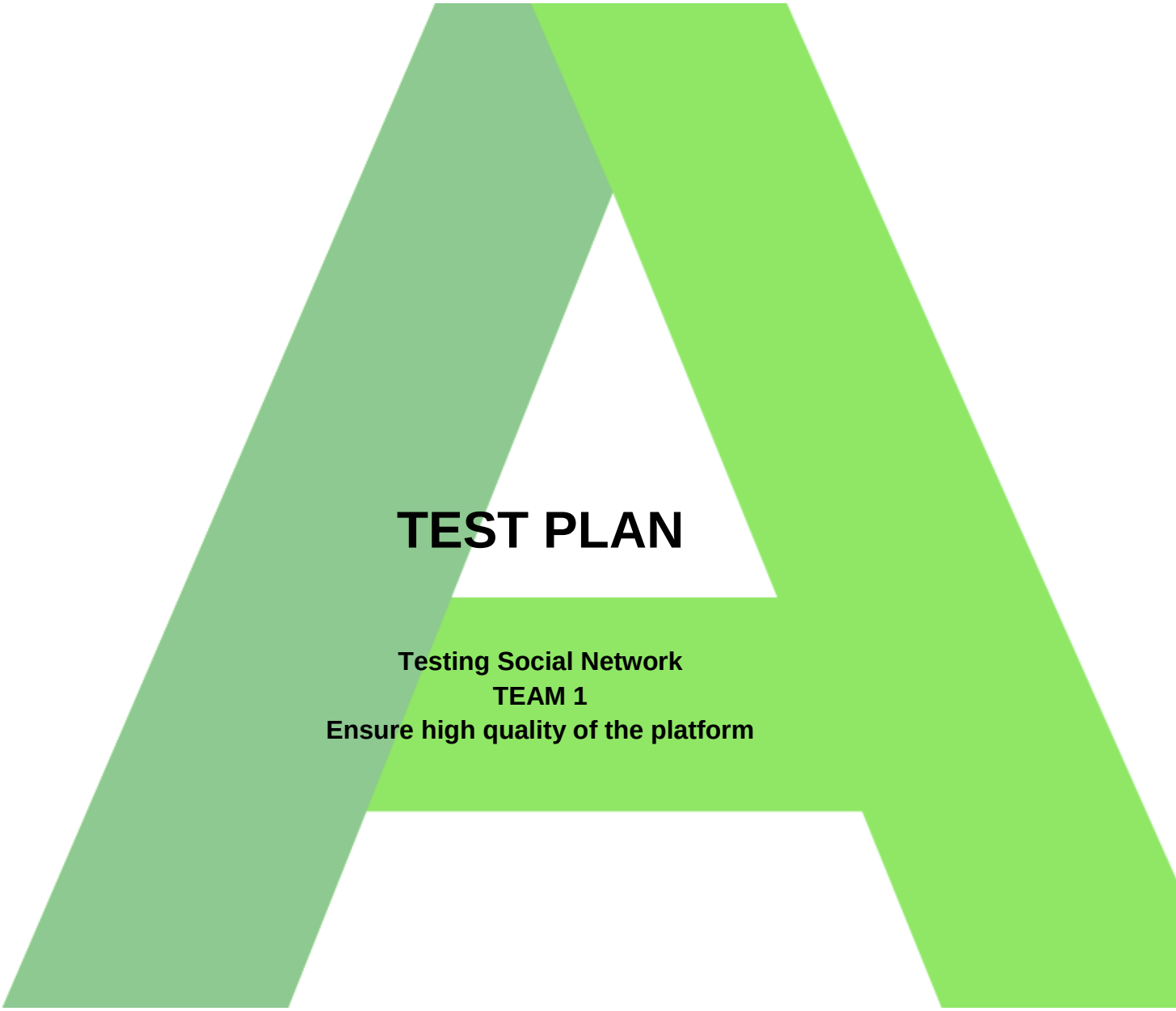


TEST PLAN

Testing Social Network
TEAM 1
Ensure high quality of the platform

Dipl. Eng. Teodora Hristova
Dipl. Eng. Ivailo Tabakov
Dipl. Eng. Ivan Petkov

Document Location			Category	Audience
Team1 OneDrive			Final Project	**Telerik Academy Alpha Trainers**
Date 19.10	**Author TH, IT & IP**	**Version I.1**		

Table of Contents

I. Introduction

In order to ensure the reliability, functionality, and quality of the WEare Social Network application, this Test Plan have been developed by Team 1 of A50 cohort to provide a structured approach to testing activities.

