./

GENESIS - Learning Outcome & Mini-project Summary Report



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

# Contents

[Contents 3](#_Toc55739428)

[List of Tables 4](#_Toc55739429)

[List of Figures 4](#_Toc55739430)

[Miniproject -1(Selenium Automation with Java) [Team] 5](#_Toc55739431)

[Modules 5](#_Toc55739432)

[Topic and Subtopics 5](#_Toc55739433)

[Objectives & Requirements 5](#_Toc55739434)

[Design 6](#_Toc55739435)

[Test Plan 7](#_Toc55739436)

[Implementation Summary 8](#_Toc55739437)

[Video Summary 9](#_Toc55739438)

[Git Link 9](#_Toc55739439)

[Git Dashboard 9](#_Toc55739440)

[Summary 10](#_Toc55739441)

[Individual Contribution & Highlights 10](#_Toc55739442)

[Challenges faced and how were they overcome. 11](#_Toc55739443)

[Miniproject -2(JavaScript and Jasmine Framework) [Team] 12](#_Toc55739444)

[Modules 12](#_Toc55739445)

[Topic and Subtopics 12](#_Toc55739446)

[Objectives & Requirements 12](#_Toc55739447)

[Design 13](#_Toc55739448)

[Test Plan 13](#_Toc55739449)

[Implementation Summary 14](#_Toc55739450)

[Video Summary 15](#_Toc55739451)

[Git Link 15](#_Toc55739452)

[Git Dashboard 15](#_Toc55739453)

[Summary 16](#_Toc55739454)

[Individual Contribution & Highlights 16](#_Toc55739455)

[Challenges faced and how were they overcome. 17](#_Toc55739456)

[Miniproject -3(Advanced Python programming) [Team] 18](#_Toc55739457)

[Modules 18](#_Toc55739458)

[Topic and Subtopics 18](#_Toc55739459)

[Objectives & Requirements 18](#_Toc55739460)

[Design 18](#_Toc55739461)

[Test Plan 19](#_Toc55739462)

[Implementation Summary 20](#_Toc55739463)

[Video Summary 20](#_Toc55739464)

[Git Link 20](#_Toc55739465)

[Git Dashboard 20](#_Toc55739466)

[Summary 21](#_Toc55739467)

[Individual Contribution & Highlights 21](#_Toc55739468)

[Challenges faced and how were they overcome. 21](#_Toc55739469)

## List of Tables

[Table 1. Unit Level Test Cases 7](#_Toc55661729)

[Table 2.Integration Level Test Cases 7](#_Toc55661730)

[Table 3.Individual 10](#_Toc55661731)

## List of Figures

[Figure 1.Sequence Diagram 6](#_Toc55739474)

[Figure 2.Object Diagram 7](#_Toc55739475)

[Figure 3.Eclipse IDE console output 9](#_Toc55739476)

[Figure 4. Git Repo Screenshot 10](#_Toc55739477)

[Figure 3.Eclipse IDE console output 15](#_Toc55739478)

[Figure 4. Git Repo Screenshot 16](#_Toc55739479)

[Figure 3.Python console output 20](#_Toc55739480)

[Figure 4. Git Repo Screenshot 21](#_Toc55739481)

# 

# Miniproject -1(Selenium Automation with Java) [Team]

## Modules

Modules linked to the miniproject–SDLC , Java and Selenium.

## Topic and Subtopics

Core Topic:

Web page automation

Selenium automation using java with ECLIPSE IDE

Sub Topics:

Testing using cucumber framework

Creation of feature files

Run configuration – Cucumber feature

## Objectives & Requirements

Objective: To automate a sample webpage and check all the functionalities and hyperlinks.

<http://www.practiceselenium.com>

Understanding Java and selenium tool.

Testing using Cucumber framework.

Requirements (High level and low level):

|  |  |
| --- | --- |
| ID | Description |
| HL\_01\_L\_01 | Low level 01 – Invoke ECLIPSE IDE  High level 01 –Creation of MAVEN repository |
| HL\_02\_L\_02 | Low level 02 – Setting up executable chrome driver  High level 02 –Launching chrome browser |
| HL\_03\_L\_03 | Low level 03 – Identifying required URL  High level 03 – Launching corresponding URL |
| HL\_04\_L\_04 | Low level 04– The Automated Testing System should include the following components: Selenium dependencies and JAVA environment required by the tests  High level 04 –Page automation and checking the functionality. |
| HL\_05\_L\_05 | 1. Low level 05 – The Automated Testing System should be invoked from BDD (cucumber framework) 2. High level 05 –Feature file creation and execution using cucumber framework. |
| HL\_06\_L\_06 | 1. Low Level 06 - The Automated Testing System should implement the tests cases for the following scenarios: 2. Given launching chrome 3. When launching URL 4. Then check functionality 5. High Level 06 – should pass all the above scenarios. |

## Design

Behavioral Diagram (Sequence Diagram)

