# Section 1: Model Architecture

This section presents the design of the proposed model to manage customer involvement in distributed teams. It discusses how a computer-based system would be able to provide the needed coordination. The Section begins with an overview of the high-level architecture for the system components needed to manage the customer involvement (Section 1.1). The data model of the system is presented in Section 1.2.

## System Architecture

The customer involvement management model proposed in this research focus mainly on activities within planning and releasing stages. Figure 1 shows a high-level architecture of the system and its components. In order to achieve the required coordination support that is needed for customer involvement, the model needs to:

* be linked with both development and testing environments.
* be integrated with the existing agile project management tools.



Figure 1. A High-Level Architecture for the Customer Involvement Management System

**Linked with Both Development and Testing Environments**

This link is needed to keep track of changes on user story states and the progress of the development and testing, as well as providing a direct link with automated testing tools to provide results of the acceptance testing. Possible customer activities that may require this linkage is modifying user stories, running acceptance tests, and deleting acceptance tests.

**Integrated with the Existing Agile Project Management Tool**

Customer involvement with the development team needs to be done directly through the same agile project management tool to provide a seamless user experience. However, the customer’s involvement points and the actions which they are allowed to perform should be well defined.

## Data Model

There are different types of data entities required in the proposed customer involvement management system. The UML class diagram shown in Figure 2‎ shows the main entities in the system and the relationships among them in an abstract way. Considering the customer as the main point of interest:

* Customer manipulates one or many user stories (initiate add, edit, delete).
* Customer manipulates one or more acceptance tests of a user story (add, edit, delete, run, change status).
* Customer gives one or more feedback.
* Customer receives notifications.
* One customer or more gives at least one prioritization of a user story.

Moreover, as an overall view of the agile development, the project manager is responsible for the releases. Each release contains multiple user stories which need to be approved by the project manager. Each user story is developed by one or more team members. In addition, each user story must have one or more acceptance tests.



Figure 2. Rational Model of the Customer Involvement Management System.

Each entity has some attributes that represent the needed data to develop the proposed model. Table ‎1 shows a summary of the data entities and their attributes. Figure 3 shows the proposed model Entity-Relationship (ER) diagram.

Table 1. Proposed Model Data Entities and Attributes

| **Data Entity** | **Attributes** |
| --- | --- |
| Release | ID, ActualStart, ActualComplete, PlannedStart, PlannedComplete, OwnerID |
| Iteration | ID, ActualSart, ActualComplete, releaseID |
| User Story | ID, Name, Description, InitialPriority, PriorityMethod, ActualStart, ActualComplete, PlannedStart, PlannedComplete, State, CustomerID, DeveloperID, testerID, IterationID, releaseID |
| Story Priority | ID, StoryID, customerID, PriorityMethod, Priority |
| Acceptance Test | ID, Name, Description, Status, Type, StoryID, customerID |
| Product Owner | ID, Name, Tel, Email, Location |
| Customer | ID, Name, Tel, Email, Location, Rank |
| Developer | ID, Name, Tel, Email, Location |
| Tester | ID, Name, Tel, Email, Location |
| Feedback | ID, Title, Details, Date, userID, ItemType, ItemID |
| Notification | ID, Message, Date, userID |



Figure 3. Proposed Model ER Diagram

# Section 2: Implementation

As a proof of concept, a web-based system has been developed to demonstrate the possibility to implement a computer-based system of the proposed approach to manage customer involvement. The system has been developed using PHP and JavaScript, which are both well-known languages. The database was developed using MySQL which is compatible and supported by the PHP. The XAMPP has been used as the local server, and the NetBeans IDE has been used as the development environment. The reason for choosing all these languages or tools is for their portability and popularity.

All the activities initiated by the customer have been implemented in order to make sure of the system feasibility. This includes adding, modifying, deleting user stories and acceptance tests by the customer. It also includes prioritization user story and running acceptance tests by the customer. Moreover, it includes providing feedback as well. However, the detailed analysis provided in this document will focus only on the three selected scenarios. The system database is first presented in Section 2.1 while the implementation of three scenarios are provided in Section 2.2, Section 2.3 and Section 2.4.

**2.1 System Database**

The backend SQL database of the data model provided in Section 1.2 has been created (Figure 4). This database is capable of demonstrating all of the customer involvement activities provided on this research, not only for the selected scenarios.

The database state of the system before applying the three scenarios is as follows:

* There is a past release (release 1)
* The current release is release 2 which contains one user story US1.
* A future release is release 3 which contains US1-Next.
* Both C1: Ahmed and C2: Sara are included as customers for the current release.



Figure 4. System Database

**2.2 Implementation of Scenario 1**

The implementation of the proposed approach steps of scenario 1: initiate adding user story and its acceptance tests is shown below:

1. C1: Ahmed initiates adding a user story US2 to the current release (Figure 5 and Figure 6):

First Ahmed will log in, according to his privilege, options will be available (e.g., release that he is authorized to work in, list of his user stories and acceptance tests).

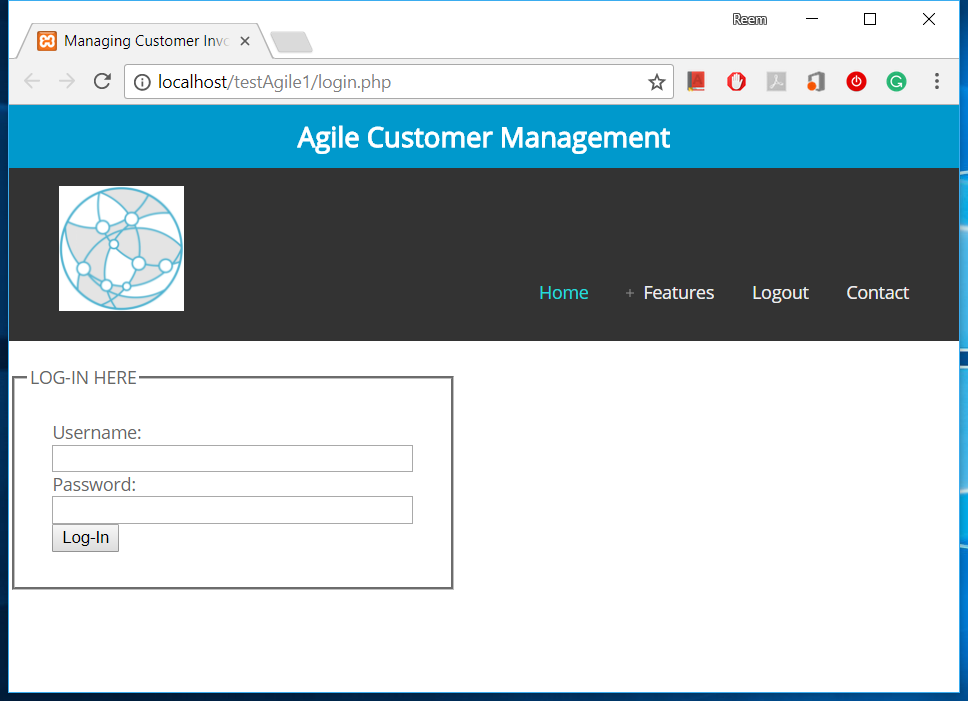


Figure 5. Customer Log-In

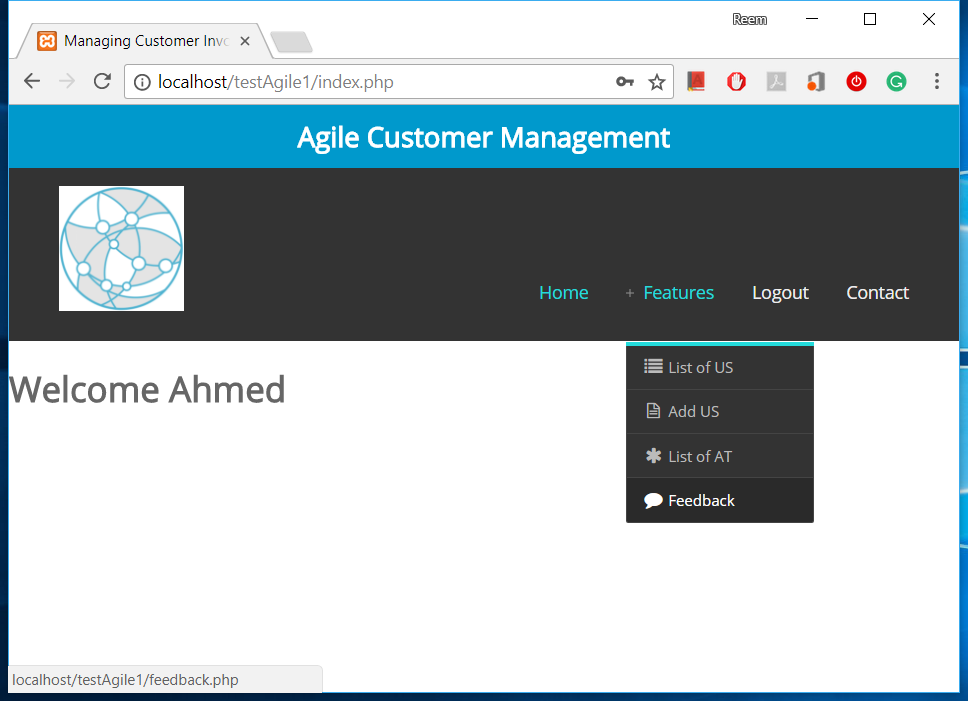


Figure 6. Customer Home Page and Main Functions

After selecting the release, the ‘add user story’ form will be displayed (Figure 7 and Figure 8).

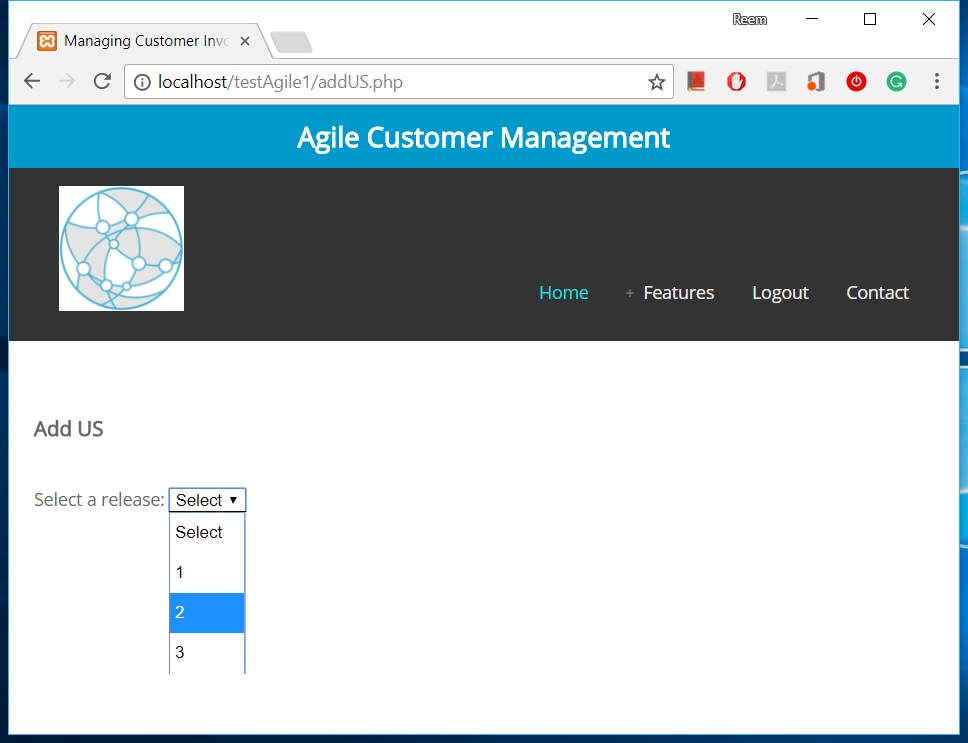


Figure 7. Select a Release to Add US

1. The system will validate US2 in terms of authority release, and time within the release.

* **Check Valid Release:**

If the previous release selected, the system will not allow adding, as shown in the below code:

// dont allow add to previous release

if ($releaseID < $currentRelease) {

$addMsg = "You can not add user story to a previous relese !";

echo "<script type='text/javascript'>

alert('$addMsg');</script>";

}

* **Check Authority:**

The system will check if logged in the customer is authorized to work on the selected release, as shown in the below code:

if (!checkAuthority($releaseID, $owner)) {

$msg = "You are not autherized to add US on this release !";

echo "<script type='text/javascript'>alert('$addMsg');</script>";

}

The code below shows the SQL statement needed to check the authority:

$query = "SELECT \* FROM releaseCustomer WHERE releaseID = $releaseID AND customerID = $userID ";

* **Check time within release:**

The SQL statement used to retrieve time within release is given below:

$query = "SELECT PlannedComplete FROM projectRelease WHERE ID = $releaseID;";

If there is still at least 24 hours, the system will allow addition as shown in the code below:

if (($plannedComplete > $currentTime) && ($timeDifference > 1 ))

return true;

else

return false;

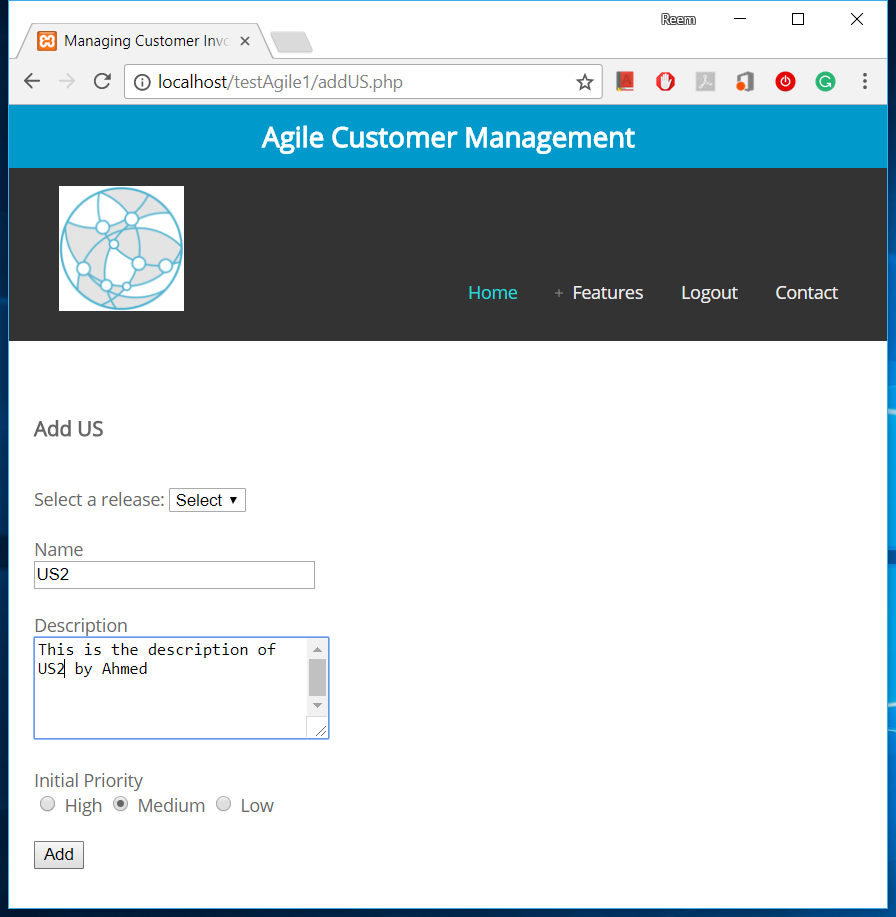


Figure 8. Add US Form.

1. Validation passed and because the addition request is on the current release, the PO:Marco approval is required. US2 state here is ‘Waiting for Approval’ (Figure 9 and Figure 10).

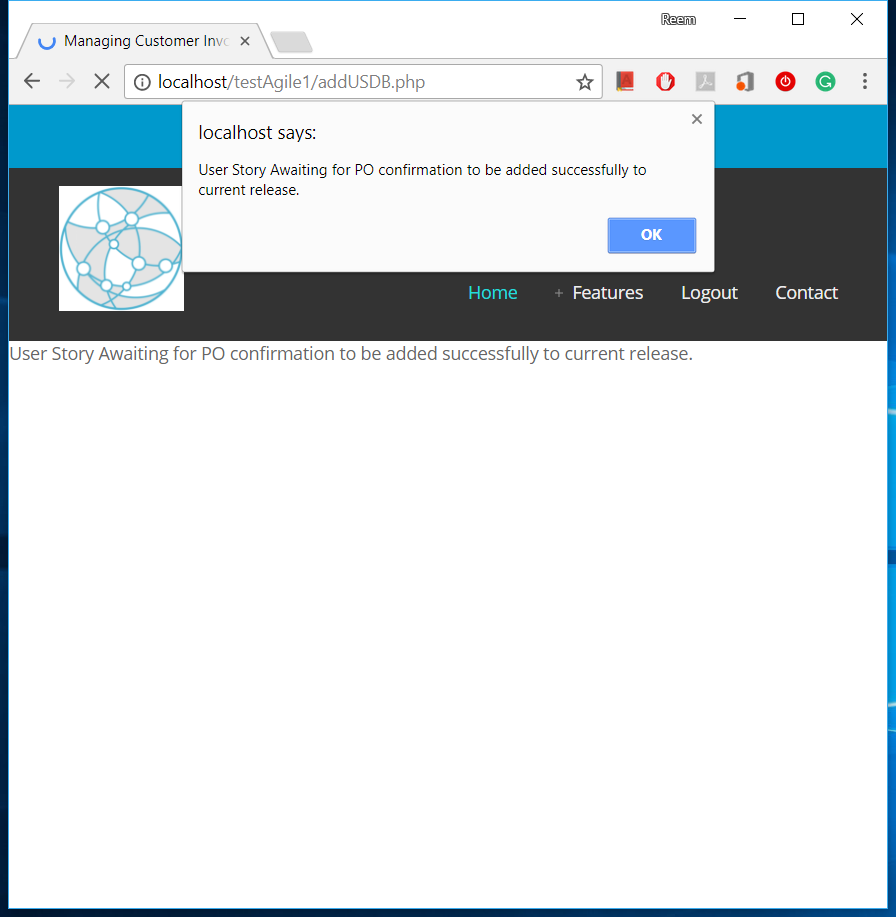
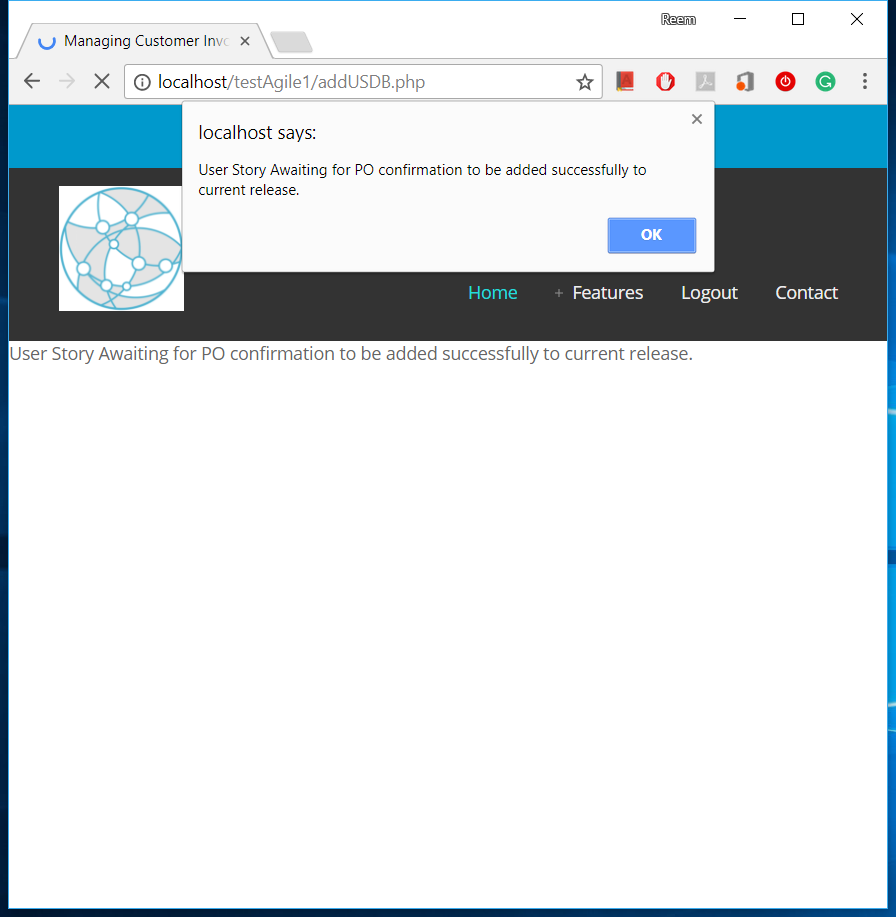


Figure 9. Add US Notification

The code below checks if current release selected, allow addition and ask for PO approval:

//check if current release selected -> ask PO approve

if ($releaseID == $currentRelease) {

$state = "Waiting for Confirmation";

$addMsg = "User Story Awaiting for PO confirmation to be added successfully to current release. ";

//notify PO

$msgPO = "$customer has added a new User Story to current release. Awaiting for your confirmation.";

notifyPM($releaseID, "US added to current release", $msgPO);

}

The SQL statement for the US insert is shown in the below query:

$query = "INSERT INTO `story`( `Name`, `Description`, `State`, `InitialPriority`, `CustomerID`, `IterationID`,`releaseID` ) "

. "VALUES ('$name','$description','$state','$priority','$owner', $iterationID, $releaseID) ";

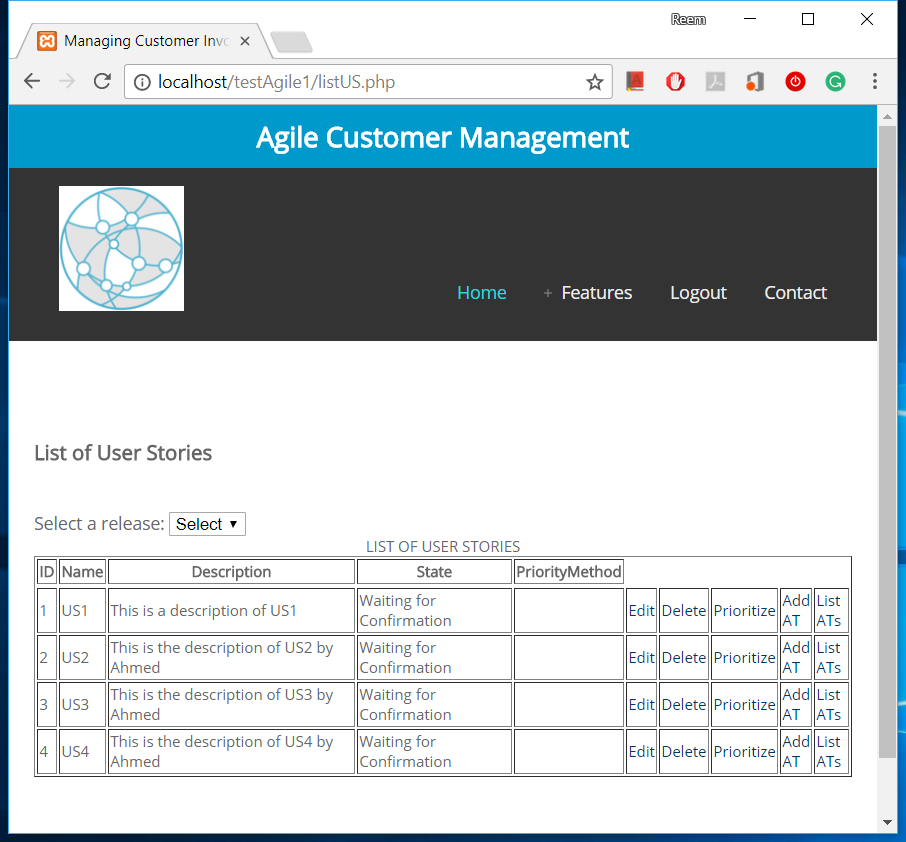
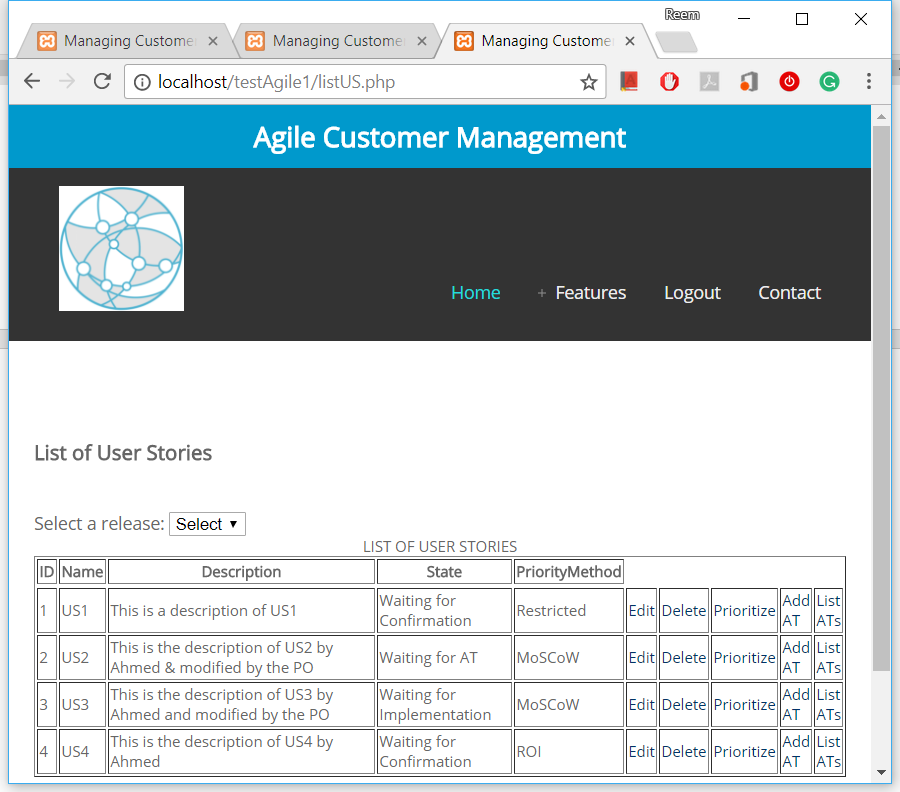


Figure 10. List of User Stories.