This Test Plan, serves as a comprehensive guide for the testing of the application, outlining the scope, objectives, and strategies.

It is a vital document for coordinating and executing testing activities for the project, ensuring its reliability and quality.

1. Purpose

The purpose of this document is to address the Quality Assurance coverage for a Social Network web application. Information from the requirements, system design and acceptance criteria are aggregated into a detailed plan for testing. It identifies the tasks and activities to be performed so that all aspects of the system are adequately tested, and that the system can be successfully implemented. The Test Plan documents the scope, content, methodology, sequence, management of, and responsibilities for test activities.

2. Definitions, Acronyms and Abbreviations

This subsection provides the definitions of all terms, acronyms, and abbreviations required to interpret properly the document.

Name	Description
Test Case (TC)	A test case is a set of conditions or variables and inputs that are developed for a particular goal or objective to be achieved on a certain application to judge its capabilities or features.
Exploratory Tests (ET)	Exploratory Testing is a type of software testing where Test cases are not created in advance, but testers check system on the fly. They may note down ideas about what to test before test execution.
Functional Tests (FT)	Functional Tests ensure that an element of the system meets the functional requirements of the solution.
Performance Tests (PT)	Performance tests ensure that the system under a particular workload performs as specified in the relevant requirements of the solution.
REST API (API)	REST API testing is a crucial process that involves evaluating the functionality, performance, and security of RESTful APIs. It verifies whether the API meets the required specifications, responds correctly to different requests, handles errors gracefully, and interacts seamlessly with other components of the software ecosystem. Each test is comprised of test actions. These are the individual actions a test needs to take per API test flow. For each API request, the test would need to take the following actions: 1. Verify correct HTTP status code. For example, creating a resource should return 201 CREATED and unpermitted requests should return 403 FORBIDDEN, etc. 2. Verify response payload. Check valid JSON body and correct field names, types, and values — including in error responses.

	3. Verify response headers. HTTP server headers have implications on both security and performance. 4. Verify correct application state. This is optional and applies mainly to manual testing, or when a UI or another interface can be easily inspected. 5. Verify basic performance sanity. If an operation was completed successfully but took an unreasonable amount of time, the test fails.

3. References

List of other documents or Web addresses to which this document refers.

Name	Comment
Test cases	Jira Board - WEARE
Defect/Bug Tracking Portal	Jira Board - BUG
Source Code Management Tool	https://github.com/TEAM-1-A50

II. Test Objective

To prove that testing is objective, the system quality should be measured. This will be achieved by measuring its compliance with the functional requirements. The requirements are covered by tests mapped to a specific type of test (Jira Board – Test cases) and based on it, they are validated and the expected results for each test being performed are measured.

The tests should ensure that all functionalities of the website are working properly, and a client would be able to connect with people, create, comment and like posts, get a feed of the newest/most relevant posts of its connections.

III. Test Approach

The approach provides the details of the levels and types of testing. The types of tests needed to validate the system are:

1. Exploratory Tests

- written as a list of tests in Jira project, ordered in test sets
- responsibility – created and performed by the QA team
- have label “ET”

2. Functional Manual Tests

- written in Jira project, ordered in test sets

- responsibility – created and performed by the QA team
- correspond to Use Case
- have label “FT”

3. REST API Tests

- written as a list of tests (Appendix A - Requirements Traceability Matrix)
- stored in QA team’s repository on GitHub
- responsibility – created and performed by the QA team
- Tools for testing – Postman & REST Assured
- written in Jira project with label PM for Postman tests, and label REST Assured for corresponding tests

IV. Features and Functions

1. Features and Functions To be Tested

- User management (register, update profile information, delete profile)
- User authentication (Sign in / out)
- Connect / Disconnect (send, receive, approve)
- Post actions (create, read, edit, delete, like/unlike)
- Comment actions (create, read)
- Profile search (by name and/or email)
- API
    - Users
        - CRUD Operations
        - Add/approve friend
        - Get news feed
    - Posts
        - CRUD operations
    - Comment posts
        - CRUD operations
    - Like/unlike posts/comments

2. Features and Functions Not To be Tested

Implemented but not included in the requirements:

- Profile search by profession
- Profile information – skills to offer
- Filter posts by skill category
- Edit own posts as a user
- Delete own posts as a user
- Edit own comments as a user
- Delete own comments as a user
- Friends list
- Users list

