**Defect Prevention**

**System;**

This web site is designed to keep track of anime that someone might be watching, has finished or plans to watch in the future, allowing it user to very easily manage all their anime. The system will allow the user to register, add anime to different lists such as watching, finished, plan to watch and dropped, search anime. While the admin manages all the anime available.

**Requirements;**

R1 - User must log in to view other users ID.

R2 - Confirmation box appears when user adds an anime to a list.

R3 - Search filter will display results within 2 seconds of searching.

R4 - Every anime will have a unique ID.

R5 - Only admin can add new anime to the database.

**Requirement level (SRS)**

Here all requirements are maps against different goals as to what we want to achieve, defects as to which we want to prevent and techniques as to how we will prevent send defects. This can help us eliminate the bugs and errors early on even before anything is implemented or designed. This leads to saving many hours of work and effort as well as resources.

**Table;**

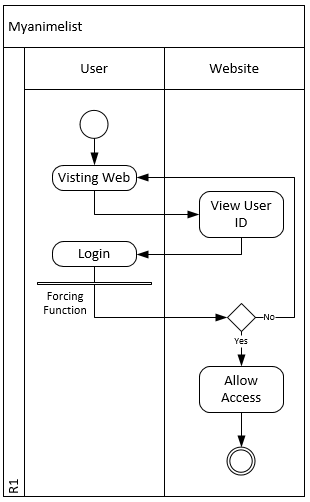
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Goal** | **Sub-Goal** | **Requirement** | **Defect** | **Prevention Technique** |
| **R1** | Security | Authenticity | User must log in to view other users ID | Unauthorized access | Forcing function |
| **R2** | Security | Data  Integrity | Confirmation box appears when user adds an anime to a list | Accidental addition | Redundancy |
| **R3** | Performance | Response Time | Search filter will display results within 2 seconds of searching | Long loading time | Standardization |
| **R4** | Maintainability | Stability | Every anime will have a unique ID | Duplication | Automation And Computerization |
| **R5** | Reliability | Fault tolerance | Only admin can add new anime to the database | False addition | Fail-Safe and Constraint |

**Design level (UC)**

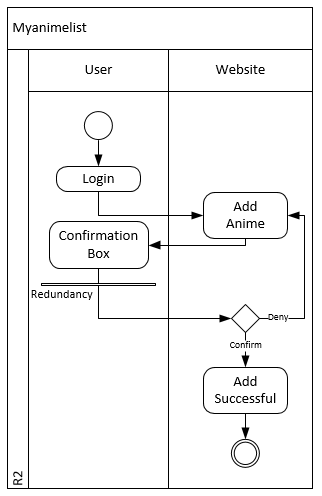
In this we add the defect preventions in activity diagram. This will help to visualize exactly where we have to add the preventions in the flow of the code.

No R3 included as it is difficult to represent performance related to time in an activity diagram.

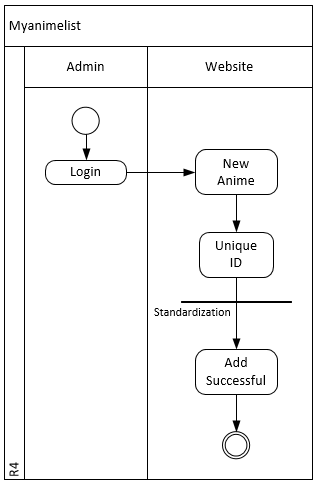
**R1;**



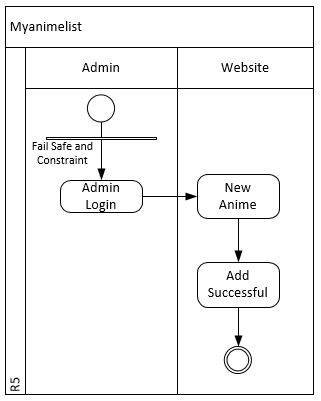
**R2;**



**R4;**



**R5;**

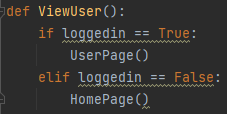


**Code level (SC)**

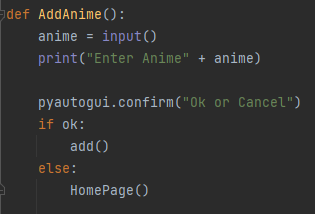
Finally the last stage of defect prevention for which we did all the previous steps in order to implement the preventions in to our code so we can have the prevention actually working.

For R3 a good search algorithm will be required to be implemented for quicker search results.

