vorbedingungen:** -

**Nachbedingungen:** -

**Auslöser:** Ein potentieller Nutzer hat (z.B. von bekannten) von Bright Impact gehört und möchte sich informieren.

**Haupterfolgsszenario:**

1. Der Nutzer kommt auf die Website
2. Das System liefert die Seite inklusive aktueller Statistiken aus
3. Die Landing-Page zeigt alles, inklusive aktueller Statistiken (z.B: wieviel wurde insgesamt gespendet), schnell, übersichtlich und ansprechend an.
4. Sollte der potentielle Nutzer sich registrieren wollen, wird korrekt auf die entsprechenden Registrierungsseiten weitergeleitet

**Erweiterungen:** 1) Ein Nutzer ist interessiert, möchte aber persönlich beraten werden. Er kann über ein Kontaktformular Kontakt zum Bright Impact Support aufnehmen, der ihn unterstützt.

**Spezielle Anforderungen:** -

**Technologie:** -

#### **2-Use-Case: Registrierung eines Spenders**

**Use Case Name:** Registrierung eines Spenders

**Level:** User-Goal, primärer Use Case

**Primärer Akteur:** (Neuer) Spender

**Stakeholders:**

1. BrightImpact: Will neue Spender gewinnen
2. Spender: Will sich unkompliziert registrieren

**Vorbedingungen:** -

**Nachbedingungen:**

1. Account des Spenders ist im System angelegt
2. Die Bestellung der Box ist gespeichert

**Auslöser:** Der Spender hat sich über Bright Impact informiert und will nun ebenfalls spenden.

**Haupterfolgsszenario:**

1. Der Spender klickt auf registrieren
2. Das System liefert Registrierungsseite mit E-Mail/Passwort aus
3. Der Spender gibt beides ein und klickt auf "Jetzt Registrieren".

4. Das System speichert die Daten (sicher) und verschickt eine E-Mail, in der der Spender seine Mail-Adresse innerhalb der nächsten 48 Stunden bestätigen muss.
5. Der Spender gibt auf der nächsten Seite seine persönlichen Informationen ein
6. Das System speichert die persönlichen Daten des Nutzers
7. Der Spender trägt die technischen Daten seiner Solaranlage ein
8. Das System speichert die technischen Daten der Solaranlage
9. Das System erstellt intern den passenden Auftrag zur Verschickung der Donation Box
10. Der Spender erhält eine Bestätigungsmail zur erfolgreichen Registrierung inkl. Informationen zum Versand.

**Erweiterungen:**

- 1) Der Spender braucht Hilfe, z.B. weil er keine Informationen über seine Solaranlagen und klickt auf "Support"
  1. Das System zeigt eine Hotline-Nummer an, die der Spender anruft. Dazu gibt es darunter ein Kontaktformular.
  2. Der Spender kann auch das Formular ausfüllen.
  3. Die Anfrage wird im System gespeichert und eine Bestätigungsmail an den Spender verschickt.
- 3) Die E-Mail Adresse existiert nicht.
  1. Das System löscht alle bereits gespeicherten Daten zum Spender
- 10) Die Solaranlage des Spender wird noch nicht unterstützt. Er wählt:
  - a. Die Box über Wetterdaten steuern zu lassen
  - b. Informiert zu werden, sobald seine Anlage unterstützt wird

**Spezielle Anforderungen:** -

**Technologie:** -

**Häufigkeit:** Regelmäßig

### 3-Use-Case: Registrierung eines Vereins

**Use Case Name:** Registrierung eines Sponsors

**Level:** User-Goal, primärer Use Case

**Primärer Akteur:** Verein (bzw. dessen Vorstand und Mitglieder)

**Stakeholders:**

- Vereinsvorstand: Will neue Spenden generieren, hält bright impact für seriös
- Vereinsmitarbeiter: Wollen die Anfrage des Vorstands möglichst leicht umsetzen und den Verein registrieren

**Vorbedingungen:** -

**Nachbedingungen:**

1. Der Vereinsaccount ist im System angelegt
2. Der Gemeinnützigkeitsnachweis ist mit dem Verein verbunden, und liegt in der Pipeline um bestätigt zu werden.

**Auslöser:** Verein überlegt sich neue Möglichkeit, spenden einzunehmen; hat sich über Bright Impact informiert und will sich registrieren.

**Haupterfolgsszenario:**

1. Der Mitarbeiter klickt auf registrieren
2. Das System liefert Registrierungsseite mit E-Mail/Passwort aus
3. Der Mitarbeiter gibt beides ein, und klickt auf "Jetzt Registrieren"
4. Das System speichert die Daten (sicher) und verschickt eine E-Mail, in der der Mitarbeiter seine Mail-Adresse innerhalb der nächsten 48 Stunden bestätigen muss
5. Der Mitarbeiter trägt auf der nächsten Seite die Informationen zum Verein ein
6. Das System speichert die persönlichen Daten des Vereins und leitet zur Ident-Seite weiter
7. Der Verein lädt den Nachweis seiner Gemeinnützigkeit hoch
8. Das System bestätigt den Empfang und speichert den Nachweis und versendet eine Bestätigungsmail an den Mitarbeiter. Das Ident-Verfahren wird im Hintergrund eingeleitet.

