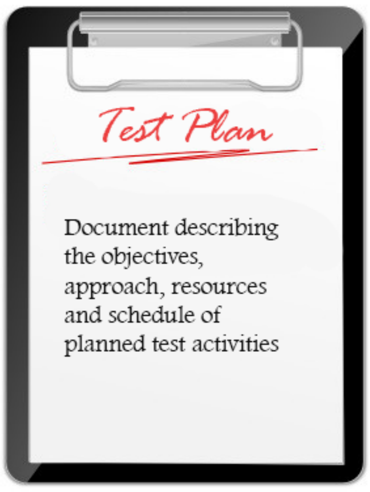
Test Plan


Test Plan
Document describing the objectives, approach, resources and schedule of planned test activities

Project:

WEAre Social Application System

Prepared by:

Filip Gargov

Kristina Duhneva

Rositsa Markova

Table of contents

1. Overview.. 3
1.1 Purpose.. 3
1.2 Objectives.. 3
1.3. Tasks.. 4
1.4. Scope of testing.. 5
2. Test types.. 5
2.1. Functional Testing.. 6
2.2.Non-functional Testing.. 6
2.2.1. Compatibility Testing.. 6
3. Functionalities.. 6
3.1. Functionalities to be tested.. 6
3.2. Functionalities not to be tested.. 8
4. Entry Criteria.. 8
5. Exit Criteria.. 9
6. VALIDATION AND DEFECT MANAGEMENT.. 9
6.1.Validation.. 9
6.2.Defect management.. 10
7. Resources.. 11
8. Schedule.. 16
9. Test environment.. 18
10. Risks and mitigations.. 20
11. Appendix:.. 20
● High-Level Test Cases.. 20
● Test Cases Template.. 20
● Bug Template.. 20
12. Approvals:.. 20

Document History

Version	Date	Author	Description of Change	Reviewed by
1	21.09	Filip Gargov Kristina Duhneva Rositsa Markova	First release of the whole document	Filip Gargov Kristina Duhneva Rositsa Markova

1. Overview

The Weare application is a social media platform designed to connect individuals based on their skills, services, and shared interests. It serves as a dynamic hub that facilitates skill and service exchange, community building, and professional networking.

Users can register and showcase their skills, talents, and services they are willing to offer, cultivating a collaborative environment where individuals can find others with complementary skills. This platform encourages the formation of a supportive and interconnected community, promoting a return to community values through cooperation.

Weare also accommodates personal interests and hobbies, allowing users to share their passions, whether it's sports, music, cooking, or handicrafts, with others who have similar interests. The application offers robust search functionality, streamlining the process of connecting with the right people.

1.1 Purpose

This document describes the test plan for the Weare project. The purpose of this document is to define the strategy, approach, roles and responsibilities of the parties involved in the test process during the construction of the respective project. It is to ensure the quality and reliability of the application by thoroughly testing its functionalities and identifying any potential issues or defects. The plan describes what will be tested, by whom and their responsibilities.

1.2 Objectives

A primary objective of testing is to assure that the system meets the full requirements, including quality requirements (functional and non-functional requirements) and fit metrics for each quality requirement. This includes verifying the correctness, reliability, usability, and performance of the Weare application.

The primary objectives of this test plan are to:

o Verify that the Weare application effectively connects users based on their skills, services, and interests.

o Assess the application's usability to ensure a seamless and enjoyable user experience.

o Identify and mitigate security risks to protect user information and privacy.

o Evaluate the application's performance and scalability.

○ Confirm that the application adheres to specified requirements and design principles.

1.3. Tasks

The testing for the Weare application will encompass a comprehensive approach, ensuring thorough evaluation of functionality, user experience, performance, integration, compatibility, and regression capabilities. The testing process will cover a wide range of aspects to validate that all parts of the application meet established standards and user expectations.

The testing objectives will be addressed through thorough testing efforts to validate that all aspects of the application perform reliably and as expected, providing users with a seamless and efficient experience. This includes evaluating key functionalities, assessing the overall user experience, testing performance under various conditions, ensuring compatibility across platforms, and conducting continuous validation to prevent disruptions caused by code changes or new features.

To accomplish the tasks outlined in the scope, we will adopt an interactive and collaborative approach:

- Daily Meetings: We will convene daily meetings to discuss progress, address concerns, and plan testing activities. These meetings will ensure that the team is aligned and that issues are promptly addressed.

