Requirements Specification

*Team CAtS (Yihong Zou, Huairui Qi, Siqi Lu, Yun-Hsuan Chang)*

[**Context**](#_qp224i9zcv2n) **2**

[**Functional Requirements**](#_o6hz6hd5fdiv) **2**

[**Activity Diagram**](#_538t3zb1pa4u) **4**

[**Use Cases**](#_icmkg3bxp530) **4**

[View CA report and download](#_rcn5vm8cr8jl) 4

[Create new issue on Git](#_9tawgtkbtqwk) 5

[User change profile](#_aaxtbv6g28l) 6

[User change setting](#_lze5wqyc8sdj) 7

[User manage project accessibility](#_h63vsco0g683) 8

[Admin login](#_nyxqi1qilemn) 9

[Register project on CAS](#_8ir157nmp3p6) 10

[Configure the controller list for project](#_4p31k356kg5p) 11

[Admins install CAS plugin on jenkins](#_fvbkfdeyp2zf) 13

[Admin configure CAS plugin on Jenkins](#_b7veq1tg7n19) 14

[View CAS result on dashboard](#_ejismm8rhklw) 15

# Context

Continuous security assessments are becoming incredibly important during every phase of the **Software Development Lifecycle** (SDLC) to ensure application security. The basic idea is that whenever new code is committed and merged into the current repository, the system will automatically conduct security tests, either locally or globally, to ensure the security of the software application.

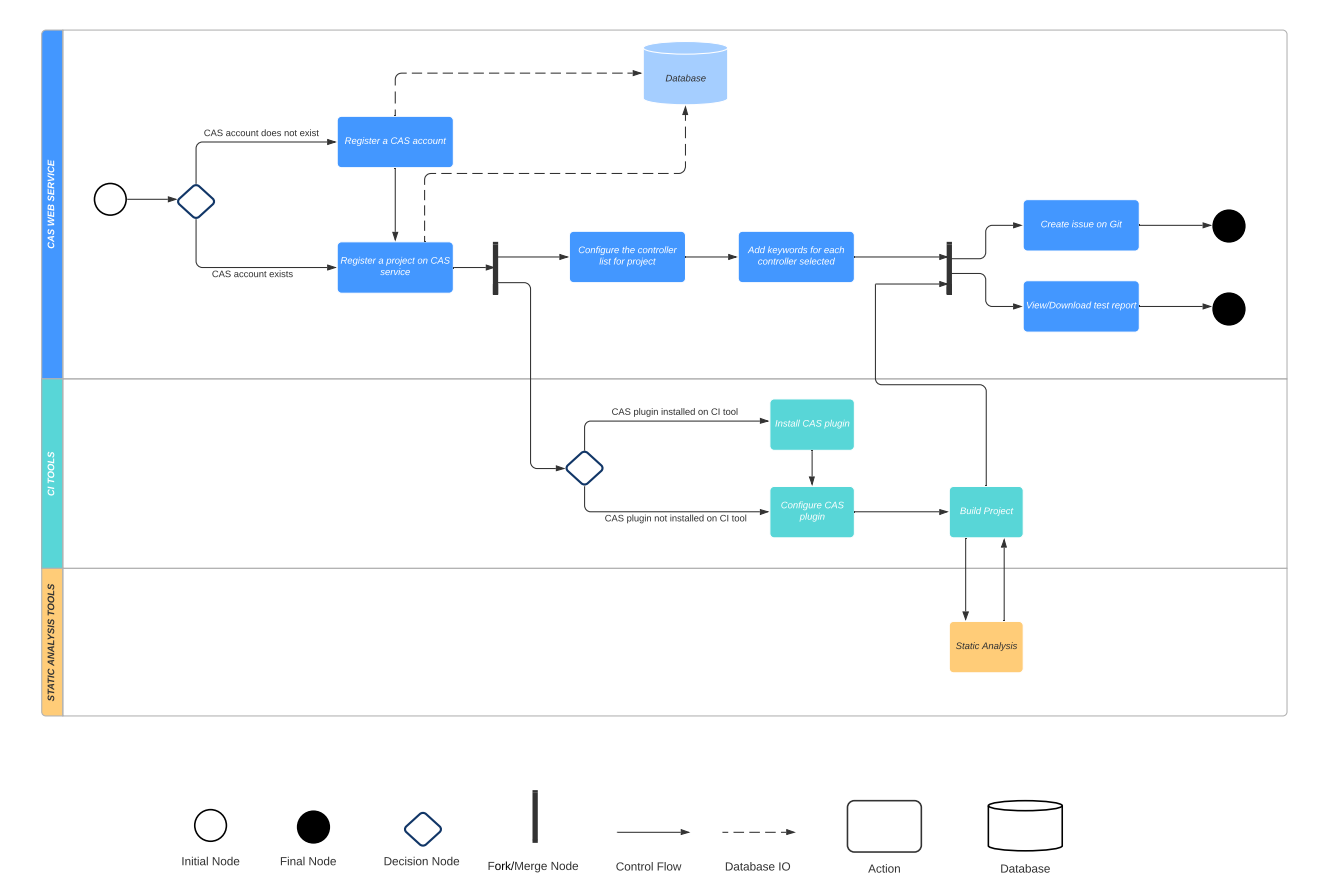
Since there does not exist any comprehensively and completely automated security evaluation service on either commercial or open source platforms, we are going to create a web-based **Continuous Authorization Service** (CAS) that automates the whole security testing process. The CAS system will implement the feature that users are able to customize their security testing preference, extract the user-focused security issues from multiple automated static testing tools, and automatically view the reports on their issue tracking system such as **Git**.