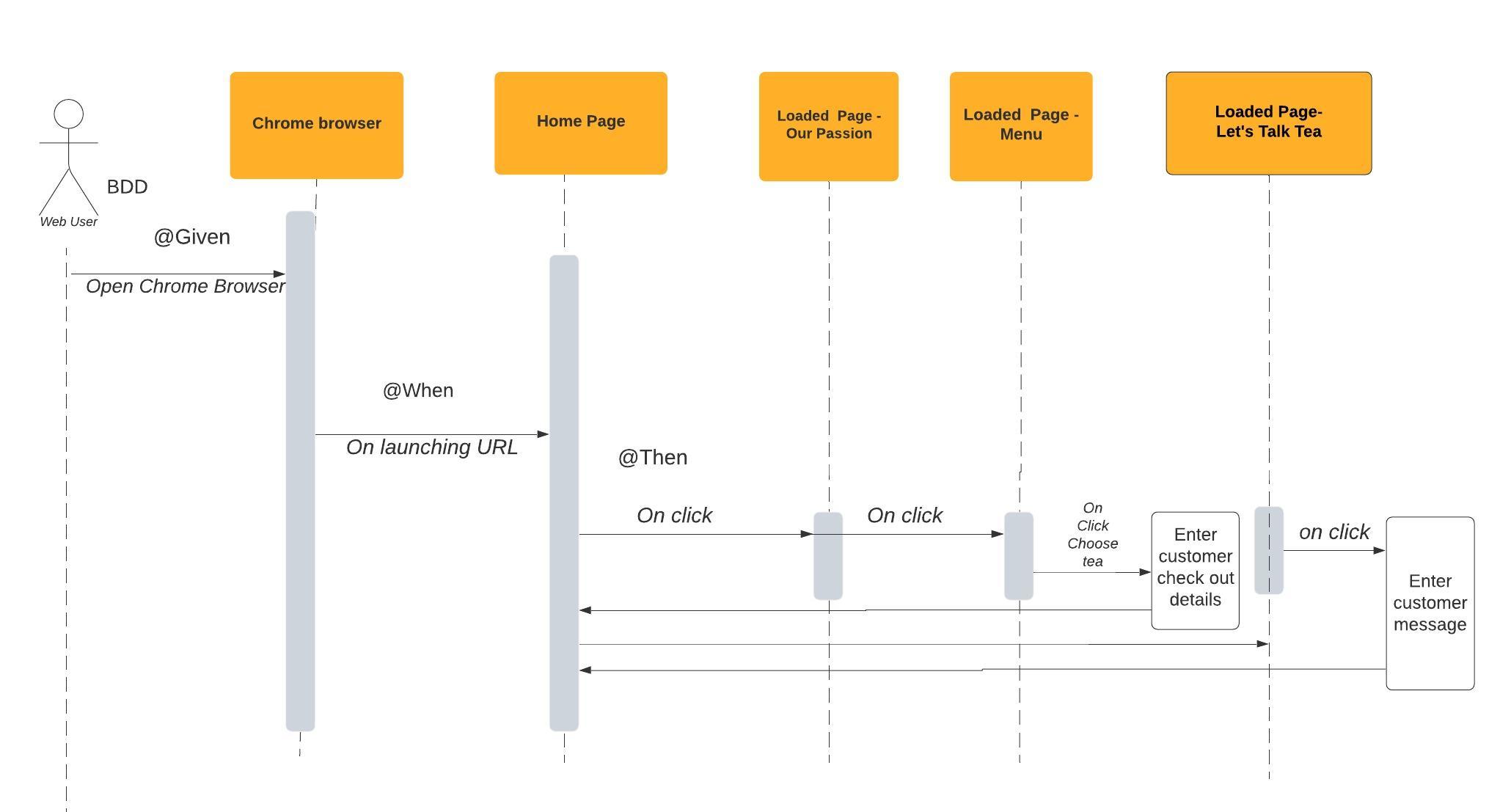


Figure 1.Sequence Diagram

Structural Diagram (Object Diagram)

