TAU INF202 Software Engineering

Individuelles Projekt

Pflichtenheft

Projektdokumentation

Version: 2023.v1.0

Status: Freigegeben

Schwarze Texte sind INVARIANT (zu behalten)!
Die blauen und roten Texte sind zu ersetzen oder zu entfernen!

SiparisTakip

Verantwortliche/r:

Kerem Şakir, e200503038@stud.tau.edu.tr

Yasin Aydın, e220503068@stud.tau.edu.tr

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

Template: Projektdokumentation.doc / Version vom 29.09.2020

Dokumentenverwaltung

Dokument-Historie

Version	Status *)	Datum	Verantwortlicher	Änderungsgrund
v0.9	Entwurf	25.02.2023	Yasin Aydın / Kerem Şakir	Die Vorlage für das “SiparisTakip”-Projekt wurde veröffentlicht.
v1.0	Freigegeben	17.04.2023	Yasin Aydın / Kerem Şakir	Für das Projekt SiparisTakip wurde die Vollversion (v1.0) freigegeben.

**) Sofern im Projekt nicht anders vereinbart, sind folgende Statusbezeichnungen zu verwenden (in obiger Tabelle und am Deckblatt):*
Dokument-Status: Entwurf / in Review / freigegeben (abgegeben)

Dokument wurde mit folgenden Tools erstellt:
Microsoft Office Word
OpenAI DALL-E
Lucidchart
SwaggerHub

Inhaltsverzeichnis

1. Einleitung

Dieses Dokument zielt darauf ab, die Funktionen zu beschreiben, die für ein prototypisches Restaurant-Auftragsverfolgungssystem erforderlich sind, und es auf klare und nützliche Weise zu erklären.

Die Anwendungsfälle und Anforderungen werden aus Sicht der Stakeholder beschrieben.

Die grafische Oberfläche zur Überwachung des Auftragsverfolgungsprozesses ist aus Sicht eines vom Mitarbeiter beschrieben.

Kapitel 2 „Ausgangssituation und Ziele“ stellt die Ausgangssituation und die Gründe für die Wahl einer Autoladestation übersichtlich dar.

Kapitel 3 „Gesamtarchitektur“ beschreibt die physische und konzeptionelle Architektur des Systems, zusammen mit den wichtigsten Subsystemen (Komponenten), Benutzern und notwendigen Kommunikationsschnittstellen. Auch weitere Anforderungen an Architektur oder Komponenten finden Sie hier.

Kapitel 4 „Funktionale Anforderungen“ beinhaltet die Beschreibung funktionaler Anforderungen durch Prozessbeschreibungen (User Stories), Use Cases (Use Cases), fachliche und fachliche Anforderungen (Requirements). Alle betriebsbezogenen Daten werden durch das Datenmodell definiert.

In Kapitel 5 „Nichtfunktionale Anforderungen“ werden funktionale Anforderungen um nichtfunktionale Anforderungen erweitert.

In Kapitel 6, Akzeptanzkriterien, werden die zu liefernden Artefakte definiert, die nicht ohne die Zustimmung der Stakeholder manipuliert werden sollten.

Kapitel 7 „Meilensteine des Projekts“ listet die wichtigsten Termine auf, die den Zeitplan und die Teilergebnisse des Projekts definieren.

Die wichtigsten Referenzen sind in Kapitel 8 „Literatur“ aufgelistet.

2. Ausgangssituation und Ziele

Einleitung

In diesem Kapitel werden wir darüber sprechen, für welche Probleme unser Programm Lösungen bietet, welche Gruppen am meisten von der Nutzung des Systems profitieren, in welcher Infrastruktur Umgebung das System eingesetzt wird, welche Einschränkungen für das System berücksichtigt werden sollten Einsatz- und Betriebsumgebung und welche Ziele mit den Lösungen erreicht werden.

