SE: Problem Statement

Bright Impact: Shine to Mine

Inhaltsangabe

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. Scenarios

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 zugehbar 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.

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: Das System muss bereit sein: Alle notwendigen Technologien sind entwickelt und einsatzbereit.
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. Die Landing-Page zeigt alles, inklusive der Statistiken, schnell, übersichtlich und ansprechend an.
3. Der Nutzer ist vom Angebot überzeugt, und beschließt sich zu registrieren
    a. Ein potentieller Spender bestellt eine Donation Box.
    b. Ein Potentieller Job-Käufer wird zur Registrierungsseite für Jobs weitergeleitet und registriert sich.
    c. Ein Verein wird zur Registrierungsseite von Vereinen weitergeleitet und registriert sich.

Erweiterungen:

1. Ein Nutzer ist interessiert, möchte aber persönlich beraten werden. Er kann über ein Kontaktformular Kontakt zu Bright Impact aufnehmen.

Spezielle Anforderungen: -
Technologie: -

Use-Case: Registrierung eines Spenders

Use Case Name: Registrierung eines Spenders
Level: User-Goal, primärer Use Case
Primärer Akteur: Neuer Spender
Stakeholders: Brightimpact, Spender
Vorbedingungen: Der Spender muss eine Solaranlage besitzen
Nachbedingungen: Die Spendenbox wird an den Spender verschickt
Auslöser: Der Spender hat sich über Bright Impact informiert und entschließt sich ebenfalls zu spenden.
Haupterfolgsszenario:

1. Nutzer informiert sich über Bright Impact-Spenden-Projekt auf der Website
2. Nutzer entschließt sich zu registrieren.
3. Nutzer trägt Daten über seine Solaranlage ein
4. Nutzer trägt seine Adressdaten ein

5. Nutzer verifiziert sich über ident-verfahren
6. Die Donation Box wird konfiguriert und verschickt

Erweiterungen:

1. Der Nutzer braucht Hilfe, z.B. weil er keine Informationen über seine Solaranlagen hat, weshalb er mit dem Bright Impact Support Kontakt aufnimmt, der ihn unterstützt.
2. Die Solaranalge des Nutzers wird nicht unterstützt. Er bekommt die Optionen:
    a. Die Steuerung der Box über Wetterdaten zu akzeptieren
    b. Informiert zu werden, sobald seine Anlage unterstützt wird.

Spezielle Anforderungen: -
Technologie: -
Häufigkeit: Regelmäßig

Use-Case: Registrierung eines Vereins

Use Case Name: Registrierung eines Spenders
Level: User-Goal, primärer Use Case
Primärer Akteur: Verein (bzw. dessen Vorstand und Mitglieder)
Stakeholders: Vereinsvorstand, Vereinsmitglieder, Brightimpact, Spender
Vorbedingungen: Verein muss existieren und bescheinigt gemeinnützig sein.
Nachbedingungen:

1. Spender können sich über die Projekte informieren
2. Die Projekte können als Spendenempfänger ausgewählt werden

Auslöser: Verein überlegt sich neue Möglichkeit, spenden einzunehmen
Haupterfolgsszenario:

1. Verein informiert sich über Bright Impact-Spenden-Projekt auf der Website
2. Verein legt Nutzerkonto auf Website an
3. Verein lädt seine Gemeinnützigkeitsbescheinigung hoch
4. Bright Impact Mitarbeiter bestätigt Gemeinnützigkeitsbescheinigung
5. Verein legt Spendenprojekte an
6. Verein lädt Bilder und Texte über Spendenprojekt hoch

Erweiterungen:

1. Der Verein braucht Hilfe, weshalb er mit dem Bright Impact Support Kontakt aufnimmt, der ihn unterstützt.

Spezielle Anforderungen: -
Technologie:
Häufigkeit: Regelmäßig

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
Vorbedingungen:

1. Solaranlage der Familie muss unterstützte Schnittstelle anbieten
2. Sie muss ein konfiguriertes Heimnetzwerk mit ausreichender Internetgeschwindigkeit besitzen

Nachbedingungen:

1. Die Donation Box muss die Informationen der Solaranlage an den Mainframe weitergeben.
2. Die Familie muss über ihre Verwaltungs-Applikation auf ihre Informationen zugreifen können.
3. Der Mainframe muss die Donation Box ansprechen können, um verschiedene Services zu starten.

