**Test Plan STC Time Management**

A system test of the prototype will be conducted.

The following features will be tested:

* Employee Time Entry
* Employee Vacation Request
* Supervisor Vacation Management
* Login
* General Performance

**Testcase 1:** automatic time input

**Method**: manual test

**Description**: Click the Start working button and wait for 0, 5, and 30 minutes and with break

**Expected Outcome**: All entries should be logged in the database with the correct parameters, entries with 0 hours work should not be added

**Actual Outcome**: 0 hours worked are still added to the database, the rest works correctly

**Consequences**: write a check to intercept those cases

**Testcase 2:** normal manual time input

**Method**: manual test

**Description**: Fill out the manual input form with plausible data

Expected Outcome: All entries should be logged in the database with the correct parameters

Actual Outcome: The entries are logged correctly

Consequences: -

**Testcase 3:** work\_ended before work\_started

**Method:** manual test

**Description:** Fill out the manual input form with impossible data, work ended before work started

**Expected Outcome:** Impossible entries should not be added, an error should be generated

**Actual Outcome:** the entry is added, no error

**Consequences:** write a check to intercept those cases

**Testcase 4:** break too big for time worked

**Method:** manual test

**Description:** Fill out the manual input form with impossible data, break bigger than time\_worked

**Expected Outcome:** Impossible entries should not be added, an error should be generated

**Actual Outcome:** the entry is added, no error

**Consequences:** write a check to intercept those cases

**Testcase 5:** SQL Injection in comment

**Method:** manual test

**Description:** Use the freeform comment field to inject sql code ('); DROP TABLE day; --) to drop table

**Expected Outcome:** sql injection should be prevented somehow

**Actual Outcome:** injection is not possible due to sqli->prepare to always treat inputs as parameters

**Consequences:** -

**Testcase 6:** request Vacation

**Method:** manual test

**Description:** use the vacation form to add a vacation entry with plausible data to the database

**Expected Outcome:** All entries should be logged in the database with the correct parameters

**Actual Outcome:** The entries are logged correctly

**Consequences:** -

**Testcase 7:** approve/reject vacation

**Method:** manual test

**Description:** log in as supervisor and approve a pending vacation request

**Expected Outcome:** status should be changed in the database

**Actual Outcome:** status is changed correctly

**Consequences:** -

**Testcase 8:** circumvent login

**Method:** manual test

**Description:** Try to circumvent login in any way

**Expected Outcome:** it should not be possible

**Actual Outcome:** it is possible to just enter the other pages without ever trying to log in

**Consequences:** create some kind of session login system to replace the login mockup

**Testcase 9:** stress-testing

**Method:**

**Description:** stresstest the server, emulate a DoS attack

**Expected Outcome:** a server shutdown should be avoided, functionality should not be affected

**Actual Outcome:**

**Consequences:**