Problemstellung (Funktionalität)

Unser System zielt darauf ab, Fehler zu vermeiden, die bei der Annahme von Bestellungen und Kontokürzungen in Geschäften wie Restaurants und Cafés auftreten können, und die Arbeit durch seine einfache und effektive Verwendung zu beschleunigen.

Stakeholder (Anwender):

Die für den Einsatz unseres Systems vorgesehene Masse ist der Dienstleistungssektor. Zum Beispiel Cafés, Restaurants, Bars und Clubs.

Systemumfeld (Einsatzumgebung)

Das Programm, das aus Komponenten besteht, die in speziellen Prozessen mit der erforderlichen Software ausgeführt werden, wird auf einem Computer ausgeführt.

Rahmenbedingung (Einschränkungen)

Die Haupteinschränkung liegt in der Wahl der Software-Tools.

- Eclipse wird als Softwareentwicklungstool verwendet,
- implementiert durch Java Framework und Spring Framework am Backend,
- JavaFX-Rich-Client-Technologie am Frontend,
- und persistente Daten werden in der SQL-Datenbank gespeichert.

Ziele (Lösung)

Es wird ein Prototyp entwickelt, der den Auftragsstatus und die Auslastung der Tische zeigt. Darüber hinaus sollten alle über das System getätigten Bestelltransaktionen und Zahlungstransaktionen, die am Ende dieser Bestellungen getätigt wurden, gespeichert werden. Erforderliche Backend-Operationen müssen mithilfe von RESTful Services (API) definiert werden. Die Persistenz der Daten sollte mit einem Datenbanksystem erfolgen.