**Erweiterungen:**

- 1) Der Mitarbeiter braucht Hilfe und klickt auf "Support"
  1. Das System zeigt eine Hotline-Nummer an, die der Spender anruft. Dazu gibt es darunter ein Kontaktformular.

2. Der Mitarbeiter kann auch das Formular ausfüllen.
  3. Die Anfrage wird im System gespeichert und eine Bestätigungsmail an den Mitarbeiter verschickt.
- 3) Die E-Mail Adresse existiert nicht
1. Das System löscht alle bereits gespeicherten Daten zum Mitarbeiter

**Spezielle Anforderungen:** -

**Technologie:** -

**Häufigkeit:** Regelmäßig

#### **4-Use-Case: Anlegen eines Projekts**

**Use Case Name:** Anlegen eines Projekts

**Level:** User-Goal, primärer Use Case

**Primärer Akteur:** Vereinsmitarbeiter

**Stakeholders:**

- Vereinsvorstand: Will, dass alle Projekte des Systems möglichst gut präsentiert werden, um maximale Spenden zu generieren.
- Vereinsmitarbeiter: Wollen die Anfrage des Vorstands möglichst leicht umsetzen.

**Vorbedingungen:**

1. Der Verein hat einen Benutzeraccount angelegt
2. Der Verein wurde erfolgreich verifiziert

**Nachbedingungen:**

1. Die Projekte sind im System angelegt und mit dem Verein verbunden

**Auslöser:**

Option 1: Der Verein erhält die Mail, dass er erfolgreich verifiziert wurde. Der zuständige Mitarbeiter will nun Projekte eintragen.

Option 2: Der Verein beginnt ein neues Projekt und möchte dafür Spenden sammeln.

**Haupterfolgsszenario:**

1. Der Mitarbeiter loggt sich auf der Organisationswebsite ein.
2. Das System bestätigt die Anmeldung und liefert die Übersichtsseite aus.
3. Der Mitarbeiter öffnet die Seite zum Anlegen eines neuen Projektes.
4. Das System liefert ein Formular mit allen Pflichtfeldern aus.
5. Der Mitarbeiter trägt einen Projekttitel und eine Projektbeschreibung ein.
6. Der Mitarbeiter trägt das Spendenziel und den Zeitraum ein.
7. Der Mitarbeiter lädt ein Header-Bild hoch.
8. Der Mitarbeiter klickt auf „Weiter“.
9. Das System bestätigt die Eingabe aller Pflichtfelder und führt eine serverseitige Zwischenspeicherung aus.
10. Das System liefert die „Editor“-Seite aus.
11. Der Mitarbeiter wählt „Karte hinzufügen“.
12. Der Mitarbeiter trägt ein, wo das Projekt durchgeführt wird.
13. Der Mitarbeiter wählt die geografischen Orte des Projekts aus, indem er eine Suchfunktion innerhalb der eingebetteten Karte nutzt.
14. Das System ruft über eine API eine eingebettete Karte eines Drittanbieters (z. B. Google Maps) auf.
15. Der Mitarbeiter gibt in ein Suchfeld die Ortsnamen oder Adressen ein. Alternativ kann er Orte direkt durch Klicken auf die Karte auswählen.
16. Das Drittanbietersystem (z. B. Google Maps) liefert die entsprechenden Kartendaten (Koordinaten, Visualisierung) zurück und zeigt den ausgewählten Ort an.
17. Der Mitarbeiter fügt weitere Inhalte hinzu, z. B. Textblöcke, Bilder, Videos, Galerien, Tabellen und externe Links. Sie alle können mit den bereits eingefügten geografischen Orten verbunden werden, um sie in den Kontext zu setzen..
18. Der Mitarbeiter organisiert die Inhalte in einem Grid und ordnet sie in verschiedenen Tabs an.
19. Der Mitarbeiter klickt auf „Vorschau anzeigen“.
20. Das System führt eine serverseitige Zwischenspeicherung aus.
21. Das System liefert die Vorschau aus.
22. Der Mitarbeiter klickt auf „Veröffentlichen“.
23. Das System speichert die Daten und bestätigt dem Mitarbeiter die Projektanlage.

