Homework 1

Group members:

Charles Assongmo Tazanou

Tyra Omolara Adeoye

Tim Poppen

Exercise 1

The list of the six keys stakeholders, followed by a power/interest grid to organise them according to their influence and engagement in the project:

a) The stakeholders:

1. **Students**:
   . As the primary end-users, students depend on the system for fair group assignments that fit into their schedules.
   . Students want a transparent and fair system to prevent schedule conflicts and maximise their chances of getting a convenient group assignment.

2. **Lecturers**:
   . They create and manage exercise groups, set session times, regulate groups sizes and address scheduling complaints.
   . They want an efficient, fair system that reduces administrative workload and considers students' availability.

3. **System administrators**:
   . They maintain and scale the system for peak performance, manage access via Shibboleth, and ensure data security.
   . They prioritize system stability, scalability, security, and seamless integration with other university systems.

4. **Developers (Students and external):**
   . They design, build, and test the system, ensuring that it meets the functional and quality standards.
   . Student developers gain experience, while external developers focus on successful delivery.

5. **University management (Computer science department):**

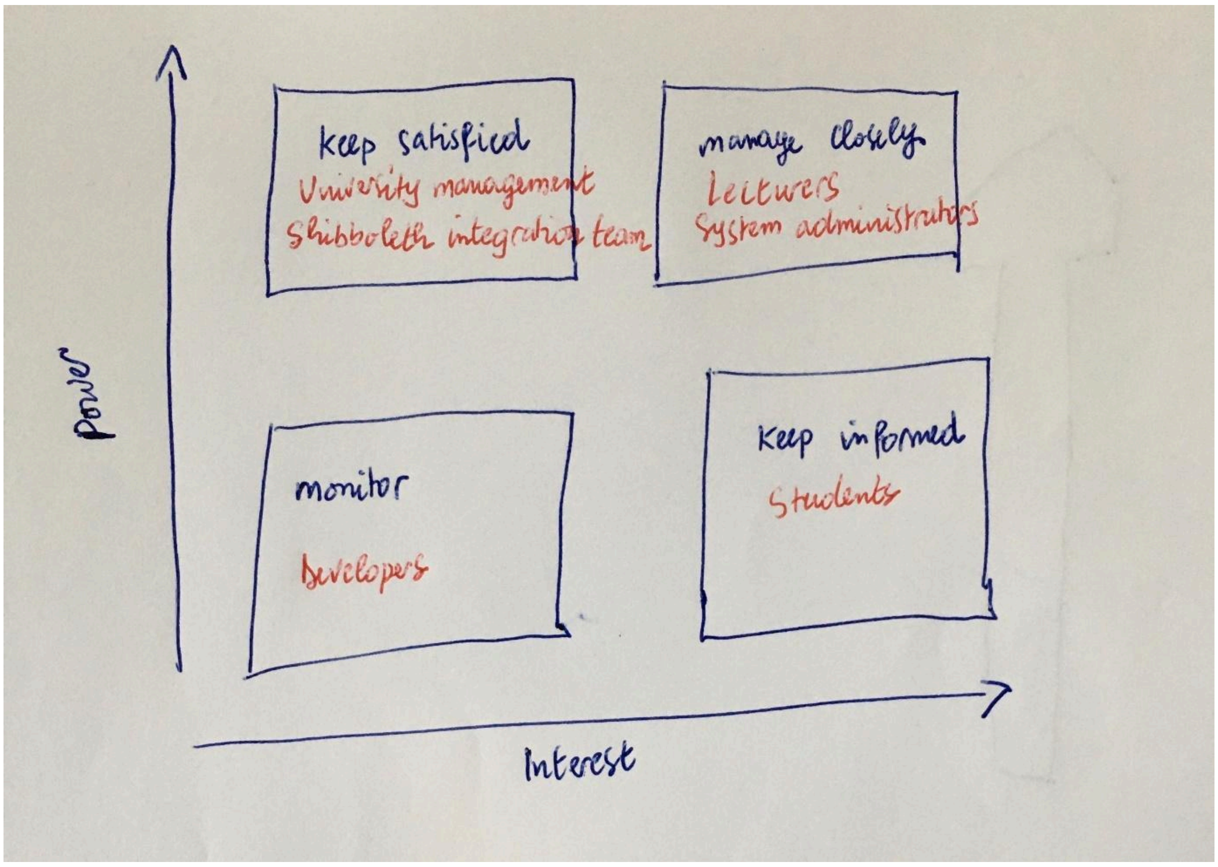
. As project sponsor, they ensure that the system aligns with academic goals and policies, overseeing funding and strategic direction.
. They are focused on project success and potential expansion but are involved mainly in oversight and approval, not direct use.