3. Gesamtarchitektur

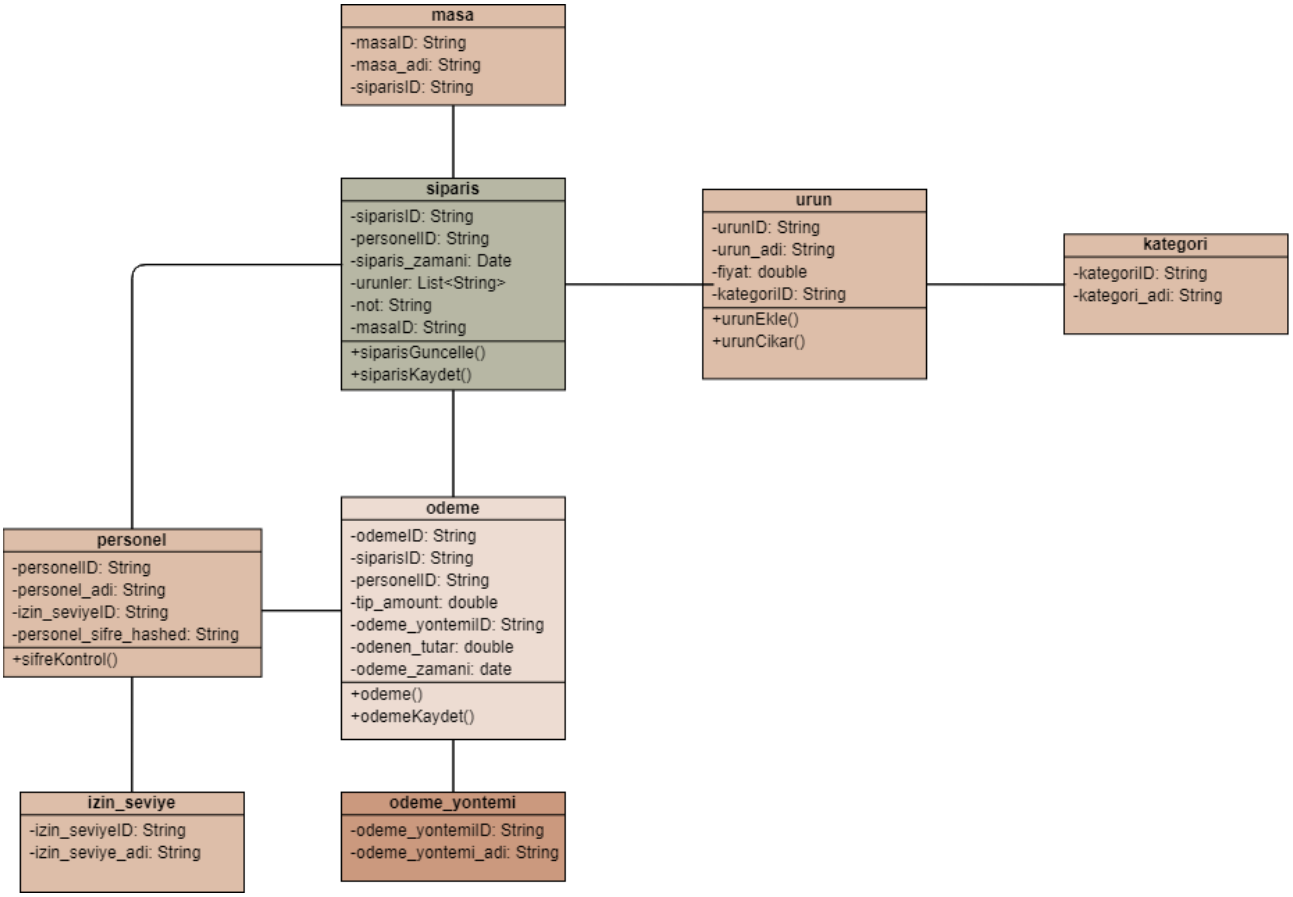
Einleitung

In diesem Kapitel werden die Systemgrenzen definiert, die mit entsprechenden Illustrationen spezifiziert werden. Die externen Schnittstellen (grafische Oberflächen) werden vom Benutzer unter Berücksichtigung der Systemgrenzen definiert. Die notwendigen Systemkomponenten und Datenstrukturen werden definiert und modelliert.

Gesamtarchitektur

Die Gesamtarchitektur wird unter folgenden Aspekten betrachtet:

- Kontext des Geschäftsprozesses, in dem das Programm verwendet wird,
- Systemarchitektur zur Identifikation autonomer Komponenten, wie beispielsweise der Auftragsabwicklung und der Zahlungsabwicklung
- Komponentenarchitektur (Architekturmuster)



Im obigen UML-Klassendiagramm werden die Klassen, Merkmale und Methoden des Auftragsverfolgungssystems im Allgemeinen ausgedrückt.

Komponente Order Tracking

Die Komponente Order Tracking (OT) verwaltet Bestellinformationen und Bestellstatus im System. Über die Serviceschnittstelle stellt OT Informationen über die Bestellung und deren Status anderen Komponenten im System zur Verfügung, z.B. für Benutzeroberflächenkomponenten und Komponenten zur Auftragsabwicklung. Der Bestellstatus wird automatisch aktualisiert, wenn Änderungen an der Bestellung durch die Bestellabwicklungskomponente vorgenommen werden, z. B. wenn neue Artikel zur Bestellung hinzugefügt oder aus dieser entfernt werden oder die Zahlung für die Bestellung eingegangen ist.

Externe Schnittstellen

Externe Schnittstellen sind:

1. Tischlayout-Schnittstelle: Es ist die Schnittstelle, die entsprechend der Tischordnung des Unternehmens gestaltet ist.
2. Auftragsschnittstelle: Dies ist die Schnittstelle, die vom Geschäftskunden erteilten Aufträge vor Mitarbeiter in das System verarbeitet werden.
3. Menüschnittstelle: Es ist die Schnittstelle, auf der neue Produkte hinzugefügt, vorhandene bearbeitet und entfernt werden können.
4. Login-Schnittstelle: Dies ist die erste Schnittstelle, über die Mitarbeiter mit ihren eigenen Passwörtern auf das System zugreifen können.
5. Zahlungsschnittstelle: Es ist die Schnittstelle, an der Gesamtschuldenbetrag des Geschäftskunden berechnet und die Zahlung empfangen wird.
6. Zahlungsverlaufsschnittstelle: Dies ist die Schnittstelle, in der Vergangenheit erhaltene Zahlungen aufgelistet werden.