**Erweiterungen:**

- 8) Es fehlen Pflichtfelder. Das System erlaubt dem Mitarbeiter nicht, weiterzugehen, und zeigt eine passende Fehlermeldung an.
- 7, 17) Der Multimediainhalt wird nicht unterstützt oder erfüllt die Mindestanforderungen nicht. Das System zeigt dem Mitarbeiter eine Warnmeldung mit Details zu den Anforderungen an.
- 16) Es treten Verbindungsprobleme mit dem Drittanbietersystem (z. B. Google Maps) auf. Das System zeigt eine Fehlermeldung an und schlägt vor, die Funktion später erneut zu nutzen.
- 16) Die Adresse oder der Ort wird nicht gefunden. Das System zeigt eine Fehlermeldung an und bietet Alternativen (z. B. ähnliche Suchergebnisse) an.
- 22) Dem Mitarbeiter gefällt die Vorschau nicht. Er klickt auf „Bearbeiten“ und wird zurück zur Editor-Seite (Schritt 10) geleitet. Alle bisherigen Änderungen bleiben erhalten.
- 22) Der Mitarbeiter wählt statt „Veröffentlichen“ die Option „Entwurf speichern“. Das System speichert die Projektseite als Entwurf und leitet den Mitarbeiter zurück zur Übersichtsseite, wo der Entwurf später bearbeitet werden kann.

**Spezielle Anforderungen:** -**Technologie:** -**Häufigkeit:** Regelmäßig**5-Use-Case: Registrierung einer neuen Donation Box**

**Use Case Name:** Registrierung einer neuen Donation Box

**Level:** User-Goal, primärer use case

**Primärer Akteur:** Spender

**Stakeholders:** Spender: Will Box ohne viel Aufwand anschließen

**Vorbedingungen:**

1. Spender hat einen Account bei BrightImpact
2. Donation Box ist bei Spender angekommen
3. Der Nutzer hat die Spender App heruntergeladen und sich dort angemeldet

**Nachbedingungen:**

1. Die Donation Box ist beim Mainframe registriert
2. Die Donation Box hat die notwendigen Daten, um auf die Solaranlage zuzugreifen

**Auslöser:** Die Donation Box wird geliefert, und der Nutzer will sie anschließen.

**Haupterfolgsszenario:**

1. Der Spender schließt die Donation Box an den Strom und schließt sie per LAN ans Netzwerk an
2. Eine gelbe LED blinkt an der Box, die signalisiert, dass die Box startet
3. Der eingebaute Computer fährt hoch und der interne Server wird gestartet
4. Die Box registriert sich beim Mainframe und fängt an grün zu blinken
5. Der Spender scannt den QR-Code auf Donationbox mit der Spender App
6. Die App kommuniziert mit dem Mainframe und verbindet die gescannte Box mit dem Nutzer
7. Der Nutzer konfiguriert den API Connector für seine Solaranlage
8. Die App bestätigt das erfolgreiche Starten der Donation Box
9. Die grüne LED der donation box leuchtet dauerhaft
10. Die Box sendet regelmäßig Statusupdates über die Solaranlage und der aktuellen Stromgewinnung um so zu entscheiden, ob Jobs angenommen werden sollen

**Erweiterungen:**

- 3) Falls kein Internet verfügbar ist, leuchtet eine rote Status-LED am Gerät auf. Der Nutzer kann auf Hilfestellungen in der Anleitung zurückgreifen oder den Support anrufen.
- 7) Verbindung zum Connector schlägt fehl. Der Spender hat 2 Möglichkeiten:
  - a) Er kann akzeptieren, dass auf lokale Wetterdaten zugegriffen wird, um abzuschätzen, wann eine Überproduktion vorliegt.
  - b) Er kontaktiert den Support.

**Spezielle Anforderungen:** -

**Technologie:** Spezielle Hardware  
**Häufigkeit:** Regelmäßig

## 6-Use-Case: Projektwahl durch Spender

**Use Case Name:** Projektwahl durch Spender

**Level:** User-Goal, primärer Use Case

**Primärer Akteur:** Spender

**Stakeholders**

- Spender: möchte generierte Spenden an gemeinnützige Organisation spenden
- Spendenorganisation: Benötigt Spenden zur Umsetzung sozialer Projekte
- Bright Impact: Ermöglicht Organisationsauswahl zur Spendenweiterleitung

**Vorbedingungen:**

1. Spender hat registrierte Donation Box
2. Spender ist auf der Spender App angemeldet und hat seine Donation Box verbunden

**Nachbedingungen**

1. Spendenorganisation erhält alle Spenden, die nach der Projektauswahl durch die Donation Box generiert werden ODER einmalig alle bisher angesammelten, nicht verteilten Spenden.
2. Der Zeitpunkt der neuen Projektauswahl ist gespeichert
3. Die Historie der unterstützten Projekte ist einsehbar
4. Der Spender kann getätigte Spenden in App einsehen

**Auslöser:** Der Spender erhält eine E-Mail, dass seine Wallet einen bestimmten Betrag erreicht hat.

**Haupterfolgsszenario:**

1. Spender öffnet Bright Impact App
2. Spender loggt sich mithilfe seiner Accountdaten ein.
3. Spender navigiert zur Projektauswahl
4. Spender sieht wieviel Geld aktuell angesammelt und nicht verteilt ist
5. System zeigt Liste verfügbarer Spendenorganisationen/Projekte
6. Spender klickt auf ein Projekt, wodurch sich Detailansicht öffnet
7. Spender wählt Projekt für seine Spenden aus
8. System überweist entsprechenden Betrag vom zentralen Spendenkonto auf das Spendenkonto der NGO und der Betrag wird von der Wallet des Spenders abgezogen
9. Spendenbeleg wird generiert und in App angezeigt