- Continuous Communication: Team members will be in constant communication throughout the day. We will utilize chat and messaging platforms to facilitate real-time collaboration and quick issue resolution.

- Work Distribution: To efficiently distribute the testing workload, we will assign specific areas or functionalities to individual team members based on their expertise. This approach allows for focused testing efforts and accountability. We will work together to validate that all parts of the application are functioning correctly. Collaborative testing sessions will involve team members cross-checking each other's work to ensure comprehensive coverage.

- Issue Tracking: We will maintain an issue tracking system to log and prioritize any defects or concerns identified during testing. This system will help us systematically address and resolve issues.

- Regular Status Checks: Periodic status checks will be conducted to assess testing progress and adjust strategies if needed. These checks will help us stay on track and meet testing goals.

- Documentation: All testing procedures, findings, and outcomes will be documented comprehensively. This documentation will serve as a reference point and provide valuable insights for future testing cycles.

1.4. Scope of testing

The scope of testing refers to the boundaries and extent of testing activities that will be performed on the Weare application. The testing activities will focus on the following areas of the application:

- User Registration and Profile Management: Testing user registration, login functionality, and password recovery processes, ensuring seamless account management.

- Skill and Service Sharing: Verifying the functionality for users to share their skills, talents, and services, fostering collaboration within the community.

- Search Functionality: Testing the search feature's effectiveness in connecting users based on complementary skills and interests.

- Hobby and Interest Sharing: Testing the ability of users to share their hobbies and interests, promoting a diverse range of skill-sharing opportunities.

- Ensuring that the Weare application functions seamlessly across various browsers, operating systems, and devices, providing a consistent user experience.

- Assessing the application's response time, scalability, and stability under varying loads, ensuring it performs optimally even during peak usage.

- Validating the implemented security measures to protect user data, including authentication, authorization, and protection against common security threats.

2. Test types

The testing of the Weare application will cover various types of testing to ensure broad coverage of its functionalities. In the case of the application, various test types are essential to verify its features and functionalities.

The following testing types will be performed:

2.1. Functional Testing

Functional testing will verify that the application's features and functionalities are working as intended. This testing type will focus on validating the correctness of the application's behavior and ensuring that it meets the specified requirements.

Smoke testing, also known as “Build Verification Testing” or “Build Acceptance Testing,” is a type of software testing that is typically performed at the beginning of the development process to ensure that the most critical functions of a software application are working correctly. It aims to identify showstopper defects early in the testing process.

Integration testing focuses on evaluating the interactions between different components, modules, or systems within the application. It aims to uncover issues related to data flow, communication, and integration points. Integration tests will verify that these components work together seamlessly and that data is correctly passed between them. This testing type ensures that the various parts of the application are integrated effectively and can function as a cohesive whole.