1. PO: Marco receives US2 addition request, as shown below (Figure 11).

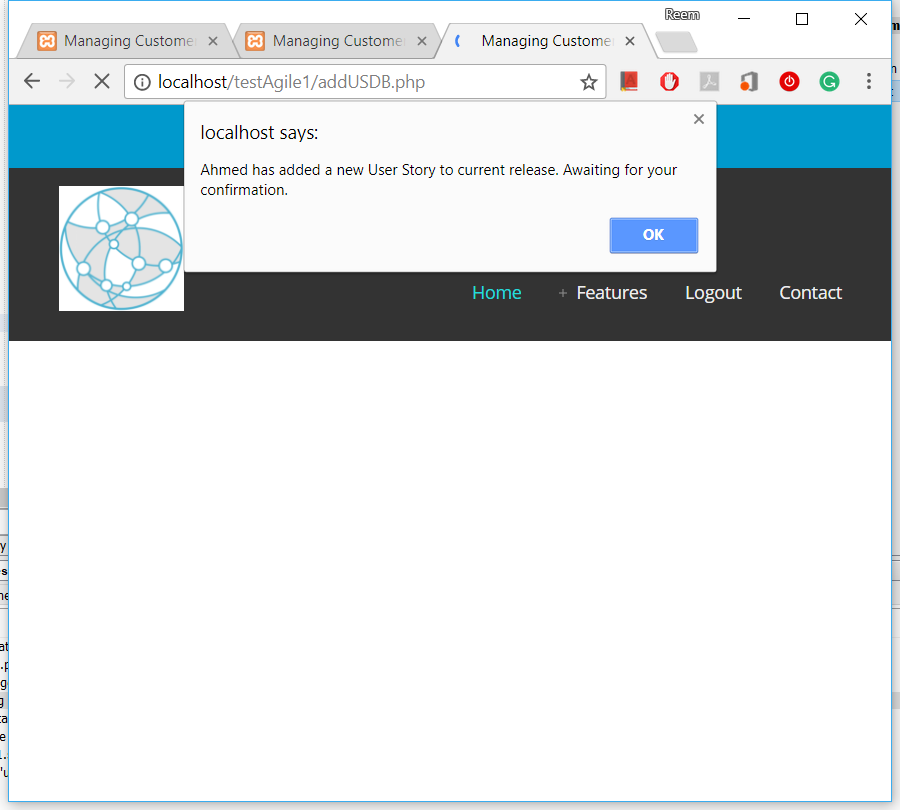
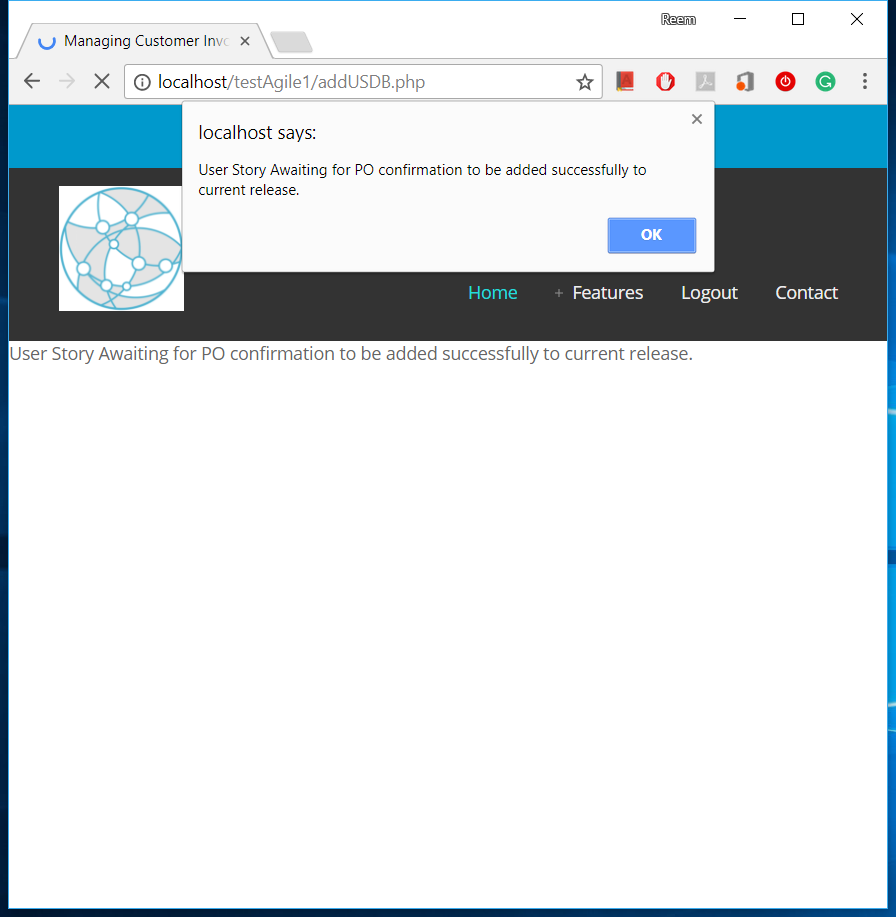


Figure 11. Request PO Approval to Add US

The below code shows sending the PO the approval request:

$msgPO = "$customer has added a new User Story to current release. Awaiting for your confirmation.";

notifyPM($releaseID, "US added", $msgPO);

1. PO: Marco accepts the addition of US2 with some modifications (Figure 12).

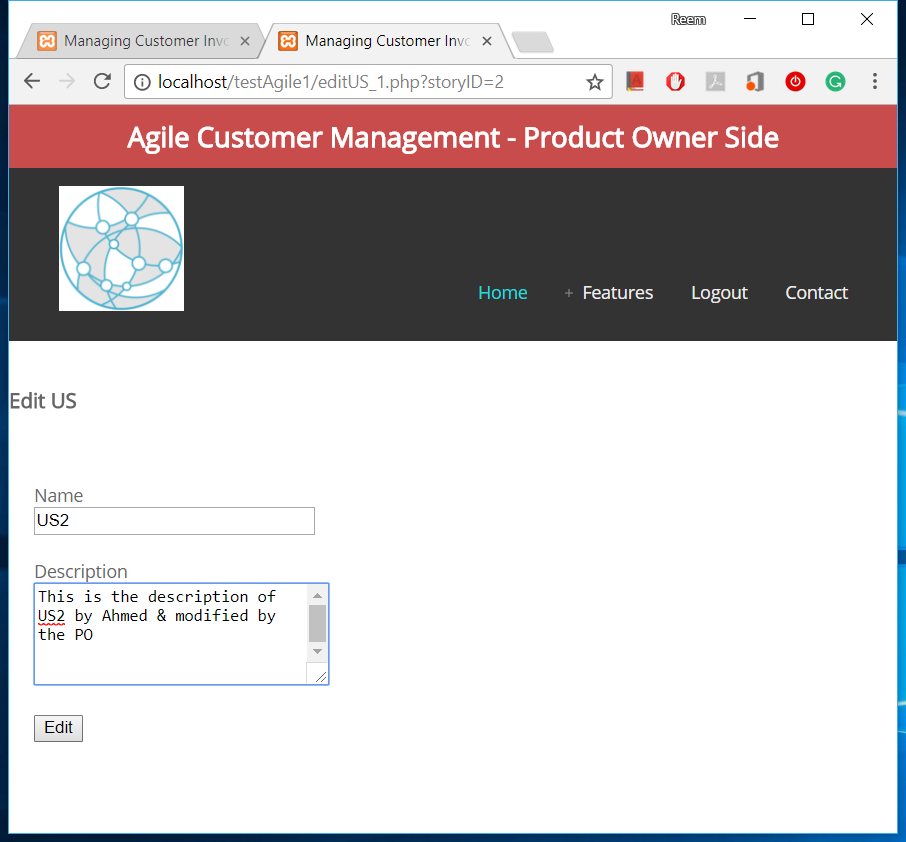


Figure 12. PO Approval to Add US with Modifications

1. C1: Ahmed receives notification about the modification and approves it (Figure 13).

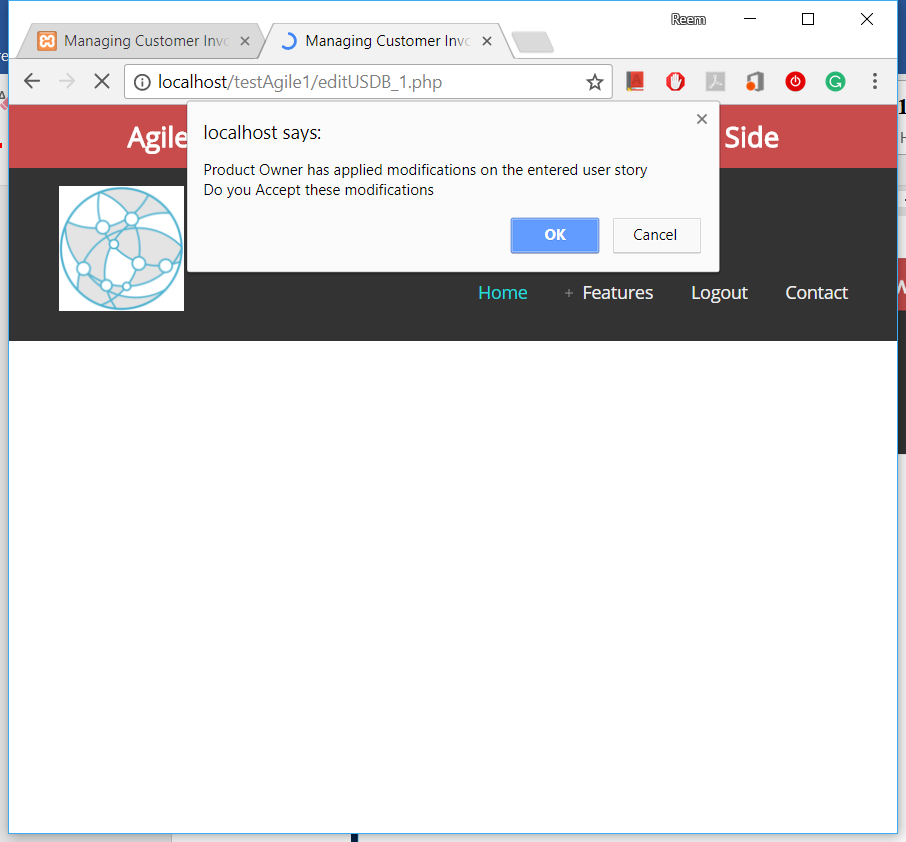
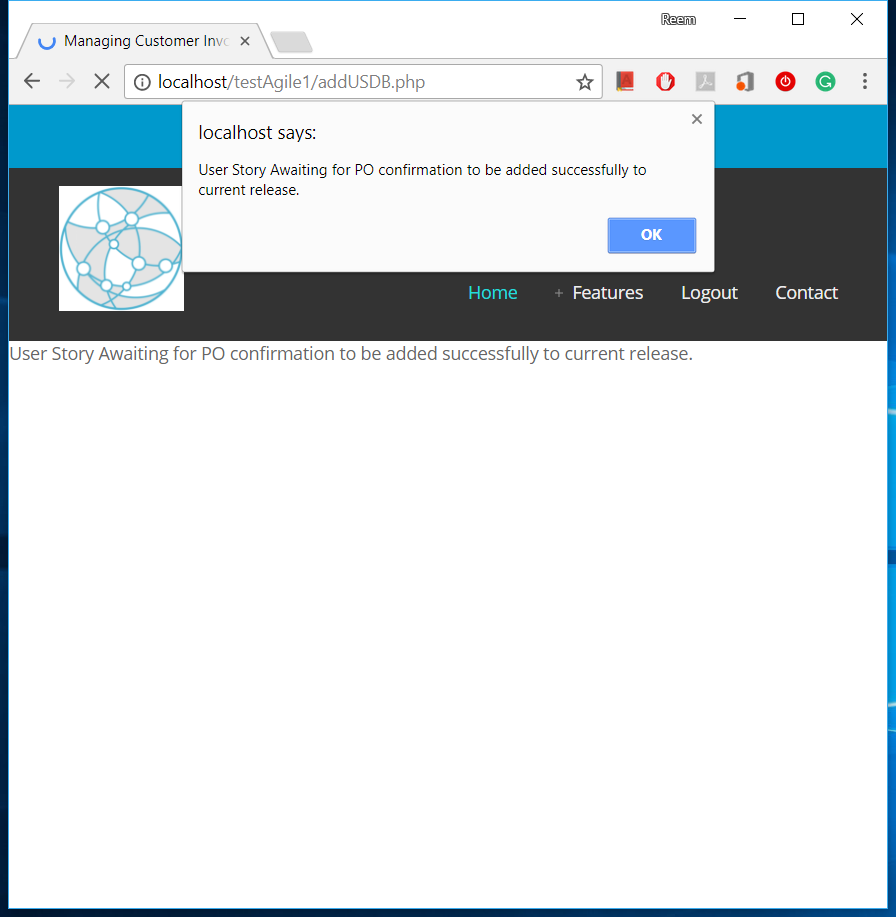


Figure 13. Request Customer Approval of PO Modifications

The below code shows the customer approval needed steps:

var msg = "Product Owner has applied modifications on the entered user story \nDo you Accept these modifications";

var customerConfirm = confirm(msg);

if (customerConfirm === true) { //apply modification

<?php $customerResponse = true; ?>

} else {// ignore modification

<?php $customerResponse = false; ?>

}

1. The system will complete the addition process and update the US2 state to ‘Waiting for Implementation’ (Figure 14).

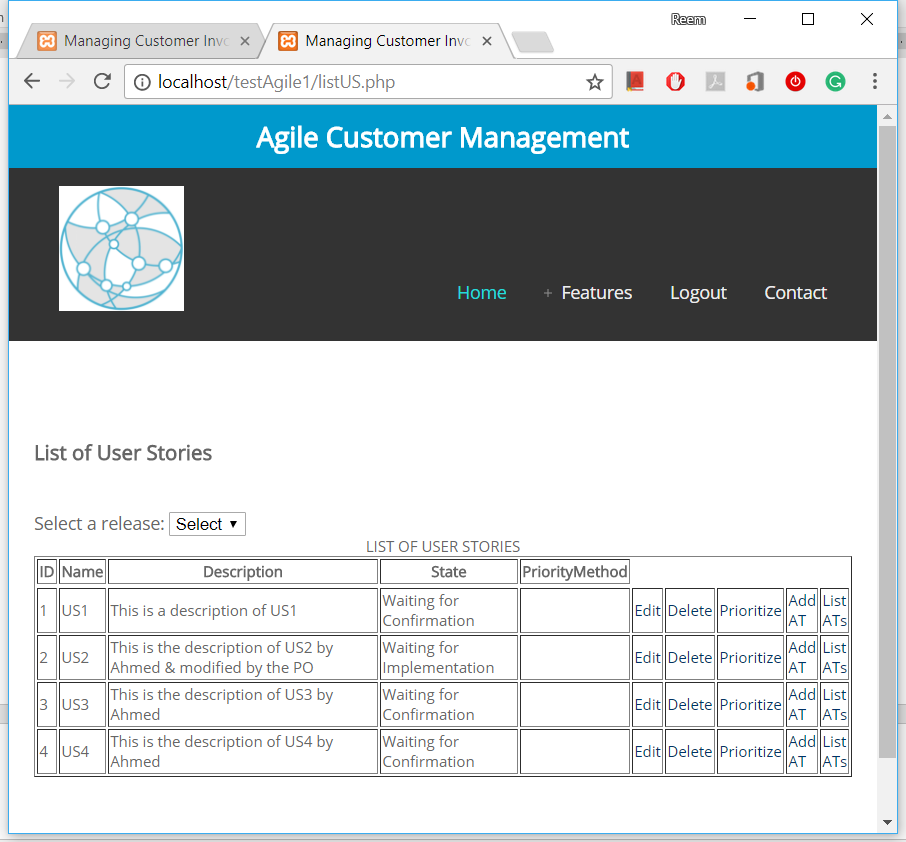
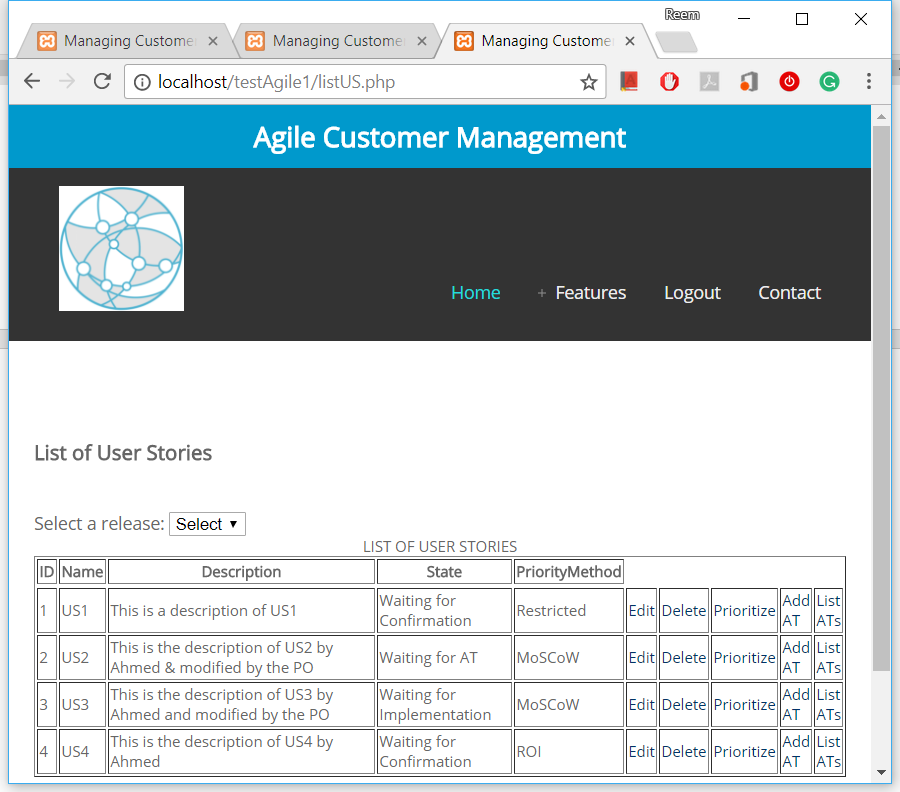


Figure 14. List of User Stories - Updated Status

The SQL statement that confirms the modification is shown below:

$state = 'Waiting for Implementation';

$query = "UPDATE `story` SET `Name` = '$name', `Description` = '$description' , `State` = '$state' WHERE `ID` = '$storyID' ;";

1. Automatically system will send a notification to C1: Ahmed, D1: Debi, and PO:Marco about the state of US2 (Figure 15).