**Erweiterungen:**

- 7a) Kein Guthaben -> keine Spende möglich. Hinweis wird angezeigt.
- 7b) Nach der Projektauswahl wählt Spender ein Projekt als "dauerhaften Spendenempfänger" aus, dann werden (bis zur nächsten Änderung) regelmäßig alle Spenden an dieses Projekt gespendet

**Spezielle Anforderungen:** -

**Technologie:** Ein Zahlungsdienstleister ist angebunden

**Häufigkeit:** Regelmäßig

**Sonstiges:** -

## 7-Use-Case: Verteile Jobs als parallele Tasks auf verschiedene Donation Boxen

**Use Case Name:** Verteile Jobs als parallele Tasks auf verschiedene Donation Boxen

**Level:** User-Goal, primärer use case

**Primärer Akteur:** Nutzer des Job Services, Bright Impact

**Stakeholders:** Nutzer des Job Services, Bright Impact

**Vorbedingungen:**

1. Customer muss sich auf der Jobseite registriert haben
2. Customer muss seine Kreditkarte hinterlegt haben

**Nachbedingungen:**

1. Der Job wurde erfolgreich abgeschlossen und der Customer hat das Ergebnis erhalten
2. Das Geld wurde auf das Zentrale Spendenkonto übertragen

3. Das Geld wurde auf die Wallets der beteiligten Spender übertragen

**Auslöser:** Customer möchte einen Job auf den Donation Boxen ausführen lassen.

**Haupterfolgsszenario:**

1. Der Customer loggt sich in seinen Account ein
2. Der Customer kann über ein Input Feld die location eines Container angeben, welchen er auf verschiedene Donation boxen verteilen möchte und gibt die Anzahl an Workern sowie den Zeitraum an
3. Das System bestätigt die Erreichbarkeit des Container und schlägt einen Preis vor
4. Der Job–User sieht die Kosten für seinen angeforderten Job und kann diesen bestätigen oder ablehnen
5. Nach Annahme scheduled der Mainframe die Jobs als einzelne Tasks auf verschiedene Worker (DonationBoxes)
6. Auch zeigt das System dem Job–User eine genaue Übersicht aller benutzten Ressourcen
7. Der Job wird erfolgreich abgeschlossen und die Ergebnisse dem Customer übermittelt.

**Erweiterungen:**

- 2) Alternativ kann der Customer auch über einen Hardware-Client eine Software Library On-Demand Jobs/Funktionen via eine Art von Cloudfunctions ausführen (Vorbild Dask)
- 5) Sollten wir gerade nicht ausreichend Rechenkapazität verfügbar sein, leiten wir Teile der Jobs an Partner-Rechenzentren weiter, um keine Kunden wegen geringer availability zu verlieren. Ein entsprechender Hinweis wird angezeigt.

**Spezielle Anforderungen:** -

**Technologie:** Remote-Protokol, Docker, Scheduling

**Häufigkeit:** Regelmäßig

## 8-Use-Case: Krypto-Mining auf Donation Boxen

**Use Case Name:** Krypto-Mining auf Donation Boxen

**Level:** Subfunction, primärer use case

**Primärer Akteur:** Donation Box

**Stakeholders:** Bright Impact: interessiert an der maximalen Nutzung verfügbarer Rechenressourcen

**Vorbedingungen:**

1. Es muss eine registrierte Donation Box geben, in deren Haushalt es aktuell Überproduktion gibt
2. Die Donation Box darf nicht mit einem Task belegt sein.
3. Der Miner muss korrekt auf der Box installiert sein, das sollte bei der Registrierung passieren.

**Nachbedingungen:**

1. Der Mining-Job wurde erfolgreich beendet
2. Der Gewinn wurde auf den Wallets verbucht
3. Alle Statistiken (wann die Box aktiv war etc.) wurden erfolgreich aktualisiert
4. Das Spendenkonto der Spender wurde um den geminten Betrag erhöht.

**Auslöser:** Die Box meldet, dass sie bereit ist, und der Mainframe hat keine Tasks, die er ihr zuteilen kann.

**Haupterfolgsszenario:**

1. Die Box verbindet sich mit einem Mining-Pool und meldet ihre Bereitschaft zum Mining
2. Der Mining-Pool sendet der Box regelmäßig Hashes zum Rechnen
3. Die Donation Box rechnet die Hashes und sendet sie an den Pool
4. Sobald das Mining-Netzwerk Kryptowährung gemined hat, werden seitens Mining-Netzwerk anteilig Crypto-Coins auf die Wallet des Mainframes transferiert
5. Diese werden zum nächstmöglichen Zeitpunkt gegen Euro gewechselt, und die Spender kriegen ihren Spendenbetrag anteilig auf ihr Wallet gutgeschrieben.

