TaskFlow - Anforderungsspezifikation

1. Produktübersicht

TaskFlow ist ein Task-Management-System, das es Benutzern ermöglicht, persönliche Aufgaben zu verwalten. Das System bietet sowohl eine lokale Datenspeicherung als auch einen Zugriff (zur Synchronisation) über eine REST-API an. TaskFlow wird hauptsächlich in Firmen eingesetzt und dient als Arbeitsmittel für die MitabeiterInnen, um ihre persönlichen Aufgaben zu planen.

2. Funktionale Anforderungen

2.1 Aufgabenverwaltung

Das System muss es Benutzern ermöglichen, Aufgaben zu erstellen, die einen Titel, eine optionale Beschreibung, eine Priorität (NIEDRIG, MITTEL, HOCH) und ein optionales Fälligkeitsdatum enthalten. Jede Aufgabe erhält automatisch eine eindeutige ID und einen Zeitstempel der Erstellung.

Benutzer müssen in der Lage sein, existierende Aufgaben zu bearbeiten und dabei alle Attribute außer der ID und dem Erstellungszeitpunkt zu ändern. Das System muss eine Funktion zur Löschung von Aufgaben bereitstellen, wobei eine Bestätigung erforderlich ist.

Der Status einer Aufgabe kann zwischen OFFEN, IN_BEARBEITUNG und ABGESCHLOSSEN gewechselt werden. Bei Statusänderungen wird automatisch ein Zeitstempel gespeichert.

2.2 Aufgabenansicht und Filterung

Das System muss eine Listenansicht aller Aufgaben bereitstellen, die standardmäßig nach Priorität und dann nach Fälligkeitsdatum sortiert ist. Benutzer können Aufgaben nach verschiedenen Kriterien filtern: Status, Priorität, Fälligkeitsdatum (überfällig, heute, diese Woche) und Suchbegriffe im Titel oder in der Beschreibung.

Eine Detailansicht für einzelne Aufgaben muss alle Informationen inklusive der Historie von Statusänderungen anzeigen. Das System soll eine Statistikübersicht bereitstellen, die die Anzahl der Aufgaben pro Status und die Anzahl überfälliger Aufgaben zeigt.

2.3 Datenpersistierung

Alle Aufgaben werden in einer Sqlite Datenbank gespeichert. Das System muss beim Start automatisch die gespeicherten Daten laden und bei jeder Änderung die Daten aktualisieren. Eine Backup-Funktion soll die aktuelle Datenbasis in eine separate JSON Datei mit Zeitstempel exportieren können.

2.4 REST-API Integration

Das System startet auch einen eingebetteten HTTP-Server, der eine REST-API bereitstellt. Die API muss folgende Endpunkte unterstützen:

- GET /api/tasks - Alle Aufgaben abrufen
- GET /api/tasks/{id} - Einzelne Aufgabe abrufen
- POST /api/tasks - Neue Aufgabe erstellen
- PUT /api/tasks/{id} - Aufgabe aktualisieren

- DELETE /api/tasks/{id} - Aufgabe löschen
- GET /api/tasks/stats - Statistiken abrufen

Die API muss JSON als Datenformat verwenden und entsprechende HTTP-Statuscodes zurückgeben.

2.5 Benachrichtigungen

Das System soll beim Start prüfen, ob Aufgaben überfällig sind oder heute fällig werden, und diese in einer speziellen Ansicht hervorheben. Optional kann eine Desktop-Benachrichtigung für überfällige Aufgaben angezeigt werden.

2.6 Einschränkungen

Eine Benutzerverwaltung ist optional. Alle Benutzer können ohne Anmeldung auf die Anwendung zugreifen.

3. Nicht-funktionale Anforderungen

3.1 Benutzerfreundlichkeit

Das Userinterface soll selbsterklärend und intuitiv verwendbar sein. Fehlermeldungen müssen klar und hilfreich sein, mit Hinweisen zur korrekten Verwendung.

3.2 Performance

Das System muss in der Lage sein, mindestens 100.000 Aufgaben ohne merkliche Verzögerung zu verwalten. Die Startzeit der Anwendung darf 2 Sekunden nicht überschreiten. Filteroperationen müssen innerhalb von 100ms Ergebnisse liefern.

3.3 Zuverlässigkeit

Das System muss robust gegen fehlerhafte Eingaben sein und darf bei ungültigen Befehlen nicht abstürzen. Die Datenspeicherung muss so erfolgen, dass bei Abbrüchen keine Datenverluste entstehen. Eine hohe Verfügbarkeit>99% muss gegeben sein, da kritische Aufgaben mit der Anwendung geplant werden.

3.4 Wartbarkeit

Der Code muss modular aufgebaut sein mit klarer Trennung zwischen den implementierten Schichten aufweisen (Präsentation, Daten, Logik ...). Eine Testabdeckung von mindestens 80% für die Geschäftslogik ist anzustreben.

3.5 Implementierung

Es stehen zwei Varianten zur Verfügung:

- Variante 1: Implementierung als Webanwendung (Spring o.ä.)
- Variante 2: Implementierung als Desktop Anwendung (Swing)

3.6 Portabilität

- Variante 1: Das System muss auf Firefox, Chrome und Safari lauffähig sein.
- Variante 2: Das System muss auf Windows, Linux und macOS lauffähig sein.

4. Technische Rahmenbedingungen

- Programmiersprache: Java 17 oder höher
- Build-Tool: Maven
- User Interface: Web - Javalin, Spring Boot o.a. oder Swing
- Datenbank: sqlite
- JSON-Verarbeitung: Jackson
- REST-Framework: Javalin oder Spring Boot (embedded)
- Testframeworks: JUnit 5, Mockito

5. Abnahmekriterien

Das System ist abnahmereif, wenn:

- Alle funktionalen Anforderungen implementiert und getestet sind
- Die Testabdeckung mindestens 80% beträgt
- Keine kritischen Bugs bekannt sind
- Eine Benutzerdokumentation vorhanden ist
- Der Code den vereinbarten Coding-Standards entspricht