In addition to the CAS web service, a plugin which enables the integration of CAS and continuous integration (CI) tools will also be a part of the final deliverables. During the development stage, the primary CI tool to integrate will be **Jenkins**. In the future, CAS wishes to support more CI tools, such as **TeamCity, Travis CI, Bamboo**, etc.

# Functional Requirements

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Priority |
| FR-01 | The system shall be able to handle users credentials properly. | Critical |
| FR-01-1 | The system shall allow users to create an account through the website interface, setting their usernames and passwords. | Critical |
| FR-02 | The system shall allow users to manage their project repositories. | Critical |
| FR-02-1 | The system shall allow users to register or delete a project. | Critical |
| FR-02-2 | The system shall allow users to configure the controller with keywords. | Minor |
| FR-02-3 | The system shall allow users to share their projects with other users. | Critical |
| FR-03 | The system shall allow users to manage the continuous authorization process of their projects properly. | Critical |
| FR-03-1 | The system shall allow the users to view and download the generated testing report from dashboard. | Critical |
| FR-03-2 | The system shall be able to track the issues automatically for a certain project along with a user provided issue tracking system. | Critical |
| FR-03-3 | The system shall be able to parse reports in both csv and xml format. | major |
| FR-03-4 | The system shall be able to parse reports for each control of a project, with respect to the control’s keywords entered by users. | major |
| FR-04 | The system shall allow users to manage their user account settings | Major |
| FR-04-1 | The system shall allow users to change their profile settings. | Minor |
| FR-04-2 | The system shall allow users to change their credential settings. | Critical |
| FR-05 | The system shall allow an admin to control and manage the users, projects and controllers. | Critical |
| FR-06 | The system shall be able to install and configure CATS on Jenkins as a plugin. | Critical |

# Activity Diagram



# Use Cases

#### View CA report and download

|  |  |
| --- | --- |
| Use case name: | View CA report and download |
| Unique case ID: | ViewCAReportAndDownload |
| Primary Actor(s) : | User |
| Secondary Actor(s): | None |
| Brief description: | Users can view the history of authentication testing of a project by using cas. |
| Preconditions: | 1. Users have a CAS account. 2. Users have already set up a project on their cas. 3. Users have internet access. |
| Flow of events: | 1. Users go to the CAS website, log in to their account. 2. On the project page, click the project name you want to check to go to the dashboard of the chosen project. 3. On the project dashboard page, click the test history you want to check. 4. Click “View Details” to view the detailed report. 5. Click “Download as PDF” to download the report to the local machine. |
| Postconditions: | 1. Users can see the test history of the authentication test for an existing project. 2. Users can see the test report of the authentication test for an existing project. 3. Users can see the report on their local computer. |
| Priority | High |
| Alternative flow or exception: | None |
| Non-behavioral requirements | None |
| Assumptions: | 1. The connection between CAS service and database works fine. |
| Source: | None |