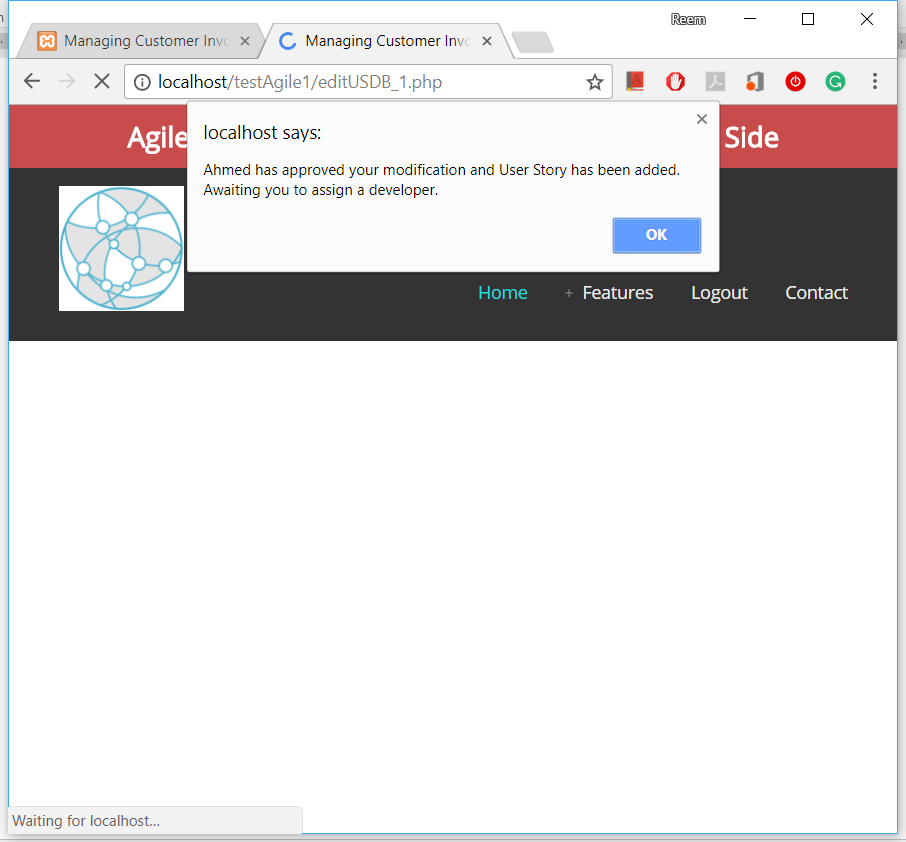
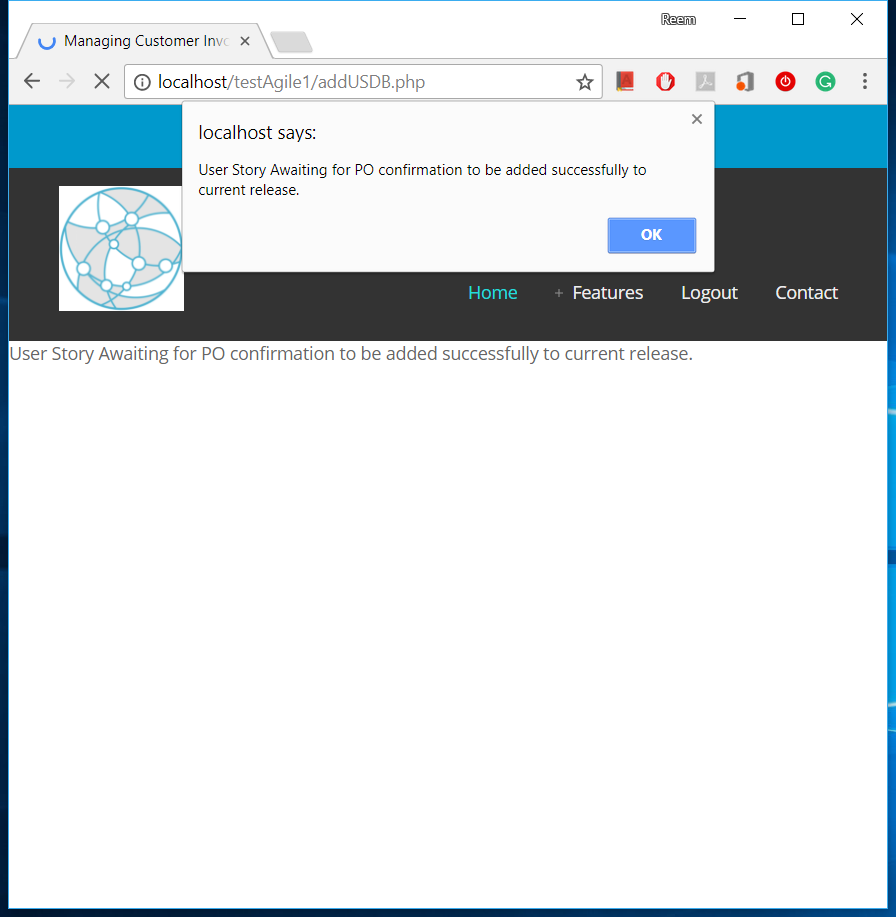


Figure 15. Notification of Customer Approval

The below code shows the notification to PO:

//notify PO

$msgPO = "$customer has approved your modification and User Story has been added. Awaiting you to assign a developer.";

notifyPM($releaseID, "US added to current release", $msgPO);

1. Concurrently after step number 3, C1:Ahmed will be allowed to add the acceptance test of the user story AT2.1 and AT2.2.

After adding the user story, an option to add its acceptance tests will be available to the customer (Figure 16 and Figure 17):

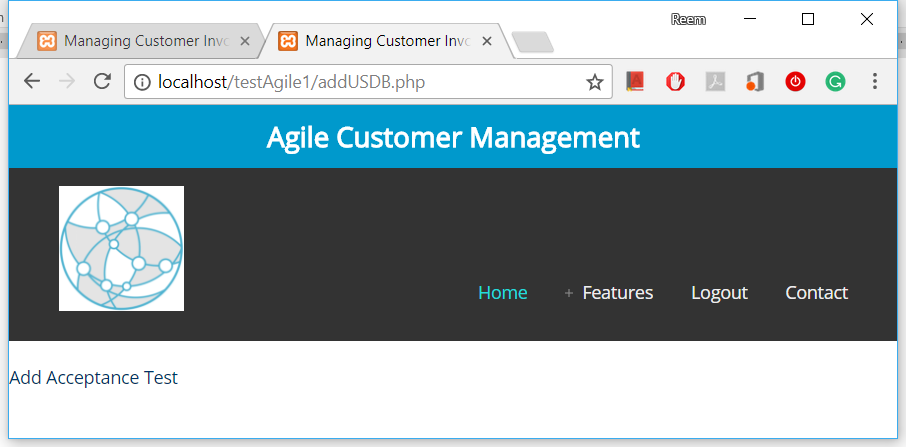
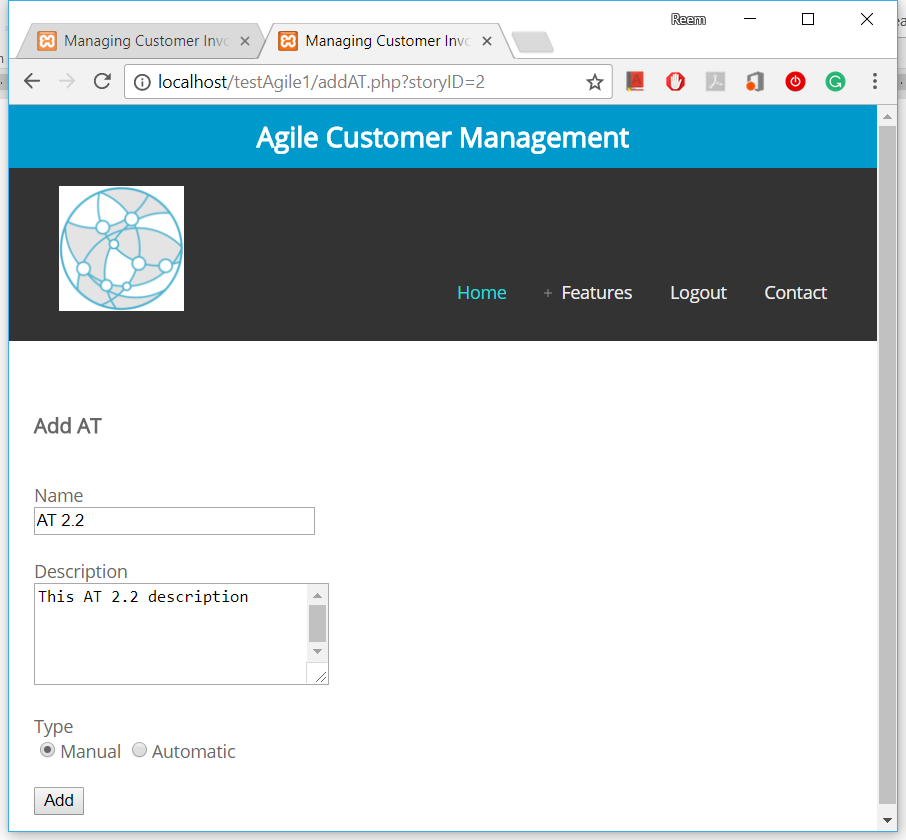


Figure 16. Add AT Option After Adding US.



**Add Acceptance Test**

Figure 17. Add AT Form

1. The same steps and validation will be applied, and PO:Marco will assign a tester T1:Jack to test and apply the ATs (Figure 18).

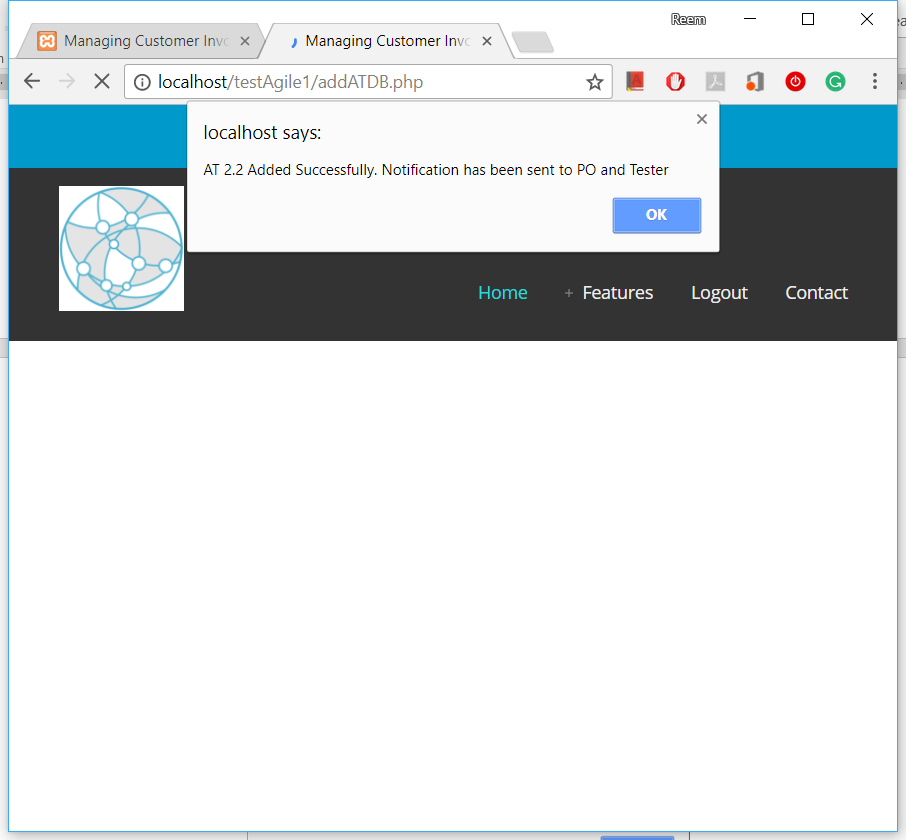
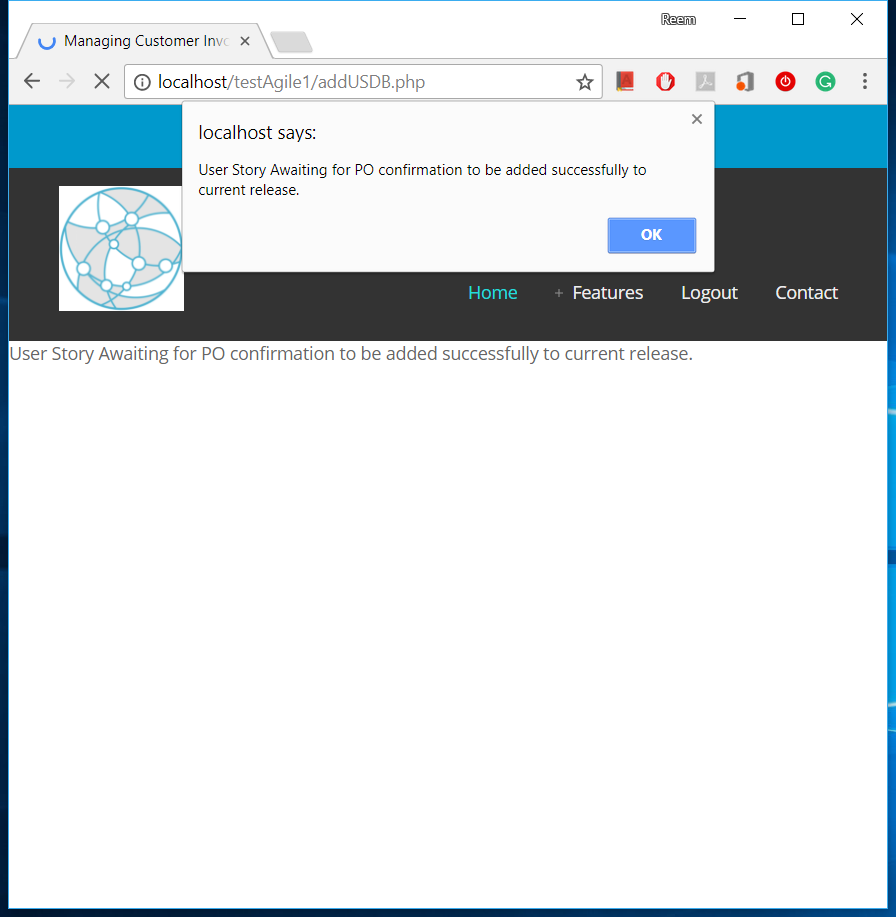


Figure 18. Successfully Adding AT Notification

The SQL statement of AT addition is shown below:

$query = "INSERT INTO `at`(`Name`, `Description`, `Status`, `Type`, `StoryID`, `customerID`) VALUES ('$name','$description','$status','$type','$storyID', $owner ) ";

1. A notification will be sent to affected users to update them with the state (C1:Ahmed, D1:Debi , T1:Jack) (Figure 19 and Figure 20).

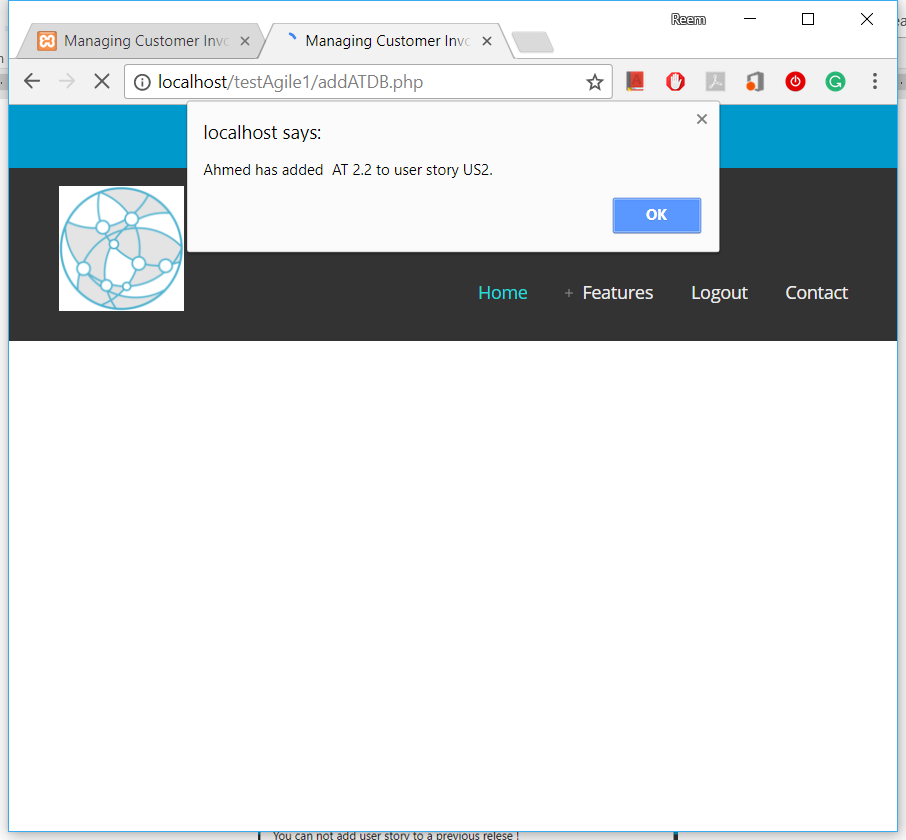
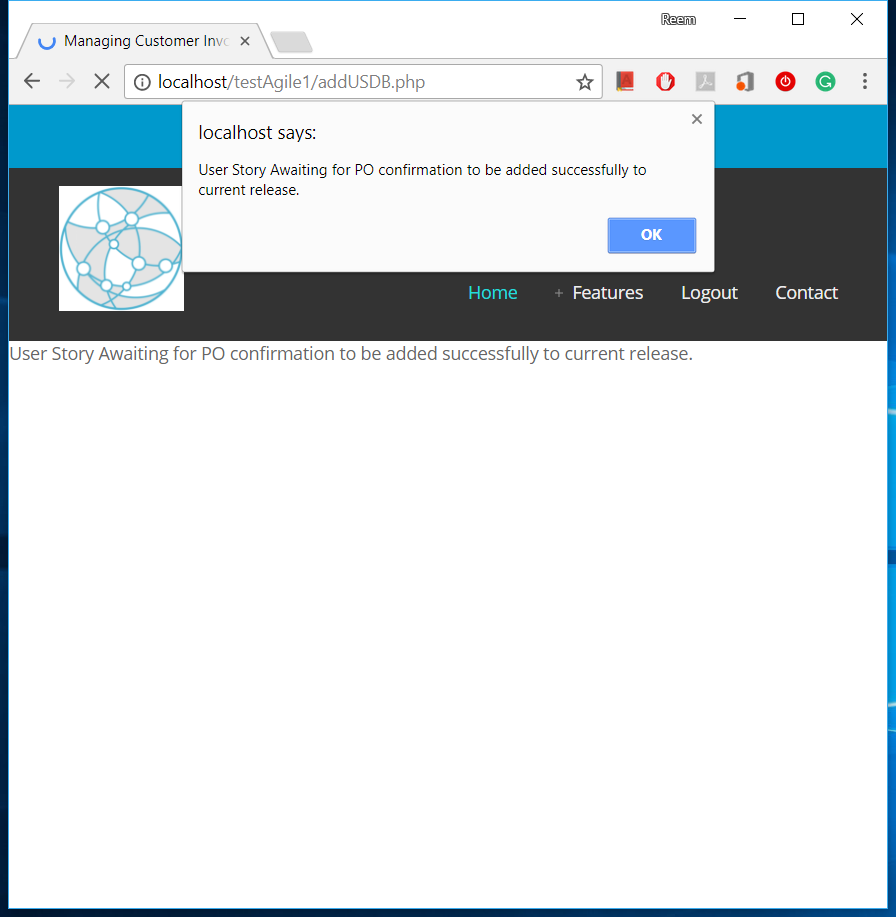
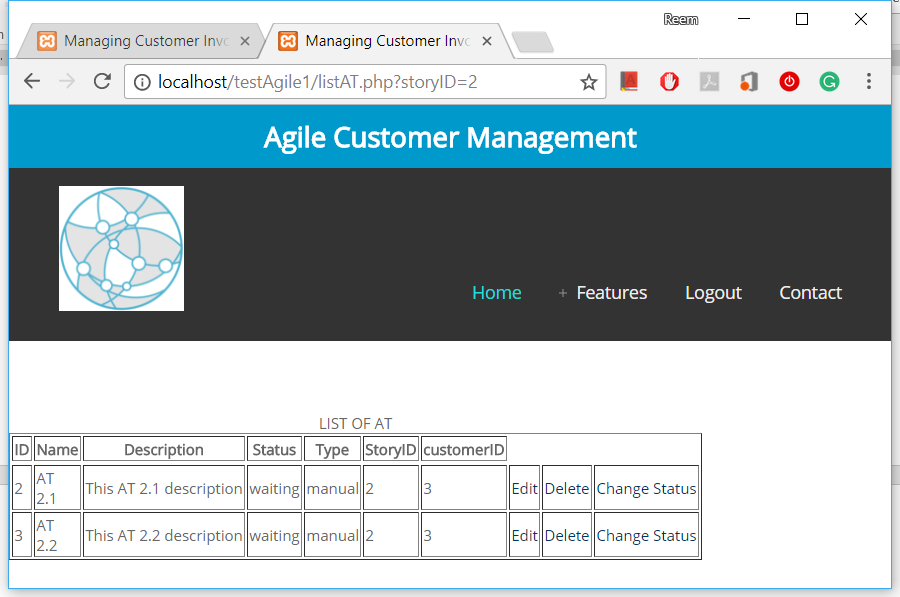


Figure 19. Adding AT Notification to Affected Team Member.



LIST OF ACCEPTANCE TESTS

Figure 20. List of ATs with Status.

**Further Possible Variant of the scenario and its Validation of Constraints:**

* System will not allow customer to add user story to a previous release (Figure 21).

