INDUSTRIAL INTERNSHIP AT SOLARTIS

by

RENU.K. S

20BCS015

Report submitted in partial fulfillment

of the requirements for the Degree

of Bachelor of Engineering in

COMPUTER SCIENCE AND ENGINEERING

Dr.Mahalingam College of Engineering and Technology

Pollachi – 642003

MAY 2023

Dr.Mahalingam College of Engineering and Technology

Pollachi – 642003

COURSE CODE: 19CSPN6003 INTERNSHIP

BONAFIDE CERTIFICATE

Certified that this Industrial Internship Training Report titled "INDUSTRIAL INTERNSHIP TRAINING AT SOLARTIS TECHNOLOGY SERVICES PRIVATE LIMITED " is the bonafide work of Ms. K.S.RENU who carried out the training work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr.G.Anupriya
Professor
Head of the Department
Department of Computer
Science and Engineering

Mrs.G.Uma Maheshwari
Assistant Professor
Internship Coordinator
Department of Computer
Science and Engineering

Submitted for the Industrial Internship Viva-Voce examination held on 13.04.2024

Examiner 1

Examiner 2

Dr.Mahalingam College of Engineering and Technology

Pollachi – 642003

COURSE CODE: 19CSPN6003 INTERNSHIP

DECLARATION

I affirm that the Industrial Internship Training report titled "INDUSTRIAL INTERNSHIP TRAINING AT SOLARTIS TECHNOLOGY SERVICES PRIVATE LIMITED" being submitted in partial fulfillment of the requirements for the award of the Degree of BACHELOR OF ENGINEERING IN COMPUTER SCIENCE is the original work carried out by me. It has not formed a part of any other project work submitted for award of any degree or diploma, either in this or any other Institution.

RENU.K.S

20BCS015

I certify that the declaration made above by the candidate is true.

Mrs.G.Uma Maheshwari

Assistant Professor

Internship Coordinator

Department of Computer Science and Engineering

TABLE OF CONTENTS

ABSTRACT

The major goal of the internship is to gain practical knowledge and skills that are most widely needed in the software industry. As technologies grow tremendously these days, the need for people with greater skill sets and real time working experiences are drastically increasing. Further, the clear and effective utilization of the knowledge obtained is much more necessary. The internship period provided the opportunity to test the skills that developed in college or previous work and see how they work in the real world.

During the internship, working in an office environment, we had the chance to observe how others operate whether it's communication, behavior or office etiquette. The Information Technology domain is meeting high range applications that must be capable of functioning in wide areas and situations with minimal changes in the underlying system. Even the development and maintenance of the application must be so rapid with effective operational capabilities. The internship period clearly depicted the various business environments like Unstable (UCI), Stable, Quality Assurance (QA), User Acceptance Test (UAT), Production and the various technologies used for the complete software development.

By experiencing hands-on testing during the internship, we got a clear idea about the responsibilities of the role. This enables us to concentrate and focus on the role with greater confidence. During the internship, there are several good chances to learn new skills that are both technical and interpersonal. The management skills required for development and real-world working employment are gained. In addition, the project and stress management techniques are clearly understood. Of all these, the importance of time management in the work environment has been acknowledged during the internship period.

LIST OF ABBREVIATIONS

SDLC	Software Development Life Cycle
STLC	Software Testing Life Cycle
FaaS	Function as a Service
PaaS	Platform-as-a-Services
SaaS	Software-as-a-Service
API	Application Programming Interface
UAT	User Acceptability Testing
ISO	International Organization for Standardization
UI	User Interface
UX	User Experience
QA	Quality Analyst
UCI	Unstable Continuous Integration
CEO	Chief executive officer

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The main goal is to redefine policy administration. With the rise of technologies like blockchain and IoT, insurance is changing. Solartis believes that lean insurance technology is critical for more robust risk coverage, more security in existing markets, and better reach into new and emerging markets. no longer limited by bulky, inflexible policy administration software. Solartis has serviced clients ranging from innovative startups to established insurers and industry leaders in challenging the status and have invited hundreds of developers to experiment with the microservices and build products that are faster, more useful, and future proofed. Solartis Insure has helped companies launch greenfield products in half the time, with lightning-fast performance.

