**Homework 1 – Requirements**

Name: Simon Butz

Ilias: sbutz3

Matrikelnummer: 7425743

**Exercise 1: Stakeholders)**

**a)**

**Professoren/Dozenten:**

**Betroffenheit und Interesse:** Dozenten sind direkt in die Organisation und Verwaltung der Übungsgruppen involviert. Sie erstellen Gruppen und legen Zeiten sowie Gruppengrößen fest. Das System vereinfacht ihre Verwaltungsaufgaben und ermöglicht eine faire Verteilung der Studierenden auf Übungsgruppen.

**Studierende:**

**Betroffenheit und Interesse:** Studierende sind die Hauptnutzer des Systems, da sie sich für Übungsgruppen anmelden müssen. Sie haben ein großes Interesse an der fairen Zuteilung, die ihren Zeitplan und eventuelle Kursüberschneidungen berücksichtigt.

**IT-Entwicklungsteam (IT Development Team):**

**Betroffenheit und Interesse**: Das IT-Entwicklungsteam setzt das EDG-System um, einschließlich Architektur, Programmierung, Integration und Tests, gemäß den Anforderungen der Stakeholder und Zeitpläne. Ihr Interesse liegt im erfolgreichen Projektabschluss innerhalb des Budgets, da dies den Projekterfolg direkt beeinflusst. Ein klarer Feedback-Prozess unterstützt sie, die Bedürfnisse der Nutzer optimal zu berücksichtigen.

**Universitätsverwaltung:**

**Betroffenheit und Interesse:** Da das System universitätsweit implementiert werden könnte, ist die Universitätsverwaltung an der zukünftigen Skalierbarkeit und den Auswirkungen auf die Ressourcen der Universität interessiert.

b)

**Ein Bild, das Text, Rechteck, Reihe, Diagramm enthält.

Automatisch generierte Beschreibung**

Dozenten/Professoren

Universitätsverwaltung

Studierende

IT-Entwicklungsteam

**Exercise 2: Requirements)**

**a) Functional Requirements:**

1. Das System ermöglicht Dozenten die Erstellung von Übungsgruppen mit Details wie Zeiten und Gruppengröße.
2. Studierende können sich im System anmelden, um verfügbare Übungsgruppen einzusehen.
3. Studierende können Zeiten markieren, in denen sie für Übungsgruppen NICHT verfügbar sind.
4. Das System berücksichtigt die angegebenen Verfügbarkeiten und Kursüberschneidungen bei der Gruppenzuteilung.
5. Das System verteilt die Studierenden automatisch auf die Gruppen, um eine faire Verteilung sicherzustellen.
6. Studierende werden über ihre Gruppenzuteilung per Benachrichtigung informiert.

**b) Quality Requirements und Quality Attribute:**

1. **Benutzerfreundlichkeit:** Die Software soll einfach zu bedienen sein, um die Interaktion für Studierende und Dozenten zu erleichtern.
2. **Zuverlässigkeit:** Das System muss zuverlässig arbeiten, um während der Registrierungsphasen stabil und verfügbar zu sein.
3. **Sicherheit (Security):** Persönliche Daten sollen geschützt sein, und der Zugriff erfolgt nur über Universitätszugänge (Shibboleth).

**c) Constraint:**

* Das System muss in Java entwickelt werden.

**d) Project Requirement:**

* Das System soll mit einem Gesamtbudget von 70.000 Euro entwickelt werden.

**e) Process Requirement:**

* Die Entwicklung soll so erfolgen, dass das System bis zum Wintersemester 2026/27 einsatzbereit ist, wobei erste Testversionen zu Beginn des Wintersemesters 2025/26 verfügbar sein sollen.