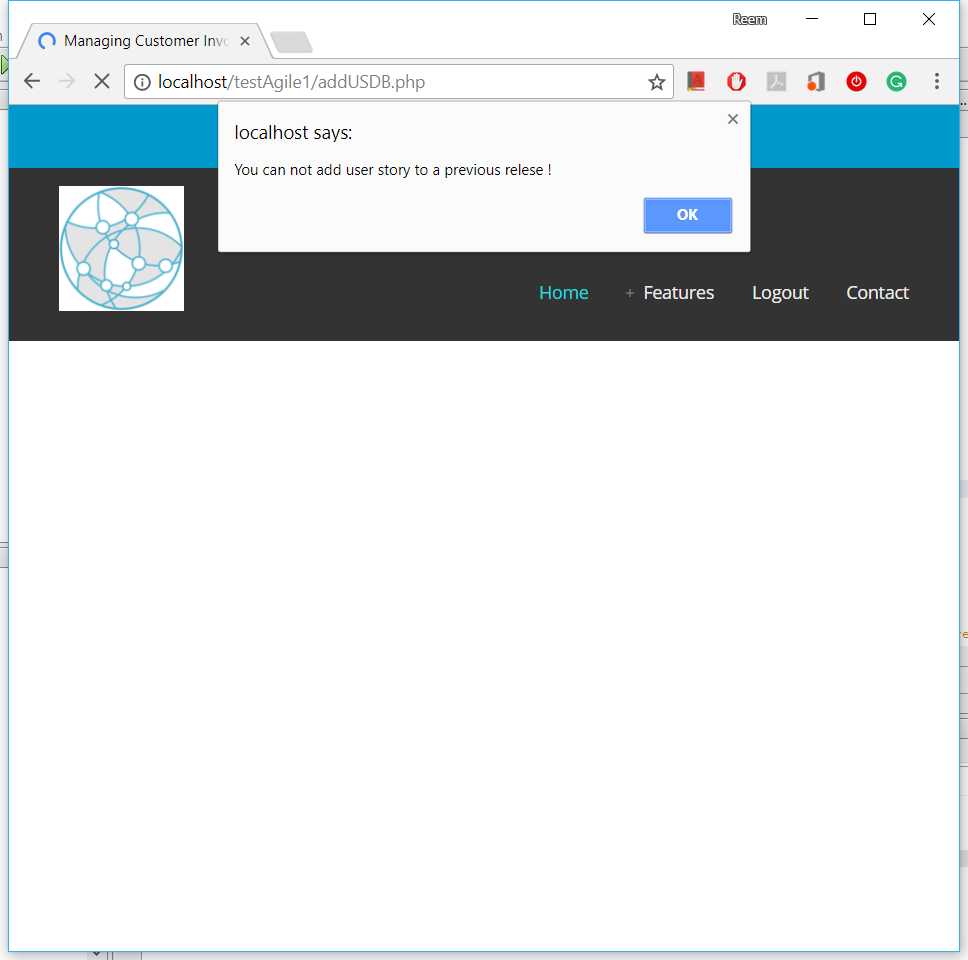
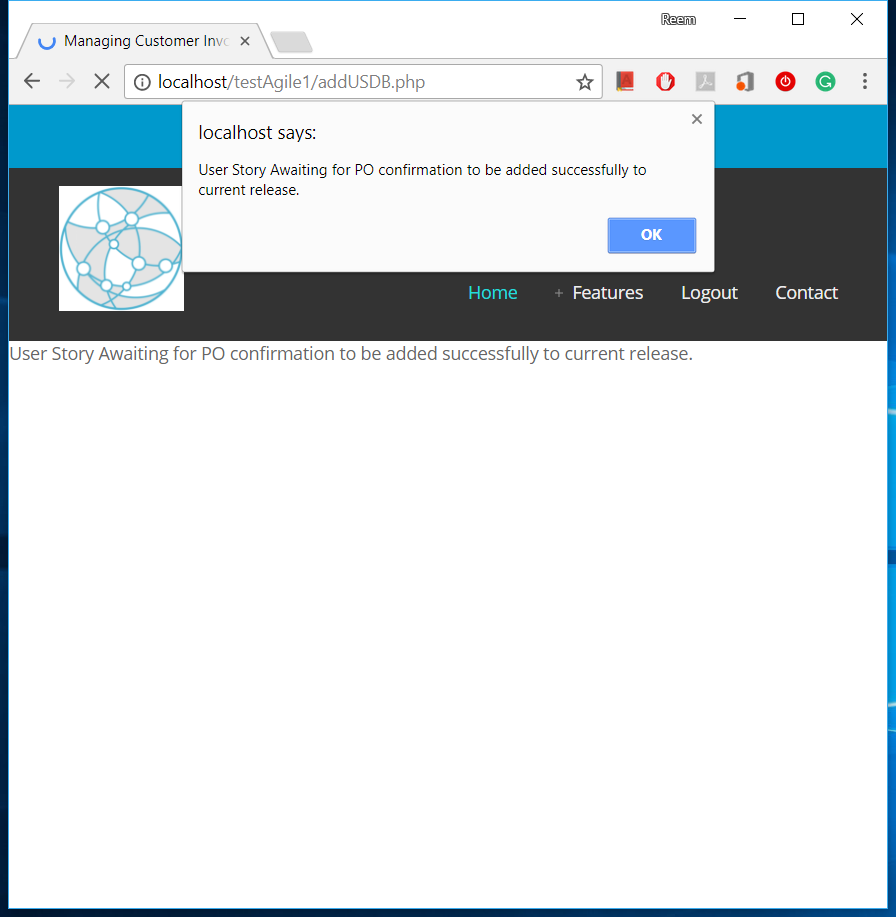


Figure 21. Prevent Adding US to Previous Release

* If future release selected, add and notify PO (as shown in the below code):

$addMsg = "User Story added successfully to next release. PO has been notified.";

$state = "Waiting for Implementation";

//notify PO

$msgPO = "$customer has added a new User Story to next release. Awaiting you to assign a developer.";

notifyPM($releaseID, "US added to next release", $msgPO);

* If user story was ‘Complete’ and customer added a new AT to it, its state will become ‘Waiting for AT’ again (as shown in the below code):

if (checkStateUS($storyID) == 'Complete') {

changeStateUS($storyID, 'Waiting for AT');

$msg = "User Story state becomes 'Waiting for AT' now";

echo "<script type='text/javascript'>alert('$msg');</script>";

}

**2.3 Implementation of Scenario 2**

The prioritization method needs to be set before customers can prioritize user stories. There are many prioritization techniques that could be used to prioritize requirements [108] [109] [110] [111] [112] [113]. In order to validate the implementation possibility of applying these techniques, two well-known prioritization methods have been selected, which were also recommended earlier on one of the conducted workshop:

* MoSCoW: Must Have, Should Have, Could Have, and Would Have
* ROI: Return on Investment.

The corresponding value of the MoSCoW drop down list is from 5 to 1 respectively, and the ROI is entered as a number representing a percentage.

The implementation of the proposed approach steps of scenario 2: Prioritize a user story is provided below:

1. PO:Marco decides the prioritization possibility and enter the prioritization method of each user story.

First the PO needs to log into the system (Figure 22):

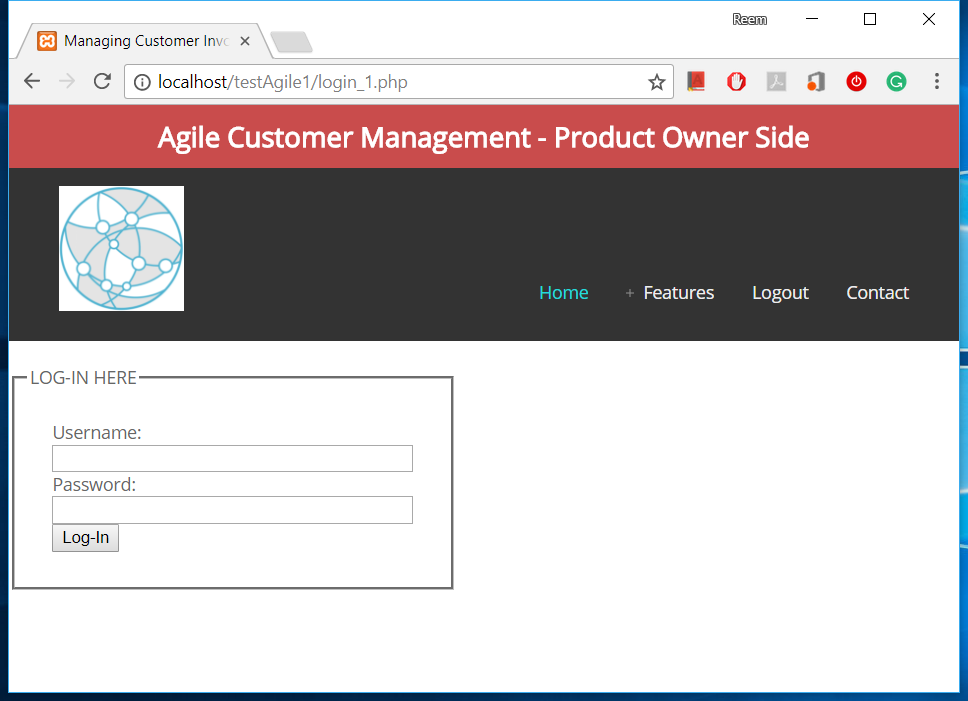
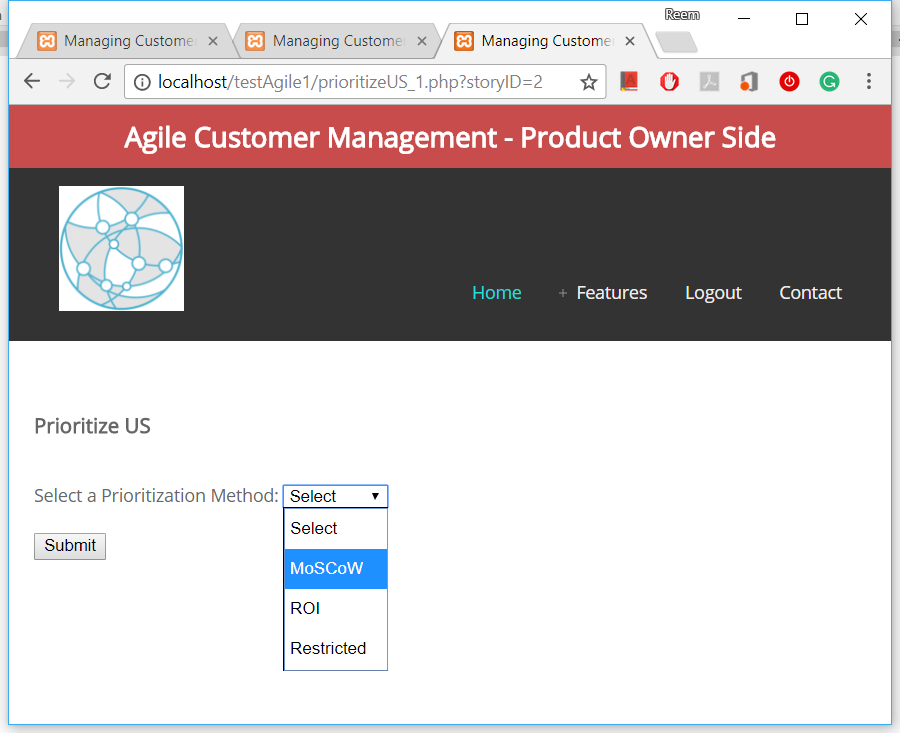


Figure 22. PO Login

The set Prioritization window will be available to him (Figure 23):



**Prioritize User Story**

Figure 23. Select Prioritization Method

This will affect the prioritization method of the user story that is displayed to the customer, who can now start prioritization (Figure 24).

The SQL statement that updates the prioritization method of the user story is shown in the code below:

$query = "UPDATE `story`

SET `PriorityMethod` = '$priorityMethod'

WHERE `ID` = '$storyID' ;";

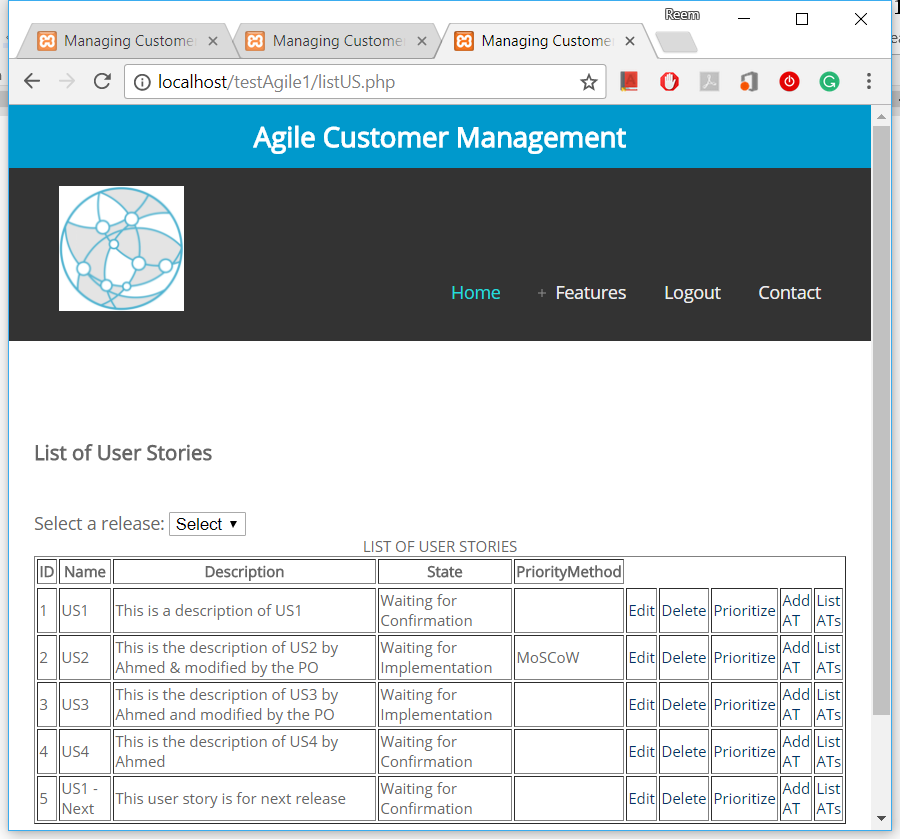
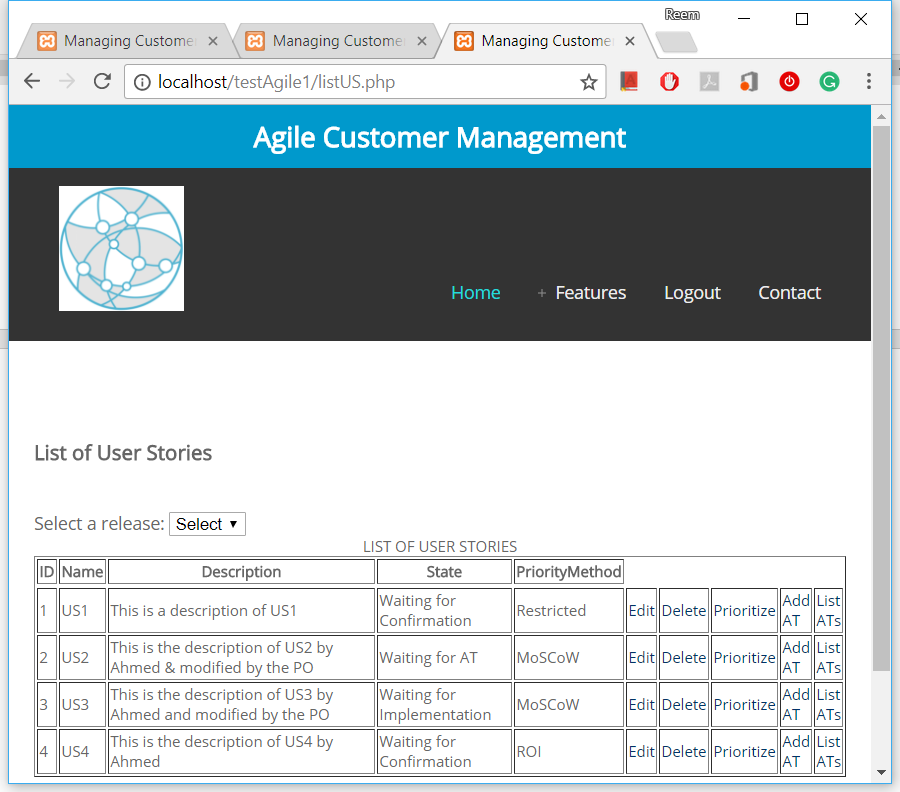


Figure 24. Prioritization Method Showed in List of USs

1. C1: Ahmed and C2:Sara will receive a notification and add their prioritization value for each user story (Figure 25).

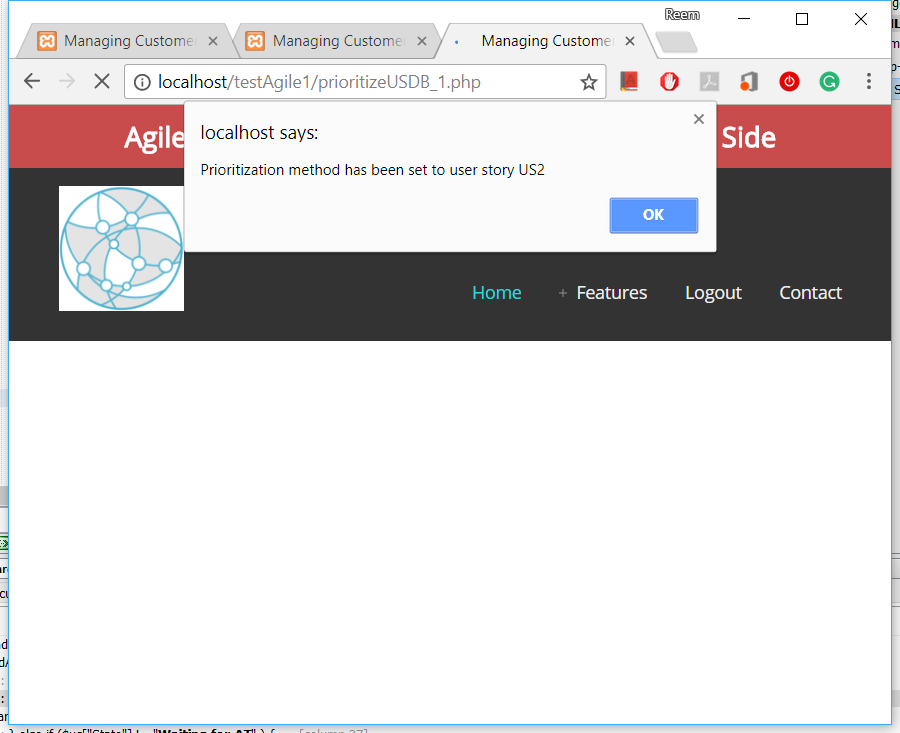
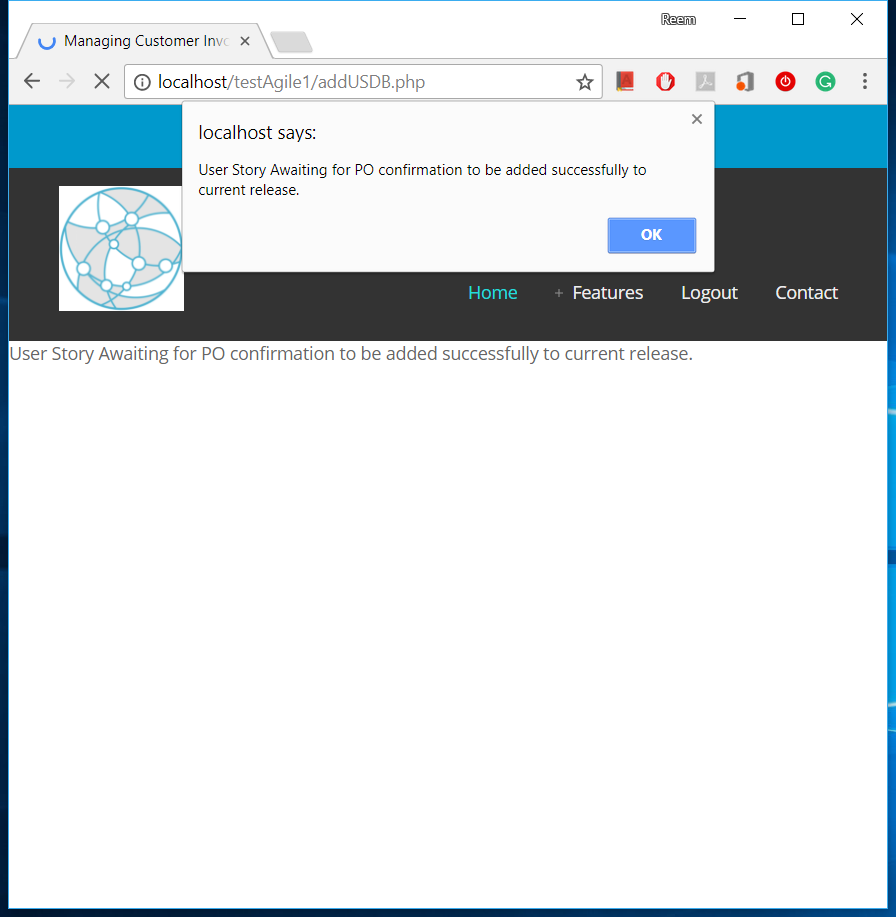
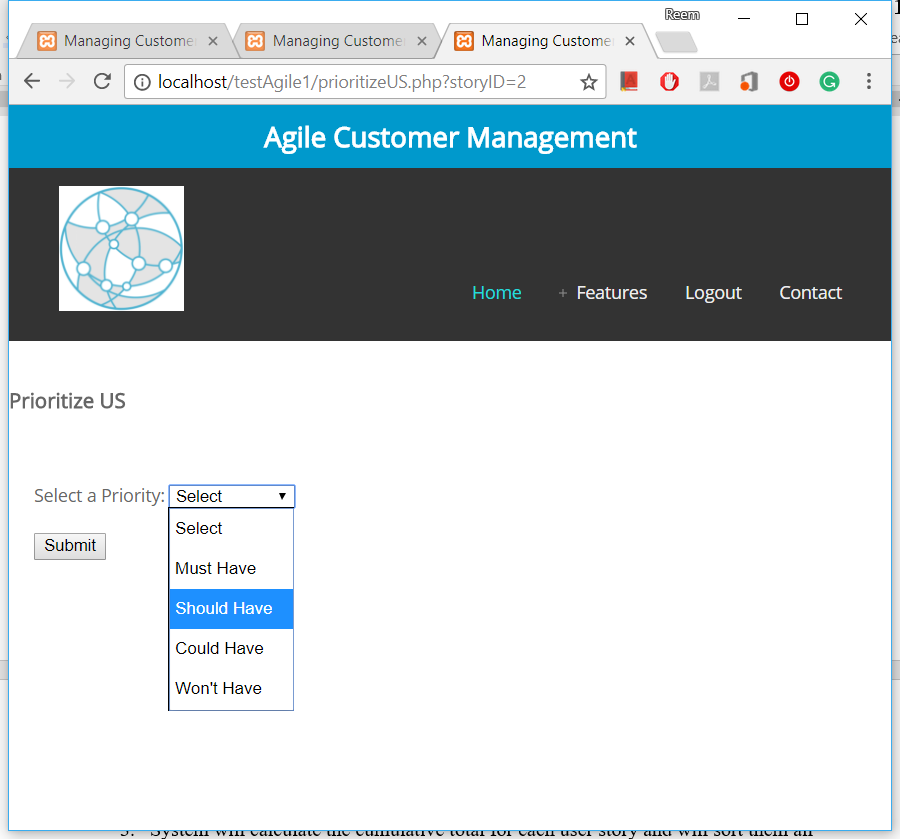


Figure 25. Sent Notification About US Prioritization Method

Only customers who are included in a specific release will be allowed to prioritize user stories of that release, even if they did not initiate it.

* **MoSCoW Method:** Customer will select one of the priority options available from the list (Figure 26). After this a notification will be sent to update affected team members about the added priority value (Figure 27).