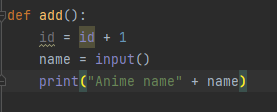
**R1;**



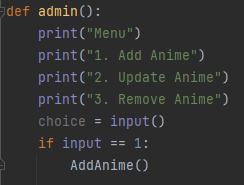
**R2;**



**R4;**



**R5;**



**Defect Detection**

After implementing defect prevention we reduce the bugs being injected but not completely prevent them from occurring hence we use defect identification. In this we try to catch any errors that slip though the previous stage.

**System;**

This web site is designed to keep track of anime that someone might be watching, has finished or plans to watch in the future, allowing it user to very easily manage all their anime. The system will allow the user to register, add anime to different lists such as watching, finished, plan to watch and dropped, search anime. While the admin manages all the anime available.

**Requirements;**

R1 - User must log in to view other users ID.

R2 - Confirmation box appears when user adds an anime to a list.

R3 - Search filter will display results within 2 seconds of searching.

R4 - Every anime will have a unique ID.

R5 - Only admin can add new anime to the database.

R6 - User can rate the anime only when logged in between 1 to 5.

**Requirement level (SRS)**

For defect detection on requirement level we use equivalence class partitioning. This technique allows to test all inputs in a variable that may lead to errors.

**Equivalence class partitioning;**

**R1 - User must log in to view other users ID.**

**Variable;**

User login

**Valid Class;**

Login = {True}

**Invalid Class;**

Login = {False}

**Test Cases;**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **User Login** | **Class** | **Expected Output** |
| **1** | True | Login = True | Access allowed |
| **2** | False | Login = False | Access denied |

**R6 - User can rate the anime between 1 to 5.**

**Variable;**

Rating

**Valid Class;**

0 ≤ Rating ≤ 5

**Invalid Class;**

Rating < 0

Rating > 5

Rating = Character

**Test Cases;**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **User Login** | **Class** | **Expected Output** |
| **1** | 2 | 0 ≤ Rating ≤ 5 | Rating added |
| **2** | -2 | Rating ≤ 0 | Rating not added |
| **3** | 8 | Rating ≥ 5 | Rating not added |
| **4** | c | Rating = Character | Rating not added |

**Checklist;**

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Checklist Point** | **Defect** |
| User must log in to view other users ID | Navigation: Is information easy to find? | Unclear instructions for the user |
| Confirmation box appears when user adds an anime to a list | Are prompts brief? | Too much information in prompts |
| Search filter will display results within 2 seconds of searching | Is the information relevant? | Results are irrelevant |
| Only admin can add new anime to the database | Does the system prevent users from making errors whenever possible? | Incorrect data input from user |
| User can rate the anime between 1 to 5 | What is this thing for? | Wrong rating average |

**Design level (UC)**

At design level we write full use cases and case table to find the error that may have been added in to the software.

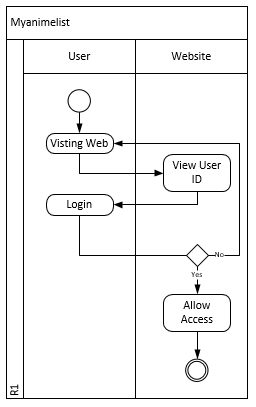
**Test case scenario;**

In test case scenario we write extended use cases and run through a complete action that a user might perform on our software.

**Use case table;**

|  |  |
| --- | --- |
| Use Case Name | R1 – Other user account access |
| Description | User must log in to view other users ID. |
| Actors | User (Primary). |
| Pre-Condition | User must be logged in to their account. |
| Post-Condition | User can view other users ID. |
| Basic Flow | 1. User Logs in to their account. 2. User is on the home page. 3. User searches for another account. 4. Account is displayed. 5. User opens and visits the other user ID. |
| Alternate Flow | 1a. User is not logged in.  1b. User searches for another account.  1c. Account is displayed.  1d. User opens but cannot visit the other user ID. |
| NFR | NFR1. Search result must be displayed under 2 sec. |

