*TAU INF202 Software Engineering*

*Individuelles Projekt*

***Pflichtenheft***

Projektdokumentation

Version: v1.0

Status: Freigegeben

Who Pays?

Verantwortliche/r:

Ahmet Emirhan Acar, e200503002@stud.tau.edu.tr

Ibrahim Onur Demir, e200503012@stud.tau.edu.tr

Stakeholder: DI. Ömer Karacan, omer.karacan@tau.edu.tr

**Dokumentenverwaltung**

**Dokument-Historie**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Status \*)** | **Datum** | **Verantwortlicher** | **Änderungsgrund** |
| v0.1 | Entwurf | 01.04.2023 | A. Emirhan Acar  I. Onur Demir | Wir haben ein Entwurf entwickelt. |
| V1.0 | Freigegeben | 15.04.2023 | A. Emirhan Acar  I. Onur Demir | Who Pays? wurde die vollversion freigegeben |

***Dokument-Status: Entwurf / in Review / freigegeben (abgegeben)***

freigegeben

**Dokument wurde mit folgenden Tools erstellt:**

Microsoft Office Word

Google Docs

Canva

Dall.e

Lucidheart

**Inhaltsverzeichnis**

[1. Einleitung 4](#_Toc132667405)

[2. Ausgangssituation und Ziele 5](#_Toc132667406)

[3. Gesamtarchitektur 7](#_Toc132667407)

[4. Funktionale Anforderungen 9](#_Toc132667408)

[5. Nichtfunktionale Anforderungen 12](#_Toc132667409)

[6. Abnahmekriterien 13](#_Toc132667410)

[7. Projekt Meilensteine 14](#_Toc132667411)

[8. Referenzen 15](#_Toc132667412)

# Einleitung

Dieses Dokument dient dazu, die zwingenden Anforderungen an das Prototyping einer Zahlungsabrechnungs- und Tracking-App zu definieren und vollständig und konsistent zu beschreiben.

Die Use Cases und Anforderungen sind aus der Sicht des Stakeholders beschrieben.

Im Kapitel 2 „Ausgangssituation und Ziele“ sind die Ausgangssituation und der Grund zur Wahl von

Autoladestation anschaulich dargestellt.

Im Kapitel 3 „Gesamtarchitektur“ sind die physikalische und die konzeptionelle Architektur des

Systems, und die wichtigsten Subsysteme (Komponenten), die Anwender und die notwendigen

Kommunikationsschnittstellen dargestellt. Hier sind auch zusätzliche Anforderungen an die

Architektur oder Komponenten zu finden.

Im Kapitel 4 „Funktionale Anforderungen“ beinhaltet die Beschreibung der funktionalen

Anforderungen durch die Ablaufbeschreibungen (User Stories), die Anwendungsfällen (Use

Cases), und technischen und fachlichen Anforderungen (Requirements). Alle betriebsrelevanten

Daten werden durch die Datenmodellen definiert.

Im Kapitel 5 „Nichtfunktionale Anforderungen“ sind die funktionalen Anforderungen durch

diejenigen Anforderungen erweitert, die keine funktionalen Anforderungen sind.

Im Kapitel 6 „Abnahmekriterien“ sind die Abgabeartefakten festgelegt, die ohne Abstimmung des

Stakeholders nicht zu manipulieren sind.

Im Kapitel 7 „Projekt Meilensteine“ sind die wichtigsten Termine ausgelistet, die den Fortschritt die

Teilergebnisse des Projektes definieren.

Im Kapitel 8 „Referenzen“ sind die wichtigsten Referenzen aufgelistet

# Ausgangssituation und Ziele

**Einleitung**

In diesem Kapitel möchten wir Ihnen erläutern, was uns dazu bewegt hat, dieses Projekt zu starten und welche Ziele wir damit verfolgen. In einer WG mit mehreren Mitbewohnern kann es schwierig sein, die Zahlungen für Miete und Nebenkosten im Überblick zu behalten. Unser Ziel ist es, mit diesem Projekt ein Tool zu schaffen, das den Überblick über Zahlungen erleichtert und damit Konflikte zwischen den Mitbewohnern vermeidet. Durch die Möglichkeit, alle Zahlungen und Schulden auf einem Dashboard zu verfolgen, können die Mitbewohner die Zahlungen und die Aufteilung der Kosten besser nachvollziehen. Mit diesem Projekt wollen wir eine effektive Lösung für dieses Problem anbieten und die Zusammenarbeit zwischen den Mitbewohnern verbessern.