**Prioritize User Story**

Figure 26. Add Prioritization Value by Customer – MoSCoW

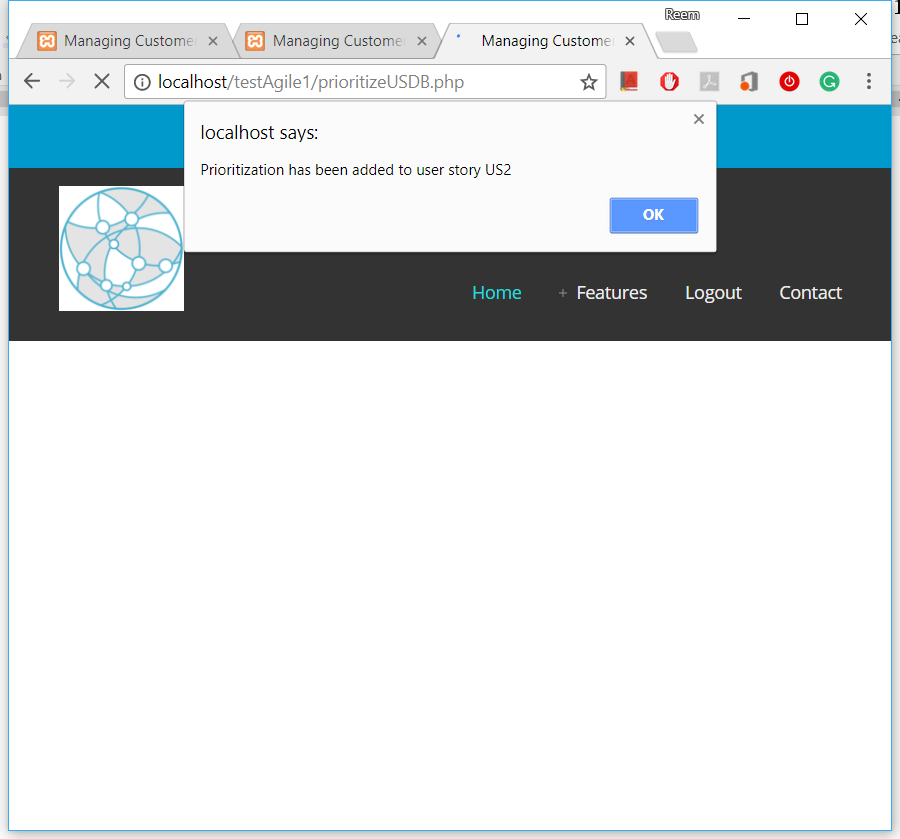
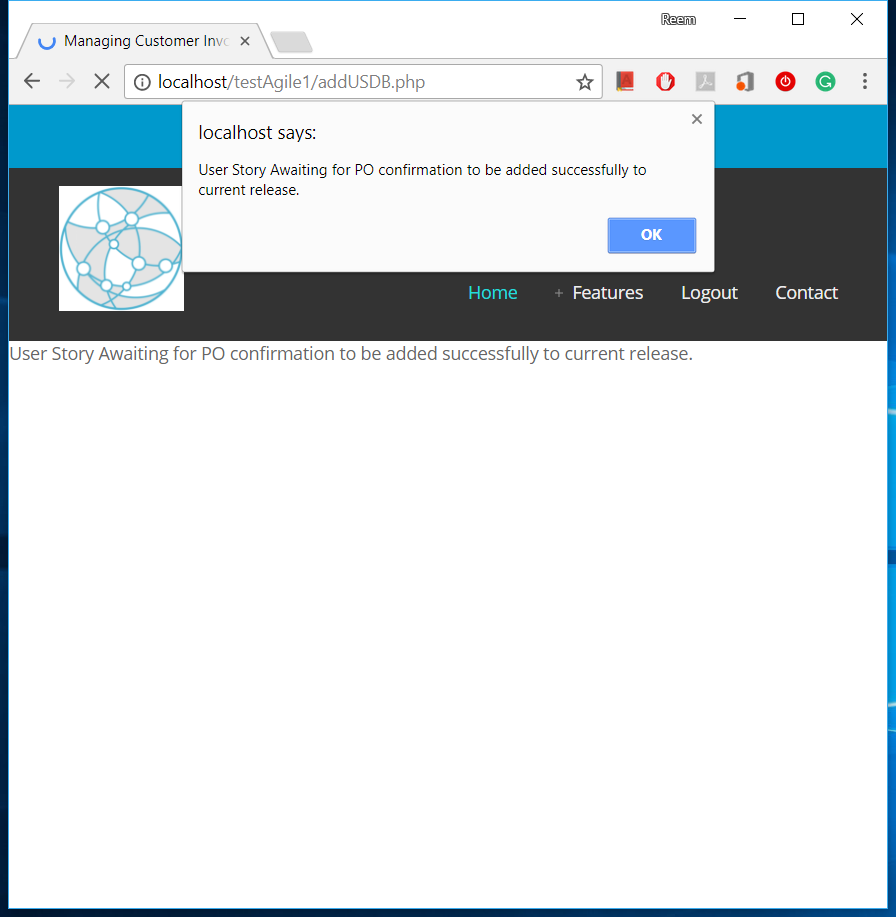
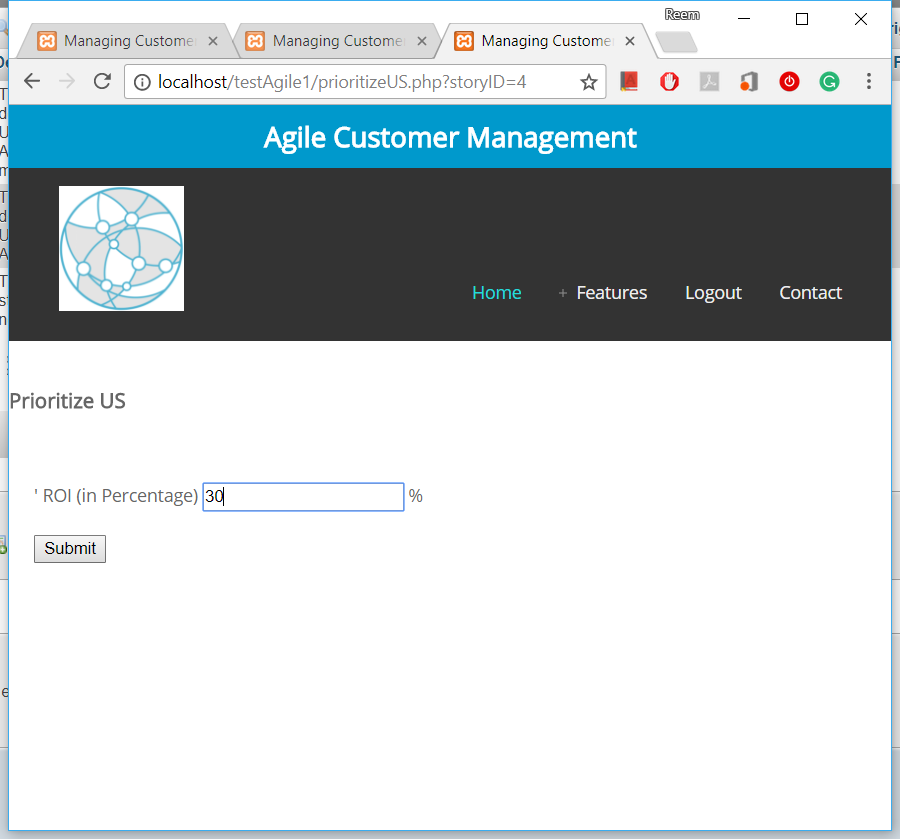


Figure 27. Sent Notification About US New Added Priority Value

* **ROI Method:** Customer will enter the percentage value of the ROI in the given box (Figure 28). After this a notification will be sent to update affected team members about the added priority value (Figure 29).



**Prioritize User Story**

Figure 28. Add Prioritization Value by Customer – ROI

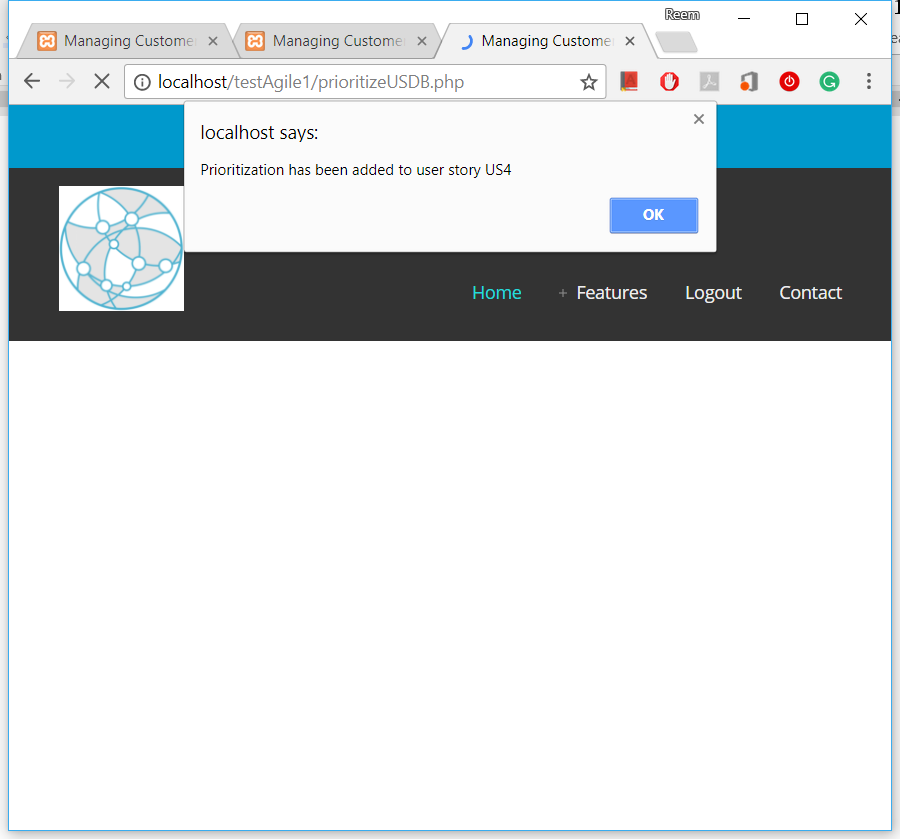
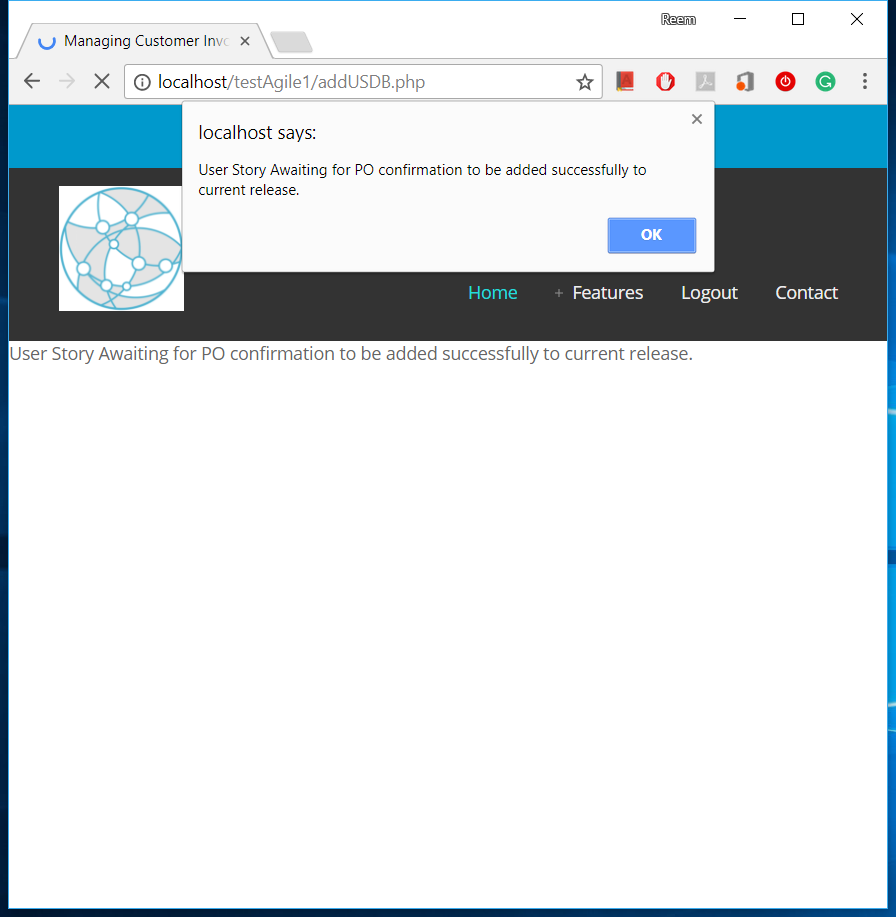


Figure 29. Sent Notification About US New Added Priority Value

The SQL statement that add a prioritization of a specific user story is given in the code below:

$query = "INSERT INTO `storypriority`

( `StoryID`, `customerID`, `PriorityMethod`, `Priority`)

VALUES ($storyID,$owner,'$priorityMethod',$priorityValue) ;";

1. System will calculate the cumulative total for each user story and will sort them all and notify the PO to take an action (Figure 30 and Figure 31 show respectively the priority value before and after current customer update).

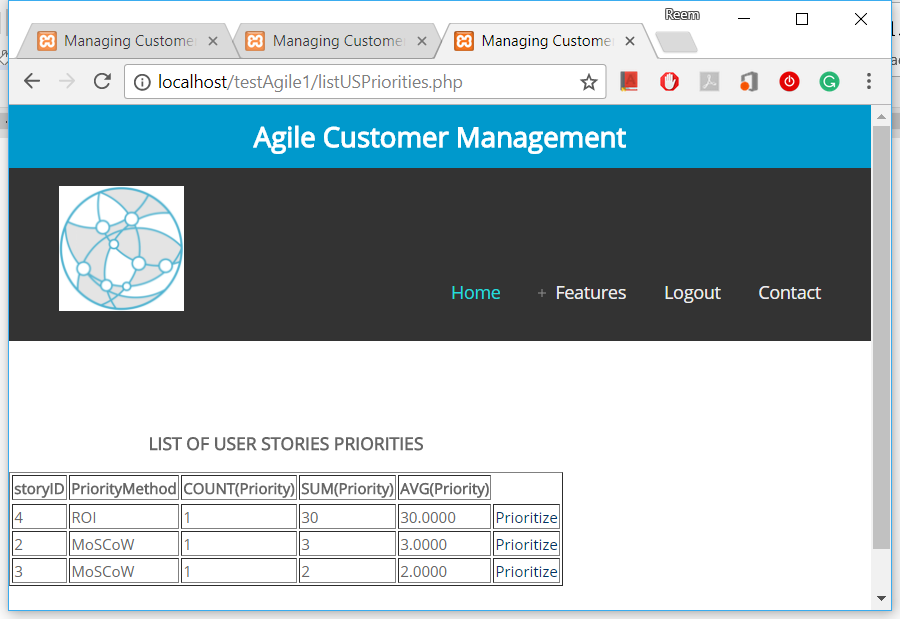
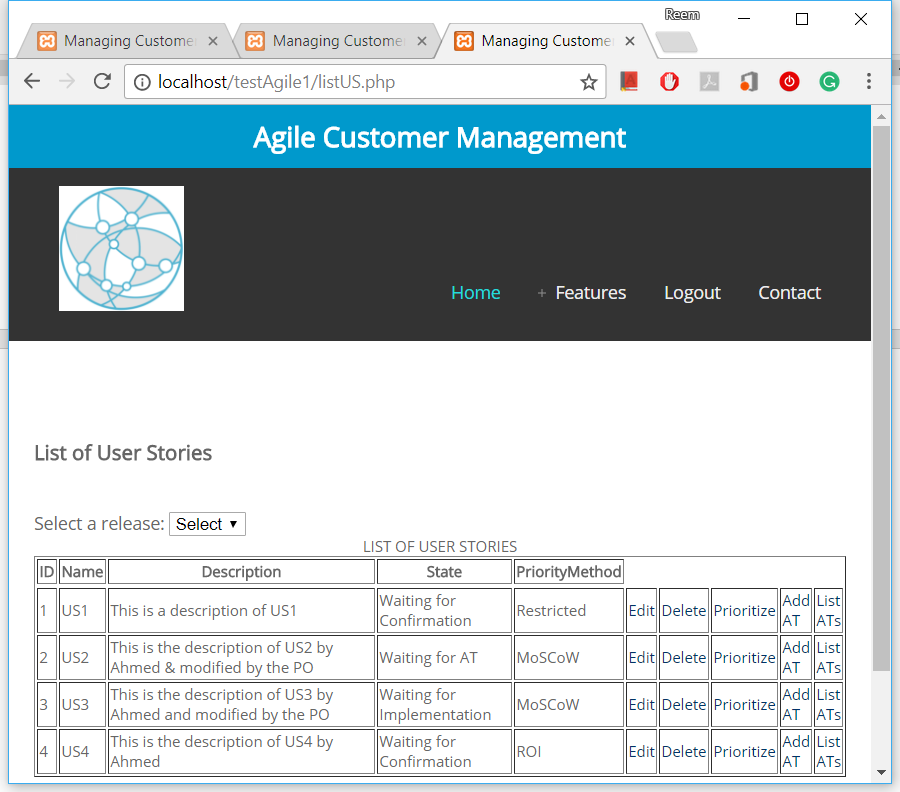
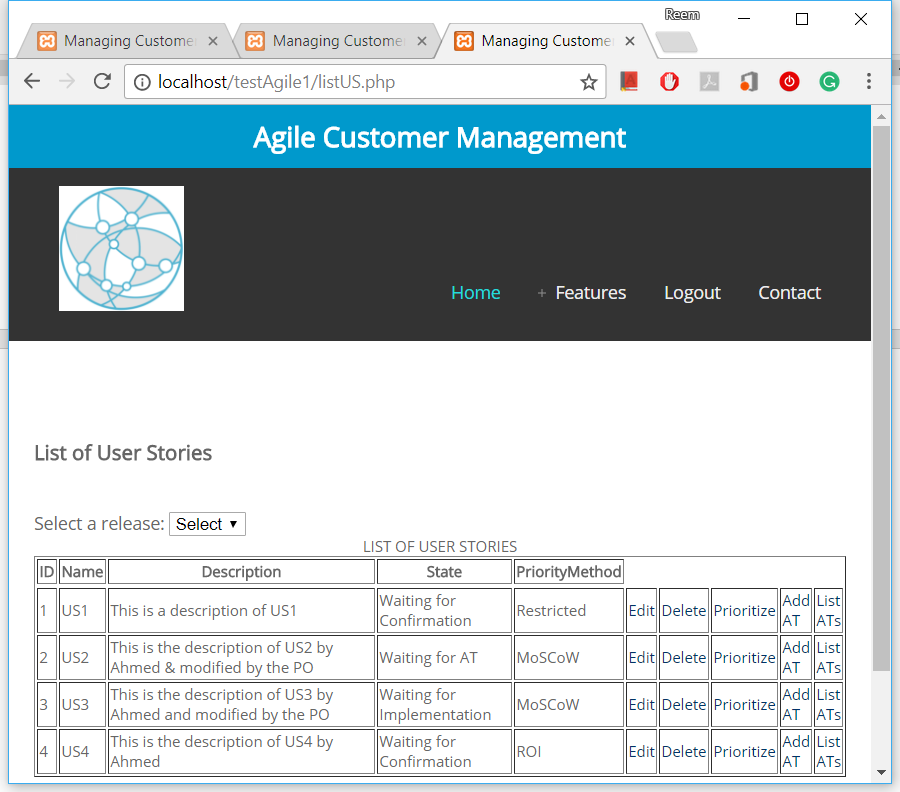
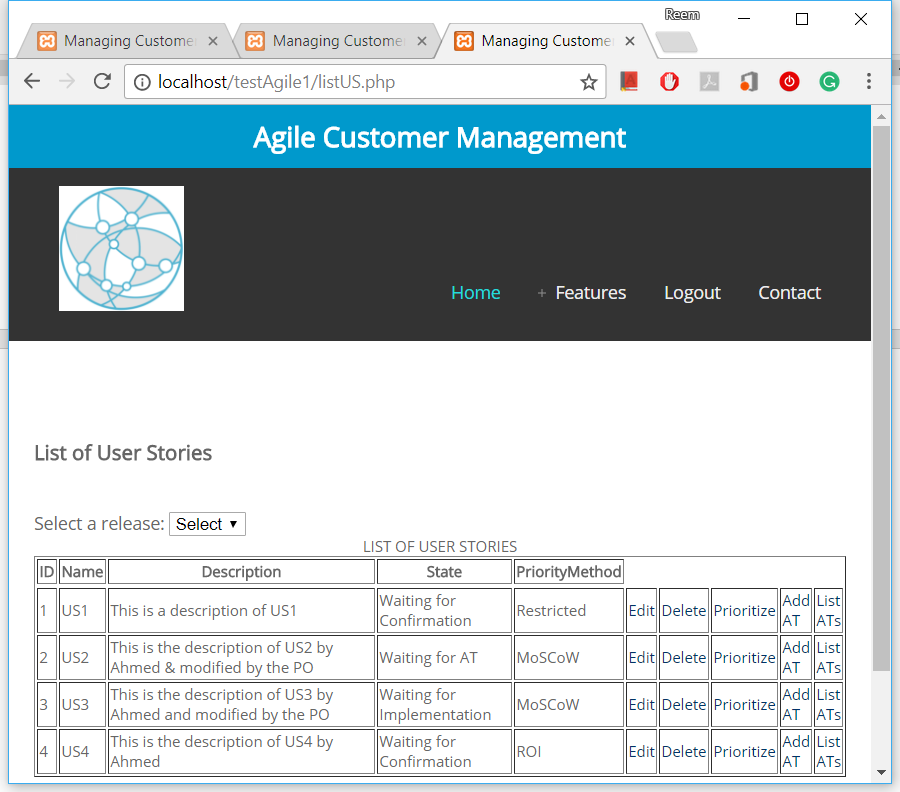
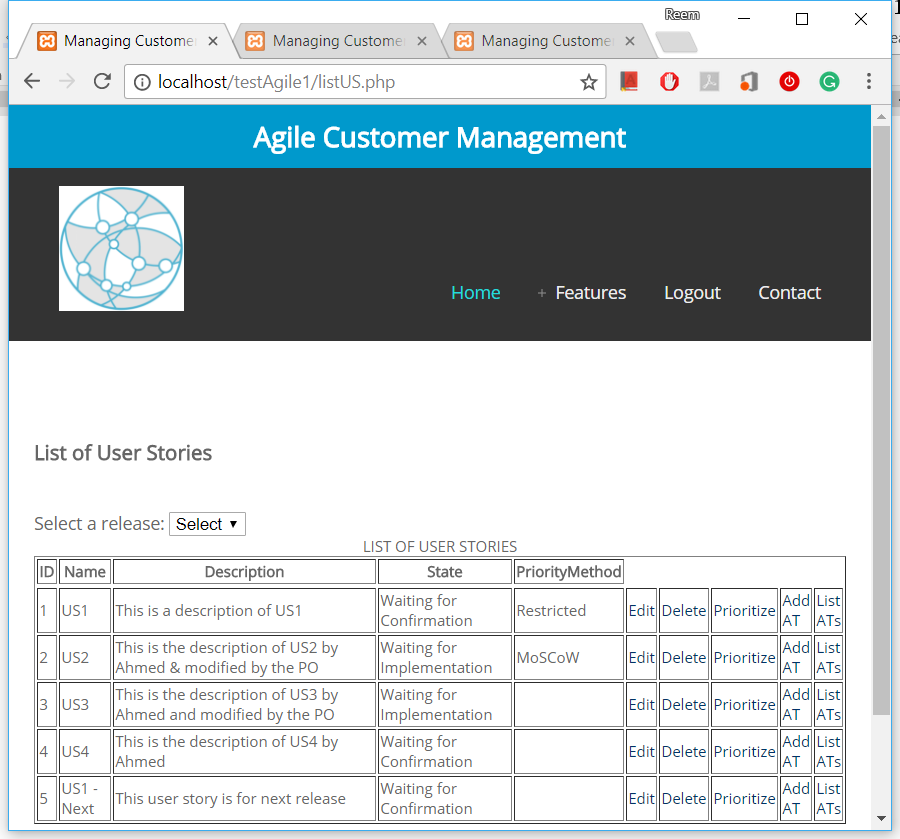