**Activity diagram;**



**Test Classes;**

**Case 1;**

User login

**Variable;**

Login

**Valid Class;**

Login = True

**Invalid Class;**

Login = False

**Case 2;**

Search result

**Variable;**

Search

**Valid Class;**

Search = Found

**Invalid Class;**

Search = Not Found

**Case 3;**

ID access

**Variable;**

Access

**Valid Class;**

Access = Allowed

**Invalid Class;**

Access = Not allowed

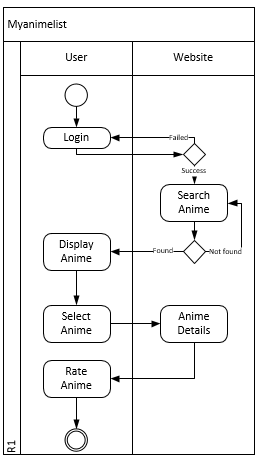
**Test case table;**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Login** | **Class** | **Search** | **Class** | **Access** | **Class** | **Result** |
| **1** | Correct login, correct search and access allowed | True | Login = True | Found | Search = Found | Allowed | Access = Allowed | User can visit other user ID |
| **2** | Incorrect login | False | Login = False | Found | Search = Found | Not allowed | Access = Not allowed | Login to view user ID |
| **3** | Search not found | True | Login = True | Not found | Search = not found | Not allowed | Access = Not allowed | User not found |

**Use case table;**

|  |  |
| --- | --- |
| Use Case Name | R6 – Rating |
| Description | User can rate the anime only when logged in between 1 to 5. |
| Actors | User (Primary). |
| Pre-Condition | 1. User must login. 2. User must visit the anime detail page. |
| Post-Condition | User submitted a rating for the anime. |
| Basic Flow | 1. User Logs in to their account. 2. User selects or searches for an anime. 3. User visits the anime detail page. 4. User enters the rating. 5. User submits the rating. |
| Alternate Flow | 1a. User is not logged in.  1b. User select or searches for another anime.  1c. User visits the anime detail page.  1d. User cannot submit a rating for the anime. |
| NFR | NFR1. Search result must be displayed under 2 sec. |

**Activity diagram;**



**Test Classes;**

**Case 1;**

User Login

**Variable;**

Login

**Valid class;**

Login = True

**Invalid class;**

Login = False

**Case 2;**

Search result

**Variable;**

Search

**Valid class;**

Search = Found

**Invalid class;**

Search = Not found

**Case 3;**

Anime rating

**Variable;**

Rating

**Valid class;**

0 ≤ Rating ≤ 5

**Invalid class;**

Rating < 0

Rating > 5

Rating = Character

**Test case table;**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Login** | **Class** | **Search** | **Class** | **Rating** | **Class** | **Result** |
| **1** | Correct login, correct search and correct rating | True | Login = True | Found | Search = Found | 2 | 0 ≤ Rating ≤ 5 | Rating submitted |
| **2** | Incorrect login | False | Login = False | Found | Search = Found | Not available | Not available | Cannot rate anime |
| **3** | Search not found | True | Login = True | Not found | Search = not found | Not available | Not available | Anime not found |
| **4** | Incorrect rating | True | Login = True | Found | Search = Found | -2 | Rating < 0 | Error incorrect rating |
| **5** | Incorrect rating | True | Login = True | Found | Search = Found | 9 | Rating > 5 | Error incorrect rating |
| **6** | Incorrect rating | True | Login = True | Found | Search = Found | v | Rating = Character | Error incorrect rating |

**Code level (SC)**

On implementation level in defect detection we use white box testing in order to check our code working and what the user will see as a result of their actions.

**White box testing;**

In white box testing we check the flow of the code line by line and further analysis the flow by creating a control flow diagram. This help use apply the path coverage technique on the code to test the actual output from the code.

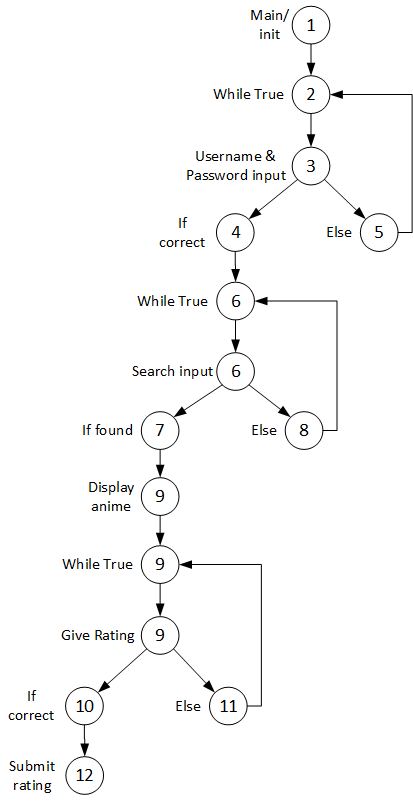
**Test case;**

R6 - User can rate the anime only when logged in between 1 to 5.