2.2.Non-functional Testing

2.2.1. Compatibility Testing

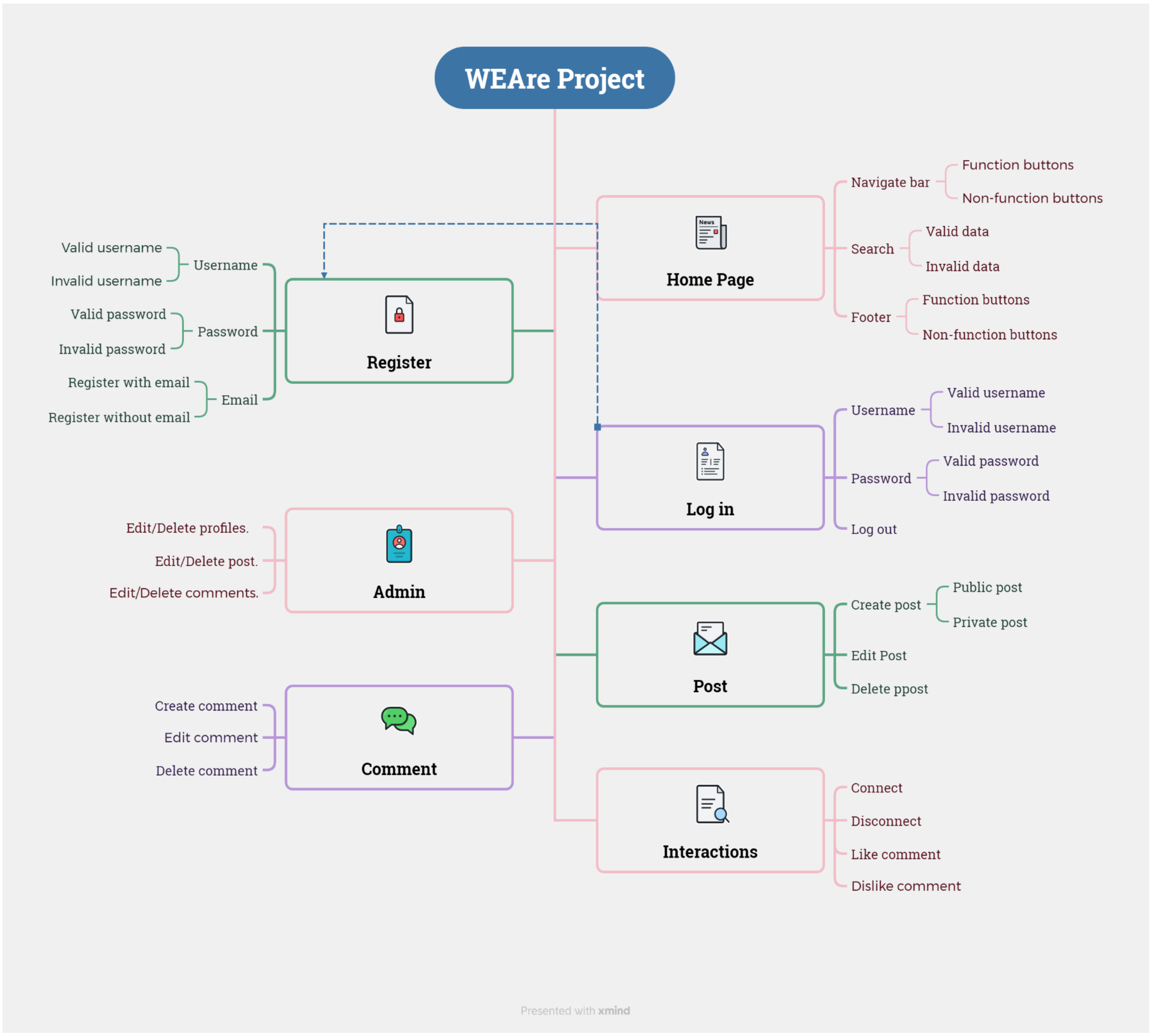
Compatibility testing will verify the application's compatibility across different platforms, devices, and web browsers. It will involve testing on various operating systems (such as Windows, macOS, iOS, Android) and popular web browsers (such as Chrome, Firefox, etc). This testing type aims to ensure that the application functions correctly and consistently across different environments.

3. Functionalities

3.1. Functionalities to be tested

- Registration of a user - Users can create new accounts by providing their email, username and password.
- Authentication - Registered users can log in to their accounts using their username and password.
- Updating Personal Profile - Users can edit and update their personal profiles, including names, email, professional information - professional category and skills, and profile picture.
- Creating a Post - Users can compose and publish posts, which can be either public or private.

- Comments on a Post: Users can add comments to posts to engage in discussions.
- Post Interaction (Like/Dislike, Edit, Delete): Users can like or dislike posts, edit their own posts, and delete their own posts.
- Public Part - The public part allows anonymous users to access the application, view public profiles, search for users, and read chronologically ordered public posts.
- Private Part - In the private part, authenticated users can log in, update their personal information, connect with other users, create posts with customizable visibility, and perform various user-specific actions.
- Administrative Part - The administrative part grants admin users special privileges to edit/delete user profiles, posts, and comments, ensuring effective management and moderation of the platform.
- Viewing Latest Posts: Users can view the latest posts in their feed, including both public and private posts if logged in.
- Search Functionalities for Professionals: Users can search for professionals based on criteria such as name, category, or skills.
- Connecting/Disconnecting with Other Users: Users can send connection requests and accept requests.



3.2. Functionalities not to be tested

- User deletion by admin
- Security testing
- Performance testing

4. Entry Criteria

4.1. The designated test environment, including hardware, software, and network configurations, should be set up and ready for testing.

4.2. The necessary testing resources, including skilled testers, testing tools, and frameworks, should be available and prepared for use.

4.3. Business requirements should be clear and available

5. Exit Criteria

5.1. 100% of Priority 1 and Priority 2 test cases have been completed.

5.2. 70% of the rest of the test cases are completed.

5.3. Planned time has run out.

5.4. Budget depleted.

5.5. Pass rate – 80%

6. VALIDATION AND DEFECT MANAGEMENT

This section outlines the processes and procedures for validating test cases and scenarios, as well as managing defects identified during testing. Effective validation and defect management are crucial aspects of ensuring the quality and reliability of the software under test.