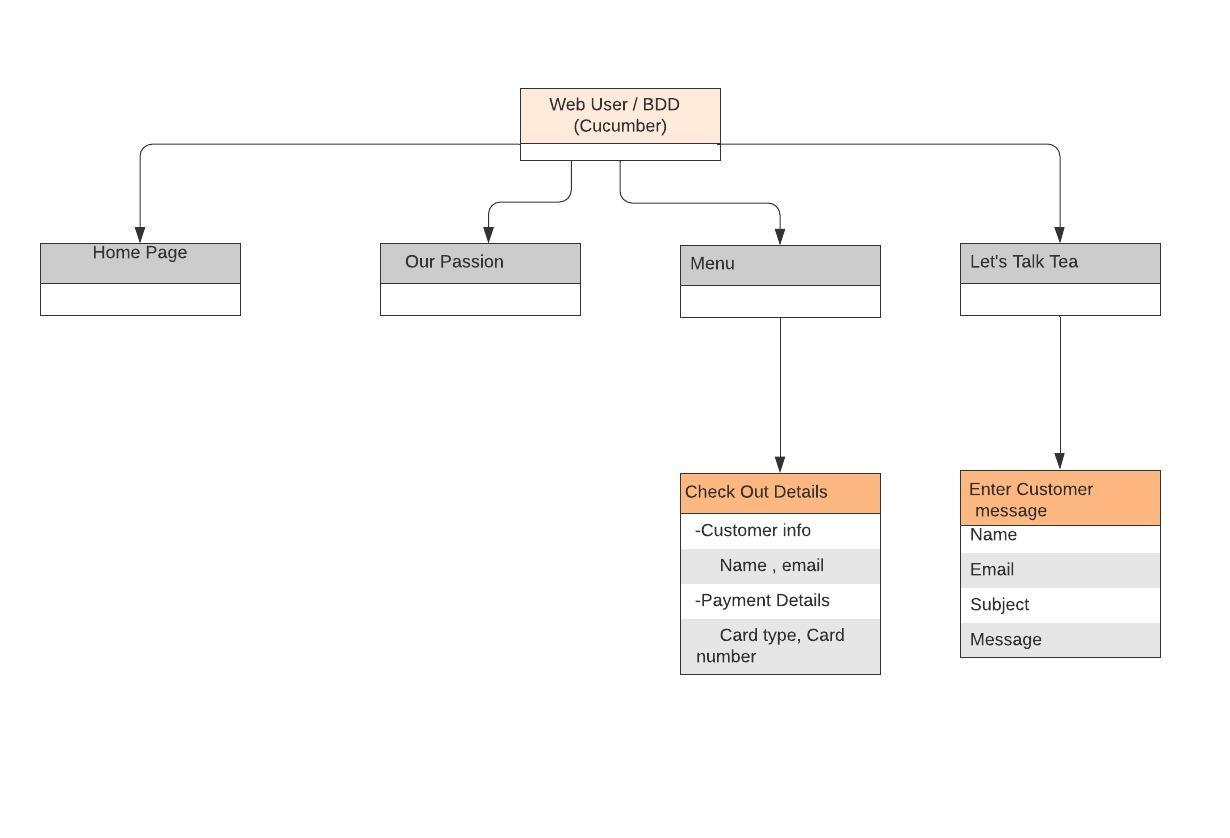


Figure 2.Object Diagram

## Test Plan

Unit Level Test Cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Precondition | Excepted input | Expected output | Actual output |
| Tc1 | Check whether chrome browser and URL is launching | Presence of executable chrome driver. | Chrome driver.exe | Browser launch | Browser Launch |
| Tc2 | Check whether ECLIPSE IDE compatible with Java environment | Set required environment variables and path. | Any java Project Creation | Successful Project execution | Successful Project execution |
| Tc3 | Check whether all Selenium dependencies( POM.xml) are added. | Maven repository | Add Selenium dependencies based on chrome version. | Invoking all selenium JARS in a MAVEN dependency library. | Invoked all selenium JARS in a MAVEN dependency library. |
| Tc4 | Check whether all cucumber dependencies( POM.xml) are added. | Maven repository | Add cucumber dependencies based on chrome version. | Invoking all cucumber JARS in a MAVEN dependency library. | Invoked all cucumber JARS in a MAVEN dependency library. |
| Tc5 | Check whether home screen is visible. | Launch chrome browser. | Corresponding Java code. | Launch home page. | Launched home page. |
| Tc6 | Check whether ‘Our Passion’ is displayed. | Launch chrome browser and home page. | Corresponding Java code | Launch the corresponding page. | Launched the corresponding page. |
| Tc7 | Check whether ‘Menu’ page is displayed | Launch chrome browser and home page. | Corresponding Java code | Launch the corresponding page. | Launched the corresponding page. |
| Tc8 | Check whether checkout details can be entered and submitted. | Launch menu page in chrome browser. | Enter customer and payment details. | Display the entered details and submit the page. | Displayed the entered details and submitted the page |
| Tc9 | Check whether ‘Let’s Talk Tea ’ page is displayed | Launch chrome browser and home page. | Corresponding Java code | Launch the corresponding page. | Launched the corresponding page. |
| Tc10 | Check whether customer message can be entered and submitted. | Launch let’s talk tea page in chrome browser. | Enter customer  Details and message | Display the entered details and submit the page. | Displayed the entered details and submitted the page |

Table 1. Unit Level Test Cases

Integration level Test cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tc1 | Check whether all page functionalities are passing the cucumber test framework. | Launch chrome browser | Corresponding Java code and cucumber tags. | Launch the corresponding page with all functionalities. | Launched the corresponding page with all functionalities. |

Table 2.Integration Level Test Cases

## Implementation Summary

Automated a web page using Selenium testing tool with JAVA to check all the page functionalities .Also incorporated Behavioral Driven Development (BDD) testing with cucumber framework. The following scenarios were tested using cucumber tags:

Feature: Page functionality checking

Scenario: functionality checking

Given launching chrome

When launching URL

Then check functionality

Output:

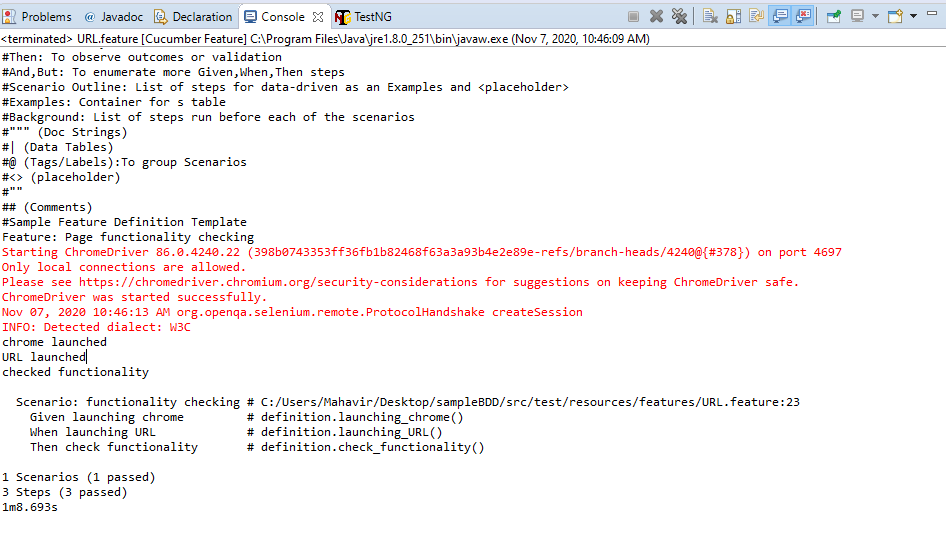


Figure 3.Eclipse IDE console output

### Video Summary

The video includes a small look into our miniproject workflow. Initially the java file with selenium dependencies is executed in ECLIPSE IDE and the page automation is done. It is followed by execution of feature file indicating BDD testing and test cases passing with cucumber framework.

[video link](https://github.com/99002550/MiniProject-Selenium-with-JAVA-/tree/main/Selenium_video)

### Git Link

[Git Repo Link](https://github.com/99002550/MiniProject-Selenium-with-JAVA-)

### Git Dashboard

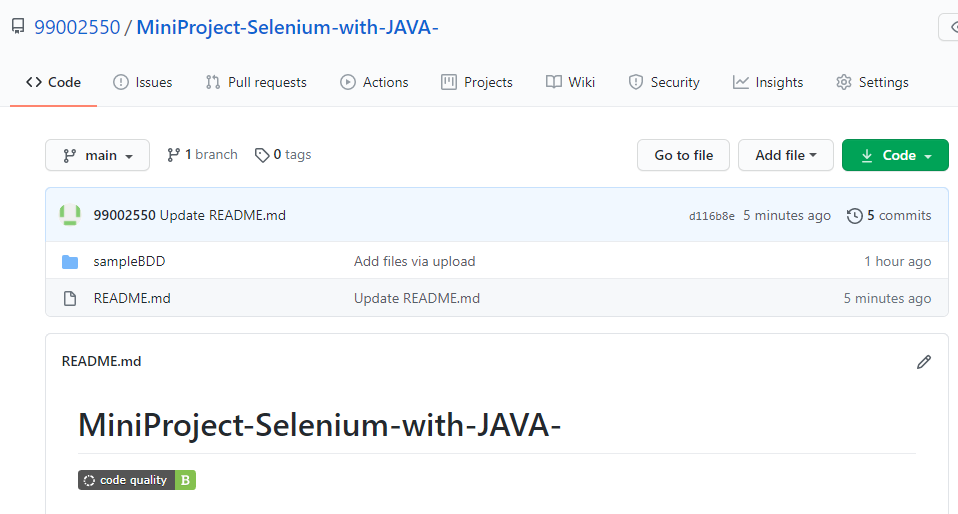


Figure 4. Git Repo Screenshot

### Summary

Automated a web page using Selenium testing tool with JAVA to check all the page functionalities. Also incorporated Behavioral Driven Development (BDD) testing with cucumber framework. The following scenarios were tested using cucumber tags:

Feature: Page functionality checking

Scenario: functionality checking

Given launching chrome

When launching URL

Then check functionality

## Individual Contribution & Highlights

|  |  |  |
| --- | --- | --- |
| S.No | Name | Contributions |
| 1 | Shahna S.S (99002550) | Done with the selenium automation of web page.  Documentation of report and ppt. |
| 2 | Riya Kuriakose (99002556) | Done with the BDD testing of automated page with cucumber framework.  Documentation of report and ppt. |
| 3 | Gowsalya Selvaraj (99002) | Done with git repository creation, readme file creation and code quality checking.  Documentation of report and ppt. |

Table 3.Individual Contributions

## Challenges faced and how were they overcome.

Challenge 1: Timeout and sync issue.

Solution: This can be avoided by including delay ie, Threadsleep ().

Challenge 2: Page Loading.