1.1 COMPANY BACKGROUND:

Solartis Technology Services Pvt. Ltd, founded in 2004 with headquarters located at Manhattan Beach, California, is an IT firm that develops software for insurance and financial institutions. Nick Richardson is the President and CEO of Solartis.

Solartis believes that lean insurance technology is critical for more robust risk coverage, more security in existing markets, and better reach into new and emerging markets. They are no longer limited by bulky, inflexible policy administration software. Now creativity is the limit. From consumer portals, agency and broker systems, and full-blown carrier underwriting platforms, the tech behind the product is rooted in an understanding of the insurance needs, from the ground up.

Solartis services clients ranging from innovative startups to established insurers. Solartis is also industry leaders in challenging the status quo and has invited hundreds of developers to experiment with their micro services and build products that are faster, more useful, and future proofed. Solartis Insure has helped companies launch Greenfield products half the time, with lightning- fast performance.

At Solartis, the goal is to redefine policy administration. With the rise of insurance technologies and disruptive new solutions, they help to lead the charge and prepare the

organization for success. Solartis enables both ensure Tech startups and trusted major insurance companies to meet and address the expectations of the insurance industry. They are an insurance technology solution designed to reduce the dependence on monolithic policy administration system and reduce business needs.

Solartis has cracked the code for automating ISO Electronic Rating Content into insurance products. They automatically update platform with the latest updates within one day, so that the customers can focus on the carrier exceptions. For all products, Solartis Insure enables product versioning, product cloning, and the reusability of component coverage configurations so can launch new P&C products in record time.

Newly configured products are provided in Solartis sandbox early on, so one can test and sign off on them. This in turn, shortens the implementation and reduces the application testing activities. Solartis insure is in production with the following business lines: Commercial Liability (General, Environmental, Professional, Special Event), Commercial Package, Crime and Fidelity, Commercial and Personal Accident and Health, Commercial Property, Workers' Compensation, Commercial and Personal Inland Marine, Homeowners / Dwelling Fire, Commercial Auto, Business Owners, Commercial and Personal Excess / Umbrella.

1.2 TRAINING OBJECTIVE

At the end of the internship, I will be able to

- Be familiar with STLC phases. From test planning and design to execution, defect reporting, and closure, each step is critical. Familiarize with test case creation, execution, and defect management. A strong grasp of STLC ensures effective collaboration with development teams and thorough testing coverage.
- Be familiar with the testing methodologies like Agile methodology and testing types especially functional testing, Usability testing, browser testing, performance testing and Security testing.
- Be familiar with **Bugzilla**, an open-source bug tracking system, excels in bug tracking and provides a cost-effective, technical solution. It allows developers to track and manage

bugs efficiently, offers customization options, and supports API interaction. However, its complexity can be challenging for some users.

- Be familiar with **JIRA**, developed by Atlassian, is a powerful project management tool widely used in software development. Beyond bug tracking, JIRA offers a comprehensive suite for agile project management, issue tracking, and collaboration. Its features include agile planning, customizable dashboards, flexible workflows, and real-time release tracking. JIRA's versatility makes it suitable for teams requiring robust project management tools.
- Be familiar with manual testing strategies including test case creation, filing bugs in the QA environment.

1.3 STUDENT WORK ASSIGNMENT:

During the internship, training on various technologies and practical sessions were given. During the initial period, an overview of the business and general work environment and the practices followed were gained using Solartis University Platform. The knowledge about the insurance workflow, different roles and their responsibilities in the company and various lines of businesses are also well gained. This is accomplished well by having a questionnaire section and healthy interaction with the managers daily, which helps clear all the doubts and assess the knowledge gained by taking up weekly assessments.

As a **QA intern**, my role involves meticulously crafting test cases for diverse requirements. These test cases serve as the critical bridge between the software's intended functionality and its real-world behavior. Let's delve into the intricacies of this process:

1. **Understanding the Requirements:**
    - **Requirement Analysis:** I immerse myself in the project's requirements, dissecting them to grasp their nuances. Whether it's a feature enhancement, a bug fix, or a new module, I dissect the details.
    - **User Stories and Use Cases:** I translate user stories and use cases into tangible scenarios. These become the foundation for my test cases.