Auslöser: Die Box wird mit einem Stromkabel und einem LAN-Kabel angeschlossen
Haupterfolgsszenario:

1. Der eingebaute Computer fährt hoch
2. Der interne Server wird gestartet
3. Die Box registriert sich beim mainframe als aktiv
4. Nutzer installiert App
5. Nutzer erstellt Account

6. Nutzer scannt QR-Code auf Donationbox und linkt somit Box zu seinem Account
7. Nutzer stellt API Connector ein für seine Solaranlage
8. Die Box sendet regelmäßig Statusupdates über die Solaranlage und der aktuellen Stromgewinnung um so zu entscheiden, ob Jobs angenommen werden sollen

Erweiterungen: (Alternative Szenarios für Erfolgs- oder Fehlerfälle)
1. Falls kein Internet verfügbar ist, leuchtet eine rote Status-LED am Gerät auf
2. Falls kein Connector verfügbar ist: Nutzer kann akzeptieren, dass auf lokale Wetterdaten zugegriffen wird, um abzuschätzen, wann eine Überproduktion vorliegt.

Spezielle Anforderungen: -
Technologie: spezielle Hardware
Häufigkeit: Regelmäßig

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 und Spendenweiterleitung.

Vorbedingungen:
1. Spender hat registrierte Donation Box
2. Spender hat sich Bright Impact App heruntergeladen
3. Spender hat einen Benutzeraccount und diesen mit seiner 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. Zeitpunkt der neuen Projektauswahl ist dokumentiert
3. Die historie der unterstützten Projekte ist einsehbar
4. Spender kann getätigte Spenden in App einsehen

Auslöser: Spender öffnet die App und navigiert zur Projektauswahl.

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 Wallet oder zentralem Spendenkonto auf das Spendenkonto der NGO
9. Spendensumme wird von angezeigten Guthaben in App abgezogen
10. Spendenbeleg wird in App angezeigt
11. Spender kann nachvollziehen, was mit Spende passiert (ist)
12. Spender schließt App

Erweiterungen: (Alternative Szenarios für Erfolgs- oder Fehlerfälle)
1. Kein Guthaben -> keine Spende möglich
2. Spender wählt ein Projekt als “dauerhaften Spendenempfänger”, dann werden (bis zur nächsten Änderung) regelmäßig alle Spenden an dieses Projekt gespendet
3. Falls Nutzer bereits eingeloggt ist, überspringe Schritt 2

Spezielle Anforderungen:
Internationalisierungen: Wählbare Sprache und Währung auf dem Display

Technologie: -
Häufigkeit: Regelmäßig
Sonstiges: -

Use-Case: Einsehen des Spendenaufkommens durch Verein

Use Case Name: Einsehen des Spendenaufkommens durch Verein
Level: User-Goal, primärer use case
Primärer Akteur: Mitarbeiter des Vereins
Stakeholders: Mitarbeiter des Vereins
Vorbedingungen:

1. Verein muss sich registriert haben
2. Verein muss Projekt(e) angelegt haben

Nachbedingungen: -
Auslöser: Verein ist interessiert an Spendeneinnahmen oder braucht die Information für die Buchhaltung
Haupterfolgsszenario:

1. Der Mitarbeiter loggt sich auf der Website ein
2. Der Mitarbeiter entnimmt das aktuelle Spendenaufkommen und Analysen zur Anzahl der Spender etc. aus dem Dashboard

Erweiterungen: -
Spezielle Anforderungen: -
Technologie: Website
Häufigkeit: Regelmäßig

Use-Case: Benutzung Spenden durch Verein

Use Case Name: Benutzung Spenden durch Verein
Level: User-Goal, primärer use case
Primärer Akteur: Mitarbeiter des Vereins
Stakeholders: Mitarbeiter des Vereins
Vorbedingungen:

1. Verein muss Nutzeraccount haben
2. Verein muss Projekt angelegt haben

Nachbedingungen:

1. Das Geld ist aufs Vereinskonto überwiesen worden
2. Das Guthaben des Vereins auf diesem Projekt ist um den abgezogenen Betrag verringert worden
3. In der Projektansicht für die Spender wird der Betrag, den der Verein für dieses Projekt abgerufen hat, aktualisiert.