V. Resource Requirements

The test environment used for assuring high quality of the solution should be considered as an execution environment for all the tests described below. In this section are described the initial resources requirements for test execution:

1. Environment

- Application Service: Computers capable of running Docker containers and connected to the internet
- Database Server: MariaDB SQL server version 10 running on a private server which is accessible by the Application Service and supporting multiple connections
- Networking: Application Service should be able to connect to the Database server
- Operating Systems: macOS, Windows
- Browsers: Chrome, Safari, Firefox, Edge

2. Software Resources

- IntelliJ IDEA ULTIMATE
- Docker Platform
- Atlassian JIRA
- Apache Maven

3. Automated Testing Tools

- Selenium WebDriver
- REST Assured

- Postman API Platform

4. Staffing

- Responsibilities

List of QA team members and their responsibilities

Company	Role	Name	Responsibility
A50T1	**Testers**		
		Teodora Hristova	**Test Plan Structure, Jira setup, Bug & Test case templates, Create Test Cases, Bug reporting, Manual Testing, Automation Testing, Test Report**
		Ivajlo Tabakov	**Test Plan Structure, Rest Assured, GitHub account setup, REST Assured framework, Manual Testing, Automation Testing, Test Report**
		Ivan Petkov	**Test Plan Structure, Test schedule, create Test Cases, Manual Testing, Automation Testing, Test Report**

VI. Defect Resolution

For defect tracking, a Jira project is used which can be found here. The defects found in the process testing cycle will be reported there.

To follow, approve and report the quality of the system proper metrics should be created. Defect tracking includes the following basic metrics for compliance based on the type of defect found.

1. Defect Status

- To do – this is the initial state of the error (defect) when it is registered in the system, or when it is decided that it needs further work (when it has been partially fixed)
- In Progress – the defect is scheduled, and the corresponding member of the team is working on it
- Done – the corresponding member of the team has resolved the problem

- Closed – nothing more to be done with this defect

2. **Test Priority**
    - High: These are critical tests that must be executed first. They cover core functionality, critical paths, and high-impact features of the software. Any failure in these tests may block further testing or deployment.
    - Medium: Tests at this level are important but not as critical as high-priority tests. They cover essential functionality that should be thoroughly tested but may not have the same impact on the overall system if they fail.
    - Low: These tests are less critical and can be executed after high and medium-priority tests. They often cover non-essential or rarely used features. Failures in these tests are less likely to block progress.
3. **Bug severity**
    - Urgent – very serious problem that blocks system functionality or stops data processing and needs an urgent reaction.
    - High - serious defects that prevent the systematic testing of certain functionality or cause incorrect data processing.
    - Normal - problems with incorrect or missing data that do not stop the system working.
    - Low - minor defects that don’t hinder or limit the functionality of the system.

The status of the system, based on the problems raised, may be followed by the Defect Tracking Report which could be extracted at any time from the defect tracking portal.

VII. Test Compliance

A defined set of **entry/exit criteria** for each stage will be created to validate that all prerequisites for a valid test have been met.

1. Entry criteria:

Based on the specific testing needs and the target stage the **entry criteria** include:

- environmental availability
- data availability
- developed and validated code which is ready to be tested
- implemented (developed) functional requirements
- implemented (developed) non-functional requirements