2. **Designing Effective Test Cases:**
    - **Scenario-Based Approach**: Each test case represents a specific scenario. I envision how users will interact with the system, considering various paths they might take.
    - **Positive and Negative Testing:** I cover both sunny-day scenarios (positive testing) and edge cases (negative testing). What happens when users input unexpected data? How does the system respond?
    - **Boundary Values:** I pay attention to boundaries. If an input field accepts values from 1 to 100, I test with 1, 100, and values just beyond these limits.
    - **Data Variations:** I explore different data inputs—valid, invalid, and borderline. This ensures comprehensive coverage.
    - **Preconditions and Postconditions**: I define prerequisites (preconditions) and expected outcomes (postconditions) for each test case. This contextualizes the test.
3. **Executing the Test Cases:**
    - **Step-by-Step Validation**: I meticulously follow the steps outlined in each test case. I observe how the system behaves.
    - **Observations and Documentation:** I document my observations. Did the system pass? Did it stumble? I capture screenshots, logs, and any anomalies.
    - **Regression Testing**: As the system evolves, I revisit existing test cases to ensure they still hold true. This guards against unintended side effects.
4. **Bug Reporting with Precision:**
    - **Detecting Defects:** When I encounter discrepancies—unexpected behaviors, crashes, or deviations—I document them.
    - **Clear Reproduction Steps**: I provide step-by-step instructions for reproducing the bug. Developers appreciate clarity.
    - **Severity and Priority**: I assess the impact (severity) and urgency (priority) of each bug. Critical issues get immediate attention.
    - **Environment Details:** I note the environment (browser, OS) where the bug occurred. Context matters.

5. **Striving for Minimal Bugs:**
    - **Quality Over Quantity:** It's not about finding countless bugs; it's about finding the right ones. I aim for minimal but impactful issues.
    - **Collaboration**: I work closely with developers, fostering open communication. Together, we refine the system.

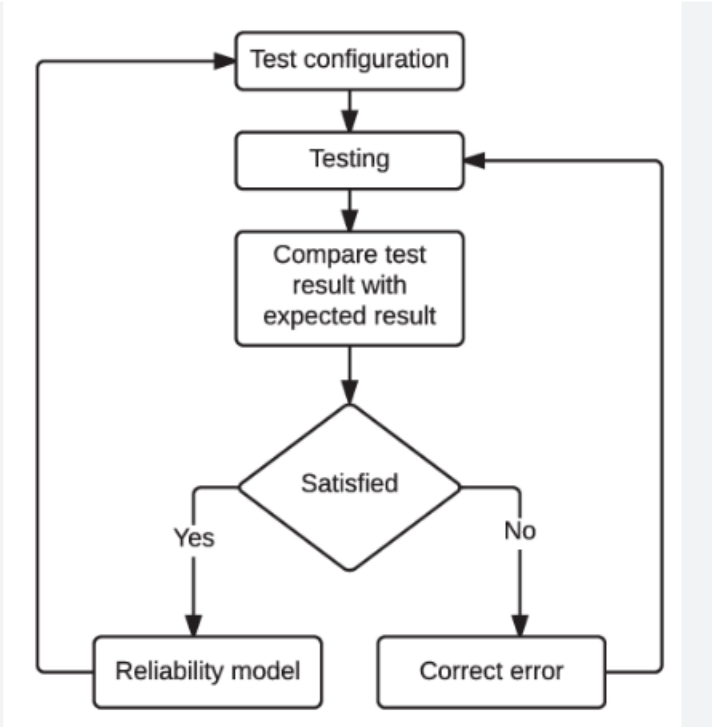

Figure 1.3.1 process flow

CHAPTER 2

TECHNICAL SECTION

2.1 SOFTWARE REQUIREMENT

Microsoft Excel serves as a valuable tool in software testing, particularly for managing and documenting test cases. Within Excel, structured templates allow testers to create well-organized test case templates, capturing essential information related to each test scenario. These templates include parameters such as test case ID, description, test steps, expected results, actual results, and test status. The benefits of using Excel for test cases are manifold: it ensures comprehensive test coverage, promotes consistency, allows reusability across projects, and facilitates traceability by linking test cases to requirements or user stories. During test execution, testers track actual results and update the test status. Additionally, Excel aids in diagnosing defects and reporting observations.