#### 

#### Create new issue on Git

|  |  |
| --- | --- |
| Use case name: | Create new issue on Git |
| Unique case ID: | CreateNewIssueOnGit |
| Primary Actor(s) : | User |
| Secondary Actor(s): | None |
| Brief description: | Users can view the history of authentication testing of a project by using cas. |
| Preconditions: | 1. Users have a CAS account. 2. Users have already set up a project on their cas. 3. Users have internet access. 4. Users have an issue tracker account (Git). |
| Flow of events: | 1. Users go to the CAS website, log in to their account. 2. On the project page, click the project name you want to check to go to the dashboard of the chosen project. 3. On the project dashboard page, click the test history you want to create issues based on. 4. Click the “Create Issues” button. 5. Enter the issue tacker information (Git Username, Git Password and Git Repo Name). 6. Click the “Update” button. 7. Login to the issue tracker account to check the issues. |
| Postconditions: | 1. Users can view the newly founded issues on their issue tracking account correlated to the new commit action. |
| Priority | Medium |
| Alternative flow or exception: | None |
| Non-behavioral requirements | None |
| Assumptions: | 1. The connection between CAS service and database works fine. |
| Source: | None |

#### 

#### User change profile

|  |  |
| --- | --- |
| Use case name: | User change profile |
| Unique case ID: | UserChangeProfile |
| Primary Actor(s) : | User |
| Secondary Actor(s): | None |
| Brief description: | Users can change the profile of their accounts. |
| Preconditions: | 1. Users have access to CAS service. 2. Users have registered for a CAS account. |
| Flow of events: | 1. Users go to the CAS website, log in to their account. 2. On the main page, click the user name on the top-left corner, which will then show the “Profile” button and “Setting” button. 3. Click the “Profile” button. 4. On the Profile Page, click “Update Info” and modify the content 5. Click the “Update” to confirm the update. |
| Postconditions: | 1. New user profile is updated to the database. 2. Users can view the new profile on their profile page. |
| Priority | Low |
| Alternative flow or exception: | None |
| Assumptions: | 1. The connection between CAS service and database works fine. |
| Source: | None |

#### 

#### User change setting

|  |  |
| --- | --- |
| Use case name: | User change setting |
| Unique case ID: | UserChangeSetting |
| Primary Actor(s) : | User |
| Secondary Actor(s): | None |
| Brief description: | Users can change the setting of their accounts. |
| Preconditions: | 1. Users have access to CAS service. 2. Users have registered for a CAS account. |
| Flow of events: | 1. Users go to the CAS website, log in to their account. 2. On the main page, click the user name on the top-left corner, which will then show the “Profile” button and “Setting” button. 3. Click the “Setting” button. 4. On the Profile Page, click “Change Password” and enter the new password. 5. Click the “Confirm” to confirm the update. |
| Postconditions: | 1. New user settings are updated to the database. 2. Users can login with new passwords. |
| Priority | High |
| Alternative flow or exception: | None |
| Assumptions: | 1. The connection between CAS service and database works fine. |
| Source: | None |

#### User manage project accessibility

|  |  |
| --- | --- |
| Use case name: | User manage project accessibility |
| Unique case ID: | UserManageProjectAccess |
| Primary Actor(s) : | User |
| Secondary Actor(s): | None |
| Brief description: | Users can share or stop sharing their project with other users. |
| Preconditions: | 1. Users have access to CAS service. 2. Users have registered for a CAS account. 3. Users have registered a project on CAS. 4. Users have to know the username of the user they want to share with. |
| Flow of events: | 1. Users go to the CAS website, log into their account. 2. On the main page, click “Own Projects”. 3. Select an already uploaded project. 4. Click the “Project Setting” button. 5. Enter the username of whom the owner wants to share with. 6. Click the “Share” button to confirm. |
| Postconditions: | 1. The user which this project shared is able to view the project on their “Shared Projects” page. 2. The owner of the project will view all the shared members of the projects. |
| Priority | High |
| Alternative flow or exception: | 1. In step 5, users can enter the username whom the owner wants to stop sharing with and in step 6, click “Confirm” to stop sharing. |
| Assumptions: | 1. The connection between CAS service and database works fine. |
| Source: | None |