Solution: Certain web pages are user specific and load different elements depending upon the previous activity of the user. It can be overcomed by giving commands like explicit waits and giving enough time to load.

Challenge 3: Difficulty in automating certain elements

Solution: Tried automating with the help of relative Xpath even though 100% automation was not possible.

### 

# Miniproject -2(JavaScript and Jasmine Framework) [Team]

## Modules

Modules linked to the miniproject–SDLC , JavaScript and Jasmine Framework.

## Topic and Subtopics

Core Topic:

Web page designing

Web page designing using HTML, CSS and JavaScript.

Sub Topics:

Testing using Jasmine Framework

Creation of spec files, source file and specrunner.

## Objectives & Requirements

Objective: To design a webpage for shopping the baby products using HTML, CSS, JavaScript in Visual studio code and testing the script using Jasmine framework. Understanding JavaScript and Jasmine Framework.

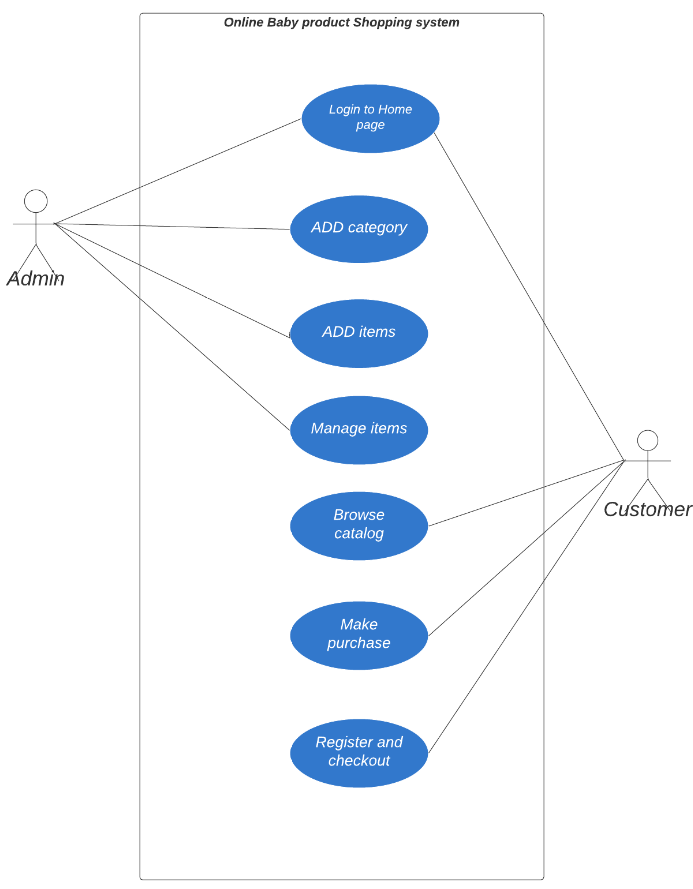
Requirements (High level and low level):

|  |  |
| --- | --- |
| ID | Description |
| HL\_01\_L\_01 | Low level 01 – Invoke Visual studio Code  High level 01 –Creation of HTML file with JavaScript and CSS. |
| HL\_02\_L\_02 | Low level 02 – Any browser of choice  High level 02 –launching the created HTML page in the default browser. |
| HL\_03\_L\_03 | Low level 03 – Invoke Jasmine framework in VS code.  High level 03 – passing of all Suites during test. |
| HL\_04\_L\_04 | Low level 04–HTML page with valid script.  High level 04 –Able to make purchases |
| HL\_05\_L\_05 | 1. Low level 05 – HTML page with valid script. 2. High level 05 –Able to register in site with valid details. |
| HL\_06\_L\_06 | 1. Low Level 06 – HTML page with valid script and corresponding regular expression. 2. High Level 06 –Throws error for invalid details. |

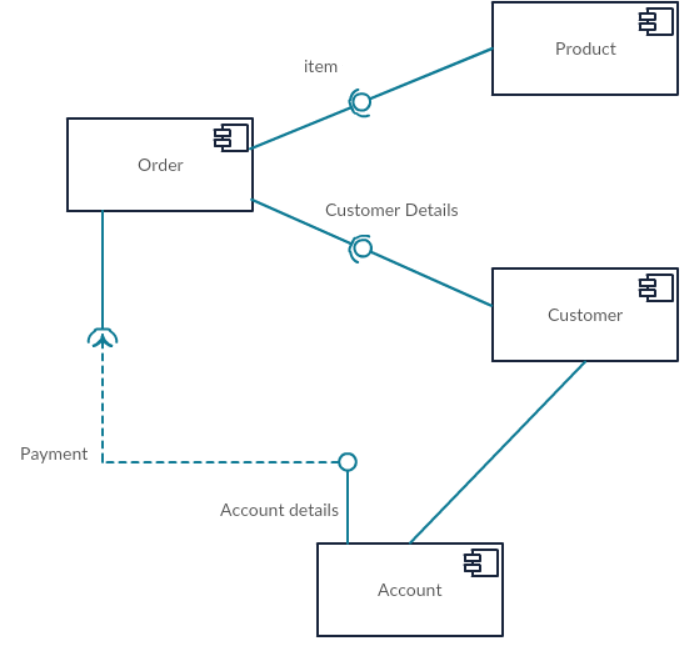
## 

## Design

Behavioral Diagram ( UseCase Diagram)