A powerful API development and testing tool, simplifies the process of working with APIs. Postman allows to create and automate test suites. can write tests for each request in a collection, chaining them together to validate complex workflows. Verify status codes and validate response data to ensure the server behaves as expected. Postman provides a user-friendly interface for creating, testing, and managing API requests. Whether a beginner or an experienced developer, Postman streamlines interactions with APIs and is essential for anyone working with web services. It offers a robust testing framework. can write test scripts to verify API behavior. Validate status codes and check response data correctness. It supports automated testing, preventing human errors and streamlining development and QA processes.

JIRA, developed by Atlassian, is a versatile software tool used for project management, issue tracking, and agile project management. It serves as a centralized platform for managing tasks, bugs, and other types of issues, allowing teams to organize and prioritize their work effectively. Whether follow Scrum, Kanban, DevOps, or other methodologies, Jira Software adapts to y workflow and integrates seamlessly with y existing tools. Trusted by over 100,000 organizations, Jira Software provides out-of-the-box features and best practices for agile teams to

develop and evolve their practices. From planning and tracking to releasing world-class software, Jira empowers teams to collaborate and achieve their best work together.

Bugzilla is an open-source tool used for issues and bugs tracking system. It is widely used as a bug-reporting tool for all types of testing functions. This tutorial introduces the readers to the basic features and usage of Bugzilla. This tutorial will guide the readers on how to utilize this tool in reporting and maintaining the bug status. It improves the quality of the product. It enhances the communication between the developing team and the testing team. It can adapt to multiple situations.

2.2 TYPES OF TESTING

2.2.1 Manual Testing

Manual testing is a technique to test the software that is carried out using the functions and features of an application. In manual software testing, a tester carries out tests on the software by following a set of predefined test cases. In this testing, testers make test cases for the codes, test the software, and give the final report about that software. Manual testing is time-consuming because it is done by humans, and there is a chance of human errors.

2.2.2 Automation Testing

Automated Testing is a technique where the Tester writes scripts on their own and uses suitable Software or Automation Tool to test the software. It is an Automation Process of a Manual Process. It allows for executing repetitive tasks without the intervention of a Manual Tester.

2.2.3 White Box Testing

White box testing techniques analyze the internal structures, the used data structures, internal design, code structure, and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing, clear box

testing or structural testing. White Box Testing is also known as transparent testing or open box testing.

2.2.4 Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the internal knowledge or implementation details of the software but rather focuses on validating the functionality based on the provided specifications or requirements.

2.2.5 Gray Box Testing

Gray Box Testing is a software testing technique that is a combination of the Black Box Testing technique and the White Box Testing technique. In the Black Box Testing technique, the tester is unaware of the internal structure of the item being tested and in White Box Testing the internal structure is known to the tester. The internal structure is partially known in Gray Box Testing. This includes access to internal data structures and algorithms to design the test cases.

2.2.6 Functional Testing

Functional Testing is a type of Software Testing in which the system is tested against the functional requirements and specifications. Functional testing ensures that the requirements or specifications are properly satisfied by the application. This type of testing is particularly concerned with the result of processing. It focuses on the simulation of actual system usage but does not develop any system structure assumptions.

2.2.7 Non - Functional Testing

Non-functional Testing is a type of Software Testing that is performed to verify the non-functional requirements of the application. It verifies whether the behavior of the system is as per the requirement or not. It tests all the aspects that are not tested in functional testing. Non-functional testing is a software testing technique that checks the non-functional attributes of the system. Non-functional testing is defined as a type of software testing to check non-functional aspects of a software application.

2.2.8 Performance Testing

Performance Testing is a type of software testing that ensures software applications perform properly under their expected workload. It is a testing technique carried out to determine system performance in terms of sensitivity, reactivity, and stability under a particular workload.