6.1.Validation

The validation of test cases and test scenarios is a critical step in our testing process.

Our approach to validation will be to execute all specified test cases in order to uncover any issues or deviations from expected behavior. Test cases will be assigned a priority level to determine the order in which they will be addressed. The priority levels are as follow:

Priority	Meaning
1 (Highest)	These issues are critical and represent the most important scenarios.
2 (High)	Issues with this priority cover important features or functionalities and would have a significant impact on the product's functionality or usability.

3 (Medium)	This priority is set to standard issues that are essential but not critical.
4 (Low)	Issues with this priority are for less critical features or scenarios that have minimal impact and can often be addressed in later phases.
5(Lowest)	Issues with this priority are with the lowest priority. They are typically used for optional or cosmetic features, and their failure has little impact on the overall product quality.

6.2.Defect management

Managing defects efficiently and effectively is integral to maintaining a robust software development lifecycle. Our defect management process consists of the following key components:

Defect Tracking: All identified defects will be tracked through a dedicated Defect Tracker, providing transparency and traceability throughout the testing phase.

Defect Classification: In addition to priority, in order to prioritize and address issues promptly, defects will be categorized also based on their severity and impact. The severity levels are defined as follows:

Severity	Impact
1 (Critical)	This bug is critical enough to crash the system, cause file corruption, or cause potential data loss. It causes an abnormal return to the operating system(crash or a system failure message appears). It causes the application to hang and requires rebooting the system.
2 (High)	It causes a lack of vital program functionality with workaround.

3 (Medium)	This Bug will degrade the quality of the System. However there is an intelligent workaround for achieving the desired functionality - for example through another screen. This bug prevents other areas of the product from being tested. However other areas can be independently tested.
4 (Low)	There is an insufficient or unclear errormessage, which has minimum impact on product use.
5(Cosmetic)	There is an insufficient or unclear error message that has no impact on product use.

7. Resources.

7.1. Roles and responsibilities

Roles	**Names**	**Responsibilities**
Test Manager	Filip Gargov Kristina Duhneva Rositsa Markova	Overall planning, coordination, and management of the testing activities. Defining the test strategy, test plan, and test schedule. Allocation of resources and monitoring their progress. Defect management and tracking.

Manual Tester	Filip Gargov Kristina Duhneva Rositsa Markova	Creation of test cases based on requirements and specifications. Execution of test cases and scripts manually. Logging defects and issues discovered during testing.
Automation Tester	Filip Gargov Kristina Duhneva Rositsa Markova	Design, development, and maintenance of automated test scripts. Execution of automated tests using testing tools or frameworks. Analysis of test results and reporting of test outcomes. Test script maintenance and enhancement as per changes in requirements. Identifying suitable areas for test automation.

7.2. Tools

The tools that will be needed for the planned activities are:

7.2.1. JIRA:

JIRA is a versatile issue and project tracking tool that will serve as our central platform for managing defects, issues, and tasks throughout the testing process. It provides a collaborative workspace for logging, tracking, prioritizing, and resolving issues, ensuring that testing-related problems are effectively documented and managed.

Testers will use JIRA to log and track defects, issues, and tasks identified during testing. It provides a structured approach to issue management, allowing testers to provide detailed information, such as steps to reproduce, expected behavior, and severity. It also helps in project planning and organization.

7.2.2. Docker Desktop:

Docker Desktop is a containerization tool used to create isolated testing environments, replicating various setups and configurations for comprehensive testing. It allows us to package the Weare application and its dependencies into containers, ensuring consistency and reproducibility of test environments.

Docker containers will be employed to create isolated testing environments, including different configurations and dependencies. This enables testers to execute tests in controlled environments and identify issues related to varying setups.

7.2.3. X-Ray:

X-Ray is a test management app integrated with JIRA, providing advanced capabilities for test planning, execution, and detailed reporting within the JIRA ecosystem. It enhances JIRA's functionality, making it a comprehensive test management tool.

X-Ray will be used for test planning and execution. Test cases will be organized and executed within X-Ray, and the tool will generate detailed test reports. It facilitates efficient management of test scenarios and results, ensuring comprehensive test coverage.

7.2.4. IntelliJ:

IntelliJ IDEA is a powerful integrated development environment (IDE) used for coding both the test automation scripts and testing the REST APIs. It offers a rich set of features for coding, debugging, and testing, making it a valuable tool for test script development.