Für Definitionen von Schnittstellen siehe das Kapitel 4 “Funktionale Anforderungen”.

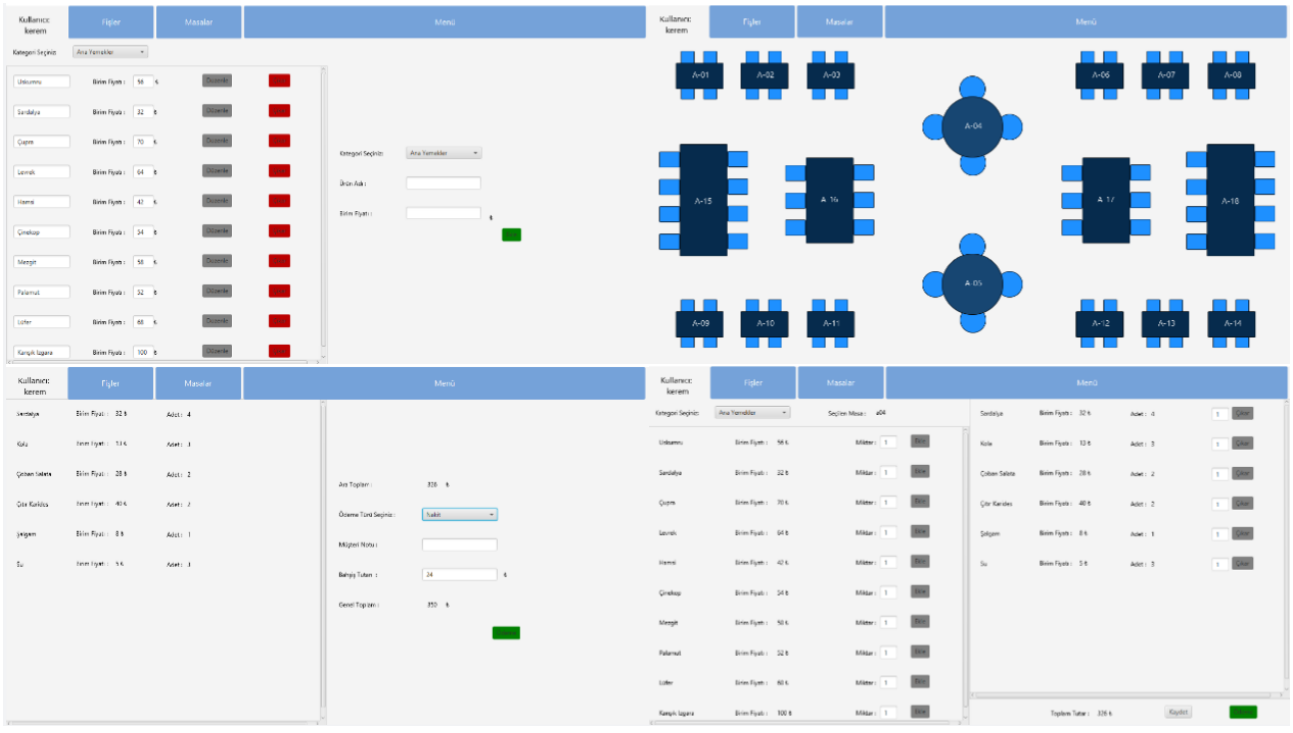
4. Funktionale Anforderungen

Einleitung

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

UI Use Cases

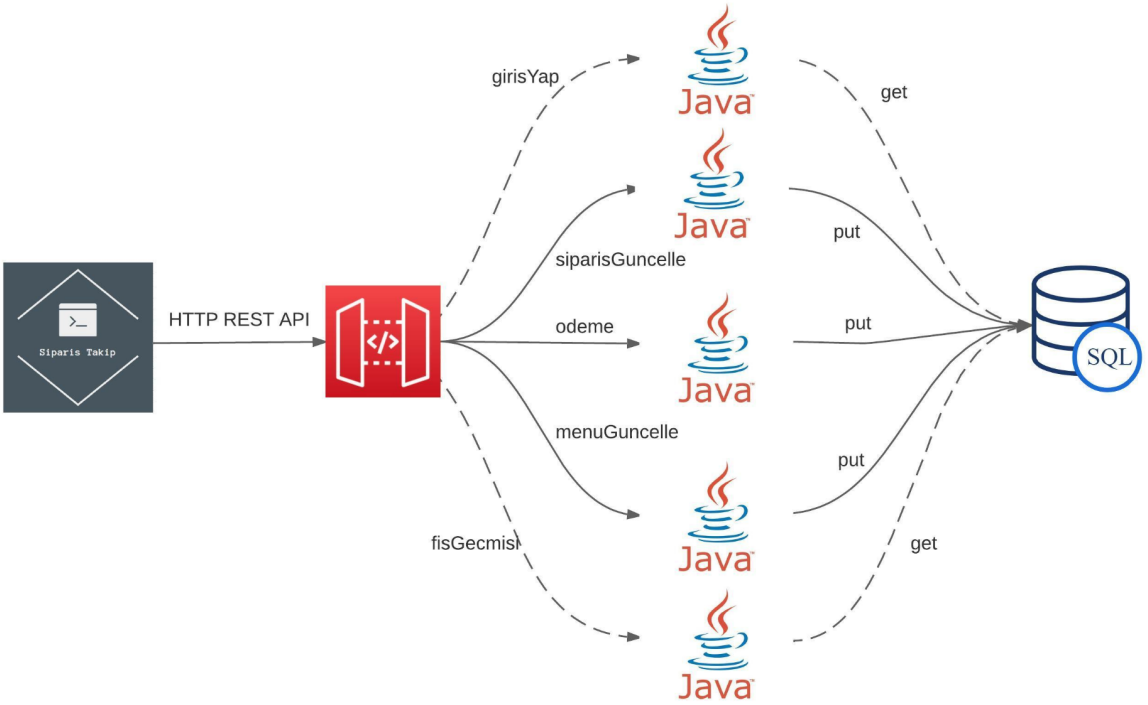
- **/UI-1/** Auf dem Anmeldebildschirm kann sich jeder Kellner mit seinem eigenen Passwort anmelden.
- **/UI-2/** In der Tische-Oberfläche werden alle Tische im System angezeigt, der zu bearbeitende ausgewählt und die Bestellungen können bearbeitet werden.
- **/UI-3/** Die Voucher-Schnittstelle hat eine Historie von zuvor erbrachten Dienstleistungen; Tischname, Ticketnummer, Zahlbetrag, Zahltag usw. kann betrachtet werden.
- **/UI-4/** Das vorhandene Menü wird von der Menüoberfläche angezeigt, es bietet die Möglichkeit zu bearbeiten und hinzuzufügen und zu entfernen.
- **/UI-5/** Die Zwischensumme des Benutzers wird auf der Zahlungsschnittstelle angezeigt und er/sie kann ein Trinkgeld hinterlassen, wenn er/sie möchte. Nach Auswahl der Zahlungsmethode zahlt der Nutzer.