Figure 30. List of All Release USs Ordered by Their Priority – Before

After C2:Sara prioritize:

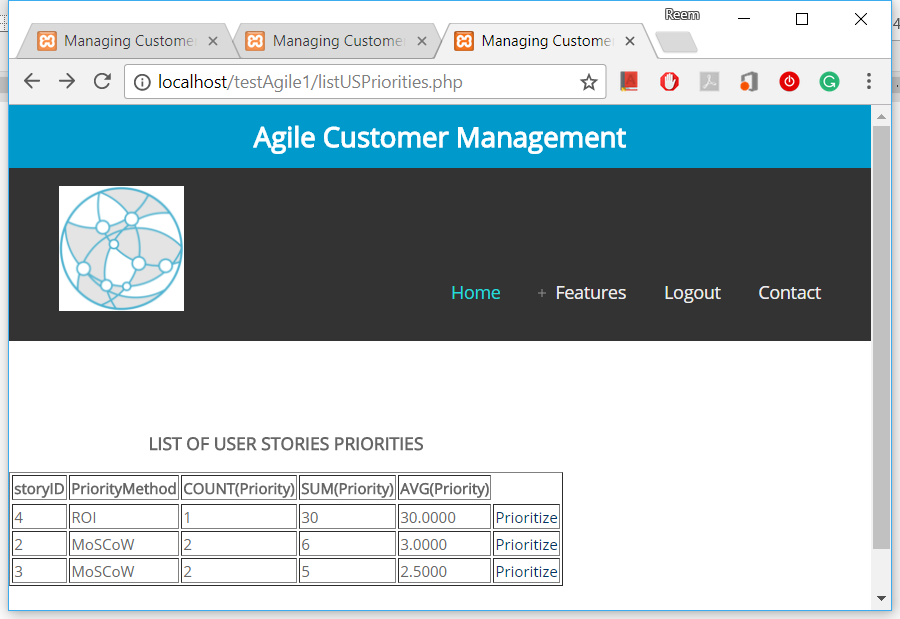
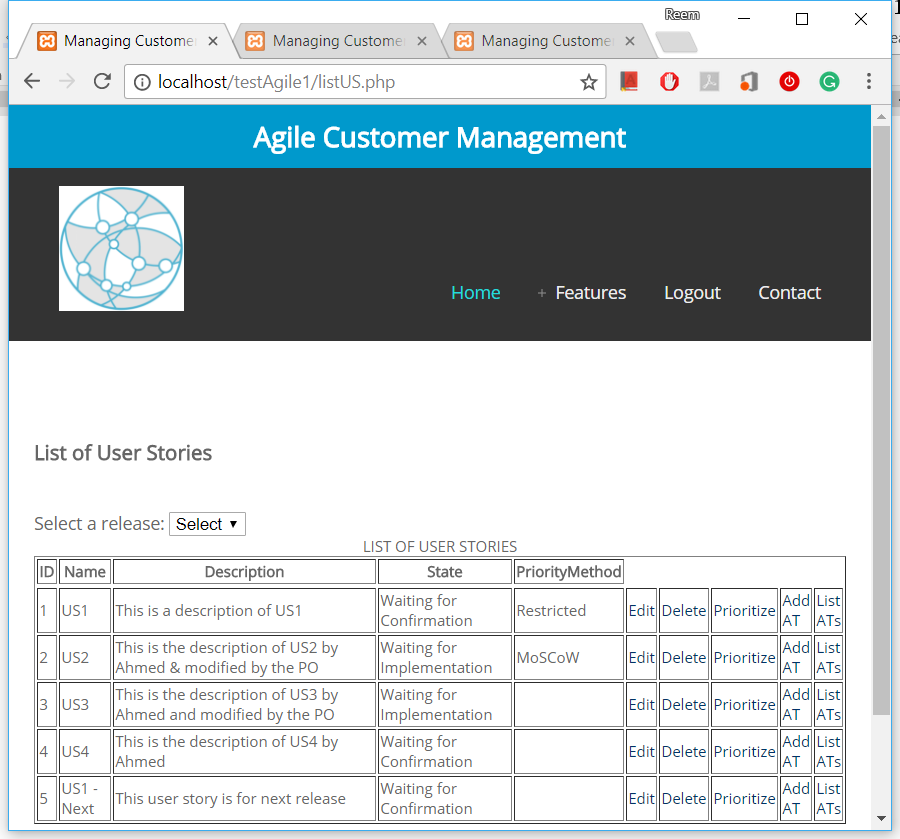
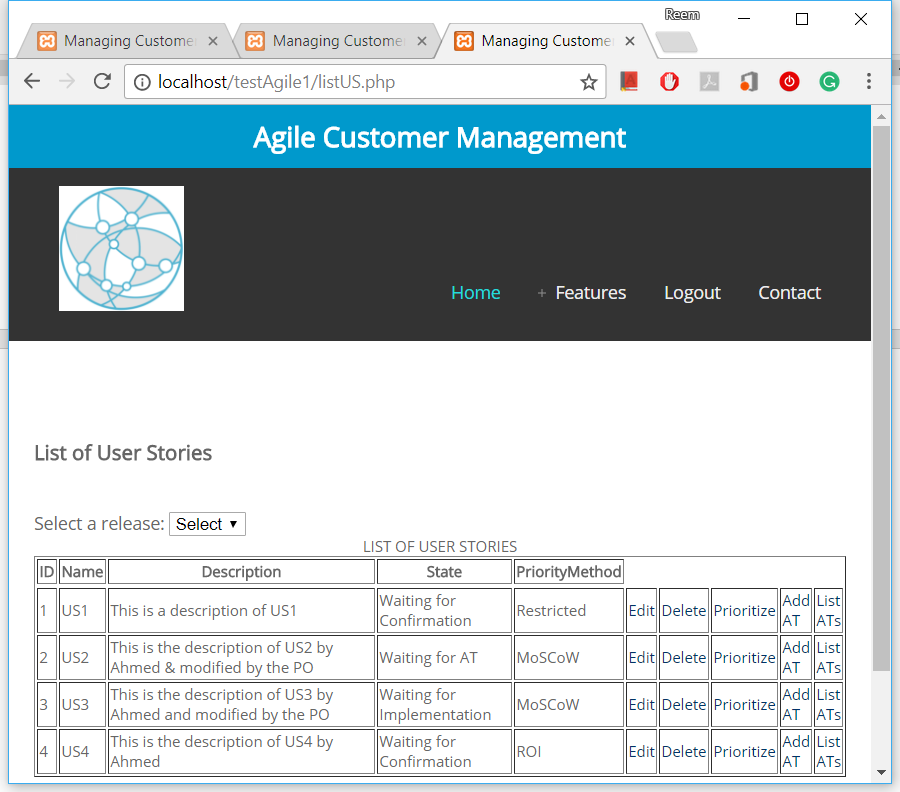
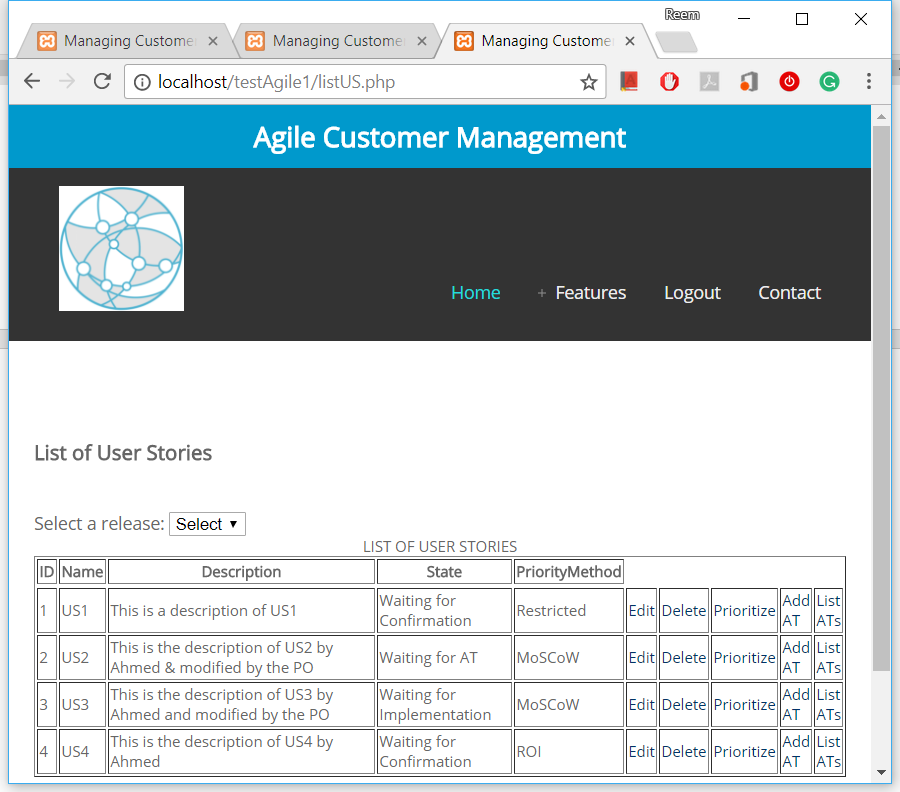
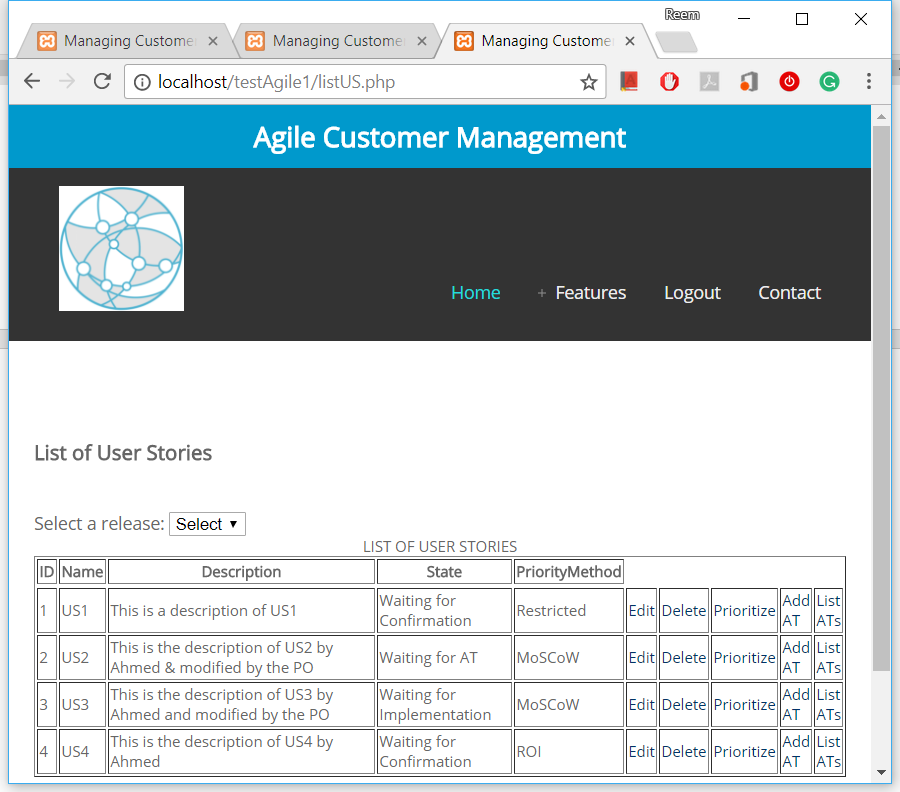


Figure 31. List of All Release USs Ordered by Their Priority – After

The SQL statement that calculate how many users add priority and total and average priority of all user stories and sort them:

$query = "SELECT storyID,PriorityMethod,

COUNT(Priority), SUM(Priority), AVG(Priority)

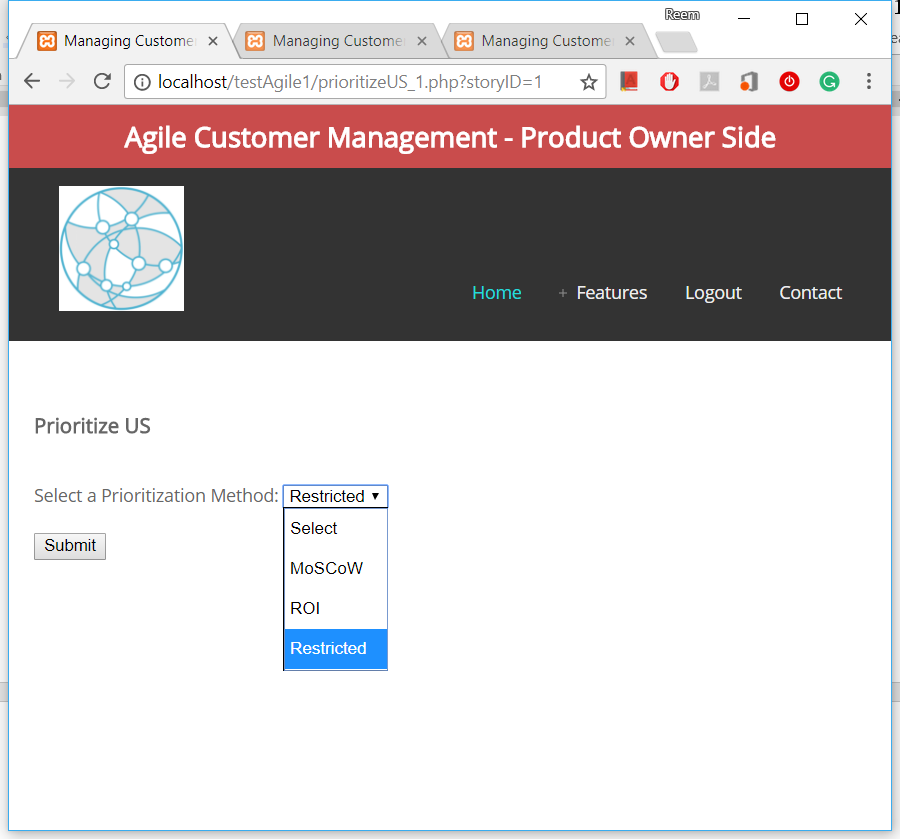
FROM storyPriority

GROUP BY storyID

ORDER BY SUM(Priority) DESC";

**Further Possible Validation of Constraints:**

* The system will not allow customer to prioritize a technically restricted user story (as shown in Figure 32, Figure 33, and Figure 34 respectively):



**Prioritize User Story**

Figure 32. Special Case for US That is not Allowe to be Prioritized.

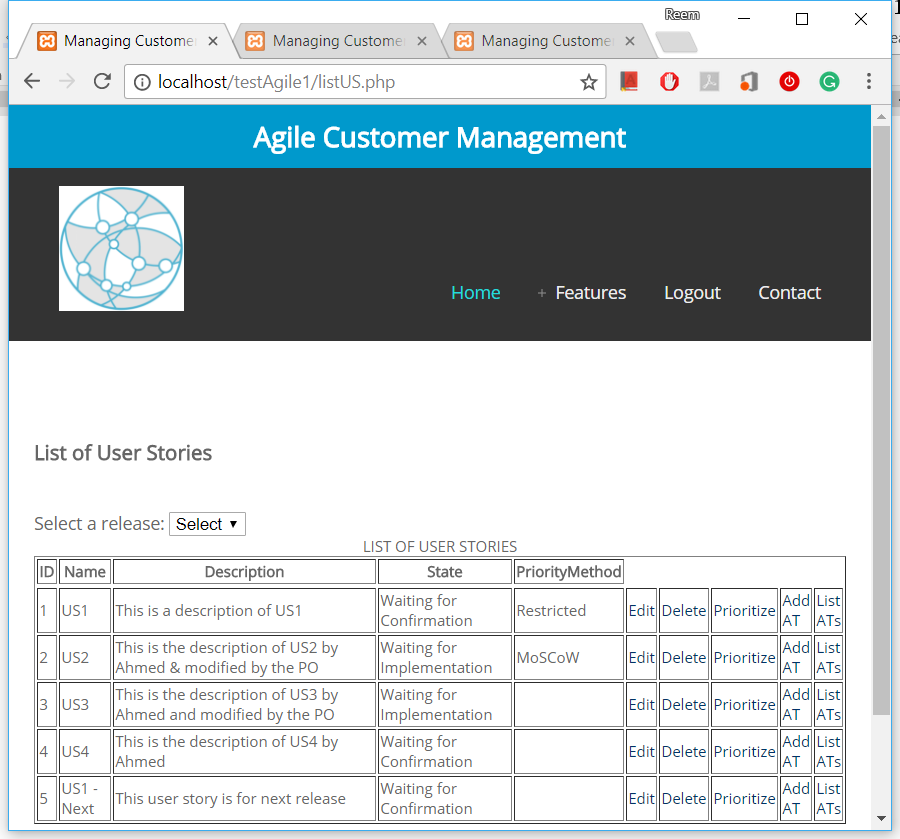
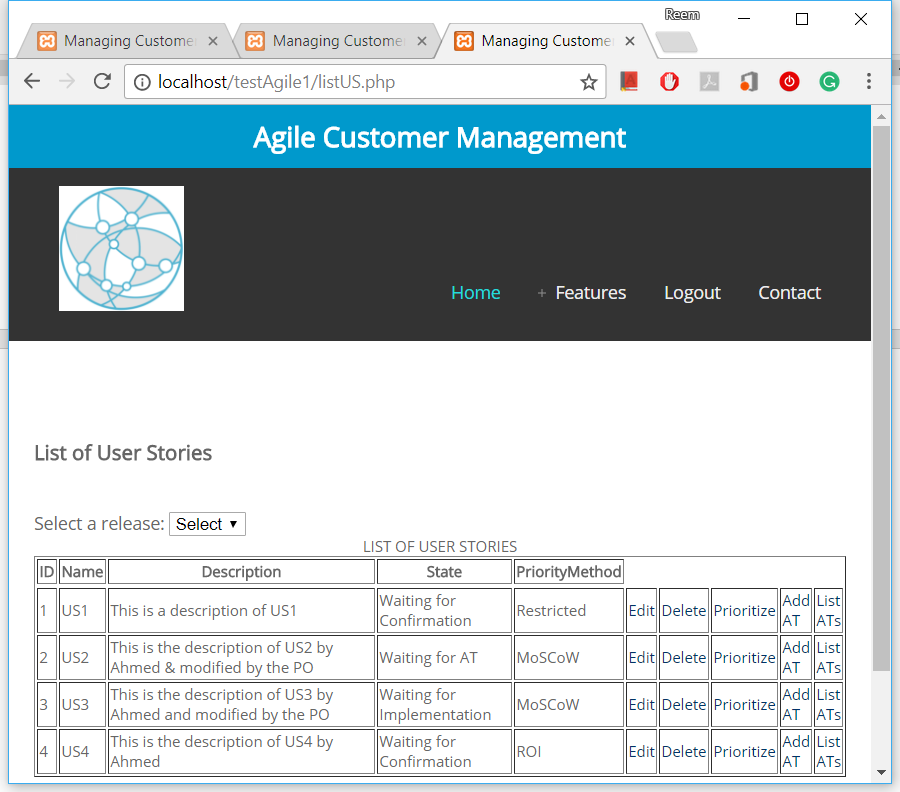


Figure 33. List of USs Showing Restricted Case

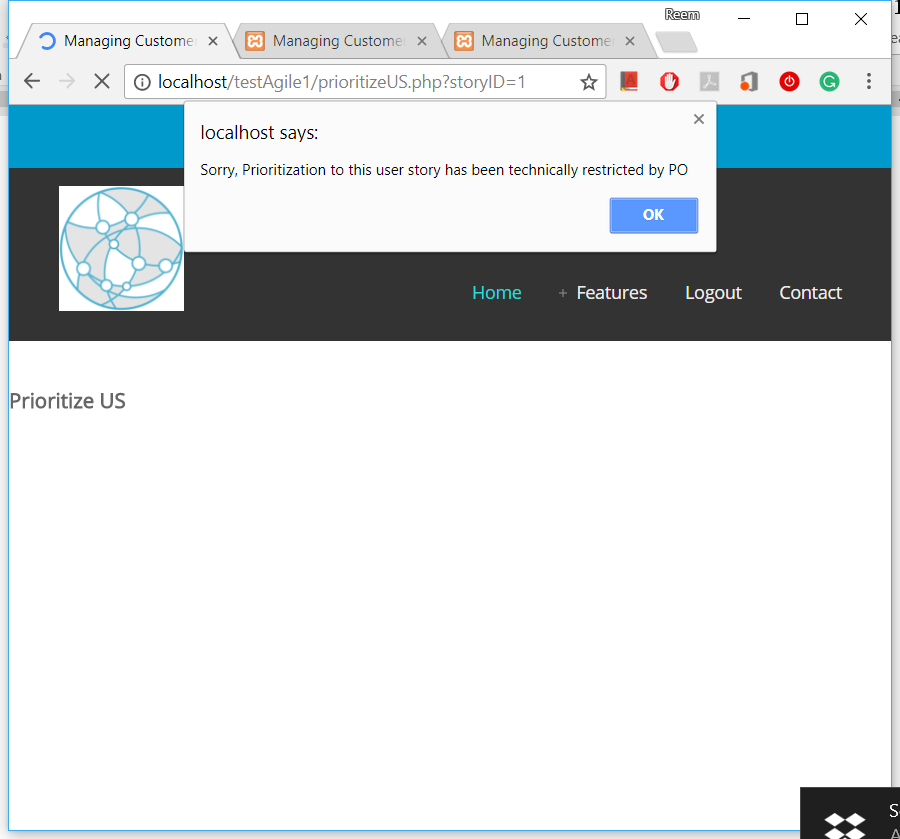
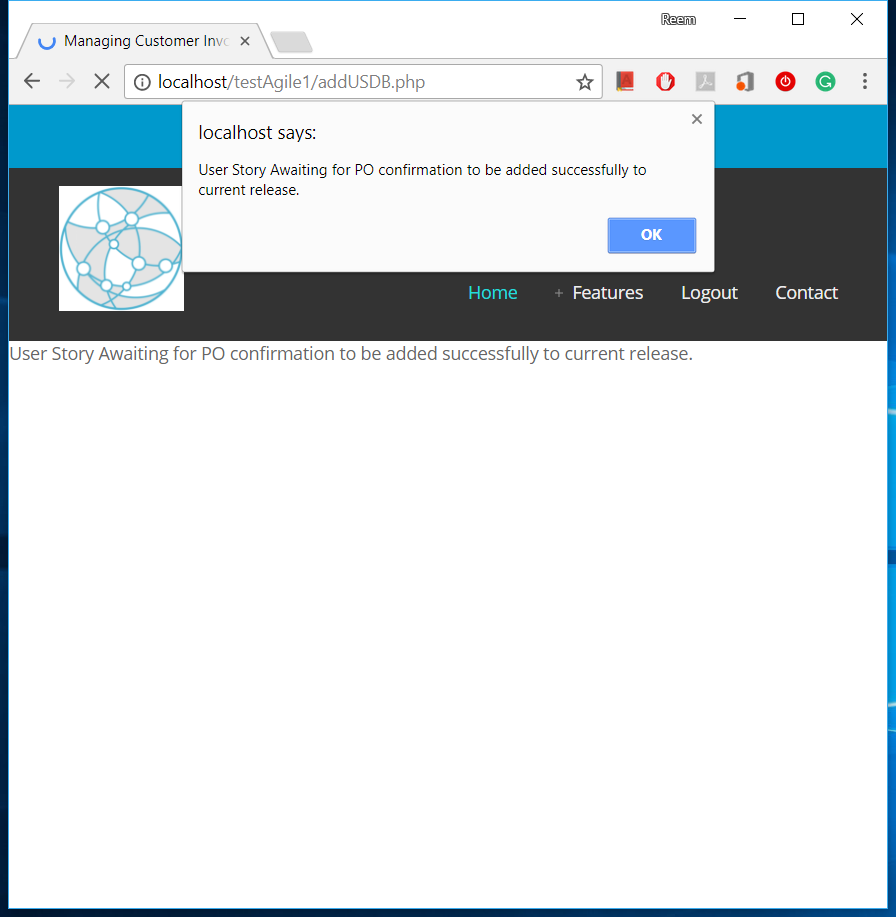