**Code;**

1. check1 = True
2. check2 = True
3. check3 = True
4. while check1 == True:
5. check1 = False
6. username = input("Enter Username: ")
7. password = input("Enter password: ")
8. if username == userlist and password == passlist:
9. while check2 == True:
10. check2 = False
11. search = input("Search anime: ")
12. if search == animelist:
13. while check3 == True:
14. check3 = False
15. print("Anime: Hunter X Hunter")
16. print("Seasons: 6")
17. rating = input("Enter Rating: ")
18. if 0 <= int(rating) <= 5:
19. print("Anime rated")
20. else:
21. print("Anime not rated")
22. check3 = True
23. else:
24. print("Anime not found")
25. check2 = True
26. else:
27. print("Username or Password Wrong")
28. check1 = True

**Control Flow Diagram;**



**Technique;**

Path coverage.

**ECP;**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Valid** | **Invalid** |
| **username** | username = Exist | username = Not exist |
| **password** | password = Match | password = Miss match |
| **search** | search = Found | Search = Not found |
| **rating** | 0 ≤ rating ≤ 5 | rating < 0, rating > 5 |

**Classes;**

**Attribute;**

username

**Valid;**

username = Exist

**Invalid;**

username = Not exist

**Attribute;**

password

**Valid;**

password = Match

**Invalid;**

password = Miss match

**Attribute;**

search

**Valid;**

search = Found

**Invalid;**

Search = Not found

**Attribute;**

rating

**Valid;**

0 ≤ rating ≤ 5

**Invalid;**

rating < 0

rating > 5

**Test Case;**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Cases** | **username** | **ECP** | **password** | **ECP** | **search** | **ECP** | **rating** | **ECP** | **Actual**  **output** |
| **T1** | Haris | username  = Exist | 1234 | password = Match | HxH | search  = Found | 4 | 0 ≤ rating  ≤ 5 | Anime Rated |
| **T2** | abc | username= Not exist | 1234 | password = Match | HxH | search  = Found | Not  available | Not  available | Username or Password Wrong |
| **T3** | Haris | username  = Exist | 0000 | password = Miss match | HxH | search  = Found | Not  available | Not  available | Username or Password Wrong |
| **T4** | Haris | username  = Exist | 1234 | password = Match | Boku No Hero | Search  = Not found | Not  available | Not  available | Anime  not found |
| **T5** | Haris | username  = Exist | 1234 | password = Match | HxH | search  = Found | -4 | rating < 0 | Anime  not rated |
| **T6** | Haris | username  = Exist | 1234 | password = Match | HxH | search  = Found | 7 | rating > 0 | Anime  not rated |

**Defect Containment**

In defect containment we analysis the information gathered in previous two stages defect prevention and defect detection. Unless we measure the gained information there is no point in spending so much time in getting all that data. We specifically gathered that data to measure our progress to see how we are performing overall in the requirement, design and implementation phases.

**System;**

This web site is designed to keep track of anime that someone might be watching, has finished or plans to watch in the future, allowing it user to very easily manage all their anime. The system will allow the user to register, add anime to different lists such as watching, finished, plan to watch and dropped, search anime. While the admin manages all the anime available.

**Requirements;**

R1 - User must log in to view other users ID.

R2 - Confirmation box appears when user adds an anime to a list.

R3 - Search filter will display results within 2 seconds of searching.

R4 - Every anime will have a unique ID.

R5 - Only admin can add new anime to the database.

R6 - User can rate the anime only when logged in between 1 to 5.

**Error Logging and Analysis**

At this stage we log all the errors by plotting a graph against our own two specified measures. This can help us visualize how many and where they are occurring while in analysis we see which paths will lead to success and which to failure.

**Error Logging;**

The measurement used for this project here is the number of error found in the previous stage and where in that stage. This means measure 1 will be errors on Y-axis and measure 2 will be the stage if the phase on X-axis.

**Requirement;**

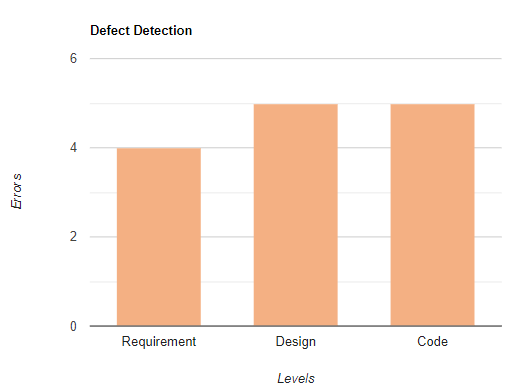
R6 - User can rate the anime only when logged in between 1 to 5.