2.2.9 Usability Testing

Design a product (say a refrigerator) and when it becomes completely ready, need a potential customer to test it to check it working. To understand whether the machine is ready to come on the market, potential customers test the machines. Likewise, the best example of usability testing is when the software also undergoes various testing processes which is performed by potential users before launching into the market. It is a part of the software development lifecycle (SDLC).

2.2.10 Compatibility Testing

Compatibility testing is software testing that comes under the non-functional testing category, and it is performed on an application to check its compatibility (running capability) on different platforms/environments. This testing is done only when the application becomes stable. This means simply this compatibility test aims to check the developed software application functionality on various software, hardware platforms, networks browser etc.

2.2.11 Load Testing

Load testing determines the behavior of the application when multiple users use it at the same time. It is the response of the system measured under varying load conditions.

2.2.12 Stress Testing

In Stress Testing, we give unfavorable conditions to the system and check how it performs in those conditions.

2.2.13 Regression Testing

Regression testing is a testing method used to ensure that changes made to the software do not introduce new bugs or cause existing functionality to break. It is typically done after changes have been made to the code, such as bug fixes or new features, and is used to verify that the software still works as intended.

2.2.14 User Acceptance Testing

User Acceptance Testing is a testing methodology where clients/end users participate in product testing to validate the product against their requirements. It is done at the client's site on the developer's site. For industries such as medicine or aerospace, contractual and regulatory compliance testing, and operational acceptance tests are also performed as part of user acceptance tests.

2.2.15 Security Testing

Security Testing is a type of Software Testing that uncovers vulnerabilities in the system and determines that the data and rescues of the system are protected from possible intruders.

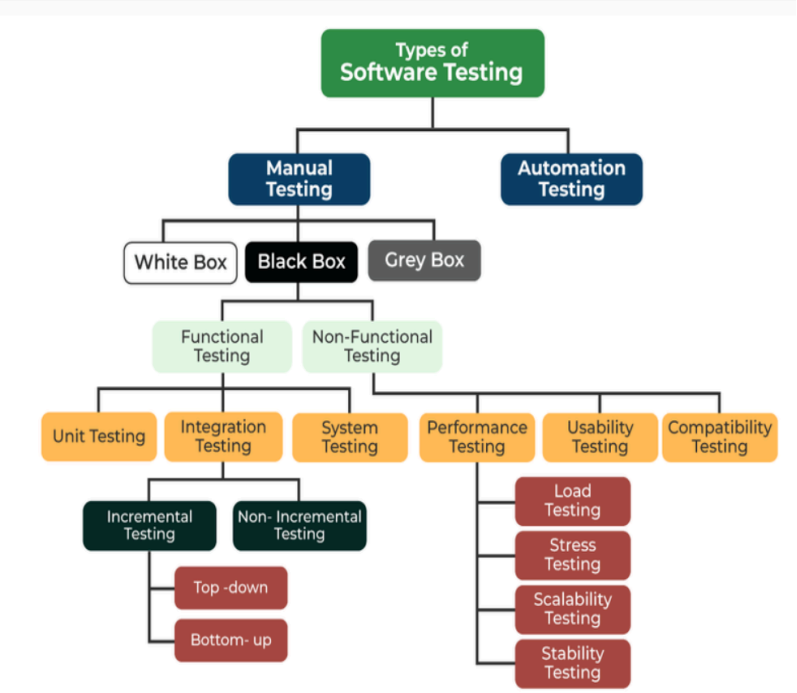

Figure 2.2.1 Types of Testing

2.3 POLICY LIFE CYCLE

Solartis Insure is an API-centric policy administration platform built from the ground up on microservice architecture. The platform is deployed in a cloud infrastructure and supports the full policy life cycle process (e.g., submission, rate, quote, refer, bind, ePay, issue, renew, endorse, cancel, reinstate, audit etc.). Solartis supports all P&C personal and commercial admitted and non-admitted lines of business. customers access microservices through Solartis provided screens or via API calls from their tech platforms.