Figure 34. Restricted US Validation Result and Notification Recieved

The system will prevent prioritize restricted user story as shown in the code below:

if ($priorityMethod == 'Restricted') {

$msg = "Sorry, Prioritization to this user story has been technically restricted by PO";

echo "<script type='text/javascript'>alert('$msg');</script>";

}

### 2.4 Implementation of Scenario 3

After the user story developed and become ready for acceptance test (Figure 35), customer will be able to perform the acceptance test run for either automatic or manual ATs (Figure 36). Below is the implementation of the proposed approach steps of scenario 3: Change acceptance test status, in which customer will perform the manual acceptance test run and change their status either to Pass or Fail:

1. C1:Ahmed will request to change the status of acceptance test AT2.1 to ‘Pass’ and AT2.2 to ‘Fail’ (Figure 37).

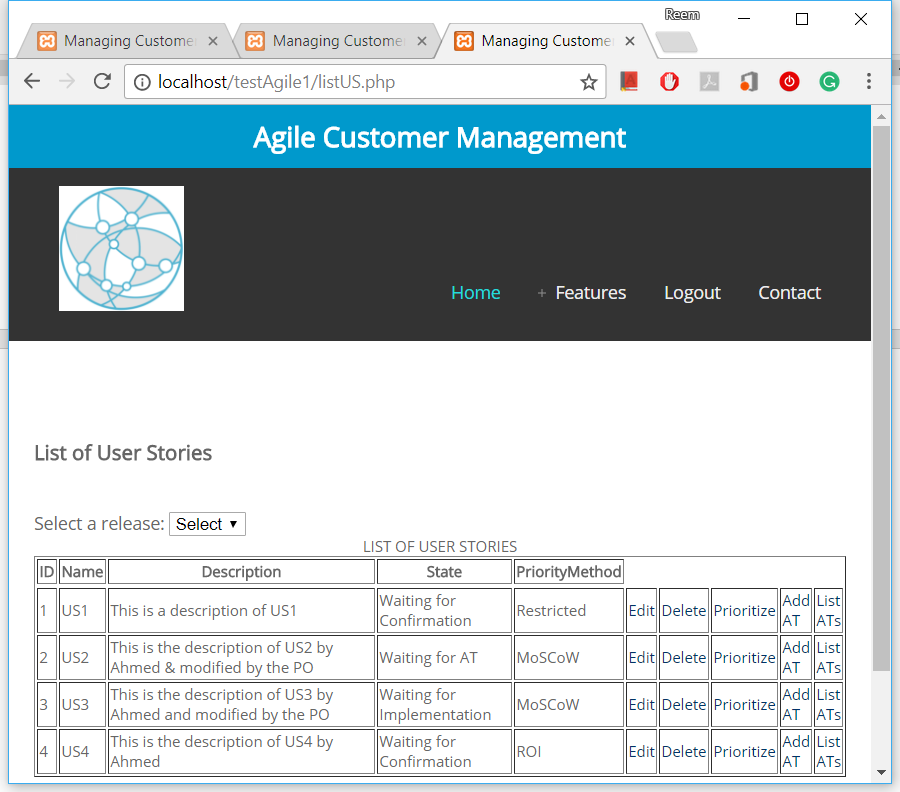
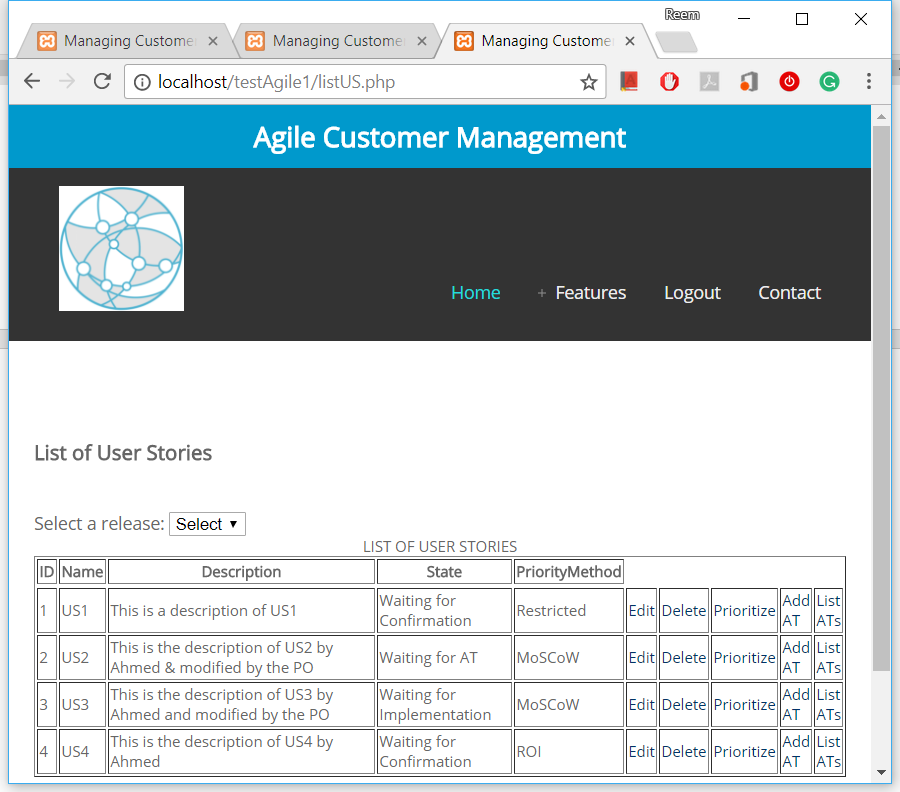
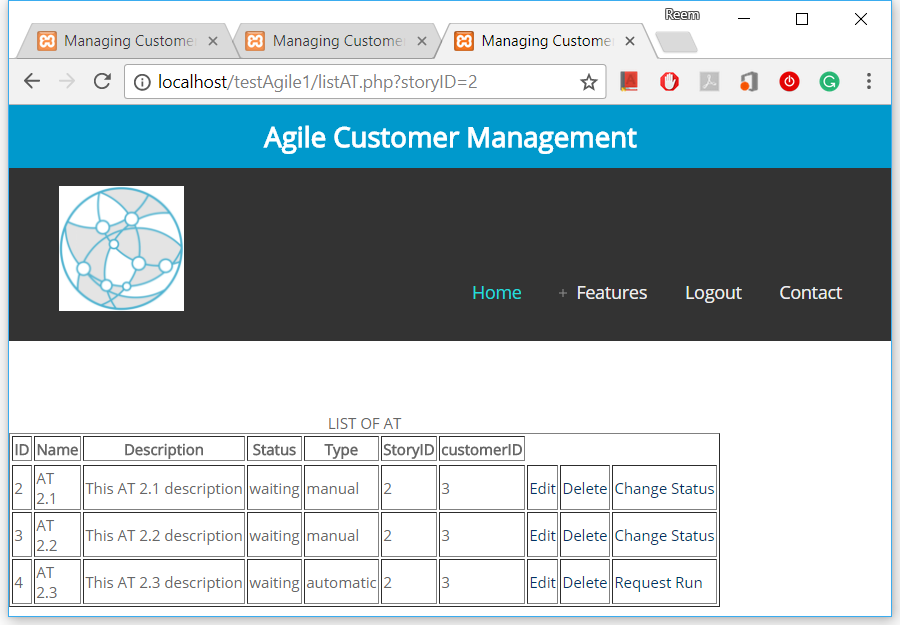


Figure 35. List of USs Showing ‘Waiting for AT’ State



LIST OF ACCEPTANCE TESTS

Figure 36. List of ATs Showing Pre-Status

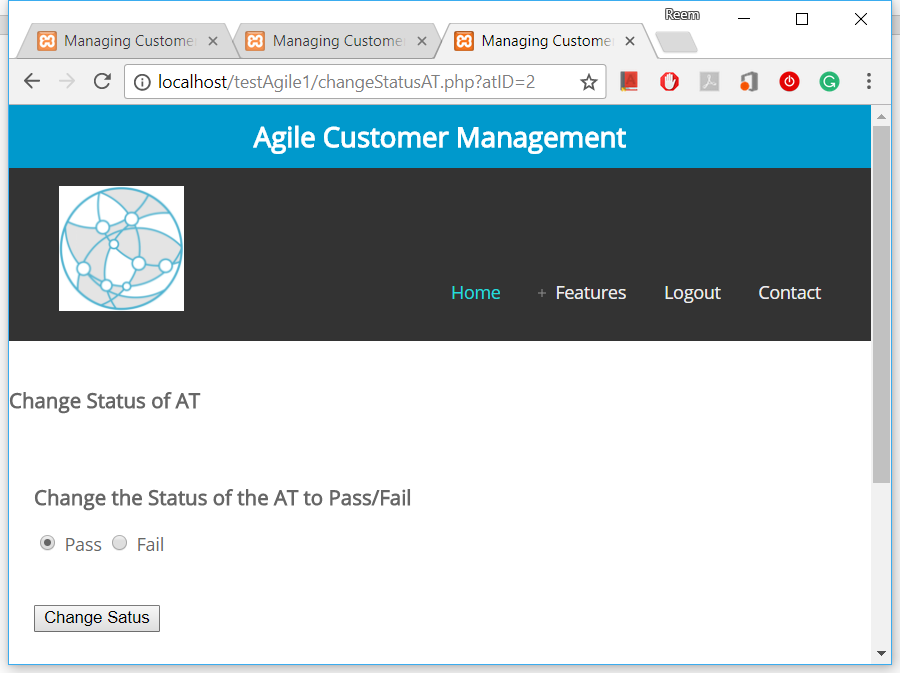


Figure 37. Changing AT Status

1. System will validate entry in terms of authentication, release, US state, and AT status.
2. Validation passed and the system will change and store the status of AT2.1 and AT2.2 and notify PO (Figure 38 and Figure 39). List of the ATs with the updated status shown in Figure 40.

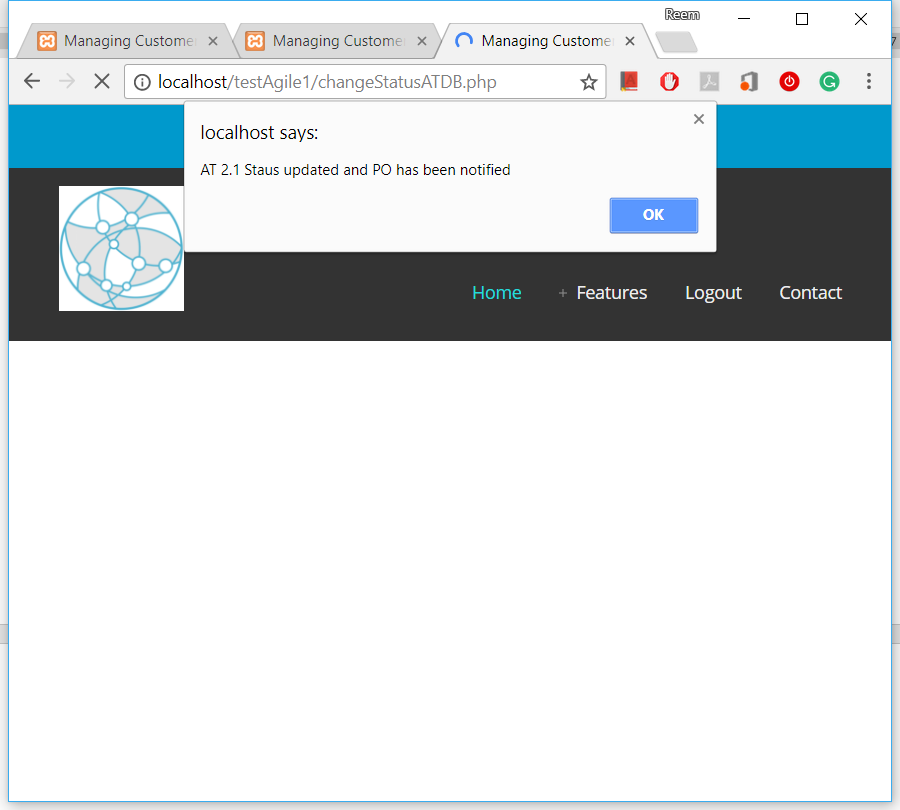
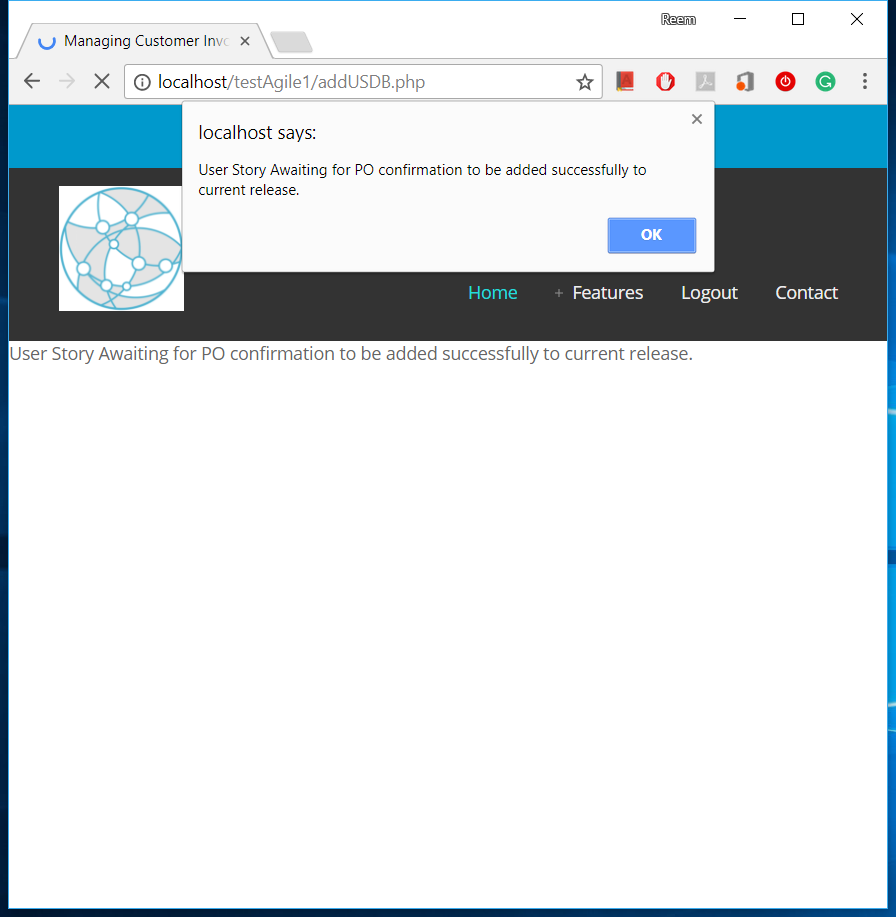


Figure 38. Notification Sent to PO - 1st AT.

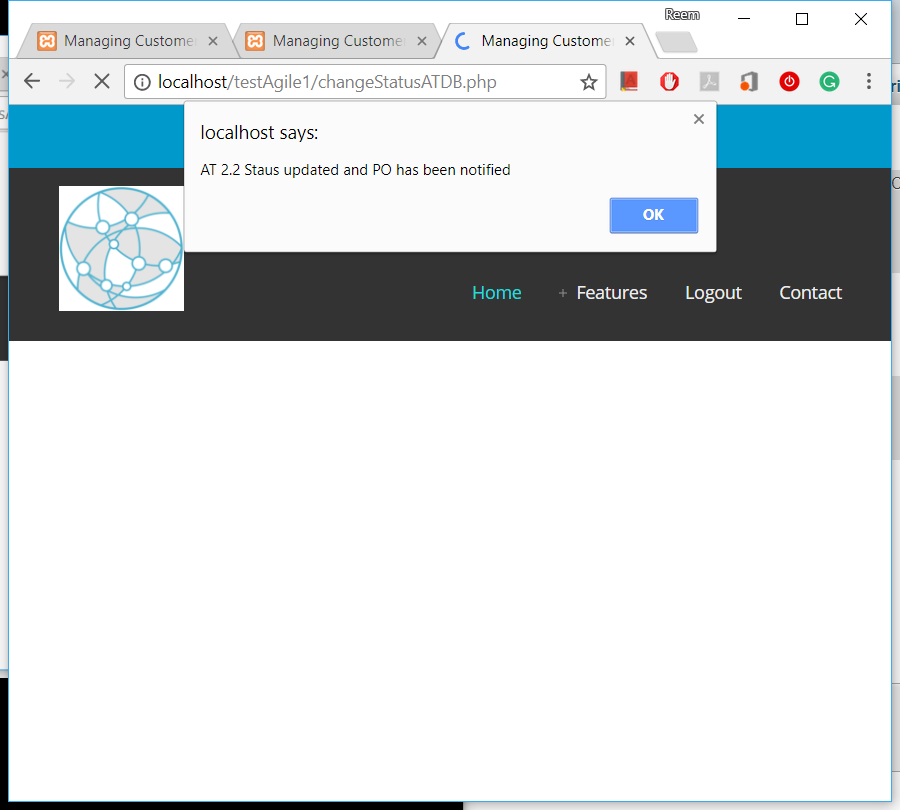
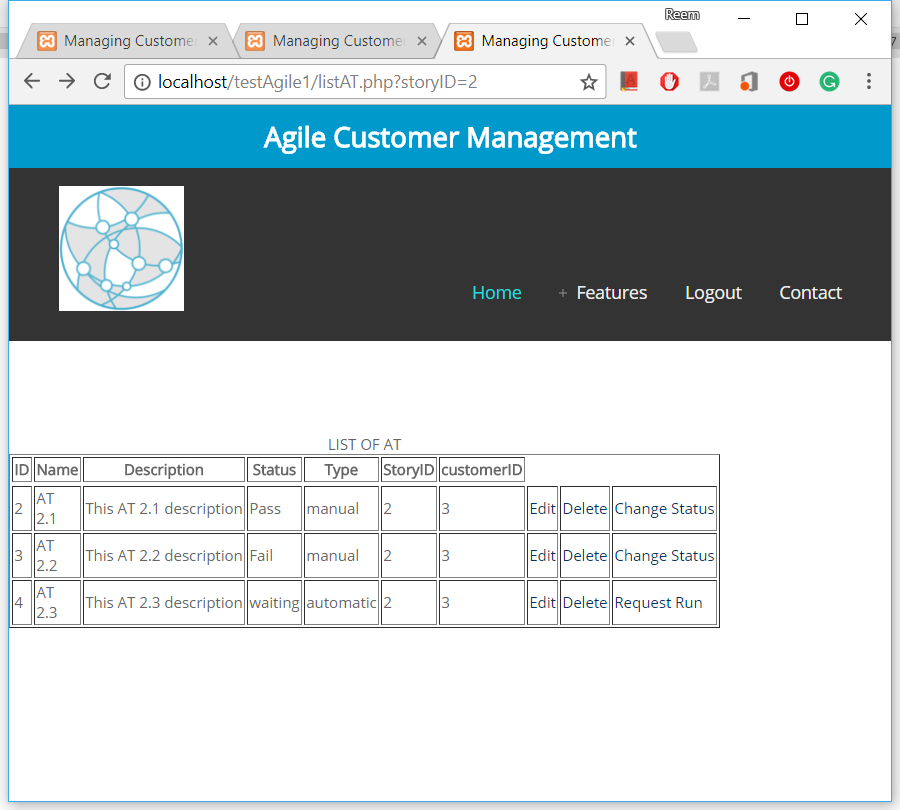


Figure 39. Notification Sent to PO - 2nd AT



LIST OF ACCEPTANCE TESTS

Figure 40. List of ATs - After Status Change

The SQL statement that update AT status is shown in the below code:

$query = "UPDATE `AT` SET `Status` = '$status'

WHERE `ID` = '$atID';";