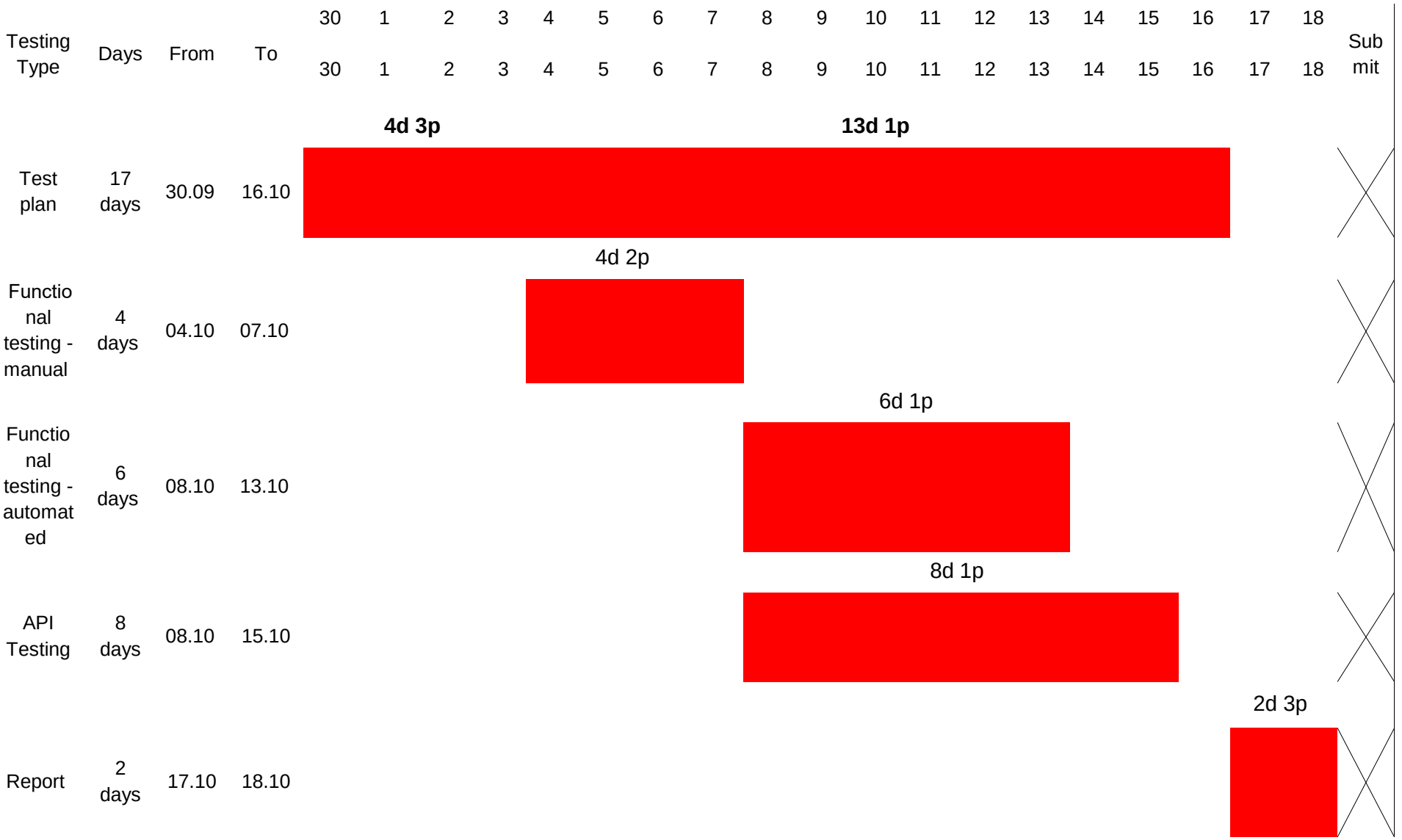
2. Exit criteria:

- Ensuring all high-priority Test Cases have passed
- Achieving complete Functional Coverage according to the scope
- Scheduled time is over

VIII. Testing schedule

Risks and their mitigation

Risk	Mitigation
Team member goes on a sick leave	Plan at least two extra person / days to tackle such situation
There is a deviation between customer requirements and test plan provisions and time is not sufficient	A detailed test plan will identify such deviations and the project management team will be able to take it to the trainers for a solution.


Testing Type
Days
From
To
30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Sub mit
Test plan
17 days
30.09
16.10
4d 3p
13d 1p
Functional testing - manual
4 days
04.10
07.10
4d 2p
Functional testing - automated
6 days
08.10
13.10
6d 1p
API Testing
8 days
08.10
15.10
8d 1p
Report
2 days
17.10
18.10
2d 3p

Appendix

In the following appendixes are shown the templates that would be populated and written as separate documents that will refer to the Test Plan.

Functionality	Requirement	Test Case Summary	Issue Type	Priority	Label	Feature
Site availability	View landing page with login, registration, profile search without authentication	Sign in, Registration and profile search should be visible on landing page	Test	High	Smoke, Automation	
User registration	**User must be able to register**	**User registration**	**Test set**		**FT**	**User management**
	User must be able to register with valid credentials	Register with valid credentials	Test	High	FT, Smoke, Automation	User management
	Username must be unique and between 2 and 20 characters	Register with invalid username – 1 character	Test	High	FT	User management
	Username must be unique and between 2 and 20 characters	Register with invalid username – 21 characters	Test	High	FT	User management
	User mustn’t register with duplicated username	Register with duplicated username	Test	High	FT	User management
	Username must be unique and between 2 and 20 characters	Register with valid credentials username – 2 characters	Test	High	FT	User management