**Problemstellung (Funktionalität)**

In einer WG mit mehreren Mitbewohnern kann es oft schwierig sein, den Überblick über die Zahlungen zu behalten. Konflikte über unbezahlte Rechnungen oder unklare Aufteilungen der Kosten können auftreten und das Zusammenleben belasten. Das Ziel dieses Projekts ist es, eine Lösung zu schaffen, die diesen Konflikten vorbeugt und ein einfaches Tool bietet, um Zahlungen und Schulden zwischen den Mitbewohnern zu verwalten. Mit diesem Tool können die Mitbewohner ihre Zahlungen einfach eintragen und einsehen, wer noch Schulden hat. Das erleichtert die Kommunikation und verbessert die Zusammenarbeit innerhalb der WG. Darüber hinaus soll das Tool die Möglichkeit bieten, Zahlungs- und Schulddaten in einer Übersicht darzustellen, um den Überblick zu erleichtern und Konflikte zu vermeiden.

**Stakeholder (Anwender):**

Alle Bürger, die von einem verheerenden Erdbeben betroffen sind, sowie Katastrophenschutzbeauftragte von Regierungen oder privaten Institutionen können dieses System nutzen.

**Systemumfeld (Einsatzumgebung)**

Unser System ist für Mitbewohner in Wohngemeinschaften gedacht, die Schwierigkeiten haben, den Überblick über ihre Zahlungen zu behalten. Die Nutzung erfordert ein internetfähiges Gerät und eine stabile Internetverbindung. Der Zugriff auf das System erfordert eine Registrierung mit gültigen persönlichen Informationen.

**Rahmenbedingung (Einschränkungen)**

* Wir müssen Benutzerinformationen für jeden Benutzer erfassen und speichern.
* Wir müssen Zahlungen und deren Details speichern.

**Ziele (Lösung)**

Für dieses Projekt wird die MySQL-Datenbank verwendet, um alle Benutzerinformationen zu speichern. Auch die Zahlungsdaten werden in dieser Datenbank gespeichert. Zahlungsdetails wie wer bezahlt hat, wie viel sie bezahlt haben und warum sie die Zahlung geleistet haben, werden von den Benutzern eingegeben. Das Programm wird diese Eingaben an die Datenbank übermitteln, und die Daten werden dort gespeichert. Wenn das Programm diese Daten benötigt, wird es diese Informationen aus der Datenbank abrufen, um sie in Zukunft zu verwenden.

Klassen werden verwendet, um die Entwicklung des Programms zu erleichtern, den schnelleren Zugriff auf die erforderlichen Daten zu ermöglichen und ein insgesamt effizientes Programm zu erstellen.

# Gesamtarchitektur

**Einleitung**

Dieses Kapital beinhaltet die Gesamtsystemarchitektur der Darstellung von dem System und deren Komponenten und die Einzelheiten von diesen Komponenten, damit die allgemeine Funktionsweise des Systems erklärt wird.

***Gesamtarchitektur***

In Gesamtarchitektur wird Einzelheiten der folgende Komponenten beschreibt:

* Benutzeroberfläche(UI)
* Datenbank

Timeline

Description automatically generated

***Abbildung 1:*** *Gesamtarchitektur*

***Benutzeroberfläche***

Die Komponente Benutzeroberfläche ermöglicht die Interaktion zwischen dem Benutzer und dem System, indem sie Anfragen weiterleitet und Antworten anzeigt.

***Datenbank***

Die Komponente Datenbank bietet eine Abstraktionsschicht, um den Datenzugriff zu ermöglichen, indem sie SQL-Abfragen oder andere Datenzugriffsbefehle generiert und ausführt.

Die Benutzeroberfläche dient als Brücke zwischen dem Benutzer und dem System, indem sie Benutzeranfragen an das System weiterleitet die Antworten anzeigt.

***Externe Schnittstellen***

***Google-Sign-In-API***

Die Google-Sign-In-API wird verwendet, um Benutzern die Anmeldung mit ihren Google-Konten zu ermöglichen.

# 

# 4. Funktionale Anforderungen

**Einleitung**

In diesem Kapitel sind Anforderungen (inklusive User Stories und Use Cases) an Gesamtsystem, aber auch an die einzelnen Systemkomponenten definiert.

**UI Use Cases**

**/UI-1/** Auf dem Anmeldebildschirm kann sich jeder Kellner mit seinem eigenen Passwort

anmelden.

**/UI-2/** In der Personenoberfläche werden alle Personen im System angezeigt, die zu bearbeitende Person ausgewählt und deren Details können bearbeitet werden.

**/UI-3/** Das vorhandene Menü wird von der Menüoberfläche angezeigt, es bietet die Möglichkeit zu bearbeiten und hinzuzufügen und zu entfernen.

**/UI-4/** Die Schnittstelle "Zahlungsverlauf" enthält eine Historie der zuvor eingegebenen Zahlungen; Rechnungen, Zahlungsdetails, Zahlungsbeträge, Zahlungsdatum usw. können angezeigt werden.

**/UI-5/** Die Schaltfläche "Berechnen" wird die notwendigen Berechnungen durchführen und dem Endbenutzer anzeigen, wer wem wie viel zahlen muss.