**Erweiterungen:**

- 1-3) Das Minen von Crypto-Coins wird unterbrochen sobald:
  - a) Der Donation Box einer neuer Task vom Mainframe zugeteilt wird
  - b) Kein überschüssiger Strom mehr verfügbar ist

**Spezielle Anforderungen:** -

**Technologie:** Miner, Docker

**Häufigkeit:** Regelmäßig



## 4. Software-Komponenten

### 4.1. Nutzerseitige Applikationen

Die Nutzerseitigen Applikationen können unabhängig voneinander entwickelt werden. Nutzerseitige Applikationen sind jene, die von Vereinen oder Endkunden verwendet werden.

#### 4.1 Donation Box

- Ist mit Solar-Anlagen Überwachungssoftware verbunden, und leitet diese Informationen an den Main-Frame weiter
- Meldet an den Mainframe, wenn sie bereit zum mining oder für einen Job ist.
- Führt zugeteilte Jobs aus, bis diese abgeschlossen sind.
- Sobald sie nicht mehr bereit ist, meldet sie das an den Mainframe. Jobs und mining werden zeitnah beendet.

#### 4.2 Client App

- Gibt Kunden allgemeine Informationen: Wie viel wurde an wen gespendet, wie viel ist die Box aktiv, ...
- Erhält Informationen über den Status der Donation-Box vom Mainframe
- Erlaubt Kunden einzustellen, an wen ihre Spenden gehen.
- Erlaubt Kunden Informationen über verschiedene Spendenprojekte, deren Spendenziele etc., einzusehen.

#### 4.3 Vereins-Verwaltungs Website

- Kommuniziert mit dem Mainframe.
- Erlaubt Vereinen, sich als Spendenempfänger zu registrieren.
- Erlaubt Vereinen, Statusupdates über ihre Projekte zu posten, die von Usern über die Client App eingesehen werden können.
- Erlaubt Vereinen, gesammelte Spenden abzurufen und auf ihr Spendenkonto zu übertragen.
- Gibt Vereinen Informationen über den Spendenverlauf, die Anzahl der Spender, ...

#### (4.4 Job Register Overview)

- Möglichkeit, neue Jobs anzumelden und zu bezahlen
- Zeigt auch Informationen über das aktuelle Scheduling an
- Zeigt den Status des Rechnernetzwerks an
- Verlauf bisheriger Jobs, ...

#### (4.5 Public Overview)

- Landing-Page: Spender, Job-Käufer und Vereine können sich über das Projekt informieren.
- Gibt aktuelle Statistiken und Informationen über die Plattform
- Erlaubt Privatpersonen sich anzumelden und eine Donation Box zu beantragen.
- Möglichkeit, Kontakt zu Bright Impact aufzunehmen.
- Leitet Job-Käufer und Vereine auf die entsprechenden Seiten weiter.

### 4.2 Serverseitige Applikation (Mainframe)

- Zentrale Anlaufstelle für Kommunikation der Komponenten
- Sammelt Informationen von Boxen und stellt diese der Client App zur Verfügung
- Aggregiert Daten und stellt diese dem Public Overview zur Verfügung
- Kümmt sich um den Verkauf von Krypto und Umwandlung in alt coins sowie Übertragung der Spenden zu Vereinen
- Verwaltet, wann Boxen aktiv sind und verteilt entweder Jobs oder aktiviert Mining.

## 4.3 Datenbank

Hier werden alle Daten gesichert gespeichert.

- Daten von den Donation boxes
- Daten über Transaktionen
- Aggregiert Nutzungsdaten
- Daten über registrierte Vereine, Nutzer, Job-Käufer
- ...

## 5. Requirements

### 5.1 Functional Requirements (FR)

Funktionale Anforderungen (FRs) definieren das grundlegende Systemverhalten unter bestimmten Bedingungen. Sie sind Produktmerkmale, die implementiert werden müssen, um es den Nutzern zu ermöglichen, ihre Ziele zu erreichen.

FR No	Beschreibung	Prio	Aufwand
FR-01	Donation Box: Die Donation Box muss einen Kryptominder steuern und ausführen können.	1	M
FR-02	Donation Box: Der Miner muss sich darum kümmern, dass der geminte Gewinn auf eine zentrale Wallet übertragen wird.	2	S
FR-03	Donation Box: Die Donation Box kann über den Mainframe zugeteilte Jobs annehmen, welche vom Benutzer selbst definiert als Container vorliegen.	4	M
FR-04	Donation Box: Die Donation Box kann über den Mainframe zugeteilte Jobs annehmen, welche wie eine Cloudfunction ausgeführt werden.	5	L
FR-05	Donation Box: Die Donation Box muss mit dem Mainframe kommunizieren können, um reguliert zu werden und Informationen auszutauschen.	1	M
FR-06	Donation Box: Die Donation Box muss verschiedene APIs von verschiedenen Solarbetrieben ansteuern können.	2	XL
FR-07	Donation Box: Die Donation Box muss mit der Solaranlage kommunizieren können, um zu erkennen, wann eine Überproduktion vorliegt.	2	S
FR-08	Donation Box: Es muss eine physische Donation Box geben, die man bei Nutzern von Solaranlagen anschließen und ins Netzwerk einbinden kann.	3	M
FR-09	Donation Box: Die Donation Box muss direkt mit dem User-Client kommunizieren können, um eine Verbindung mit dem WLAN herstellen zu können.	5	M
FR-10	Donation Box: Falls kein Connector verfügbar ist, kann aus lokalen Wetterdaten bestimmt werden, wann der Strom für Jobs benutzt wird	5	M
FR-11	User-Client: Endnutzer können sich über den User-Client anmelden können.	2	M
FR-12	User-Client: Der User-Client muss mit dem mainframe kommunizieren, um Statusabfragen über die Donation-Box zu ermöglichen, welche anschließend detailliert in einem Dashboard angezeigt werden.	1	M
FR-13	User-Client: Der User-Client stellt detailliert verschiedene Projekte der verschiedenen Spendenorganisationen vor, an welche Nutzer spenden können. Dazu zählen z.B. Spendenziele und durch Vereine bereitgestellte Updates.	1	M
FR-14	User-Client: Durch die Auswahl verschiedener Spendentöpfe können die	4	M