Auslöser: Verein möchte die erhaltenen Spendengelder nutzen
Haupterfolgsszenario:

1. Der Mitarbeiter loggt sich auf der Website ein
2. Der Mitarbeiter geht auf das Projekt, welches er umsetzen möchte und beantragt eine Überweisung des Geldes auf das Vereinskonto
3. Anschließend kann er noch eine Dankesnachricht schreiben
4. Nun hat der Mitarbeiter ebenfalls die Möglichkeit, Updates zum Projekt in Form von Bildern und Texten hochzuladen, um die Spender auf dem Laufenden zu halten.

Erweiterungen:

1. Der Mitarbeiter kann einstellen, dass das Geld automatisch in regelmäßigen Abständen (entweder zeitlich oder nach dem Erreichen eines bestimmten Betrags) auf das Vereinskonto übertragen wird.

Spezielle Anforderungen: -
Technologie: Website
Häufigkeit: Regelmäßig

Use-Case: Verteilte parallele Jobs auf verschiedene Donation Boxen

Use Case Name: Verteilte parallele Jobs 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. Job-User muss sich auf der Jobseite registriert haben
2. Job-User muss seine Kreditkarte hinterlegt haben

Nachbedingungen: -
Auslöser: Job-User möchte einen Job auf den Donationboxen ausführen lassen. **Haupterfolgsszenario**:

1. Der Job-User loggt sich in seinen Account ein
2. Der Job-User kann über ein Input Feld die location eines Container angeben, welchen er auf verschiedene Donation boxen verteilen möchte
3. Nach einer Initialen Bestätigung kann der Job-User entscheiden, wie viele Worker er buchen möchte und für welchen Zeitraum
4. Der Job–User sieht die Kosten für seinen angeforderten Job und kann diesen bestätigen oder ablehnen
5. Nach Annahme scheduled das System die Jobs auf verschiedene Worker
6. Auch zeigt das System dem Job–User eine genaue Übersicht aller benutzten Ressourcen

Erweiterungen:

1. Alternativ kann der Job-User auch über einen Hardware-Client eine Software Library On-Demand Jobs/Funktionen via eine Art von Cloudfunctions ausführen (Vorbild Dask)
2. 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

Use-Case: Krypto-Mining auf Donation Boxen

Use Case Name: Krypto-Mining auf Donation Boxen
Level: Subfunction, primärer use case
Primärer Akteur: Bright Impact
Stakeholders: Bright Impact
Vorbedingungen:

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

Nachbedingungen:

1. Der Mining-Job ist erfolgreich beendet
2. Der Gewinn liegt auf den Wallets
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 Jobs, die er ihr zuteilen kann.
Haupterfolgsszenario:

1. Die Box verbindet sich mit einem Mining-Pool
2. Die Box beteiligt sich am Krypto-Mining
3. Sobald Kryptowährung gemined wurde, wird diese auf das Wallet am Mainframe übertragen.
4. Dieser Verkauft die Währung zum nächstmöglichen Zeitpunkt, und trackt den Ertrag gleichverteilt auf die Spendenkonten der Box-Besitzer, die zum Zeitpunkt des Einkommens am Mining beteiligt waren.
5. a) Sobald kein überschüssiger Strom mehr verfügbar ist, wird das Mining beendet.
   b) Sobald ein Job verfügbar ist, wird das Mining beendet und der Job geladen.

