# Testplan outline

This outline describes test methods for the smallest pieces of logic in your code via the integration of sub-systems to the usability, scalability, and security of the total system.

Note:

* This document does not describe how to implement these tests
* You must do this research yourself

Hint: for more details and inspiration query “software testing fundamentals”

## Testing your code’s building blocks

### Unit Testing

Write tests to guarantee that essential, basic functions always produce the expected answer. When these tests fail, you detect that something has changed badly.

**Example**:

extractVAT21(nettValue):VAT

extractVAT21Test() {

// Checks the 21% case

assert(extractVAT21(12.10) == 2.10)

}

### Algorithm Testing

Write tests to prove that essential algorithms produce the expected output.

**Example**:

findShortestPath(startPoint, endPoint):path

findShortestPathTest() {

path = findShortestPath({0,0}, {9, 7});

// Check all steps within the path object

assert(path[0] == {1,1});

assert(path[1] == {2,1});

...

}

## Testing modules and their integration

### Testing individual modules

Write a script that touches one module and verify that it produces the expected outcome.

**Example**: a script that logs in and checks if an authentication cookie is set

### Testing multiple modules in sequence

Write a script that touches multiple modules and verify that the combination produces the expected outcome.

**Example**: a script to create an account and to check that it exists in the database

1. *Testing interfaces*

Write tests to verify that the data structures from a module conform to a specified data format (the data contract).

**Example**:

Ensure that the json description of an account API-call equals the specified structure.

## Stress or load testing

1. Write automatic scripts to run multiple logins, up- and downloads from an image library, … and monitor the server’s CPU, Memory, IO and Networking usage

**Note**: be aware of browser and framework caching layers (cold starts and hot starts)

1. Nice to do: use the load tests above to estimate the maximum system load for each functionality. For this you need to perform measurements in sequence, e.g. 1x, 10x, 100x, 1000x requests

## Testing the functional requirements defined in the Requirements Specification Doc

1. Write test scenarios to verify that the FR’s and Use Cases are properly implemented. Write these tests with a software tester in mind.

**Example**: start with the preparation for the pre-condition(s) of the Use Case, follow the happy, alternative and exceptional paths and use the details of the FR’s

## Usability testing

1. Write down multiple scenarios and observe how the Product Owner / client navigates through the steps. Write down where they got stuck and how to fix this.

**Example**:

SCENARIO\_1: Creating an account

Go to app.client.com and create an account. Write down every step you take. If a step feels too complex or unclear: write it down.

## Security & Privacy measures

1. Verify that your requests are sent over HTTPS, especially the sign-up and login requests. Use the built-in browser Dev tools or a “packet sniffer” (e.g. [Fiddler](https://www.telerik.com/fiddler)) for this.
2. Verify that that your database connection string is safely stored and not accessible from outside. You can do this by properly setting the file rights of the scripts containing the database logic.
3. Open your database connection(s) and verify that names, email addresses are encrypted (not visible in plain text), and passwords are hashed.

## Maintainability

1. Verify that your code is readable with proper naming (adding a coding convention helps) and has useful comments. Also, verify that files are organized in a modular manner and that server and database setups (scripts) are properly documented.

**Example**:

Let another team perform an audit to check if your codebase is properly set up.

User Acceptance testing

1. Verify the end-to-end business flow which is done by the client or end users. This is the final phase of testing before the application goes to production.   
     
   **Example:**  
     
   Think as an unknown user to the system. Identify test scenarios and test cases. Make use of use cases and real-live data.