	Gewinne an verschiedene Organisationen gleichzeitig ausgeschüttet werden.		
<b>FR-15</b>	User-Client: Nach einer Spende kann der User die Nutzung eines jeden einzelnen gespendeten Euros detailliert nachverfolgen	<b>5</b>	<b>L</b>
<b>FR-16</b>	User-Client: Nutzer müssen Einsicht haben, wieviel ungespendetes Geld aktuell auf ihrem Spendenkonto liegt.	<b>1</b>	<b>S</b>
<b>FR-17</b>	Vereins-Website: Die Vereinswebsite muss eine Option für Vereine bieten, sich anzumelden und zu verifizieren	<b>1</b>	<b>M</b>
<b>FR-18</b>	Vereins-Website: Vereine müssen Spendenziele/Projekte durch einen Editor aufstellen können.	<b>3</b>	<b>S</b>
<b>FR-19</b>	Vereins-Website: Vereine müssen Statusupdates in Form von Texten, Bildern und neuen Spendenzielen zu ihren Projekten hinzufügen können.	<b>4</b>	<b>M</b>
<b>FR-20</b>	Vereins-Website: Vereine müssen alle bisher für ein Projekt gesammelten Spenden auf ihr Konto übertragen können. Entweder manuell oder automatisch.	<b>2</b>	<b>M</b>
<b>FR-21</b>	Vereins-Website: Jeder ausgegebene Euro muss wieder in der Application mitsamt Rechnung hochgeladen werden.	<b>4</b>	<b>L</b>
<b>FR-22</b>	Public-Overview: Es muss eine Landing-Page geben, die Spender, Vereine und Job-Käufer gleichermaßen anspricht. Dazu gehört eine Echtzeit-Übersicht zu allgemeinen Statistiken, z.B. wie viele Spenden generiert wurden und wie viele Vereine & Spender bereits registriert sind.	<b>4</b>	<b>M</b>
<b>FR-23</b>	Public-Overview: Es muss möglich sein, dass Spender sich über die Seite registrieren und eine Donation Box beantragen.	<b>4</b>	<b>S</b>
<b>FR-24</b>	Public-Overview: Es gibt eine call-to-action Weiterleitung zur Vereins-Website bzw. Job-Register damit sich die entsprechenden Nutzer dort registrieren können	<b>4</b>	<b>S</b>
<b>FR-25</b>	Public-Overview: Es gibt eine Möglichkeit, Kontakt zu Bright Impact aufzunehmen.	<b>5</b>	<b>S</b>
<b>FR-26</b>	Job-Website: Die Job-Website muss eine Option für Vereine bieten, sich anzumelden und zu verifizieren	<b>4</b>	<b>M</b>
<b>FR-27</b>	Job-Website: Ein Nutzer kann eine Kreditkarte hinterlegen, um später anfallende Kosten für die Jobs zu decken	<b>4</b>	<b>L</b>
<b>FR-28</b>	Job-Website: Ein Nutzer kann durch einen Link auf ein Repo/Container einen Job hochladen, welcher dann ausgeführt wird	<b>4</b>	<b>L</b>
<b>FR-29</b>	Job-Website: Ein Nutzer kann über eine Code-Library Cloudfunctions vergleichbar zu Dask verteilen	<b>5</b>	<b>L</b>
<b>FR-30</b>	Job-Website: Für jeden ausgeführten Job erhält der Nutzer eine genaue Kostenübersicht	<b>4</b>	<b>S</b>
<b>FR-31</b>	Mainframe: Das mainframe stellt eine gesicherte REST-API Schnittstelle zur Verfügung, welche von den einzelnen Subkomponenten angesteuert werden kann	<b>1</b>	<b>M</b>
<b>FR-32</b>	Mainframe: Das Mainframe speichert alle wichtigen Personen-/ und Applicationbezogenen Daten, wie welche Box von welchem Nutzer gerade aktiv ist	<b>1</b>	<b>M</b>
<b>FR-33</b>	Mainframe: Das Mainframe trackt, wer wie viele Jobs bearbeitet hat	<b>1</b>	<b>M</b>
<b>FR-34</b>	Mainframe: Das mainframe kann die angenommenen Jobs den verschiedenen Donation Boxen zuweisen	<b>4</b>	<b>L</b>