Testers will use IntelliJ IDEA to write, debug, and maintain test automation scripts using Selenium WebDriver for UI testing and Java code for testing REST APIs. It provides a user-friendly interface and tools for efficient coding and testing.

7.2.5. MySQL:

MySQL is a widely used open-source relational database management system (RDBMS) known for its robustness, speed, and scalability. It is a popular choice for storing and managing structured data in various applications, ranging from small-scale websites to large enterprise systems. MySQL offers a comprehensive set of features for data storage, retrieval, and manipulation, including support for complex queries, indexing, and transaction management.

MySQL is the chosen relational database management system (RDBMS) for our project, primarily because the application's data is stored and retrieved using MySQL databases. This selection aligns our testing environment with the database system employed in the production environment.

7.2.6. GitHub:

GitHub is a web-based platform for version control and collaboration, serving as the primary source code management tool. It provides version control, collaboration, and documentation for code repositories.

GitHub will host the test automation code repository, offering version control capabilities to track changes, collaborate with team members, and maintain documentation. It ensures code integrity and collaboration among testers.

7.2.7. Selenium WebDriver:

Selenium WebDriver is a widely-used tool for automating web application testing. It provides a programming interface to interact with web elements and simulate user interactions in a web browser.

Selenium WebDriver will be used to automate UI testing of the Weare application. Test scripts written in Java will utilize Selenium to interact with web elements, perform actions, and verify expected outcomes, ensuring the correctness of the user interface.

7.2.8. Postman:

Postman is a widely-used API testing tool that simplifies the process of testing RESTful APIs. It provides a user-friendly interface for designing, executing, and automating API tests, making it an essential tool for validating the functionality and reliability of APIs.

Testers will employ Postman to perform comprehensive API testing of the Weare application. Postman allows for the creation of test suites, request collections, and the execution of API requests with various parameters. It helps validate API endpoints, assess response data, and ensure that the RESTful services of the application are working correctly. Detailed test reports and automation options within Postman will aid in efficient API testing.

7.2.9. RestAssured:

RestAssured is a Java-based library that simplifies API testing and automation. It integrates seamlessly with Java and provides a fluent interface for writing readable and maintainable API tests. It is particularly valuable for validating RESTful services and handling JSON and XML responses.

Testers will utilize RestAssured to write and execute API tests for the Weare application. This library will be employed to interact with RESTful APIs, send HTTP requests, and validate the responses. RestAssured's capabilities for handling JSON and XML data will be essential for verifying the correctness of API responses. It will complement Postman in ensuring the reliability of the application's API endpoints.

7.2.10. Gradle:

Gradle is a build automation tool known for its flexibility to build software. A build automation tool is used to automate the creation of applications. The building process includes compiling, linking, and packaging the code.

Newman

Newman is a free and open-source tool. It provides powerful capabilities to run the Postman collections, leveraging super-useful capabilities of Postman like Tests, Assertions, Pre-request scripts, etc and running the collection through the command line.

7.2.11. Maven:

Maven is a popular build automation tool used for managing project dependencies, building, and packaging Java applications. It is similar to Gradle but uses a different build configuration approach based on XML files (POM.xml).

It provides a structured way to specify project configurations, including dependencies and plugins. Maven's compatibility with Java 11 ensures that our test automation project is built and executed using the correct Java version. This ensures consistency and reliability in our testing environment.

7.2.12. JUnit:

JUnit is a widely-used Java framework for writing and running unit tests. It provides annotations and assertions for defining test cases and verifying expected behavior in Java code. JUnit is an essential tool for ensuring the correctness of individual components and methods in our test automation project.

In our testing strategy, JUnit will be used to write and execute unit tests for specific components of our test automation framework. These tests will validate the functionality of critical code segments, helping to identify and rectify any issues at an early stage of development. JUnit's integration with tools like IntelliJ IDEA ensures efficient test development and execution.

7.2.13. Java:

Java version 11 is the specific version of the Java programming language that we will use for developing and executing our test automation scripts.

Our choice of Java version 11 ensures that our test automation code is written in a stable and well-supported environment. It offers features that enhance code readability and maintainability. Additionally, it aligns with industry best practices for Java development and testing, contributing to the reliability and robustness of our test suite.

