TAU INF202 Software Engineering

Individuelles Projekt

Pflichtenheft

Projektdokumentation

Version: 2023.v0.9

Status: Entwurf

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

Bestandsführungssystem (Inventory management system)

Verantwortliche/r:

Muhammed El Ibrahim, e160501309@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	13.03.2023	M. El Ibrahim	Einführung in das System hinzugefügt Definierte Projektziele Aktualisierung der Entwicklungstools auf Java, SQL und JavaFX

**) 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

tool-name

Inhaltsverzeichnis

1. Einleitung

Dieses Dokument umreißt die Anforderungen für die Entwicklung einer neuen Softwareanwendung, die mit Java, SQL und JavaFX entwickelt werden soll. Das Ziel des Projekts ist die Erstellung einer Desktop-Anwendung, die eine benutzerfreundliche Schnittstelle für ein Bestandsmanagementsystem bereitstellt. Das System soll so konzipiert werden, dass Benutzer Bestände verwalten, Bestellungen verfolgen und Berichte generieren können. Es wird sich um eine Desktop-Anwendung handeln, die eine Datenbank zur Speicherung von Bestandsdaten nutzt. Ziel dieses Projekts ist es, eine effiziente und benutzerfreundliche Lösung für das Bestandsmanagement zu bieten, die den Anforderungen unserer Kunden entspricht.

Wir werden daran arbeiten, die Softwareanwendung zu entwerfen, zu implementieren, zu testen und bereitzustellen. Das Projekt wird einem strukturierten Entwicklungsprozess folgen und bewährte Verfahren der Softwareentwicklung einhalten, um sicherzustellen, dass die Anwendung von hoher Qualität, zuverlässig und skalierbar ist.

Die Benutzeroberfläche wird intuitiv und einfach zu navigieren sein, mit klarer und präziser Beschriftung und Organisation der Daten. Das System wird auch robuste Sicherheitsmaßnahmen haben, um sensible Bestandsdaten zu schützen.

Die folgenden Abschnitte umreißen die Ausgangssituation und Ziele, die Gesamtarchitektur, funktionale und nicht-funktionale Anforderungen, Akzeptanzkriterien, Projektmeilensteine und Referenzen für das Projekt.

2. Ausgangssituation und Ziele

In diesem Kapitel werden die Ausgangssituation und der Grund zur Wahl des Projektthemas anschaulich dargestellt.

Einleitung

Dieses Kapitel zielt darauf ab, eine detaillierte Beschreibung der Ausgangssituation und Ziele des Projekts für das Bestandsmanagement-System zu liefern. Das Bestandsmanagement-System ist eine Softwareanwendung, die Organisationen dabei hilft, ihren Bestand effizient zu verwalten. Dieses System wird mit der Java-Programmiersprache und SQL für die Datenbankverwaltung sowie JavaFX für die Benutzeroberfläche entwickelt.

Problemstellung (Funktionalität)

Das manuelle Bestandsmanagement-System ist ein zeitaufwendiger Prozess, der viel Mühe erfordert, um Bestandsniveaus, Verkäufe und Einkäufe zu verwalten. Außerdem kann es zu ungenauer Bestandsverfolgung führen, was zu verlorenen Verkäufen, verschwendeter Zeit und verringerte Effizienz führen kann. Die Problemstellung des Bestandsmanagement-Systems besteht darin, eine Softwareanwendung zu entwickeln, die den Bestandsmanagementprozess automatisiert und genaue Bestandsverfolgung, Verkaufs- und Einkaufsdaten bereitstellt.

Stakeholder (Anwender):
Es wird beschrieben, welche Zielgruppen durch den Einsatz des neuen Systems am meisten profitieren.

Systemumfeld (Einsatzumgebung)

Das Bestandsmanagement-System wird in einer verteilten Umgebung eingesetzt, in der die Server- und Client-Komponenten auf separaten Maschinen eingesetzt werden. Die Server-Komponente wird auf einem Windows-Server-Betriebssystem eingesetzt, während die Client-Komponente auf Windows-Desktops eingesetzt wird.

Rahmenbedingung (Einschränkungen)

Das Bestandsmanagement-System muss die folgenden technischen Anforderungen erfüllen:
Das System muss mit der Java-Programmiersprache entwickelt werden.
Das Datenbankmanagementsystem muss SQL verwenden.
Die Benutzeroberfläche muss mit JavaFX entwickelt werden.
Das System muss mit Windows-Betriebssystemen kompatibel sein.
Das System muss skalierbar sein, um große Mengen an Bestandsdaten zu verarbeiten.
Das System muss sicher sein und gegen unbefugten Zugriff geschützt sein.

Ziele (Lösung)

Das Ziel des Bestandsmanagement-Systems besteht darin, den Bestandsmanagementprozess zu automatisieren und genaue Bestandsverfolgung, Verkaufs- und Einkaufsdaten bereitzustellen. Darüber hinaus zielt das System darauf ab, die organisatorische Effizienz zu verbessern, die Kosten für das manuelle Bestandsmanagement zu reduzieren und den Umsatz zu steigern, indem Stockouts verhindert werden.

3. Gesamtarchitektur

Um die Visualisierung der Anwenderanforderungen zu ermöglichen, soll Gesamtsystemarchitektur illustriert werden, die die Sichtweise des Anwenders repräsentiert und nicht die technische Sichtweise des Systemanalytikers beziehungsweise des Systemarchitekten.

Es soll eine konzeptionelle Architektur unter der Berücksichtigung der Kommunikation (Interaktionen) mit den externen Systemen erstellt werden (Anm.: Vergleich mit einer technischen Systemarchitektur).