<b>FR-35</b>	Mainframe: Das mainframe kann im Falle einer ausgefallenen Donation Box den Job reschedulen	<b>5</b>	<b>L</b>
--------------	---	----------	----------

## 5.2 Non-functional Requirements (NFRs)

Nicht-funktionale Anforderungen (NFRs) beschreiben Aspekte des Systems, die nicht direkt mit seinem funktionalen Verhalten verbunden sind. Sie konzentrieren sich auf die Qualitätsmerkmale der Lösung, wie Skalierbarkeit, Zuverlässigkeit, Sicherheit oder Benutzerfreundlichkeit.

### NFR Klassen:

**Process:** Übergabe, Implementierung & Technik, Standards

**Product:** Benutzbarkeit, Zuverlässigkeit, Sicherheit, Performance, Umfang

**External:** Juristische oder Ökonomische/Kosten Beschränkungen, Interoperabilität

### Legal

NFR No.	Non-Functional Requirement Description	Typ

### Documentation

NFR No.	Non-Functional Requirement Description	Typ
NFR 01	Der Gedankenprozess ist durch das Hinzufügen der Hausaufgaben dokumentiert.	Process ▾
NFR 02	Die Systemarchitektur ist dokumentiert	Process ▾
NFR 03	Inline Dokumentation des Codes. Einhaltung von Clean Code (z.B. aussagekräftige Namen)	Process ▾

### Development

NFR No.	Non-Functional Requirement Description	Typ
NFR 04	CI/CD: Die Entwicklung wird unterstützt durch CI/CD Tooling	Process ▾
NFR 05	Fehlerfreiheit: Die Fehlerfreiheit des Codes ist sicherzustellen.	Process ▾
NFR 06	UI: Modern, aber einfach zu verstehen	Product ▾
NFR 07	Deployment: Das deployed Produkt muss stabil laufen und skalierbar sein	Product ▾

## 6. Target Environment (Vorschläge)

NFR No.	Non-Functional Requirement Description	Typ
NFR 08	Vereinswebsite, Job-Client, General Overview: Diese Clients sollen eine Webbasierte Anwendung sein. Da es sich hier überwiegend um UI handelt sind TS Frameworks zu empfehlen.	Product ▾
NFR 09	User-App: Die Userapp soll eine Handyapp kompatibel mit IOS und Android sein	Product ▾
NFR 09	Mainframe: Der mainframe sollte verteilt/horizontal (dynamisch) skalierbar sein, um die hohe Anfrage Last stemmen zu können	Product ▾
NFR 10	Donation-box: Falls Datenbank benötigt wird, soll diese in-memory sein	Product ▾

## 7. Deliverables und Abnahmekriterien

### 7.1 Deliverables

- Source Code für Donation Box, Client-App, Vereinswebsite und mainframe (+ weitere Websites, wenn verfügbar) und ein laufendes Deployment
- Documentation (inline code documentation + README)

### 7.2 Abnahmekriterien

- Vereine können sich registrieren und Spendenprojekte einstellen
- Benutzer können ihre Donation-Box anschließen und diese auf ihren Namen registrieren
- Die Donation Box kann sich mit der Solaranlage verbinden und ist nur aktiv, wenn genügend Strom vorhanden ist
- Boxen können Krypto minen, was dann in einer Wallet gesammelt wird
- Der Nutzer kann auswählen, wohin das Geld gespendet werden soll - die Spendenorganisation erhält dann anschließend das Geld auf ein von ihr hinterlegtes Konto/Wallet
- Die Anwendung läuft selbst mit ~1000 Donation Boxen stabil

## 8. Appendix

### 8.1 LF Tooling

- Clients: Angular & Flutter
- Mainframe: NestJS
- Donation-Box: Python FastAPI
- Datenbank: Postgres und DuckDB
- VSC & CI/CD: GitHub und GitHub Actions oder GitLab und GitLab Runners, Cloudflare Pages
- Ticket System: Linear
- Documentation: Outline/Google Docs
- Deployment: Docker Compose/K8S

### 8.2 Glossar

- Zeit Einschätzungen FRs::
  - S: 0.5 - 1.0 week
  - M: 1.0 week
  - L: 1.0-2.0 weeks
  - XL: >2.0 weeks
  - ?: hard to measure :)
- Client:
  - Beschreibt hier alles, was am Ende im Browser/beim Kunden auf dem Endgerät geladen wird.
- Server:

