SE 216 – SOFTWARE PROJECT MANAGEMENT
SOFTWARE MEASUREMENTS DOCUMENT

PROJECT NAME: MediShare

GROUP NUMBER and MEMBERS: Bartu Erdem, Batuhan Can, Erkan Efe Yeşilçimen, Emin Buğra Aksoy, Samet Tolga Esen, Hüseyin Atacan Akgün

Questions to identify measurements:
1. How reliable is this application? 2. What accomplished during certain period of time? 3. How much has the architecture of this system changed ? 4. How well does the project meet user requirements and expectations? 5. What lessons have been learned from project successes and challenges? 6. How effective was the sprints ? 7. Has everyone done their part?
Measurement storage and collection:
1. Number of Tests **When:** Before each increment. **Format:** Real number data. **How:** Entered into the “Number of” table by developers. **2. Number of Errors:** **When:** Before each sprint review. **Format:** Real number data. **How:** Add to the “Number of” table by developers.

3. Time spent on project:

When: Before each sprint.

Format: Real number data.

How: Total number of hours will be entered to the "Duration table" by project leader with the help of time recording logs.

4. Lines of code:

When: Before each increment.

Format: Real number data.

How: Entered into the "Number of" table by the project leader.

5. Number of classes methods:

When: Before each increment.

Format: Real number data.

How: Entered into the "Number of" table by developers.

6. Number of Added Requirements:

When: After each sprint review.

Format: Real number data.

How: Entered into the "Number of" table by project leader.

7. Time spent on sprint retrospective:

When: After each sprint retrospective.

Format: Real number data.

How: Entered into the project duration by project leader.

8. Number of Git Commits :

When: After each sprint review.

Format: Real number data.

How: Add to the "Number of" table by developer.

9. Stakeholder Reviews:

When: After each increment.

Format: Simple text**.**

How: Entered into the specified user review sheet by developers.

Sprint	1	2
Tests	80	100
Errors	8	4
Lines of Code	1627	2433
Methods	8	19
Classes	1	2
Changed Requirements	0	4
Git commits	24	37

Sprint	Time spend on project	Time spend on retrospective
1	30 days	2 Hours
2	28 days	1 Hour

Measurement Type	Description	Example Measurements
Total number of tests for each sprint	The number of tests conducted to enhance and increase the reliability of our program will be documented in the project table prior to each sprint review. Based on the test results, we will determine whether the program meets the requirements. We will also compare the number of these tests with the number of newly added components and functions to achieve a close ratio. If the ratio is lower than this, we will review the tests and add new ones after discussing with stakeholders	100, 25, 50, 48

Total number of errors after each test phase	The test errors will be reviewed, and the task distribution will be reassessed using git commits. For instance, if an individual consistently receives errors, it may indicate that the workload is too much for one person, or that someone else should take over their tasks.	0, 1, 20, 18
Lines of code	To evaluate the project's progress, we will review the number of codes before each sprint review. This will provide insight into the amount of work completed and the efficiency of resource utilization.	1000, 2000, 1453, 10191
Updates on the codes (Git commits)	Git commits will be used to evaluate task distribution and performance of an individual. They will also be used for identifying who received an error during testing.	47, 39, 26, 16
Number of methods added in that sprint	This number will be updated prior to each sprint review. This will help estimate the project's progress and determine the number of tests required.	18, 22, 10, 5
Number of changed requirements after each sprint	By examining this number, we can determine the effectiveness of our planning and identify areas that require attention in future planning. Additionally, it provides insight into our ability to meet user needs.	0, 4, 5
Stakeholder reviews	After each sprint, user reviews will be kept in a separate file. The file will contain the user's name and feedback. Based on this feedback, requirements will be added and written in different colors, while recommendations will be written in a separate color.	

	The color red indicates that a requirement must be implemented based on stakeholder feedback. Blue indicates that a requirement has been successfully implemented after the feedback, and green indicates considerations for improving the usability of the developed application. This approach will demonstrate the extent to which we meet users' needs and can serve as a reference for future projects.	
Time spend on Project	This data will provide information on individual contributions and sprint planning and adherence. We will use this information to optimise project progress by adjusting sprint duration and task allocation. We will attempt to progress the project optimally by re-planning sprint durations and workload distribution.	30 days, 28 days, 21 days
Time spend on sprint retrospective	By examining the time spent in the sprint retrospective, it will become apparent whether the sprint planning was effective. If the retrospective period is lengthy, it suggests that the sprint planning was flawed from the outset. This indicates that the developers may not have fully grasped the Scrum framework or that there may be miscommunication between them.	2 hours, 1 hour

Stakeholder Review Sheet Example:

Stakeholder A:

- This requirement must be added.
- This requirement was added by a user and completed during the project.
- This part requires attention.

Stakeholder B:

- This requirement must be added
- This requirement was added by a user and completed during the project.
- This part requires attention.

Stakeholder C:

- This requirement must be added.
- This requirement was added by a user and completed during the project.
- This part requires attention.