Darüber hinaus, in der Gesamtsystemarchitektur sollten die zukünftigen Systembestandteile (Komponente) identifiziert und festgeschrieben werden.

Einleitung
Eine kurze Beschreibung, was dieses Kapitel beinhaltet.

Gesamtarchitektur
Die Illustration der konzeptionellen Architektur durch die funktionellen Blöcken in free-style oder in UML-Diagrammen soll erstellt werden, wobei die wichtigste Komponente und Schnittstellen hervorgehoben sind.

Komponente <x>
Die hervorgehobenen Komponenten und Komponentenschnittstellen sollen hier beschrieben werden.

Externe Schnittstellen
Die hervorgehobenen Systemschnittstellen sollen hier beschrieben werden.

4. Funktionale Anforderungen

Funktionale Anforderungen beschreiben die Gesamtfunktionalität eines Systems, die die Zielgruppe erwartet, um mit Hilfe des Systems ein Problem zu lösen.

Die Anforderungen werden aus Ablaufbeschreibungen (User Stories) zur Nutzung des Systems abgeleitet. Aus diesem Grund, die Beschreibung der funktionalen Anforderungen erfolgt durch die Anwendungsfällen (Use Cases). Ein Anwendungsfall beschreibt die Interaktion zwischen dem Anwender (oder einem externen System) und dem zu realisierenden System abläuft.
Die Gesamtheit der Anwendungsfälle definiert das Systemverhalten und hilft enorm ein System aus der Sicht eines Anwenders zu definieren.

Die technischen und fachlichen Vorgaben werden als einzelnen Anforderungen definiert und den Use Cases zugeordnet. Solche Anforderungen sollen kurz, prägnant und direkt (active voice) formuliert werden

Darüber hinaus, im Rahmen der funktionalen Anforderungen wird ein erstes Datenmodell erstellt, das sowohl ein fachliches, aber auch schnittstellenspezifisches Datenmodell beinhaltet. Das letztere ist besonders wichtig, wenn die Systemfunktionalität auch als API zur Verfügung gestellt werden soll.

Einleitung
Eine kurze Beschreibung, was dieses Kapitel beinhaltet.

UI Use Cases
Ein repräsentativer Entwurf der graphische User Interface soll schematisch mit Hilfe von UI-Techniken wie Wireframes (dt. Drahtgittermodelle oder Funktionslayout) skizziert werden (Skizzen für bildschirmbasierte Anwendungen)

API Use Cases
Ein API Use Case fungiert als eindeutige RESTful-Schnittstelle, die eine Anforderung vom Client an den Server weiterleitet. Die API stellt die Systemfunktionalität bereit und dadurch den Zugriff auf die Geschäftsdaten durch a RESTful Service Interface (API-Gateway).

Technischen und fachliche Anforderungen
Die einzelnen Anforderungen, die in Use Cases nicht explizit erfasst werden können, sollen kurz, prägnant und direkt (active voice) formuliert werden. Diese Anforderungen sind für den Anwender transparent wie softwaretechnische Vorgaben.

Datenmodel
Ein fachliches und schnittstellenspezifisches Datenmodell ist notwendig um die persistenten Daten (Datenbank) und API spezifische Daten (API-Gateway) zu definieren.

Anmerkung: Alternativ zur API-Schnittstelle kann eine CLI-Schnittstelle realisiert werden.

5. Nichtfunktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben Anforderungen an das System, die nicht-fachlicher Natur sind, z.B. Qualitätsanforderungen, Sicherheitsanforderungen oder Performanceanforderungen. Solche Anforderungen sollen im Architekturentwurf berücksichtigt werden.

Im Allgemeinen werden diejenigen Anforderungen, die keine funktionalen Anforderungen sind, als nicht-funktionalen Anforderungen deklarieret.

Nicht-funktionale Anforderungen sollen messbar beschrieben sein.

Einleitung
Eine kurze Beschreibung, was dieses Kapitel beinhaltet.

Nicht-funktionale Anforderungen an die Systemarchitektur (Architekturmuster, Deployment)
Die nicht-funktionalen Anforderungen an die Systemarchitektur sollen hier aufgelistet sein.

Nicht-funktionale Anforderungen an die Entwicklungsumgebung
Die nicht-funktionalen Anforderungen an die Entwicklungsumgebung sollen hier aufgelistet sein.

Nicht-funktionale Anforderungen an die Entwicklungswerkzeuge (Sprache, IDE, Frameworks)
Die nicht-funktionalen Anforderungen an die Entwicklungswerkzeuge sollen hier aufgelistet sein.

Nicht-funktionale Anforderungen an die Teststrategie (Qualitätssicherung)
Die nicht-funktionalen Anforderungen an die Teststrategie sollen hier beschrieben sein.

6. Abnahmekriterien

Die Abnahmekriterien sind durch den Stakeholder definiert und sie dürfen nur mit Zustimmung des Stakeholders neu definiert, geändert oder erweitert werden.

Das Projekt wird mit den folgenden Artefakten abgegeben:

- Dokumentation:
  - Pflichtenheft: *file-name.docx*
- Software
  - Link zu GitHub Projekt: *git-hub-link*
- Evidenz:
  - System/Software-Demo via Videoclip: *videoclip-link*

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

7. Projekt Meilensteine

Diese Kapitel beinhaltet die wichtigsten Meilensteine, die der Stakeholder festgelegt hat.

8. Referenzen

Diese Kapitel beinhaltet die wichtigsten Literaturquellen aufgelistet.