API Use Cases

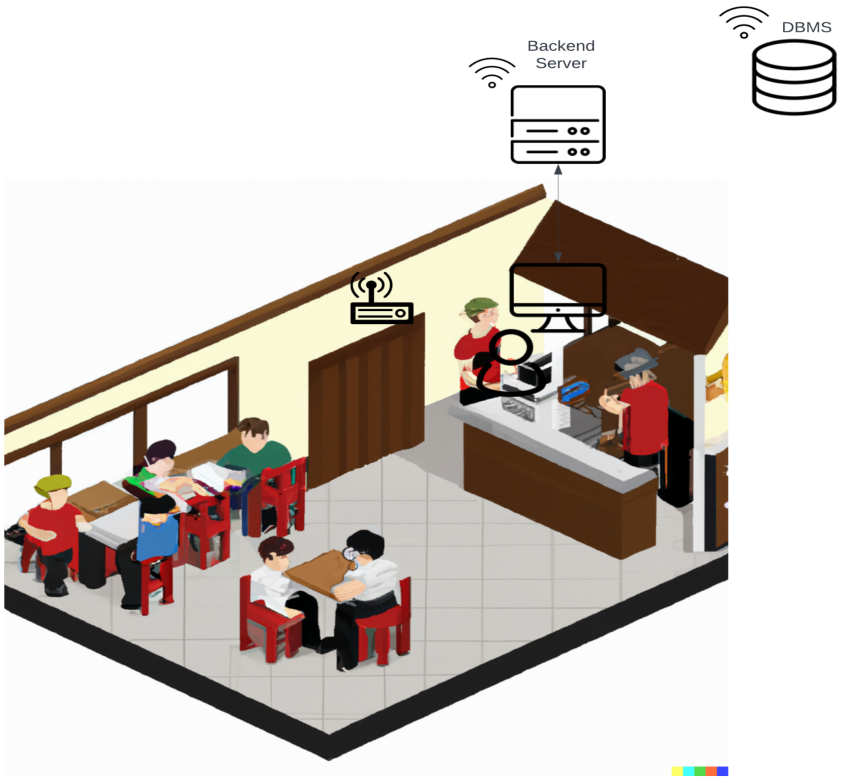
- **/API-1/** Stakeholder stellen REST-APIs in Swagger bereit. Die Schnittstelle ist in Swagger als RESTful Web Service definiert:
  - Projekt: https://app.swaggerhub.com/apis/javamudavimleri/SiparisTakip/1.0.0
  - Swagger Dokument: https://app.swaggerhub.com/apis-docs/javamudavimleri/SiparisTakip/1.0.0
  - Shared: https://app.swaggerhub.com/apis/javamudavimleri/SiparisTakip/1.0.0

- **/API-2/** Wie REST-APIs müssen auch interne APIs in Zusammenarbeit mit Stakeholdern entwickelt werden.



Technischen und fachliche Anforderungen

Order Tracking System



- **/SYS-1/** Nicht-Datenbankprogramme können veraltete Daten nach dem Neustart nicht behalten, daher ist eine SQL-Datenbank wie in unserem System erforderlich.

- **/SYS-2/** Ein Auftragsverfolgungssystem sollte eine verteilte Architektur haben. Die Back-End-Serverkomponente ist die zentrale Komponente, die die restlichen Komponenten steuert (Master-Slave-Beziehung), siehe Abbildung für die physische Systemarchitektur.

- **/SYS-3/** Der Datenaustausch zwischen OT-Schnittstelle und DBMS-Komponenten sollte nur über den Backend-Server erfolgen.

- **/SYS-4/** Der Resident steuert alle mit dem Netzwerk verbundenen Komponenten über die OT-Schnittstelle, die nur mit dem internen Server kommuniziert.

- **/SYS-5/** DBMS-Komponenten sollten nur mit dem Backend-Server interagieren.

Datenmodel

- **/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/** Das Datenmodell sollte die Parameter der API-Schnittstelle berücksichtigen.

- **/DAT-4/** Datenmodelle sollten in UML-Klassendiagrammen modelliert werden.