Structural Diagram (Component Diagram)



## Test Plan

Unit Level Test Cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Precondition | Excepted input | Expected output | Actual output |
| Tc1 | Check whether chrome browser and URL is launching | Presence of executable chrome driver. | Chrome driver.exe | Browser launch | Browser Launch |
| Tc2 | Check whether ECLIPSE IDE compatible with Java environment | Set required environment variables and path. | Any java Project Creation | Successful Project execution | Successful Project execution |
| Tc3 | Check whether all Selenium dependencies( POM.xml) are added. | Maven repository | Add Selenium dependencies based on chrome version. | Invoking all selenium JARS in a MAVEN dependency library. | Invoked all selenium JARS in a MAVEN dependency library. |
| Tc4 | Check whether all cucumber dependencies( POM.xml) are added. | Maven repository | Add cucumber dependencies based on chrome version. | Invoking all cucumber JARS in a MAVEN dependency library. | Invoked all cucumber JARS in a MAVEN dependency library. |
| Tc5 | Check whether home screen is visible. | Launch chrome browser. | Corresponding Java code. | Launch home page. | Launched home page. |
| Tc6 | Check whether ‘Our Passion’ is displayed. | Launch chrome browser and home page. | Corresponding Java code | Launch the corresponding page. | Launched the corresponding page. |
| Tc7 | Check whether ‘Menu’ page is displayed | Launch chrome browser and home page. | Corresponding Java code | Launch the corresponding page. | Launched the corresponding page. |
| Tc8 | Check whether checkout details can be entered and submitted. | Launch menu page in chrome browser. | Enter customer and payment details. | Display the entered details and submit the page. | Displayed the entered details and submitted the page |
| Tc9 | Check whether ‘Let’s Talk Tea ’ page is displayed | Launch chrome browser and home page. | Corresponding Java code | Launch the corresponding page. | Launched the corresponding page. |
| Tc10 | Check whether customer message can be entered and submitted. | Launch let’s talk tea page in chrome browser. | Enter customer  Details and message | Display the entered details and submit the page. | Displayed the entered details and submitted the page |

Table 1. Unit Level Test Cases

Integration level Test cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tc1 | Check whether all page functionalities are passing the cucumber test framework. | Launch chrome browser | Corresponding Java code and cucumber tags. | Launch the corresponding page with all functionalities. | Launched the corresponding page with all functionalities. |

Table 2.Integration Level Test Cases

## Implementation Summary

Automated a web page using Selenium testing tool with JAVA to check all the page functionalities .Also incorporated Behavioral Driven Development (BDD) testing with cucumber framework. The following scenarios were tested using cucumber tags:

Feature: Page functionality checking

Scenario: functionality checking

Given launching chrome

When launching URL

Then check functionality

Output:

Figure 3.Eclipse IDE console output

### Video Summary

The video includes a small look into our miniproject workflow.

### Git Link

### Git Dashboard

Figure 4. Git Repo Screenshot

### Summary

Automated a web page using Selenium testing tool with JAVA to check all the page functionalities. Also incorporated Behavioral Driven Development (BDD) testing with cucumber framework. The following scenarios were tested using cucumber tags:

Feature: Page functionality checking

Scenario: functionality checking

Given launching chrome

When launching URL

Then check functionality

## Individual Contribution & Highlights

|  |  |  |
| --- | --- | --- |
| S.No | Name | Contributions |
| 1 | Shahna S.S (99002550) | Documentation of report and ppt. |
| 2 |  | Documentation of report and ppt. |
| 3 |  | Done with git repository creation, readme file creation and code quality checking.  Documentation of report and ppt. |

Table 3.Individual Contributions

## Challenges faced and how were they overcome.

Challenge 1:

Solution:

Challenge 2:

Solution:

Challenge 3:

Solution:

# Miniproject -3(Advanced Python programming) [Team]

## Modules

Modules linked to the miniproject–SDLC and Advanced Python programming.

## Topic and Subtopics

Core Topic:

An application of python programming language.

Sub Topics:

A patient record monitoring using python

## Objectives & Requirements

Objective: To develop an application of python language for making a patient health record monitoring.

Understanding python programming and working with multifile in python and building them.