#### Admin login

|  |  |
| --- | --- |
| Use case name: | Admin login |
| Unique case ID: | AdminLogin |
| Primary Actor(s) : | Admin |
| Secondary Actor(s): | None |
| Brief description: | Login as admin and manage the CAS service. |
| Preconditions: | 1. Have the admin credential. |
| Flow of events: | 1. On the Login page of CAS website, click “Login as admin”. 2. On the admin login page, enter the username and password for the admin user. 3. Click “Log in” and type the admin’s credential. |
| Postconditions: | 1. Login as administrator of the CAS and manage the system. |
| Priority | High |
| Alternative flow or exception: | 1. After step 3, admin can maintain the User, Group, Projects and Controls. 2. Admin can click “VIEW SITE” on the right corner of the page to view the normal CAS web page. |
| Assumptions: | 1. The connection between CAS and database is working fine. |
| Source: | None |

#### 

#### Register project on CAS

|  |  |
| --- | --- |
| Use case name: | Register project on CAS |
| Unique case ID: | RegisterProjectOnCAS |
| Primary Actor(s) : | User |
| Secondary Actor(s): | None |
| Brief description: | Users can register a new project on CAS. |
| Preconditions: | 1. Users have access to CAS service. 2. Users have registered for a CAS account. |
| Flow of events: | 1. Users go to the CAS website, log in to their account. 2. On the main page, users click on “New Project” to go to the registering page. 3. On the registering page, users input the basic information of the project.    1. Users type the “Project Name”, “Project Description” into different input boxes respectively.    2. Users click on the register button to finish the registration. The page will automatically go to the project information page. 4. The project information page would show all the information including “Project Name”, “Project Description”. The page will also show the automatically generated random “Project ID” used for connection to the CI platform (E.g. Jenkins). |
| Postconditions: | 1. A new project is registered. 2. The project information is stored. 3. Project IDs are automatically generated. |
| Priority | High |
| Alternative flow or exception: | 1. In Step 3.2, if the user left any input boxes blank, stay on the same page and show “please fill in the blank” notices on the blank boxes. |
| Non-behavioral requirements | None |
| Assumptions: | 1. The project information includes “Project Name”, “Project Description”. 2. Users need “Project ID” to build an encrypted connection to the CI platform. |
| Source: | None |

#### 

#### Configure the controller list for project

|  |  |
| --- | --- |
| Use case name: | Configure the controller list for project |
| Unique case ID: | ConfigureProjectControllers |
| Primary Actor(s) : | User |
| Secondary Actor(s): | None |
| Brief description: | Users configure the controllers that they are concerned with for a certain project. |
| Preconditions: | 1. Users have access to CAS service. 2. Users have registered for a CAS account. 3. The project has been registered on CAS. |
| Flow of events: | 1. Users go to the CAS website, log in to their account. 2. Go to the page of “Own Projects”. 3. Choose the name of the project that users want to configure. 4. Click “Config Controls” to go to the page which shows all provided controllers in tree structure. 5. Select some classes of controllers to get all the controllers of these classes selected. 6. Click a controller class to make the controllers of this class expand below it, then select some of them. 7. Input the keywords of a certain controller in the search box then find the controller the users want, then select it. 8. After selecting all the controllers to be tested, click the “Save” button. |
| Postconditions: | 1. The controllers to be tested are configured for this project. 2. If some new tests are completed, the dashboard will show the result of these selected controllers. |
| Priority | High |
| Alternative flow or exception: | None |
| Non-behavioral requirements | 1. The list of provided controllers to be selected should be shown in a clear view so that the users can browse and find them easily. |
| Assumptions: | 1. Users are familiar with the controllers. |
| Source: | None |