	Username must be unique and between 2 and 20 characters	Register with valid credentials username – 20 characters	Test	High	FT	User management
	Username must be unique and between 2 and 20 characters	Register with valid username length with a special symbol	Test	High	FT	User management
	Password must be at least 8 symbols and should contain capital letter, digit and special symbol (+, -, *, &, ^, …)	Register with valid credentials – pass with 8 characters containing capital letter, digit and special symbol	Test	High	FT	User management
	Password must be at least 8 symbols and should contain capital letter, digit and special symbol (+, -, *, &, ^, …)	Register with invalid credentials – 7 characters containing capital letter, digit and special symbol	Test	High	FT	User management
	Password must be at least 8 symbols and should contain capital letter, digit and special symbol (+, -, *, &, ^, …)	Register with invalid password – 8 characters, without capital letter, with digit and with special symbol (+, -, *, &, ^, …)	Test	High	FT	User management
	Password must be at least 8 symbols and should contain capital letter, digit and special symbol (+, -, *, &, ^, …)	Register with invalid password – 8 characters, with capital letter, without digit and with special symbol (+, -, *, &, ^, …)	Test	High	FT	User management
	Password must be at least 8 symbols and should contain capital letter, digit and special symbol (+, -, *, &, ^, …)	Register with invalid password – 8 characters, with capital letter, with digit and without special symbol (+, -, *, &, ^, …)	Test	High	FT	User management
	Password must be at least 8 symbols and should contain capital letter, digit and special symbol (+, -, *, &, ^, …)	Register without confirming password	Test	High	FT	User management
	Email must be valid email and unique in the system.	Register with invalid credentials – invalid email	Test	High	FT	User management

	Email must be valid email and unique in the system.	Register with invalid credentials - duplicated email	Test	High	FT	User management
User Authentication	**User must be able to login**	**Sign in**	**Test set**		**FT**	**User authentication**
	User must be able to login with valid credentials	Sign in with valid credentials	Test	- High	FT, Smoke, Automation	User authentication
	User mustn't login with invalid username	Sign in with a username that has not registered	Test	High	FT	User authentication
	User mustn't login with invalid password	Sign in with incorrect password	Test	High	FT	User authentication
	User mustn't login with empty password	Sign in with username and empty password	Test	High	FT	User authentication
Sign out	**User must be able to logout**	**Sign out**	**Test set**		**FT**	**User authentication**
	When registered user signs out the 'Back button' of the browser should not navigate back into the system	Verify the sign out functionality	Test	High	FT, Automation	User authentication
		Signing out of one session ends all other session by same user	Test	High	FT	User authentication
		Successful redirection after signing out	Test	High	FT	User authentication
		User session URL is not accessible after signing out	Test	High	FT	User

						authentication
View user info	**View public profile information**	**Visible public profile information**	**Test set**		**FT**	**User management**
	View public profile information	User is able to see public profile info	Test	Low	FT	User management
	Set Public / Private profile info	User is able to set public / private profile info	Test	Medium	FT	User management
	View own profile information	User is able to see their own username, name, email, birthday, location, friends list and bio on their own profile	Test	Low	FT, Automation	User management
Edit / Update User profile	**Update user profile info (name, profile picture, picture visibility (public/connections))**	**User is able to update their personal information:**	**Test set**		FT	User management
	Update First and Last name	User is able to update their first name and last name	Test	High	FT	User management
	Update profile picture	User is able to update their profile picture	Test	High	FT	User management
	Update profile picture privacy	User is able to update their picture privacy	Test	High	FT	User management
Additional Profile Personalization	**Optional: Change password and email, and additional info like age, nationality etc.**	**User is able to update additional personal information:**	**Test set**		**FT**	**User management**
		Update users' location	Test	Low	FT	User management

		Update user's email	Test	Low?	FT	User management
		Update user's birthday	Test	Low	FT	User management
		Update user's gender	Test	Low	FT	User management
Send / Approve profile connection	**Connect / Disconnect with another user**	**User should be able to connect and disconnect with other users**	**Test set**		FT	**Connect / Disconnect**
	Registered user must be able send a connection request	Send a connection request	Test	High	FT, Automation	Connect / Disconnect
	Registered user must be able to receive connection requests	Verify connection request is received	Test	High	FT, Automation	Connect / Disconnect
	Registered user must be able to approve connection	Approve a connection request	Test	High	FT, Automation	Connect / Disconnect
	Registered user must be able to reject connection	Reject connection request	Test	Low	FT	Connect / Disconnect
	Registered user must be able to disconnect	Disconnect from a connection	Test	Medium	FT	Connect / Disconnect
Profile search	**Profile search based on name and/or email**	**User should be able to find other user(s) via the profile search bar**	**Test set**		**FT**	**Profile Search**
		Search by first name with valid input	Test	High	FT, Automation	Profile Search
		Search by last name with valid input	Test	High	FT, Automation	Profile Search