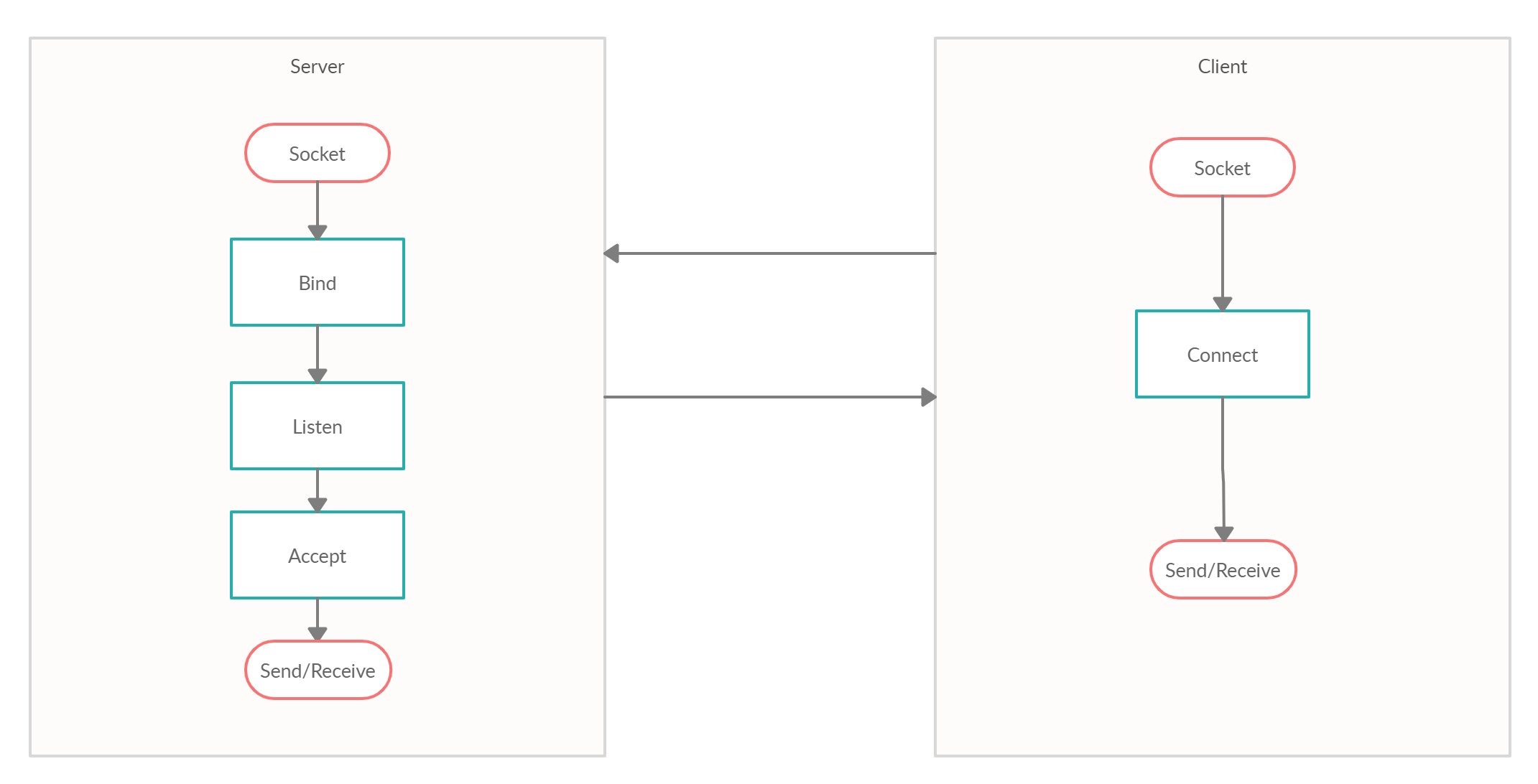
Understanding with the file operations.

Requirements (High level and low level):

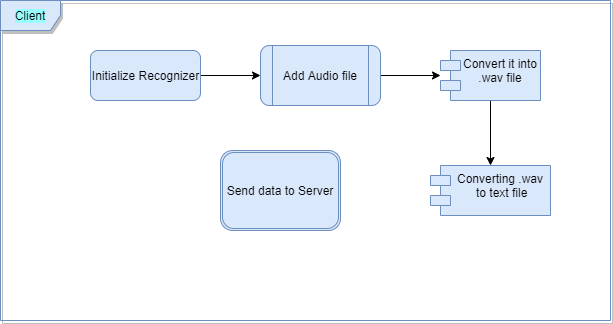
|  |  |
| --- | --- |
| ID | Description |
| HL\_01\_L\_01 | Low level 01 – current production version of Python  High level 01 –python compatibility with notepad++ |
| HL\_02\_L\_02 | Low level 02 – command prompt and python shell  High level 02 –Able to add new patient details |
| HL\_03\_L\_03 | Low level 03 – command prompt and python shell  High level 03 – Able to edit existing patient details |
| HL\_04\_L\_04 | Low level 04– command prompt and python shell  High level 04 – Able to search for existing patient details as requested |
| HL\_05\_L\_05 | 1. Low level 05 – command prompt and python shell 2. High level 05 – Able to display existing patient details as requested |
| HL\_06\_L\_06 | 1. Low Level 06 – command prompt and python shell 2. High Level 06 – Able to exit from the log on demand |