Since every company has unique policy administration needs, PAS make policy life cycle microservices available in three different delivery options are,

2.3.1 Function as a Service (FaaS)

a. Solartis Insure is an API-centric policy administration platform built from the ground up on microservice architecture. The platform is deployed in a cloud

infrastructure and supports the full policy life cycle process (e.g., submission, rate, quote, refer, bind, ePay, issue, renew, endorse, cancel, reinstate, audit etc.).

b. Solartis supports all P&C personal and commercial admitted and non-admitted lines of business. customers access microservices through Solartis provided screens or via API calls from their tech platforms.

2.3.2 Platform-as-a-Services (PaaS)

c. This “Solartis Inside” option provides stateful sales and policy life cycle microservices (accessed through APIs) without the user interface. This option allows y IT team to develop and design the user interface and business flow in-house.

d. sales and policy life cycle microservices are accessed via an XML or JSON request and the return response provides the requested transaction/information. This option allows IT team (or digital partners' IT team) to control the project. It reduces the development time and the cost and minimizes project risks as all policy admin functionality and rating content comes pretested and ready to go. Average time to Release Candidate /UAT (work and work) 75 days (about 2 and a half months).

2.3.3 Software-as-a-Service (SaaS)

This “total solution” option provides all the needed sales and policy life cycle capabilities (microservices) including the user interface, business flow, required integrations, and reporting database. This option is currently used in consumer portals, broker/agent systems, and full-blown insurance carrier underwriter systems. The average time to Release Candidate (UAT) is 90 days (about 3 months).

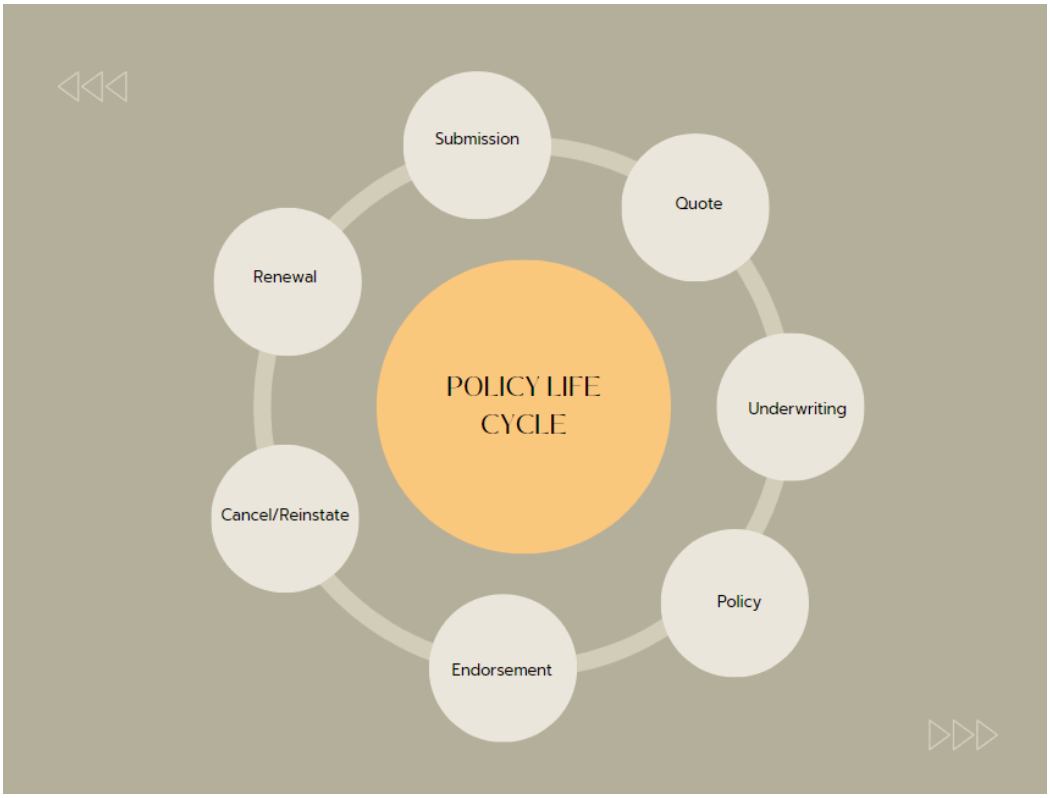

Figure 2.3.1 Policy Life Cycle

2.4 Builder

The ultimate toolkit for business users to create and manage Personal and Commercial P&C insurance products. All Solartis insurance products are developed and maintained in the Solartis Builder. This toolkit is built specifically for insurance, and just like Solartis Insure, with true cloud services and no legacy applications.