- Beschreibt den Teil der Applikation, welche isoliert vom Endbenutzer in einem geschützten Umfeld läuft. Übernimmt Verifikation von Anwendern und Datenbankzugriff.
- Krypto-Währung:
  - Kryptowährungen sind digitale oder virtuelle Währungen, die auf kryptografischen Technologien wie der Blockchain basieren, um sichere und dezentrale Transaktionen zu ermöglichen. Sie bieten eine Alternative zu traditionellen Währungen, ermöglichen Peer-to-Peer-Transaktionen und sind bekannt für ihre Volatilität sowie das Potenzial zur Revolutionierung des Finanzwesens.
- Krypto-Miner:
  - Ein Crypto Miner ist ein Computerprogramm, das Kryptowährungen wie Bitcoin durch die Lösung komplexer kryptografischer Aufgaben "schürft". Dabei validiert der Miner Transaktionen innerhalb des Netzwerks, fügt sie der Blockchain hinzu und erhält im Gegenzug neu generierte Coins oder Transaktionsgebühren als Belohnung.
- Blockchain:
  - Eine Blockchain ist eine dezentrale, digitale Datenbank, die Informationen in Blöcken speichert, die chronologisch miteinander verknüpft sind. Sie bietet Transparenz, Sicherheit und Unveränderlichkeit und wird häufig für Kryptowährungen, Smart Contracts und andere Anwendungen genutzt, bei denen Vertrauen und Nachverfolgbarkeit entscheidend sind.
- Cloud-Functions:
  - Cloud Functions sind serverlose, ereignisgesteuerte Code-Snippets, die automatisch in der Cloud ausgeführt werden, um Aufgaben wie Datenverarbeitung, API-Erstellung oder Integrationen auszuführen.
- Datenbank:
  - Ein System, welches es erlaubt, effizient und sicher Daten zu schreiben, lesen und zu modifizieren. Meistens ACID-compliant (Atomicity, Consistency, Isolation, Durability)
- Angular
  - Ein JavaScript Frontendframework, welches stark opinionated ist. Durch HTML-Templating können JavaScript Methoden injektieren werden, was wiederum DOM-Manipulation ermöglicht. Sehr robust mit einem großen Ökosystem [<https://angular.dev/>]
- PostgreSQL (Postgres)
  - Ein leistungsstarkes, objektrelationales Open-Source-Datenbankmanagementsystem, das für seine Stabilität, Skalierbarkeit und umfassende SQL-Unterstützung bekannt ist. Es bietet fortschrittliche Funktionen wie Transaktionen, Parallelverarbeitung, Unterstützung für JSON-Daten, Erweiterbarkeit und starke Sicherheitsmechanismen, wodurch es ideal für eine Vielzahl von Anwendungen ist [<https://www.postgresql.org/>]
- In-Memory Datenbank - DuckDB
  - DuckDB ist ein leistungsstarkes analytisches im prozess existierende Datenbanksystem. Es ist darauf ausgelegt, schnell, zuverlässig, portabel und benutzerfreundlich zu sein. DuckDB bietet einen umfangreichen SQL-Dialekt, der weit über grundlegendes SQL hinausgeht. DuckDB unterstützt beliebige und geschachtelte korrelierte Unterabfragen, Window-Funktionen, Kollationen, komplexe Datentypen (Arrays, Strukturen, Maps) und mehrere Erweiterungen, die SQL einfacher nutzbar machen [<https://github.com/duckdb/duckdb>]
- VSC
  - Kurzform für Version Control System: Beschreibt Programme, welche den Programmcode tracken und zwischen verschiedenen Programmieren synchronisieren können
- CI/CD
  - Kurzform für Continuous Integration and Continuous Deployment: Beschreibt den Prozess des Streamlinen von der Integration neuen Codes bis zur Auslieferung zum Kunden
- GitHub/GitLab
  - Online VSC Provider, welcher außerdem weitere Services anbietet [<https://github.com/>, <https://about.gitlab.com/>]
- GitHub Actions/Gitlab Runners
  - Cloud-Worker, welchen Jobs zugewiesen werden können. Diese kümmern sich meistens um Integration tasks, wie das Bauen der App oder das Laufen von Tests, sind aber häufig auch für das Deployment zuständig
- Cloudflare Pages:
  - Von Cloudflare bereitgestellte Plattform zum Deployen von Single Sign On Pages: Kostenlos und pro Branch nutzbar, um so Previews rendern zu können [<https://pages.cloudflare.com/>]

- Outline:
  - Ein opensource Wiki, welches von mir (Simon Okutan) privat gehostet wird  
[\[https://www.getoutline.com/\]](https://www.getoutline.com/)
- Docker Compose
  - Ein Tool zum Orchestrieren mehrerer parallel Laufender Container. Über eine YAML File werden verschiedene Deployments definiert, welche dann alle über ein einheitliches Interface bedient werden können. Mittlerweile ein Bestandteil von Docker
- K8S
  - Eine Kubernetes Version, welche es erlaubt, robust mehrere Docker parallel laufen zu lassen. Dies umfasst die Abstraktion in Pods und erlaubt mehrere verschiedene Deployment-Strategien un effizientes Load-Balancing. Mehr Konfigurationsoverhead als Docker-Compose, dafür aber auch deutlich umfangreicher.

### 8.3 Quellen

- Layout inspiriert vom JST Practical Course designed by Amin Ben Saad