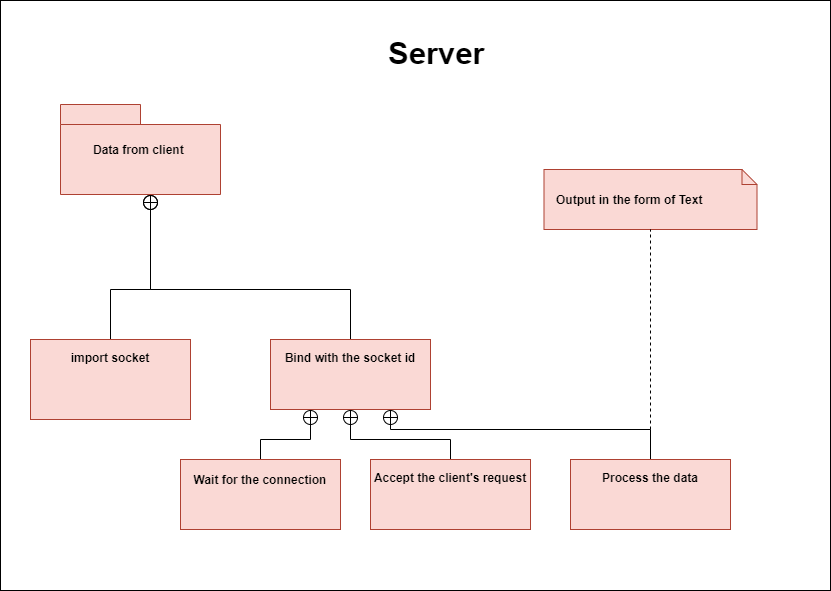
## Design

Behavioral Diagram ( UseCase Diagram)



Structural Diagram (Component Diagram)





## Test Plan

Unit Level Test Cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Precondition | Excepted input | Expected output | Actual output |
| Tc1 | Check whether python environment is set, is compatible and the path is set. | current production version of Python | Setting up the path | Indication in command window | Indication in command window |
| Tc2 | Check whether “add to patient list” function is executing. | Set required environment variables and path with current version of python. | Corresponding ID and Adding patient information to the list | List should be updated successfully | List has been updated successfully |
| Tc3 | Check whether “search in existing patient list” function is executing. | Set required environment variables and path with current version of python. | Corresponding ID and Search for a patient with full name | Display details of the requested patient | Displayed details of the requested patient |
| Tc4 | Check whether “Display all existing patient list” function is executing. | Set required environment variables and path with current version of python. | Corresponding ID to display the patient details | Display details of the all the patients in the list | Displayed details of the all the patients in the list |
| Tc5 | Check whether “modify a corresponding patient details” function is executing. | Set required environment variables and path with current version of python. | Corresponding ID and patient full name to modify | Able to modify existing details. | Existing details can be modified. |
| Tc6 | Check whether ‘Exit’ function is executed. | Set required environment variables and path with current version of python. | Corresponding ID to exit from the log | Exit the application. | Exiting the application. |

Table 1. Unit Level Test Cases

Integration level Test cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tc1 | Check whether all the functions in unit test plan when integrated are executing correctly. | Set required environment variables and path with current version of python. | Corresponding ID to details | All functions executing correctly. | All functions executing correctly. |

Table 2.Integration Level Test Cases

## Implementation Summary

An application for Patient health record monitoring with python programming language is developed. Here the user is able to create a new patient detail contains Name. Address, Disease status and ID. The new list gets updated on adding data. This data can be modified, the list can be displayed and a search action can be performed by giving corresponding Full name of the patient.

These actions are performed with the help of corresponding ID for each functions.

Enter 1. To Add Contacts

Enter 2. For Searching a Contact

Enter 3. For Modifying a Contact

Enter 4. To Display Contacts

Enter 5. To Exit

Output:

Figure 3.Python console output

### Video Summary

The video includes a small look into our miniproject workflow.

### Git Link

### Git Dashboard

Figure 4. Git Repo Screenshot

### Summary

An application for Patient health record monitoring with python programming language is developed. Here the user is able to create a new patient detail contains Name. Address, Disease status and ID. The new list gets updated on adding data. This data can be modified, the list can be displayed and a search action can be performed by giving corresponding Full name of the patient.

These actions are performed with the help of corresponding ID for each functions.

Enter 1. To Add Contacts

Enter 2. For Searching a Contact

Enter 3. For Modifying a Contact

Enter 4. To Display Contacts

Enter 5. To Exit

## Individual Contribution & Highlights

|  |  |  |
| --- | --- | --- |
| S.No | Name | Contributions |
| 1 | () | Done with coding.  Documentation of report and ppt. |
| 2 | Shahna S.S (99002550) | Done with coding.  Documentation of report and ppt. |
| 3 | () | Done with git repository creation, readme file creation and code quality checking.  Documentation of report and ppt. |

Table 3.Individual Contributions

## Challenges faced and how were they overcome.

Challenge 1: Indentation issues.

Solution: Use 4-space indents and avoiding all hard tab characters.

Challenge 2: Cross import of Modules.

Solution: Doing a selective import only in the functions where it is needed.