- **/DAT-5/** Programme, die keine Datenbank haben, können beim Neustart die vorherigen Daten nicht enthalten, daher wird wie in unserem System eine SQL-Datenbank benötigt.

5. Nichtfunktionale Anforderungen

Einleitung

Dieses Kapitel definiert nicht-funktionale Anforderungen an das System als Ganzes sowie an einzelne Systemkomponenten. Achten Sie besonders auf die Qualität der Software (Testing).

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

- **/SYS-1/** Die Kommunikation zwischen Systemkomponenten folgt den Einschränkungen der REST-Architektur und interagiert mit RESTful-Webservices.
- **/SYS-2/** Die Verbindung von Systemkomponenten mit den daran angeschlossenen Geräten (z. B. eine Systemkomponente mit Kassenterminal) ist fiktiv. Ist zur Realisierung einer benutzerdefinierten Option eine Kommunikation erforderlich, muss die Schnittstelle des angeschlossenen Gerätes „emuliert“ werden oder das Gerät als Emulation in einem eigenen Prozess laufen.
- **/SYS-3/** Bereitstellungsarchitektur: Alle Komponenten des Systems (OT-Schnittstelle, DBMS, Spring) müssen in einem eigenen Betriebssystemprozess ausgeführt werden, sodass das System über eine verteilte Architektur verfügt.

Nicht-funktionale Anforderungen an die Entwicklungsumgebung

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

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

- **/TOL-1/** Persistente Daten müssen in einer SQL-Datenbank gespeichert werden.
- **/TOL-2/** Der Backend-Server muss mit dem Java Framework implementiert werden.
- **/TOL-3/** Die Frontend-Anwendung muss mithilfe der JavaFX-Rich-Client-Technologie implementiert werden.

Nicht-funktionale Anforderungen an die Teststrategie (Qualitätssicherung)

- **/TEST-1/** Alle Use Cases sollen getestet und berichtet werden.
- **/TEST-2/** Alle User Stories müssen dokumentiert und getestet werden.
- **/TEST-3/** Alle Anforderungen müssen getestet und berichtet werden.
- **/TEST-4/** Jeder Systemdienst sollte in einer Blackbox angekreuzt werden, siehe ISTQB-2018.
- **/TEST-5/** Komponenten der OT-Schnittstelle (GUI) können unabhängig vom System getestet werden.

6. Abnahmekriterien

Das Projekt wird mit den folgenden Artefakten abgegeben:

- Dokumentation:
  - Pflichtenheft: *INF202-SiparisTakip-Pflichtenheft-2023.v1.0.docx*
- Software
  - Link zu GitHub Projekt: *https://github.com/Gruppe1-Fulya/siparistakip*
- Evidenz:
  - System/Software-Demo via Videoclip:*www.youtube.com/watch?v=KT7KDGQwkGI*

Anm.: Die Abgabetermine der Projetartefakts werden durch den Stakeholder festgelegt!

7. Projekt Meilensteine

Folgende Meilensteine sind verbindlich definiert:

- Meilenstein #1: Das Lastenheft ist fertiggestellt und mit dem Stakeholder abgestimmt.
- Meilenstein #2: Das Pflichtenheft ist fertiggestellt und mit dem Stakeholder abgestimmt.
- Meilenstein #3: (An diesem Meilenstein ist die Architektur im Vordergrund.)
    - Komponenten-basierte Architektur ist entworfen und komponentenweise implementiert.
    - Die externen Schnittstellen (Webservices) wurden entworfen/implementiert.
    - Die GUI Komponente ist ansatzweise fertig.
- Meilenstein #4: (An diesem Meilenstein ist das Testen im Vordergrund.)
    - Die Test-Cases wurden aus den Use Cases abgeleitet und ansatzweise beschrieben.
    - Die Testumgebung ist vorbereitet.
- Meilenstein #5
    - Das Projekt ist per Vereinbarung abgegeben.

8. Referenzen