Erweiterungen: -
Spezielle Anforderungen: -
Technologie: Miner, Docker
Häufigkeit: Regelmäßig

4. Software-Komponenten

4.1. Nutzerseitige Applikationen

Die Nutzerseitgen 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 Rechennetzwerks 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
- Kümmert 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.		
FR-15	User-Client: Nach einer Spende kann der User die Nutzung eines jeden einzelnen gespendeten Euros detailliert nachverfolgen	**5**	**L**
FR-16	User-Client: Nutzer müssen Einsicht haben, wieviel ungespendetes Geld aktuell auf ihrem Spendenkonto liegt.	**1**	**S**
FR-17	Vereins-Website: Die Vereinswebsite muss eine Option für Vereine bieten, sich anzumelden und zu verifizieren	**1**	**M**
FR-18	Vereins-Website: Vereine müssen Spendenziele/Projekte durch einen Editor aufstellen können.	**3**	**S**
FR-19	Vereins-Website: Vereine müssen Statusupdates in Form von Texten, Bildern und neuen Spendenzielen zu ihren Projekten hinzufügen können.	**4**	**M**
FR-20	Vereins-Website: Vereine müssen alle bisher für ein Projekt gesammelten Spenden auf ihr Konto übertragen können. Entweder manuell oder automatisch.	**2**	**M**
FR-21	Vereins-Website: Jeder ausgegebene Euro muss wieder in der Application mitsamt Rechnung hochgeladen werden.	**4**	**L**
FR-22	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.	**4**	**M**
FR23	Public-Overview: Es muss möglich sein, dass Spender sich über die Seite registrieren und eine Donation Box beantragen.	**4**	**S**
FR-24	Public-Overview: Es gibt eine call-to-action Weiterleitung zur Vereins-Website bzw. Job-Register damit sich die entsprechenden Nutzer dort registrieren können	**4**	**S**
FR-25	Public-Overview: Es gibt eine Möglichkeit, Kontakt zu Bright Impact aufzunehmen.	**5**	**S**
FR-26	Job-Website: Die Job-Website muss eine Option für Vereine bieten, sich anzumelden und zu verifizieren	**4**	**M**
FR-27	Job-Website: Ein Nutzer kann eine Kreditkarte hinterlegen, um später anfallende Kosten für die Jobs zu decken	**4**	**L**
FR-28	Job-Website: Ein Nutzer kann durch einen Link auf ein Repo/Container einen Job hochladen, welcher dann ausgeführt wird	**4**	**L**
FR-29	Job-Website: Ein Nutzer kann über eine Code-Library Cloudfunctions vergleichbar zu Dask verteilen	**5**	**L**
FR-30	Job-Website: Für jeden ausgeführten Job erhält der Nutzer eine genaue Kostenübersicht	**4**	**S**
FR-31	Mainframe: Das mainframe stellt eine gesicherte REST-API Schnittstelle zur Verfügung, welche von den einzelnen Subkomponenten angesteuert werden kann	**1**	**M**
FR-32	Mainframe: Das Mainframe speichert alle wichtigen Personen-/ und Applicationbezogenen Daten, wie welche Box von welchem Nutzer gerade aktiv ist	**1**	**M**
FR-33	Mainframe: Das Mainframe trackt, wer wie viele Jobs bearbeitet hat	**1**	**M**
FR-34	Mainframe: Das mainframe kann die angenommenen Jobs den verschiedenen Donation Boxen zuweisen	**4**	**L**

FR-35	Mainframe: Das mainframe kann im Falle einer ausgefallenen Donation Box den Job reschedulen	**5**	**L**

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	Mainfraime: Der mainfraime 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 genügen 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 BoxenStabil

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, effizent und sicher Daten zu schreiben, lesen und zu modifizieren. Meistens ACID-complient (Atomicity, Consistency, Isolation, Durability)
- Angular
  - Ein JavaScript Frontendframework, welches stark opinionated ist. Durch HTML-Templating können JavaScript Methoden injekten 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 Continious Integration and Continious Deployment: Beschreibt den Prozess des Streamlinen von der Integration neunen 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 Deploymen 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/]
- 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 Kubernetis 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