**Exercise 3: Requirements Validation)**

1. Anforderungen präziser formulieren:
   * Details hinzufügen, die unklare Begriffe spezifizieren (z. B. was genau „faire Verteilung“ bedeutet oder wie „Benutzerfreundlichkeit“ gemessen wird).
   * Wenn möglich, spezifische Werte oder Schwellen hinzufügen (z. B. Verfügbarkeitsrate, Performance-Werte, Zufriedenheitsraten).
2. Konsistenz sicherstellen:
   * Anforderungen so formulieren, dass keine widersprüchlichen Erwartungen entstehen. Beispielsweise sollten Anforderungen zu Benutzerfreundlichkeit und Sicherheit aufeinander abgestimmt sein, da zusätzliche Sicherheitsfunktionen oft die Bedienbarkeit beeinflussen.
   * Einheitliche Begriffe und eine klare Terminologie verwenden, um Missverständnisse zu vermeiden.
3. Überprüfbarkeit verbessern:
   * Anforderungen so konkret wie möglich gestalten, damit klare Erfolgskriterien entstehen, die sich testen oder messen lassen (z. B. „Das System soll eine Verfügbarkeit von 99,5% haben“ statt nur „zuverlässig sein“).
   * Testbare Qualitätsanforderungen definieren (z. B. „Usability-Tests mit einer Zufriedenheitsrate von mindestens 80%“).
4. Gültigkeit und Relevanz prüfen:
   * Anforderungen regelmäßig mit den Stakeholdern abstimmen, um sicherzustellen, dass sie wirklich ihren Bedürfnissen entsprechen.
   * Nur die Anforderungen aufnehmen, die einen klaren Mehrwert für das Projekt haben und sich im Rahmen des Budgets und Zeitplans umsetzen lassen.
5. Eindeutige Anforderungen durch Beispiele und Szenarien:
   * Beispiele, User Stories oder Szenarien hinzufügen, die zeigen, wie eine Funktion konkret genutzt wird oder wie das System in verschiedenen Szenarien reagieren soll.
   * Dies hilft, die Anforderungen greifbarer und verständlicher zu machen.
6. Meilensteine und Fortschrittskontrolle einbauen:
   * Insbesondere bei zeitkritischen Anforderungen Zwischenziele (Meilensteine) definieren, die den Fortschritt überprüfbar machen und bei Verzögerungen rechtzeitig reagieren lassen.
   * Fortschrittsberichte und regelmäßige Reviews festlegen, um die Anforderungen kontinuierlich an die Projektrealität anzupassen.

**Exercise 4: Use Case)**

**Use Case Name:**  
Student Registration and Scheduling for Exercise Groups

**Primary Actor:**  
Student

**Stakeholders and Interests:**

1. **Student:** Needs to register for exercise groups that align with their schedule and preferences.
2. **Lecturers:** Require students to be enrolled in exercise groups with minimal scheduling conflicts.
3. **Course Coordinators:** Need to manage lab and exercise group enrollments to maximize course availability.
4. **System Administrators:** Need the system to function smoothly, supporting fair and conflict-free scheduling.

**Preconditions:**

* The student is enrolled in one or more courses with associated exercise or lab groups.
* The student has university credentials to access the system (assumed integration with Shibboleth for authentication).

**Postconditions:**

* The student is either successfully assigned to all requested exercise groups or receives a notification about unresolved conflicts, with possible next steps.

**Main Success Scenario (Basic Flow):**

1. The student logs into the Exercise Group Distribution (EGD) system using university credentials.
2. The student views the list of enrolled courses and their available exercise or lab groups.
3. The student enters their availability (or marks times they are unavailable).
4. The system processes the student’s availability and course enrollment, attempting to assign them to conflict-free exercise groups.
5. If all exercise groups are assigned without conflicts, the student receives a confirmation with their full schedule via email and/or system notification.
6. The student logs out.

**Extensions (Alternative Flows):**

1. **Conflict with Exercise Group Assignment:**
   * *Condition:* The system cannot find a conflict-free schedule for one or more exercise groups.
   * *Steps:*
     1. The system notifies the student of the conflict and lists the conflicting courses/groups.
     2. The student receives guidance to contact the respective course coordinators or instructors for manual adjustment.
2. **Limited Group Availability:**
   * *Condition:* Only a few exercise groups are available, and they conflict with each other or the student’s availability.
   * *Steps:*
     1. The system assigns the student to the exercise group with the highest priority (e.g., earliest registration date or based on course requirements).
     2. The system provides the student with the option to contact course coordinators or instructors to resolve scheduling conflicts.
3. **Manual Adjustment and Notification (Assumed):**
   * *Condition:* The student successfully resolves the scheduling conflict externally (e.g., through discussion with an instructor).
   * *Steps:*
     1. The course coordinator manually adjusts the student’s schedule in the system.
     2. The system updates the student’s schedule and sends a confirmation notification or email.

**Assumptions:**

* The system supports student authentication through the university’s Shibboleth system.
* Instructors or course coordinators have the ability to manually adjust schedules for exceptional cases.
* The system sends automatic email or in-app notifications to students regarding scheduling outcomes and conflicts.

**Special Requirements:**

* The system must provide user-friendly interfaces for availability input and conflict notifications.
* High system availability during peak registration periods (e.g., start of semester).
* Data protection protocols are in place to secure students' personal information and course schedules.