#### Admins install CAS plugin on jenkins

|  |  |
| --- | --- |
| Use case name: | Admins install CAS plugin on jenkins |
| Unique case ID: | JenkinsInstallCASPlugin |
| Primary Actor(s) : | Admin |
| Secondary Actor(s): | None |
| Brief description: | Admins install the CAS plugin on their Jenkins service. |
| Preconditions: | 1. Admins have set up a Jenkins server. |
| Flow of events: | 1. Admins log on the the Jenkins GUI 2. Select “Manage Jenkins” 3. Select “Manage Plugins” 4. Click the “Advanced” tab 5. Go to “Upload Plugin”. 6. Select “cats-plugin”. 7. Click “Upload”. 8. Wait until the installation progress page displays “Success” |
| Postconditions: | 1. Cats plugin has been installed on Jenkins |
| Priority | High |
| Alternative flow or exception: | In event flow step 8, admins can also do:   1. Check the box “Restart Jenkins when installation is complete and no jobs are running”. 2. Wait until the download progress page displays “Please wait while Jenkins is restarting ...” 3. Login to Jenkins system again.   Then the postconditions can be reached with this alternative flow. |
| Non-behavioral requirements | None |
| Assumptions: | Admins are familiar with Jenkins configuration processes. |
| Source: | Own experience using Jenkins and Jenkins plugins. |

#### Admin configure CAS plugin on Jenkins

|  |  |
| --- | --- |
| Use case name: | Admin configure CAS plugin on Jenkins |
| Unique case ID: | JenkinsConfigCASPlugin |
| Primary Actor(s) : | Admin |
| Secondary Actor(s): | None |
| Brief description: | Admins have installed CAS plugin on Jenkins |
| Preconditions: | 1. Admins have set up a Jenkins server. 2. Admins have installed CAS plugin on Jenkins. 3. Admins have logged on to Jenkins GUI. 4. Admins have set up a CAS server. 5. Admins have registered for a CAS account. |
| Flow of events: | 1. Go to Jenkins system configuration 2. Under “CAS servers”, click “Add CAS” and provide: 3. CAS server name 4. CAS server URL 5. CAS server authentication token 6. Create a Jenkins job 7. Go to the job configuration 8. Under “Build”, click “Add build step”, and select “Execute CAS check” 9. Provide the following information to Jenkins: 10. Project ID generated by CAS 11. Project authentication token 12. Save build configuration. |
| Postconditions: | 1. Jenkins automatically invokes CAS check after every build of a project. |
| Priority | High |
| Alternative flow or exception: | None |
| Non-behavioral requirements | None |
| Assumptions: | Users are familiar with Jenkins build and configuration processes. |
| Source: | Own experience using Jenkins and Jenkins plugins. |

#### View CAS result on dashboard

|  |  |
| --- | --- |
| Use case name: | View CAS result on dashboard |
| Unique case ID: | ViewCASDashboard |
| Primary Actor(s) : | User |
| Secondary Actor(s): | None |
| Brief description: | Users can view the testing results of their existing project by visiting the dashboard. |
| Preconditions: | 1. Users have a CAS account. 2. Users have already set up a project on their cas. 3. Users have internet access. |
| Flow of events: | 1. Users go to the CAS website, log in to their account. 2. On the project page, click the project name you want to check to go to the dashboard of the chosen project.. 3. View the dashboard on the existing project page. |
| Postconditions: | 1. Users can see the test results of an existing project by visiting the dashboard page for that project. |
| Priority | High |
| Alternative flow or exception: | 1. On the dashboard page, users can click on each testing case to view the detailed information of that case. Possible information includes:    1. Case name.    2. Controller associated with this case.    3. Start time for that case.    4. End time for that case.    5. Output for that test case.    6. Error message for that test case if it fails. |
| Non-behavioral requirements | 1. The dashboard should show a list of test cases associated with their names, start time, end time and test results. Users should be able to click on each test case to view detailed information of that test case. |
| Assumptions: | 1. The connection between CAS and Jenkins works fine. |
| Source: | None |