6. **Shibboleth integration team:**

   . They handle integration with Shibboleth, ensuring that access is restricted to authorised users and data protection standards are upheld.

   . Their focus is on secure, compliant integration rather than the system's functionality or operations.

b) Power/interest grid:



Exercise 2

a) Six functional requirements

1. **User registration:** The system shall allow students to register for all exercise groups they wish to attend for multiple courses in one semester.

2. **Exercise group creation:** The system shall allow lecturers to create exercise groups, specifying session times and the maximum number of students per group.

3. **Time availability input:** The system shall allow students to mark time slots where they are unavailable due to conflicts with other courses.

4. **Automated group assignment:** The system shall automatically distribute students across exercise groups based on their available time slots and course conflicts.

5. **Notification system:** The system shall notify students of the exercise groups they have been assigned to.

6. **Manual administration:** The system shall provide a mechanism for manual administration to handle cases where students cannot be assigned to any group.

b) 3 quality requirements and their respective quality attributes

1. The system shall be able to scale to handle thousands of students, especially during peak registration periods. **(Scalability)**
2. The system shall protect personal data from unauthorised access and ensure access control through university credentials. **(Security)**

3. The system shall be easy to use for both students and lecturers, with an intuitive user interface. **(Usability)**

c) 1 constraint

Java development: The system shall be developed using java to allow student developers from the department of computer science to participate.

d) 1 project requirement

Budget limitation: The total budget for the system development shall not exceed 70. 000 €

e) 1 process requirement

Student Participation in Development: The development process shall involve students as developers and testers.

Exercise 3

a) Functional requirements

1. User registration

- **Precision**: Partially fulfilled. "Register for all exercise groups" could be ambiguous. Are students registering for groups or expressing interest based on their availability?
- **Consistency**: Fulfilled. No conflicting information in the current context.
- **Verifiability**: Not fulfilled. It is unclear how we will verify that the system allows registration for multiple courses.
- **Validity**: Valid. This is a necessary feature for the system.
- **Improvement**: The system shall allow students to register their interest in all exercise groups for their courses by selecting their availability for the semester.

2. Exercise group creation

- **Precision**: Partially fulfilled. "Create exercise groups" could be more specific in terms of what details are needed (e.g. session time, capacity, location).
- **Consistency**: Fulfilled. No conflicting information in the current context.
- **Verifiability**: Partially fulfilled. Needs more details on what information needs to be entered.
- **Validity**: Valid. This is a necessary feature for the system.
- **Improvement**: The system shall allow lecturers to create exercise groups by specifying session time, location and the maximum number of students allow per group.

3. Time availability input

- **Precision**: Partially fulfilled. The wording “mark time slots” lacks specific details (e.g., format, granularity of time).

- **Consistency**: Fulfilled. Consistent with other requirements.

- **Verifiability**: Partially fulfilled. Needs more clarity on how marking availability would be implemented.
- **Validity**: Valid. This is essential for conflict resolution in the system.

- **Improvement**: The system shall allow students to mark unavailable time slots for group sessions by selecting specific time ranges in a weekly schedule.

4. Automated group assignment

- **Precision**: Partially fulfilled. “Distribute students” is vague; the criteria to distribution should be clearer.
- **Consistency**: Fulfilled. No conflicting information in the current context.

- **Verifiability**: Not fulfilled. No clear way to verify if the system minimizes conflicts.

- **Validity**: Valid. This is a necessary feature for the system.

- **Improvement**: The system shall automatically assign students to exercise groups based on their marked availability and course conflicts, aiming to minimize unassigned students.

5. Notification system

- **Precision**: Partially fulfilled. It should be specified how notification is sent (e.g., email, system alerts).

- **Consistency**: Fulfilled. No conflicting information in the current context.

- **Verifiability**: Partially fulfilled. More details are needed on how notifications will be verified.

- **Validity**: Valid. Students need to be informed on their assignments.

- **Improvement**: The system shall notify students on their assigned exercise groups via both email and system alerts.

6. Manual administration

- **Precision**: Not fully fulfilled. “Mechanism for manual administration” lacks detail (e.g., who handles it, what tools they use).

- **Consistency**: Fulfilled. No conflicting information in the current context.

- **Verifiability**: Not fully fulfilled. Needs more details on how manual assignments will be tracked.
- **Validity**: Valid. Manual intervention will be required for unassigned students.