1. C1:Ahmed, T1:Jack, and PO:Marco are notified about the change (Figure 41 and Figure 42).

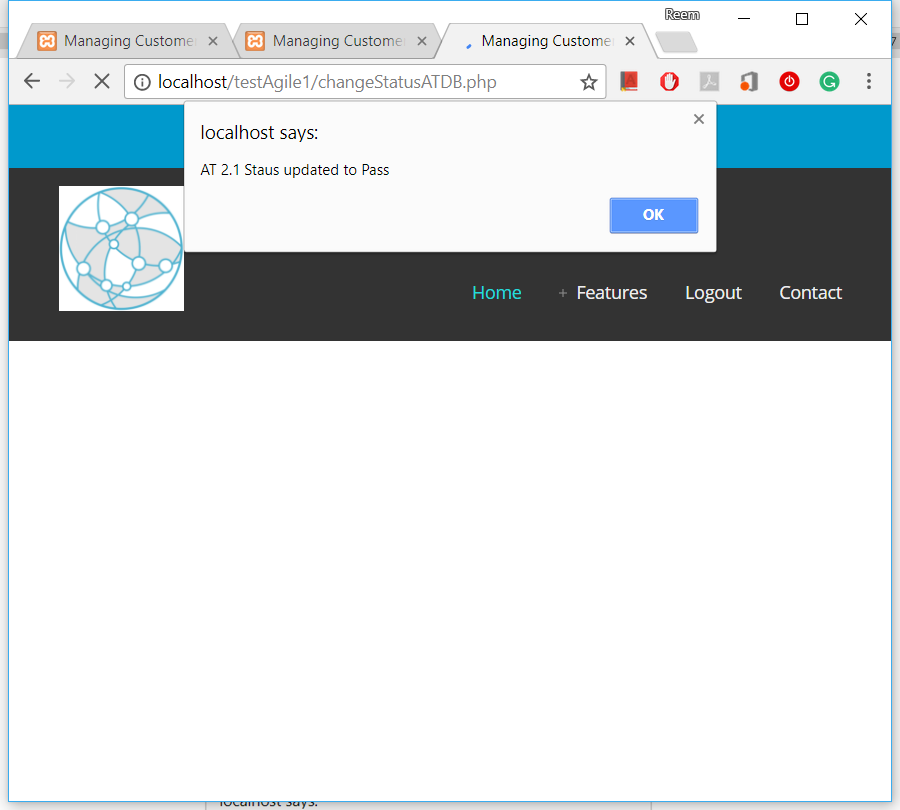
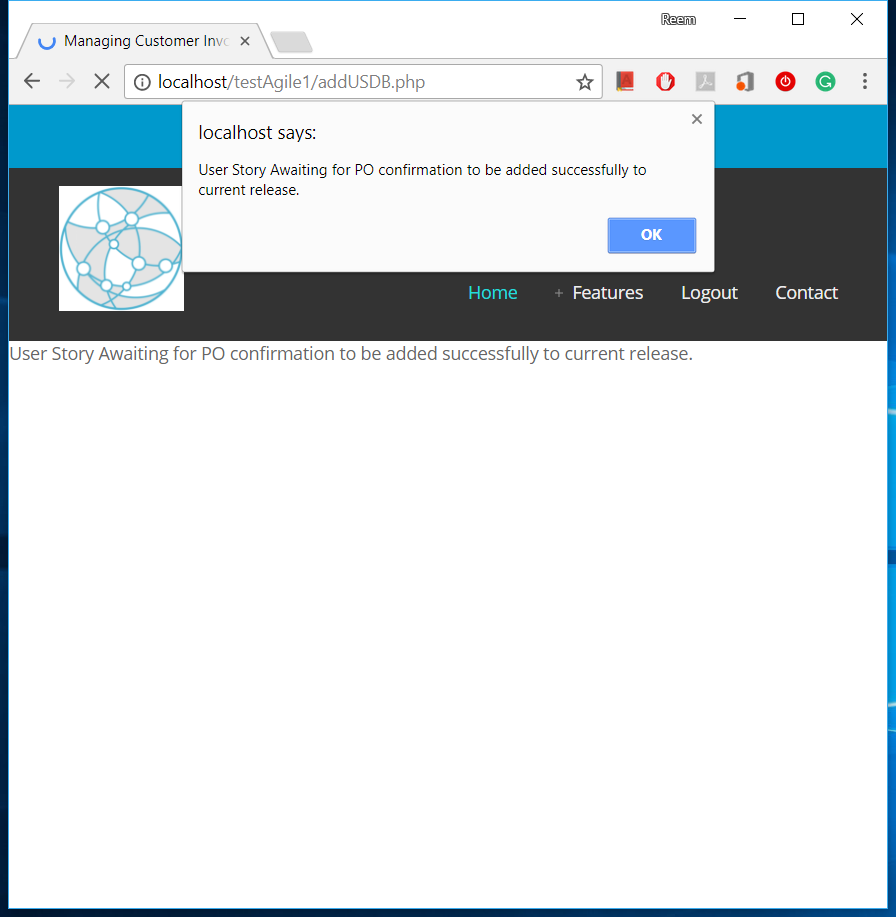


Figure 41. Notification of the New AT Status - 1st AT

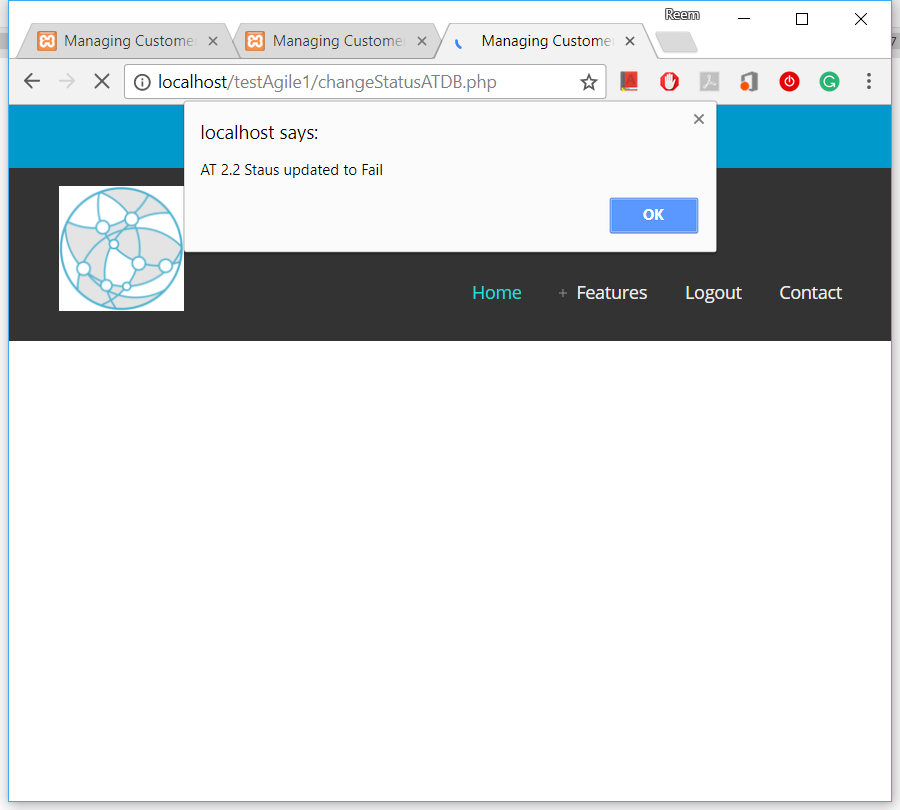
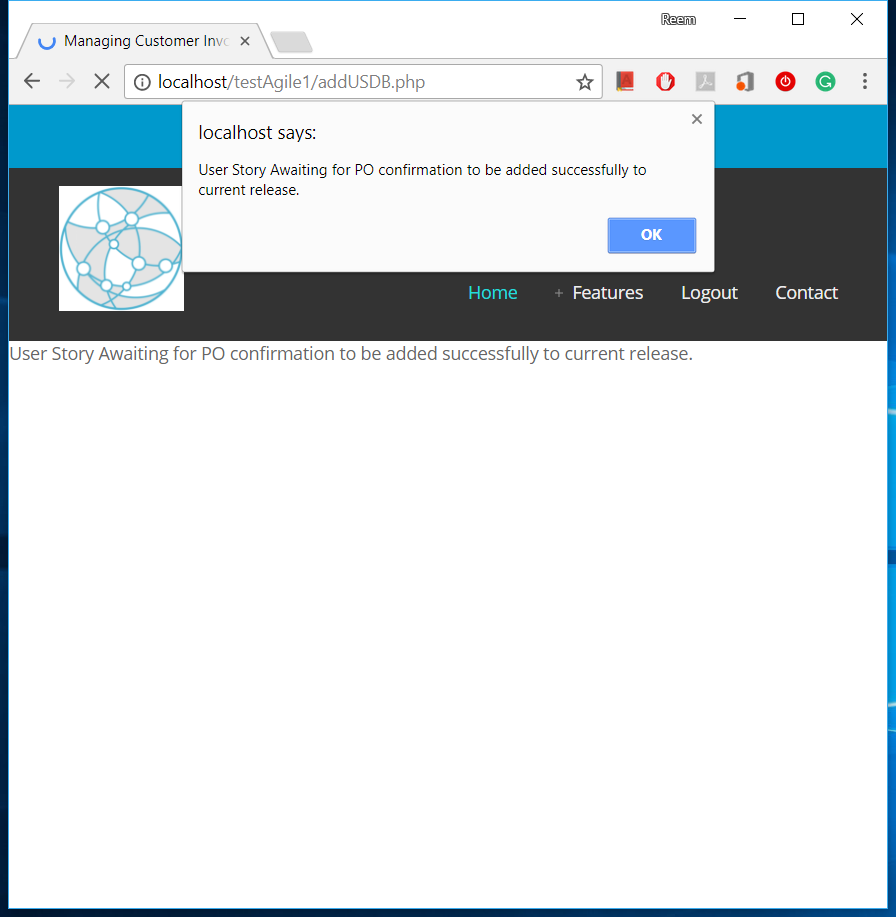


Figure 42. Notification of the New AT Status - 2nd AT

**Further Possible Scenario Validation of Constraints:**

* If the customer change AT of a Complete user story, its state will return back to ‘Waiting for AT’ as shown in the below code:

//Validate US state

if ($us["State"] == "Complete") {

changeStateUS($storyID, 'Waiting for AT');

$msg = "User Story state becomes 'Waiting for AT' now <br>";

echo "<script type='text/javascript'>alert('$msg');</script>";

}

* If the customer tries to run acceptance test of an active user story which is not ready for AT yet (Figure 43), the system will not allow the AT run as shown in the below code and a notification will be given (Figure 44):

else if ($us["State"] != "Waiting for AT") { // all other cases

$msg = "US is not ready for AT yet, check later and you will be notified when it is ready.";

echo "<script type='text/javascript'>alert('$msg');</script>";

}

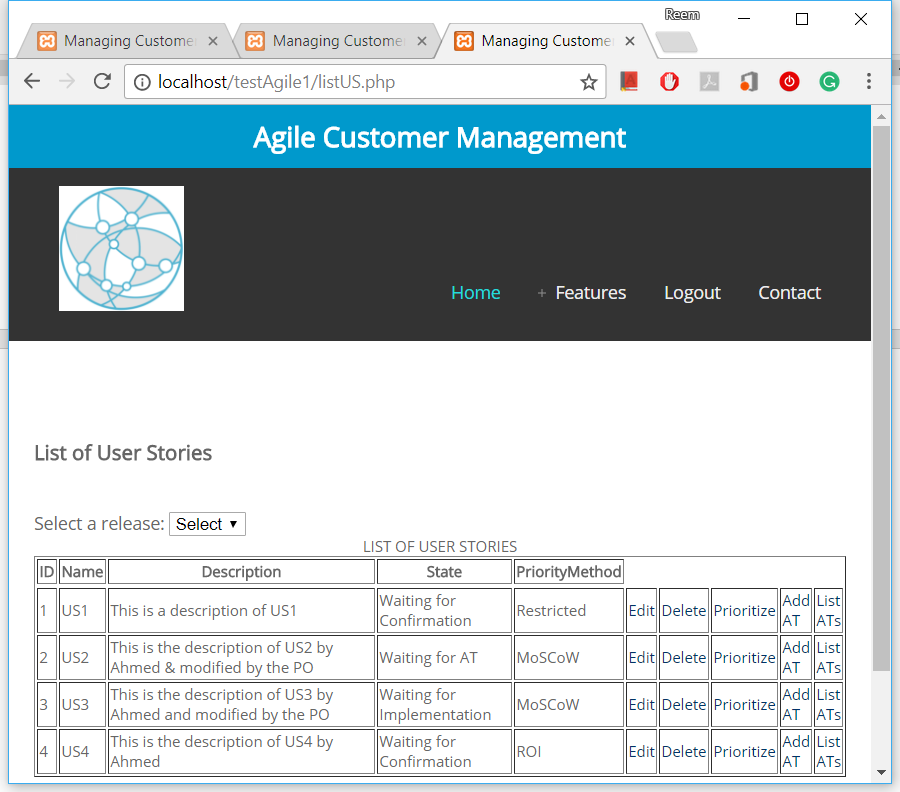
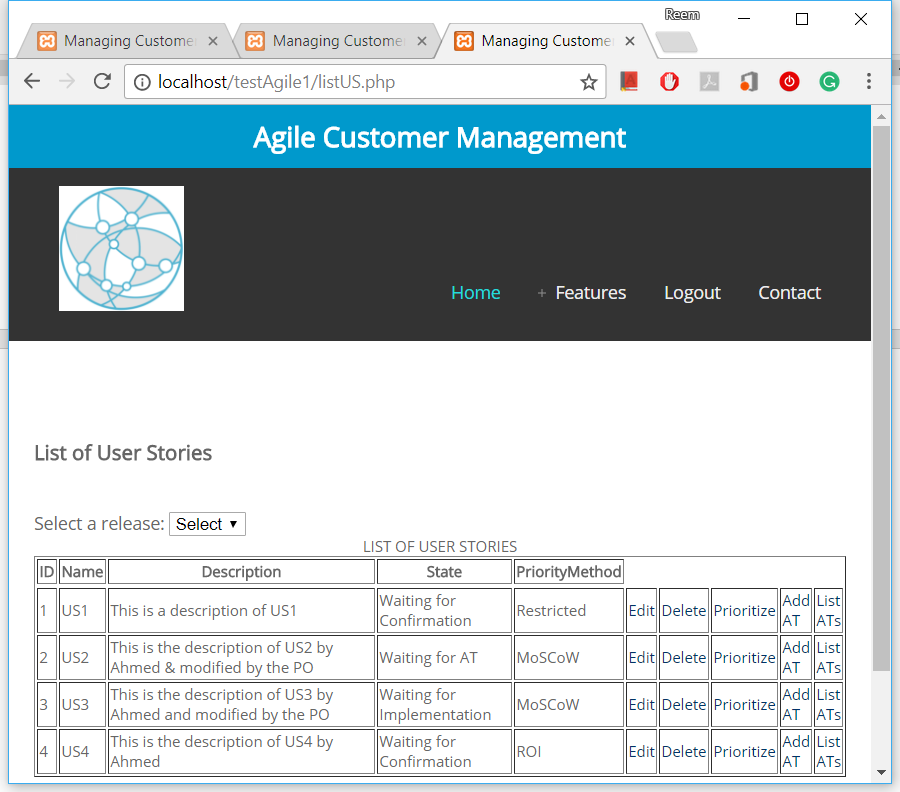


Figure 43. List of USs Showing the ‘Waiting for Implementation’ State

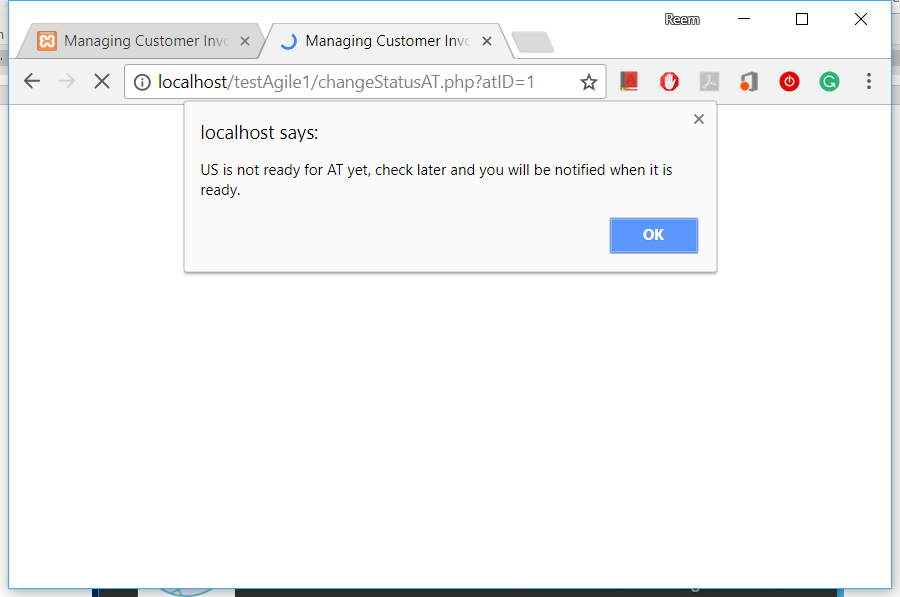
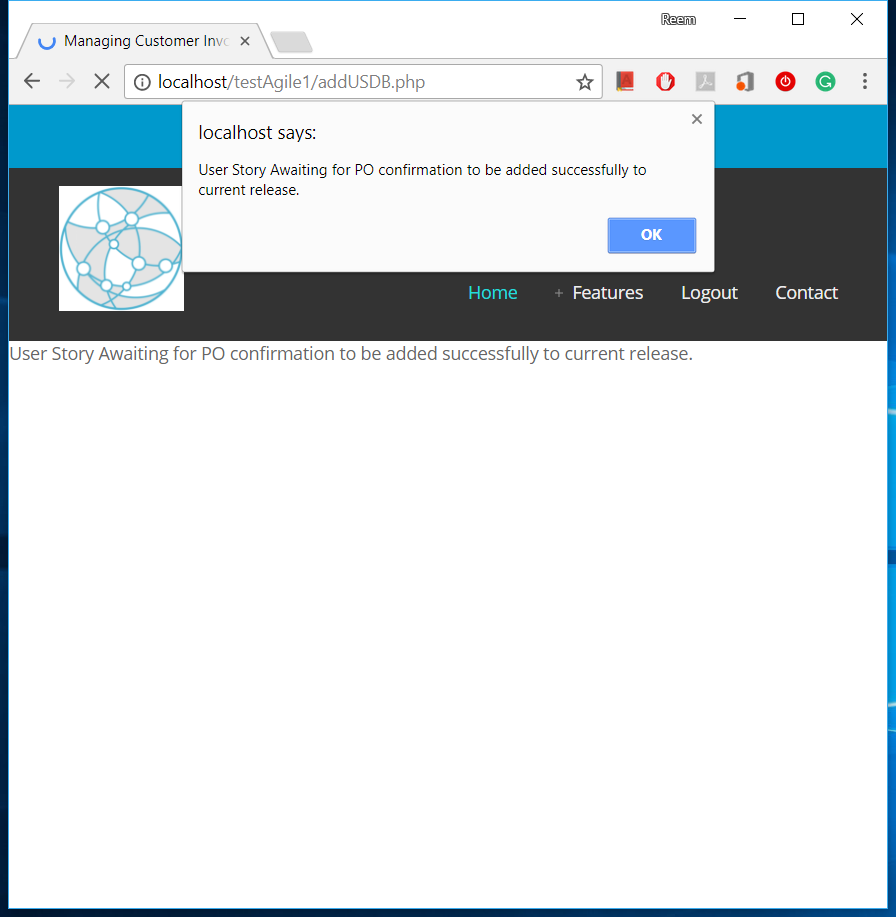


Figure 44. US Validation and Notification When Try to Make ATs for Incomplete US

**Further Possible Delay Detection:**

* If the customer change AT status which is already ‘Pass’ to ‘Fail’, this action may cause a delay and he will be notified about this (Figure 45):

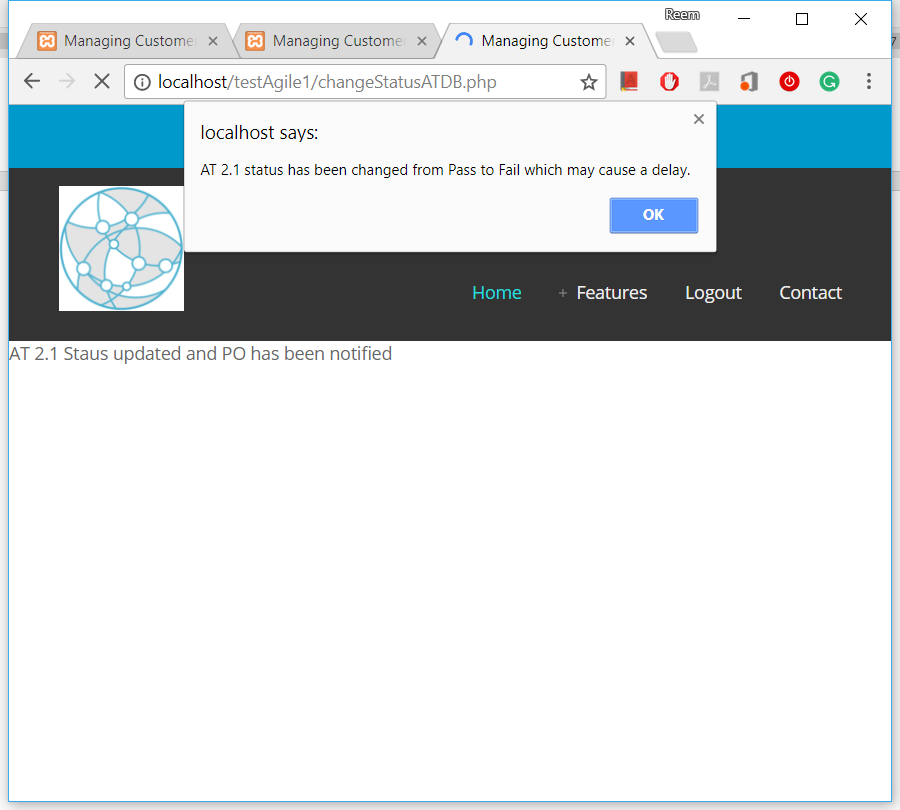
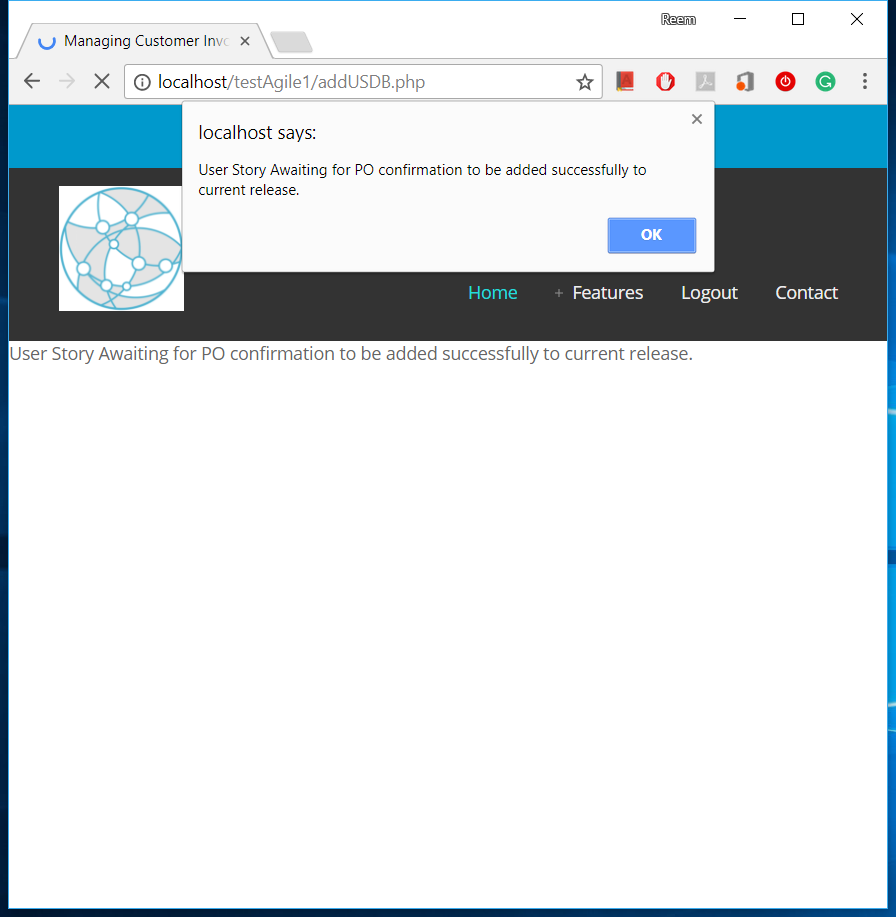


Figure 45. Notification about a Possible Delay Detection

* The SQL statement that check the possible consequences of changing the status of a ‘Pass’ acceptance test again to ‘Fail’ is given in the code below:

if ($previousATStatus == "Pass" && $status == "Fail") {

$msg = "$atName status has been changed from Pass to Fail which may cause a delay.";

echo "<script type='text/javascript'>alert('$msg');</script>";

notifyPM($releaseID, "AT Fail after Pass", $msg);

}