Easily configure and maintain personal or commercial proprietary insurance products. Solartis Builder streamlines insurance product management for users without technical expertise, including insurance product managers, business analysts, and actuarial and regulatory staff.

Solartis electronically consumes the ISO Commercial Lines content into platform so there is no need to reinvent the wheel. Simply add carrier rate and rule exceptions, add carrier proprietary documents and forms, then test and deploy. With Solartis Builder, business users can change,

test, and deploy into production within a single environment without any assistance from DevOps.

Solartis Builder Components & Capabilities Overview are

2.4.1 Create Insurance Products

a. New insurance products can be either configured from scratch or can be created from other product content or bureau-based content like ISO ERC.
b. Solartis digitally consumes all the ISO ERC countrywide & state files (and all its versions) into the Solartis platform. (i.e., States, coverages, rate tables, UI/UX pages, and their attributes, algorithms, forms selection rules, and stat codes). Solartis then uses the Solartis Builder to configure y carrier exceptions, eligibility, and referral processing rules as well as y carrier-specific policy documents and their associated forms. All initial release testing and deployment to production are performed in the Solartis Builder by Solartis.

2.4.2 Maintain Product(s)

An MGA / Carrier uses the Solartis Builder to manage all aspects of their insurance products; they can adopt or create new product versions, maintain states, coverages, rate tables, lookup tables, UI/UX pages and their attributes (metadata), all policy documents and their associated forms, as well as the processing rules and algorithms.

2.4.3 Test and Promote

Solartis has one environment for development, testing, and production. Changes are created, tested, and deployed into production using the Solartis Builder by business users without any assistance from DevOps.

2.4.5 Activity Log

Solartis Builder provides a full audit trial of all changes made to insurance products, as well as who performed the changes and when.

2.5 WORKING

1. **Requirement Analysis:**

   Requirement Analysis is the first step of the Software Testing Life Cycle (STLC). In this phase the quality assurance team understands the requirements like what is to be tested. If anything is missing or not understandable then the quality assurance team meets with the stakeholders to better understand the detailed knowledge of requirements.

2. **Test Planning:**

   Test Planning is the most efficient phase of the software testing life cycle where all testing plans are defined. In this phase manager of the testing, team calculates the estimated effort and cost for the testing work. This phase gets started once the requirement-gathering phase is completed.

3. **Test Case Development:**

   The test case development phase gets started once the test planning phase is completed. In this phase the testing team notes down the detailed test cases. The testing team also prepares the required test data for the testing. When the test cases are prepared then they are reviewed by the quality assurance team.