- **Improvement**: The system shall provide administrative tools for lecturers or system administrators to manually assign students to exercise groups in case of conflicts or unassigned students.

b) quality requirements

. Scalability

- **Precision**: Partially fulfilled. The requirement states the system should handle thousands of students but lacks specific performance metrics.
- **Consistency**: Fulfilled. No conflicting information in the current context.

- **Verifiability**: Partially fulfilled. Needs measurable criteria to test scalability.

- **Validity**: Valid. Essential for future adoption university-wide.

- **Improvement**: Specify expected performance under peak load (e.g., "The system shall support up to 10,000 concurrent users with response times under 2 seconds").

. **Security**

- **Precision**: Partially fulfilled. “Protect personal data” and “access control” need specifics on how this will be done (e.g., encryption, authentication).

- **Consistency**: Fulfilled. No conflicting information in the current context.

- **Verifiability**: Not fully fulfilled. Needs clear criteria for testing security features.

- **Validity**: Valid. This is required for protecting student data.

- **Improvement**: The system shall use encrypted connections and authenticate users via shibboleth to ensure data protection and secure access.

. Usability

- **Precision**: Not fully fulfilled. “Easy to use” is too vague. Usability metrics should be defined clearly.
- **Consistency**: Fulfilled. No conflicting information in the current context.
- **Verifiability**: Partially fulfilled. No clear metrics to evaluate usability.
- **Validity**: Valid. Usability is key for students and lecturers.

- **Improvement**: The system shall have a user interface that allows users to perform all tasks (group selection, availability input, notification) with no more than 3 clicks, and usability testing should result in at least 90% task completion rates by first-time users.

c) Constraints

. Java development

- **Precision**: Fulfilled. Clear that the system must be developed in Java
- **Consistency**: Fulfilled. No conflicting information in the current context.
- **Verifiability**: Fulfilled. Easy to verify if the codebase is written in java
- **Validity**: Valid. Matching the university’s goal of involving students in the development process.

d) Project requirements

. Budget limitation

- **Precision**: Fulfilled. The 70.000€ limit is clear.
- **Consistency**: Fulfilled. No conflicting information in the current context.
- **Verifiability**: Fulfilled. The project’s expenses can be tracked.
- **Validity**: Valid. This is a standard project constraint.

e) Process requirements

. Student Participation in Development

- **Precision:** Partially fulfilled. "Involve students as developers and testers" lacks specific details, such as the extent of their roles and tasks during development.

- **Consistency:** Fulfilled. No conflicting information in the current context.

- **Verifiability:** Partially fulfilled. There is no clear means to verify that student involvement meets desired levels, such as specific roles, hours, or stages.

- **Validity:** Valid. Student involvement in development is valuable for both system feedback and educational purposes.

- **Improvement:** The development process shall include a defined number of students in roles as developers and testers, specifying tasks, required skills, and measurable involvement across development stages.

Exercise 4

- **Title:** Fulfilled. Manage exercise group enrolment.
- **Actor:** Student
- **Preconditions:**
  . The student has valid university credentials for login.
  . The student is enrolled in courses that require exercise groups.
- **Trigger:**
  . The student opens the EGD system to manage exercise group enrolment.
- **Main success scenario:**
  1.Student: Logs into the EGD system using their university credentials.
  2.System: Verifies the student's credentials.
  3.System: Displays a list of available exercise groups for the student's enrolled courses.
  4. Student: Inputs their availability by entering time slots they are not available due to other commitments.
  5. System: Checks for scheduling conflicts against the entered availability and available exercise groups.
  6. System:
    - If no conflicts are found:
      . Assigns the student to the appropriate exercise group.
      . Send a confirmation email with the assigned schedule.
    -If conflicts are found:
      . Notifies the student of the conflicts.
      . Suggests contacting relevant lecturers for resolution.
  7.Student: Contact the relevant lecturers to discuss possible adjustments.
  8.Lecturer: Provides a solution to the student regarding the conflicting exercise group.

9. System: Updates the student's schedule based on the resolution and sends a final confirmation email.

- **Alternative paths:**

  2a1 [Student has entered incorrect login data]
  System: Displays an error message indicating that the login failed
  2a2 Student: Confirms the error message
   Return to step **1**
  3. [No exercise groups available]
  System: Displays a message indicating there are no available exercise groups for the enrolled courses.
   Use case ends
  4. [Conflicts exists]
  System: Identifies scheduling conflicts between the student's availability and the available exercise groups.
  System: Notifies the student of the conflict and provides information on how to contact lecturers.
   Use case continues to step **7**