Timeline

Description automatically generated

***Abbildung 4:*** Benutzeroberfläche

**API Use Cases**

Diagram

Description automatically generated

***Abbildung 6: Use-Case diagramm***

**/UC-1/** Die Use-Cases, die auf der Benutzeroberfläche gewährleistet werden, werden in der Teil “UI Use Cases” erklärt.

**/UC-2/** Wenn sich ein Benutzer mit seinem Google-Konto anmelden möchte, wird die Google-Sign-In-API verwendet, um die Anfrage abzuschließen.

**Technischen und fachliche Anforderungen**

**Google Sign-In APIs:**

**/GSIA-1/** Anmeldeanfrage des Nutzers wird an Google gesendet.

**/GSIA-2/** Konto wird mit Google-Informationen erstellt.

**/GSIA-3/** Neues autorisiertes Konto wird zu Who Pays hinzugefügt? Benutzerdatenbank.

**Datenmodell**

**● /DAT-1/** Ein DBMS sollte ausgewählt und eingerichtet werden, um die Daten für jede Komponente zu speichern.

**● /DAT-2/** Relevante Systemparameter werden in einer externen Datenbank gespeichert. Die Interessengruppen werden später eine Liste der Parameter bereitstellen.

**● /DAT-3/** Bei der Erstellung des Datenmodells sollten die Parameter der API-Schnittstelle berücksichtigt werden.

**● /DAT-4/** Um Datenmodelle darzustellen, empfiehlt es sich, UML-Klassendiagramme zu verwenden.

**● /DAT-5/** Ohne eine Datenbank können Programme nach einem Neustart keine früheren Daten wiederherstellen. Daher ist, wie in unserem System, eine SQL-Datenbank erforderlich.

# 

# 5. Nichtfunktionale Anforderungen

**Einleitung**

In diesem Kapitel sind die nicht-funktionalen Anforderungen ans Gesamtsystem aber auch an die einzelnen Systemkomponenten definiert. Es wird besonders auf die Software Qualität Wert gelegt (Testing).

**Nicht-funktionale Anforderungen an die Systemarchitektur (Architekturmuster, Deployment)**

* **/SYS-1/** Funktionalitäten(Abrufe und Abspeichern) werden in verschiedenen Klassen gewährleistet und implementiert.
* **/SYS-2/** Das ganze System wird auf dem lokalen Host laufen(Datenbankserver, Webseite, usw.)

**Nicht-funktionale Anforderungen an die Entwicklungsumgebung**

* **/DEV-1/** Die Entwicklungsumgebung ist frei wählbar!

**Nicht-funktionale Anforderungen an die Entwicklungswerkzeuge (Sprache, IDE, Frameworks)**

* **/TOL-1/** Verwendete Computersprachen im Projekt sind Java, html, css und javascript.
* **/TOL-2/** Verwendete Framework im Projekt ist Spring Boot.
* **/TOL-3/** Verwendete Database im Projekt ist MySQL Server.
* **/TOL-4/** IDE’s, die bei der Vorbereitung des Projekts verwendet werden, sind IntelliJ und Visual Studio Code.

**Nicht-funktionale Anforderungen an die Teststrategie (Qualitätssicherung)**

* **/TEST-1/** Es sollte getestet werden, ob es sich mit dem Api-Schlüssel verbinden kann und ob die Karte auf der Webseite angezeigt werden kann.
* **/TEST-2/** Es sollte getestet werden, ob beim Anklicken der Gebäude auf der interaktiven Karte die Gebäudeinformationen in der Infobox angezeigt werden.
* **/TEST-3/** Alle Use Cases, User Stories und Anforderungen sollen getestet und berichtet werden.
* **/TEST-4/** Es sollte getestet werden, ob sich die vom Benutzer in der Präsentationsschicht geänderten Informationen auch in der Datenbank geändert haben.
* **/TEST-5/** Es sollte getestet werden, ob die geänderten Informationen in der Datenbank für andere Benutzer sichtbar sind.

# 6. Abnahmekriterien

Das Projekt wird mit den folgenden Artefakten abgegeben:

* Dokumentation:
  + Pflichtenheft: INF202-who-pays-pflichtenheft-2023.docx
* Software
  + Link zu GitHub Projekt: https://github.com/AEmirhann/WhoPays
* Evidenz:
  + System/Software-Demo via Videoclip: im Gange

# 7. Projekt Meilensteine

* Meilenstein #1: Wir haben untersucht, wie dieses Projekt erstellt werden kann, diskutiert, welche Programmiersprache verwendet werden soll und wie die Datenbank verwaltet wird.
* Meilenstein #2: Wir haben eine UI-Demo erstellt, um eine Vorstellung von der Benutzeroberfläche zu haben.
* Meilenstein #3: Wir haben besprochen, welche Klassen wir benötigen und welche Daten wir speichern müssen.

# 

# 8. Referenzen

https://developers.google.com/api-client-library/java/google-api-java-client/oauth2