SUPER THERMOSTAT PROJECT

Team Smart Thermo

“Smarter everyday”

Team members:

A. Roman

A. Slotboom

E. Thissen

Goal: create a working thermostat application based on the client wishes.

Rotterdam

11 04 2018

Document V 1.0

1 DEFINITION OF DONE

From the scrum.org glossary:

Definition of Done: a shared understanding of expectations that the Increment must live up to in order to be releasable into production. Managed by the Development Team.

The following are our requirements for the acceptance of a user story as done:

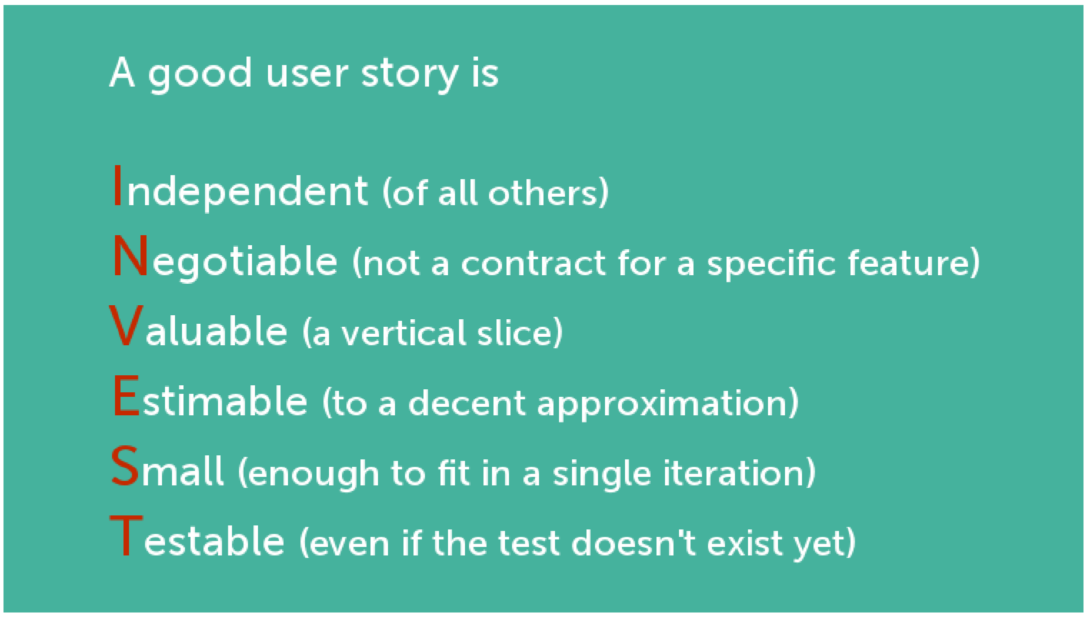
1. Code builds without warnings (technical task)[1]
2. Code unit tested (technical task)
3. Documentation updated (user story)
    3.1. Move user story to done
4. Build pushed to demo server (user story)
    4.1. Build and test with new user story
    4.2. Build can be showed during a demo

[1] https://www.atlassian.com/blog/jira-software/8-steps-to-a-definition-of-done-in-jira

2 DEFINITION OF READY

The definition of ready for this project is based on the following image from themanifesto.co.uk website.

1. **Independent**: the user story is a stand-alone story, meaning it is not dependent on other stories to be able to be built in the current sprint.
2. **Negotiable**: user story can be changed if the team feel it is not adequate.
3. **Valuable**: user story add value to the product.
4. **Estimable**: user story can be estimated up to a certain degree of certainty. If user story is too big it need to be split into multiple user stories.
5. **Small**: user story can be built in 1 sprint.
6. **Testable**: user story is testable in the build.


A good user story is

Independent (of all others)

Negotiable (not a contract for a specific feature)

Valuable (a vertical slice)

Estimable (to a decent approximation)

Small (enough to fit in a single iteration)

Testable (even if the test doesn't exist yet)

Figuur 1 https://manifesto.co.uk/the-definition-of-ready/

3 EPICS

1. **Need to have (MVP):**
   - 1.1. Change temperature as user
   - 1.2. Change temperature as admin
   - 1.3. View temperature as user
   - 1.4. View all lodge's temperature as admin (with lodge numbers)
   - 1.5. Set communication between thermostat and database
   - 1.6. Set communication between user environment and database
   - 1.7. Set communication between admin environment and database
   - 1.8. Set up server
   - 1.9. Set max and min temperature limits
   - 1.10. Make UI thermostat user
   - 1.11. Make UI thermostat admin
   - 1.12. Set time interval for temperature logging (log every 1 hour, total 24 hours)
   - 1.13. Display current temperature (every 60 seconds)
   - 1.14. Log output data from thermostat to database(current temperature, status etc.)
   - 1.15. Keep temperature constant as desired (% pump and burner)
   - 1.16. Security measure: stop warming when temperature does not increase after X time
   - 1.17. Set thermostat activity based on user movements
2. **Nice to have:**
   - 2.1. View CO2 footprint as user
   - 2.2. View CO2 footprint as admin
   - 2.3. View gas m3 usage as user
   - 2.4. View gas m3 usage as admin
   - 2.5. View cost (yearly) in euro as user
   - 2.6. View cost (yearly) in euro as admin
   - 2.7. View CO2 footprint compared to average (user is either above or below)
   - 2.8. View graph with temperature as user
   - 2.9. View graph with temperature as admin
   - 2.10. Ability to program the desired temperature at a certain time

4 USER STORIES

1. **User story 001:**
   - 1.1. **As a**:
   - 1.2. **I want**:
   - 1.3. **So that**:
   - 1.4. Acceptance criteria
     - 1.4.1. **I know this is done when**:
   - 1.5. **Estimation**:

5 QUESTIONS

1. Efficiency of pump and burner in the element, how much % each for optimal use? Which ratio and relation.
2. What is the ‘gas use’ in the emulator? Is it cumulative or rate of use?
3. Gas use as Euros? Which price do we use?
4. Is the ability to time program the thermostat by the user a must have?
5. Application server needed or only Java standalone executable? Which server is advisable?