8. Schedule

№	**Activity**	**Timeframe**	**Responsible person**
1.	**Setup environment**	**14.09-16.09**	**All**
2.	**Test Planning**	**17.09-20.09**	**All**
3.	**Writing high-level tests**	**20.09-21.09**	
3.1.	Comment functionalities Interaction functionalities		Filip Gargov
3.2.	Post functionalities		Kristina Duhneva
3.3.	Register Login Logout		Rositsa Markova
4.	**Writing Test Cases in Jira/X-ray**	**22.09-25.09**	
4.1.	Comment functionalities Interaction functionalities		Filip Gargov
4.2.	Home page-search field Post functionalities		Kristina Duhneva
4.2.	Login Logout Register		Rositsa Markova

5.	**Manual execution of test cases**	**25.09-26.09**	
5.1.	Comment functionalities Interaction functionalities		Filip Gargov
5.2.	Home page-search field functionalities Post functionalities		Kristina Duhneva
5.3.	Login Logout Register		Rositsa Markova
6.	**API testing using Postman**	**27.09-05.10**	
6.1.	User functionalities		Filip Gargov
6.2.	Comment functionalities Connection functionalities		Kristina Duhneva
6.3.	Post functionalities Skill functionalities		Rositsa Markova
7.	**Running Postman collections using Newman**	**05.10-06.10**	**All**
8.	**API testing using Rest Assured**	**06.10-19.10**	
8.1.	User functionalities Skill functionalities		Filip Gargov
8.2.	Comment functionalities Post functionalities		Kristina Duhneva

8.3.	Connection functionalities		Rositsa Markova
9.	**Web testing using Selenium WebDriver**	**06.10-19.10**	
9.1.	Admin functionalities Comment functionalities		Filip Gargov
9.2.	Personal profile functionalities Homepage functionalities		Kristina Duhneva
9.3.	Register functionalities Login functionalities Post functionalities Connection functionalities		Rositsa Markova

9. Test environment

The test environment for Weare application is about the hardware, software, and network configurations required to conduct the testing activities effectively.

- Operating System: Application should be compatible with popular operating systems like Windows, macOS, iOS, and Android.

- Web Browsers: Application should be compatible with major web browsers such as Mozilla Firefox, Google Chrome Safari and Microsoft Edge

- Internet Connectivity: The application should be secure, stable and reliable and should have adequate bandwidth and low latency.

- Test Devices: Different Laptops, Desktop computers and mobile devices.

Testing devices:

Device 1:

- Processor 12th Gen Intel(R) Core(TM) i7-1260P 2.10 GHz

- Installed RAM 16.0 GB
- System type 64-bit operating system, x64-based processor
- Windows Specifications: Windows 11 Home, version 22H2
- Browser: Google Chrome, version: 117.0.5938.149
- Database - MySQL, version: 5.5.62-0ubuntu0.14.04.1

Device 2:

- Processor: 12th Gen Intel® Core™ i7-12700H
- Installed RAM: 32.0 GB DDR5
- Storage: Up to 3TB M.2 PCIe 4.0 SSD
- System type: 64-bit operating system, x64-based processor
- OS Name: Zorin OS 16.3 based on Ubuntu 20.04 LTS
- Browser: Opera One 102.0.4880.78, Chromium: 116.0.5845.188
- Database: MySQL 5.5.62-0ubuntu0.14.04.1

Device 3:

- Processor Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz 2.00 GHz
- Installed RAM 8.00 GB
- Storage Up to 1TB M.2 PCIe 4.0 SSD
- System type 64-bit operating system, x64-based processor
- Windows Specifications Windows 11 Pro
- Browser: Google Chrome, version 117.0.5938.149
- Database - MySQL, version: 5.5.62-0ubuntu0.14.04.1

10. Risks and mitigations

10.1. Risk of potential delays in the project schedule due to unforeseen circumstances, such as additional development work or unexpected dependencies on external systems.
Mitigation: Regular monitoring and tracking of project progress and early identification of potential delays

10.2. Change in Requirements: Changes in requirements during the testing phase can impact the scope and timelines of testing activities.
Mitigation: Establishing a well-defined change management process to evaluate and prioritize requirements changes

10.3. Resource Constraints: Limited availability of resources, such as skilled testers or test environments impacting the testing timeline and coverage.
Mitigation: Proper resource allocation and planning, leveraging test automation to optimize testing efforts, and exploring possibilities for resource enhancement if needed.

11. Appendix:

- High-Level Test Cases
- Test Cases Template
- Bug Template
- Detailed Test Cases

12. Approvals:

1. Diana Gospodinova …………………………………………………………..

2. Nikolay Avramov …………………………………………………………..