Visual Code Review

Group 5

Daniel Andersson Tenninge

Gustaf Carleson

Johan Björk

Patrik McKiernan

Functional and Non-functional requirements

Functional requirements:

- The system shall intercept commits to the Source Control Management (SCM) system.
- The system shall put intercepted code up for review if a policy states it needs reviewing.
- The system shall store all commits that have been up for review.
- The system shall display code that is up for review and allow reviewing.
- The system shall only let developers review code where a policy states they are authorized to do so.
- The system shall allow reviewers to accept or reject a pending commit and supply a note to the developer explaining their decision.
- The system shall log the actions of the reviewer. That includes the name of the reviewer, the action taken, the date and the commit.
- The system shall commit the reviewed and approved code to the SCM repository.
- The system shall notify the developer of whether or not his/her code needs reviewing.
- The system shall notify the developer of whether or not his/her code got accepted or not.
- The system shall authenticate users of the system.
- The system shall allow the management of user accounts.
- The system shall display all branches and users of the system available for policy setting.
- The system shall allow setting of policies for users and branches in the SCM system.

Non-functional requirements:

- The system shall not use any third party code that limits sale of the system.
- The system shall be available at least 99 out of 100 times when the SCM system is available (hardware failure excluded).
- The mean time between failures in the software system (hardware failure excluded).
    - The system shall not loose or corrupt a commit more than once in a million.
    - The system shall not allow accidental reviews by developers with insufficient review rights more than once in a million.
- The users of the system shall be able to use the web page of the system after two hours of training. With this training, users of the system will make no more than two errors per day, on average.
    - An error here implies that the user makes the system do something he did not intend.
- The system shall be capable of handling at least as many users as the underlying SCM system.

Use cases

Use Case UC1: Submit code to the repository.

Primary Actor: Developer
Stakeholder and Interests:

- Developer: Wants to submit code to the repository using the code review process with as little extra work as possible. Wants to be notified whether or not his code is accepted or rejected.
- Software company: Quality control the code in the system by enforcing review policies.
- Project leader: Wants to be able to check on how the system development is progressing.
- The person responsible for setting policies: Wants the system to enforce the review policies.

Preconditions: Developer is identified and authenticated and has checked out source code from the Source Code Management (SCM) repository. Source code review policies are set for parts of the system that needs policies.
Success guarantee (post conditions): The code review policies are upheld. Code is stored by the system awaiting review if necessary otherwise committed to the repository. The developer is notified of whether or not his code was committed or is now up for review.

Main Success Scenario (or Basic flow):

1. Developer submits code to the repository.
2. The system stores the submit.
3. The system enforces the policies set for the code submitted.
4. The developer receives notification about the actions taken by the system.

Extensions (or Alternate flows)

2.
    a. There are no policies set for the code submitted.
        i. Code is committed to the repository.
    b. The policies state that the code needs reviewing.
        i. Code is stored awaiting review.
3.
    a. The code did not need to be reviewed.
        i. Developer is notified that the code got committed.
    b. The code needs reviewing.
        i. Developer is notified that the code needs reviewing.

Special requirements:

- No special requirements.

Technology and Data variation list:

- No variations in technology and data.

Frequency of Occurrence: Could be nearly continuous.

Open Issues:
- There are no open issues.

Use Case UC2: Reviewing code.

Primary Actor: Reviewer
Stakeholder and Interests:
- Reviewer: Wants to review submitted code and either accept or reject it based on the code quality. When rejecting code he wants to send the developer a note letting him/her know what was insufficient with the code.
- Software company: Quality control the code in the system by enforcing review policies.
- Project leader: Wants to be able to check on how the system development is progressing.
- The person responsible for setting policies: Wants the system to enforce the review policies.

Preconditions: Reviewer is identified and authenticated and has logged in to the code review web page.

Success guarantee (post conditions): The code review policies are upheld. Code is either committed or rejected based on the actions taken by the developer after reviewing.

Main Success Scenario (or Basic flow):
1. The system displays a list of code up for review based on the reviewers' authorization level.
2. The reviewer selects one item from the list.
3. The system displays the code for the reviewer.
4. The reviewer checks the quality of the code and accepts it if it is up to standard.
5. The system records the action taken by the reviewer.