**Error Log;**

In defect detection we found the most errors, a total of 14 errors overall. These narrow down to 4 errors in requirement level, 5 errors in design level and 5 errors in code level.

|  |  |
| --- | --- |
| **Defect Detection** | |
| **Level** | **Errors** |
| Requirement Level | 4 |
| Design Level | 5 |
| Code Level | 5 |

**Error Graph;**



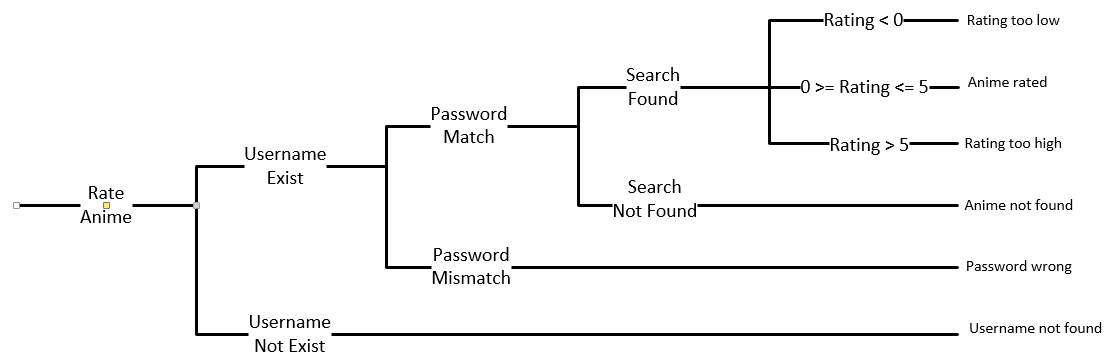
**Analysis;**

In analysis we used event tree to determine the paths and the results of these paths. Weather they lead to success or failure.

**Requirement;**

R6 - User can rate the anime only when logged in between 1 to 5.

**Event Tree Analysis (ETA);**



**Project Review**

**Task;**

In this project we were to use the same requirements from 1 project to complete 3 phases of software quality assurance. These phases included:

* Defect Prevention.
* Defect Detection.
* Defect Containment.

**System;**

This web site is designed to keep track of anime that someone might be watching, has finished or plans to watch in the future, allowing it user to very easily manage all their anime. The system will allow the user to register, add anime to different lists such as watching, finished, plan to watch and dropped, search anime. While the admin manages all the anime available.

**Requirements;**

R1 - User must log in to view other users ID.

R2 - Confirmation box appears when user adds an anime to a list.

R3 - Search filter will display results within 2 seconds of searching.

R4 - Every anime will have a unique ID.

R5 - Only admin can add new anime to the database.

R6 - User can rate the anime only when logged in between 1 to 5.

**Defect Prevention**

**Requirement Level;**

**Requirement Table;**

Here all requirements are maps against different goals as to what we want to achieve, defects as to which we want to prevent and techniques as to how we will prevent send defects. This can help us eliminate the bugs and errors early on even before anything is implemented or designed. This leads to saving many hours of work and effort as well as resources.

**Design Level;**

**Activity Diagram;**

In this we add the defect preventions in activity diagram. This will help to visualize exactly where we have to add the preventions in the flow of the code.

**Code Level;**

**Implementation;**

Finally the last stage of defect prevention for which we did all the previous steps in order to implement the preventions in to our code so we can have the prevention actually working.

**Defect Detection**

**Requirement Level;**

**Equivalence class partitioning (ECP);**

For defect detection on requirement level we use equivalence class partitioning. This technique allows to test all inputs in a variable that may lead to errors.

**Checklist;**

Checklist provides a set of pre-defined questions standard for all project. Achieving a positive answer for as many question as possible will lead to a better resulting software.

**Design Level;**

**Test case scenario;**

In test case scenario we write extended use cases and run through a complete action that a user might perform on our software.

**Black box testing;**

In black box we input values and check the expected outcome without looking into the workings or flow of the input that resulted in such outcome.

**Code Level;**

**White box testing;**

In white box testing we check the flow of the code line by line and further analysis the flow by creating a control flow diagram. This help use apply the path coverage technique on the code to test the actual output from the code.

**Defect Containment**

**Error Logging;**

**Error graph;**

The measurement used for this project here is the number of error found in the previous stage and where in that stage. This means measure 1 will be errors on Y-axis and measure 2 will be the stage if the phase on X-axis.

**Analysis;**

**Event Tree analysis (ETA);**

In analysis we used event tree to determine the paths and the results of these paths. Weather they lead to success or failure.