		Search by full name with valid input	Test	High	FT, Automation	Profile Search
		Confirm case-insensitive search	Test	Low	FT	Profile Search
		Search by name for a non-existing user	Test	Low	FT	Profile Search
		Search by email with valid input	Test	High	FT	Profile Search
		Search by email with invalid input	Test	Low	FT	Profile Search
		Search by name and email	Test	High	FT	Profile Search
Create post	**Create a public post (must contain text, optional: picture)**	**User should be able to create a public post**	**Test set**		**FT**	**Post Actions**
		Create a public post with valid input	Test	High	FT, Automation	Post Actions
		Create a public empty post	Test	High	FT	Post Actions
		Create a public post with invalid input – with 1001 characters	Test	Low	ET	Post Actions
		Create a public post with a picture	Test	Low	FT	Post Actions
		Create a public post with a video	Test	Low	ET	Post Actions
		Create a public post with a song	Test	Low	FT	Post Actions
		Create a public post with a location	Test	Low	FT	Post Actions
	Create a connections-only post (must contain text, optional: picture)	Create a private post	Test	High	FT	Post Actions

View a post	**View different posts**	**User should be able to view posts**	**Test Set**		FT	Post Actions
	View public feed chronologically	View public posts as anonymous user	Test	High	FT	Post Actions
		NOT be able to view private posts as an anonymous user	Test	High	FT	Post Actions
	View personalized feed in chronological order	User should be able to view a personalized post feed, where they can see a feed formed from their connection's posts	Test	High	FT	Post Actions
		Posts should be ordered in chronological order when there are no interactions with them	Test	Medium	FT	Post Actions
	Optional: Complex feed generation	Posts with interactions should appear higher than posts with no interactions	Test	Mediu m		Post Actions
		Posts of new connections should be placed higher(???)	Test	Mediu m	FT	Post Actions
Like post	**Like / unlike a post**	**Registered user like/dislikes post**	**Test set**	**Mediu m**	**FT**	Post Actions
		Like a post (validate likes count)	Test	Medium	FT, Automati on	Post Actions
		Unlike a post (validate likes count)	Test	Medium	FT, Automati on	Post Actions
Create comment	**Comment on a post**	**Comment on a post**	**Test set**		**FT**	**Comment Actions**
		Create a comment with valid input	Test	High	FT, Automati on	Comment Actions
		Create an empty comment	Test	High	FT	Comment Actions
		Create a comment with invalid input with 1001 characters	Test	Low	ET	Comment Actions

	Show newest N comments under a post and older posts	Newest comments should be shown under the post	Test	Medium	FT	Comment Actions
Admin	**Administrative functions**	**Admin user must have administrative access**	**Test set**		FT	**Admin Actions**
	Admin can view and access admin panel	View and access admin panel	Test	High	FT	Admin Actions
	Edit other users' profile	Edit another user's profile information	Test	High	FT	Admin Actions
	Delete other users' profile	Delete another user's profile as admin	Test	High	FT	Admin Actions
	Edit other users' posts	Edit the content of another user's post as admin	Test	High	FT, Automation	Admin Actions
		Edit the visibility of another user's post as admin	Test	High	FT	Admin Actions
	Delete other users' posts	Delete another user's post as admin	Test	High	FT, Automation	Admin Actions
	Edit other users' comments	Edit the text on a comment as admin	Test	High	FT, Automation	Admin Actions
	Delete other users' comments	Delete another user's comment as admin	Test	High	FT, Automation	Admin Actions
API						Profile Search
	Users - CRUD operations	Create user			FT	
		Read user				Profile Search
		Update user			FT	
		Delete user				Profile Search

	Add/approve friend	Send connect invite & approve it			FT	
	Get news feed	Get news feed				Profile Search
	Posts - CRUD operations	Create post			FT	
		Read post				Profile Search
		Update post			FT	
		Delete post				Profile Search
	Comment posts	Create comment			FT	
	Like / unlike posts/comments	Like/ unlike posts				Post Actions
		Like/ unlike comments			FT, Automation	Feature