Extensions (or Alternate flows)
2.
    a. The reviewer accepts the code.
    b. The reviewer rejects the code.

Special requirements:
- No special requirements.

Technology and Data variation list:
- No variations in technology and data.

Frequency of Occurrence: Could be nearly continuous.

Open Issues:
- There are no open issues.

Use Case UC3: Review action response.

Primary Actor: The system under design
Stakeholder and Interests:
- System under design: Responds to the actions taken by the reviewer by either committing the code or discarding it. Notifying the developer of the actions taken with a note from the reviewer.
- Software company: Quality control the code in the system by enforcing review policies.
- Project leader: Wants to be able to check on how the system development is progressing.
- The person responsible for setting policies: Wants the system to enforce the review policies.
- Developer submitting code: Wants to be notified whether or not his/her code got committed with a note from the reviewer.

Preconditions: Reviewer has reviewed a piece of code and either accepted or rejected it.

Success guarantee (post conditions): The code review policies are upheld. Code is committed to the repository, discarded or kept for further reviewing.

Main Success Scenario (or Basic flow):
1. The system checks the action from the reviewer.
2. The system logs the reviewers' action.
3. The system notifies the developer with the supplied note from the reviewer.

Extensions (or Alternate flows)
1.
    a. The reviewer has accepted the code.
        i. The system compares the code review state with the code policy.
            1. The code policy requirement is met.
                a. The system commits the code to the repository.
                b. The system removes the code from further reviewing.
            2. The code policy requirement is not met.
                a. The system updates the code review state.
                b. The system keeps the code for further reviewing.

b. The reviewer has rejected the code
    i. The system removes the code from further reviewing.

Special requirements:
- No special requirements.

Technology and Data variation list:
- No variations in technology and data.

Frequency of Occurrence: When a reviewer has taken an action in the review process..

Open Issues:
- There are no open issues.

Use Case UC4: Setting up policies.

Primary Actor: Person responsible for setting policies (PRP).
Stakeholder and Interests:
- The PRP: Wants the system to enforce the review policies.
- Software company: Quality control the code in the system by enforcing review policies.
- Project leader: Wants to be able to check on how the system development is progressing.

Preconditions: PRP is logged in and authenticated with permissions to set policies.

Success guarantee (post conditions): Eventual changes are saved.

Main Success Scenario (or Basic flow):
1. The PRP selects to either set a policy for developer or for a branch.
2. The PRP is presented with entities that policies can control.
3. The PRP selects the appropriate entity.
4. The PRP saves or discards the changes.

Step 2-4 repeats until the PRP is content.

Extensions (or Alternate flows)
1.
    a. The PRP selects developers
        i. Entities are populated with a list of all developers in the system.
    b. The PRP selects branches
        i. Entities are populated with a list of all branches the PRP can set policies on.
3.
    a. There is an existing policy on this entity
        i. The previous policy is shown and the PRP can make changes or delete the policy.

b. There is no existing policy
    i. The PRP can set a new policy.

Special requirements:
- No special requirements.

Technology and Data variation list:
- No variations in technology and data.

Frequency of Occurrence: When the PRP needs to set a new policy.

Open Issues:
- There are no open issues.

Use Case UC5: User management.

Primary Actor: System administrator
Stakeholder and Interests:
- Employee: That his user, if he needs one, has the permissions he requires.
- Software company: That each employee that needs a user, has one, and that the user has permissions according to the company policy.

Preconditions: The system administrator is logged in and authenticated.

Success guarantee (post conditions): Eventual changes are saved.

Main Success Scenario (or Basic flow):
1. The administrator gets a listing of users in the system.
2. The administrator selects an action.
3. The administrator saves or discards the changes.

Step 2-4 repeats until the administrator is content.

Extensions (or Alternate flows)
2.
    a. The administrator selects to add a new user
        i. The administrator sets the properties on the user.
    b. The administrator selects a user from the list to delete
        i. The user is removed from the system.
    c. The administrator selects a user from the list to edit
        i. The administrator edits the properties for the user.

Special requirements:
- No special requirements.

Technology and Data variation list:
- No variations in technology and data.

Frequency of Occurrence: When the administrator needs to edit users.

Open Issues:

- There are no open issues.