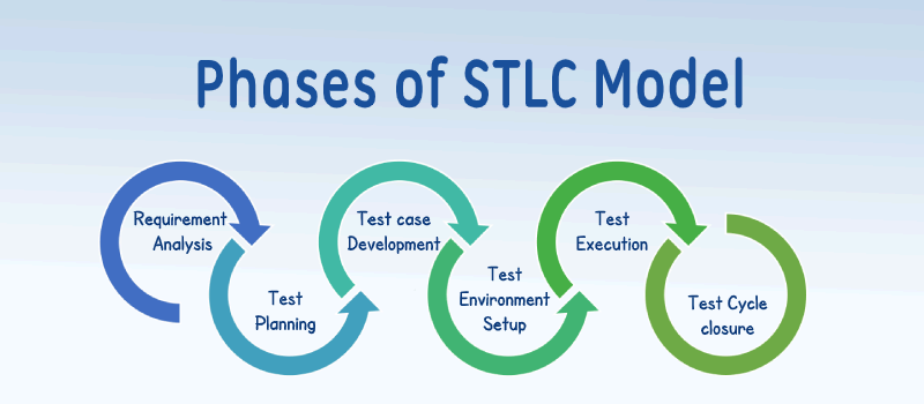

Figure 2.4.1 STLC

	A	B	C	D	E
1	**TestNo**	**Test Description**	**Test Case Steps**	**Test Data**	**Expected Result**
2	TC_001	To Check Whether the state attribute operates according to the specified requirements in safari browser	1. Open the safari browser and enter the policy creation URL 2. Click the state field. 3. Select the required state value.	State : CA or AK or VT	1. All applicable states should be present. 2. VT should not be present.
3	TC_002	To Check Whether the website page opens in 20 seconds	1. Open the policy creation page using the provided URL.	N/A	The policy creation page should be opened before 20 seconds
4	TC_003	To check whether all the attribute values are loaded in 20 seconds.	1. Open the policy creation page using the provided URL. 2. Select the required values for the fields.	State : AK or CA Sport: Cricket or Baseball Exposure: 100 or 200 Coverage : Liquor Liability Coverage or Hired Auto Coverage or both.	All the fields should be loaded before 20 seconds
5	TC_004	To check whether the time taken by the system to generate a policy and respond to user requests are less than 20 seconds under different load conditions	1. Open the policy creation page using the provided URL. 2. Select the required values for all the fields. 3. Proceed to generate the policy. 4. Policy document generation.	State : AK or CA Sport: Cricket or Baseball Exposure: 100 or 200 Coverage : Liquor Liability Coverage or Hired Auto Coverage	The policy document should be generated before 20 seconds.

Figure 2.4.2 Test Case Sample 1

Testcase No	Assumptions	Testcase Criteria	Testcase Steps	Expected Result	Actual Result	Testcase Result
TC_001	User is accesing the currency converter tool	Labels/colour coding/Fonts are as per user specification/requirements	1. Open the currency converter tool. 2. Check for the controls.	The controls should be as per the user specification/ design	The control are as per the user specification	PASS
TC_002	User is accesing the currency converter tool	Amount input only accepts the numeric values	1. Open the currency converter tool. 2. Enter the values other than number (special characters/space/alphabets)	The amount field should not accept the values other than numbers.	The amount field is not accepting the values other than numbers.	PASS
TC_003	User is accesing the currency converter tool	Check whether the From and To dropdown are prepopulated with currencies as needed in the specification document	1. Open the currency converter tool. 2. Select/open the dropdown for the from and To	The values should be populated in the dropdowns are as per the user specification	The values are populated in the dropdowns are as per the user specification	PASS
TC_004	User is accesing the currency converter tool	Check whether the "Add to the site "button works as per the expectation	1. Open the currency converter tool. 2. Click on the Add to site	Click on the add to site button should help user add this particular tool	Click on the add to site button is helping user add this	PASS

Figure 2.4.3 Test Case Sample 2

Testcase No	Assumptions	Testcase Criteria	Testcase Steps	Expected Result	Actual Result	Testcase Result
TC_001	School admission form is available to test	Check if the font, font style, colour, background,logo,images, text are as per the requirement	1. Open the school admission form. 2. Check for font, font style, colour, background,logo,images,text	The font, font style, colour, background,logo, images,text should be as per the requirement	The font, font style, colour, background,logo,images,text are as per the requirement	PASS
TC_002	School admission form is available to test	Check if the first name, last name are taking only alphebet values	1. Open the school admission form. 2. Enter the numeric/special charatcers in the text field	The system should flag an error		
TC_003	School admission form is available to test	Check if the dropdown has	1. Open the	The values in the dropdown are as		

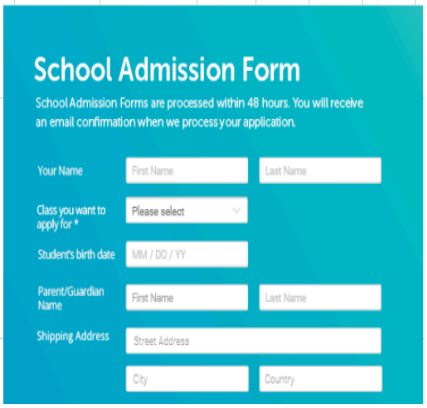

Figure 2.4.4 Test Case Sample 3

Testcase No	Assumptions	Testcase Criteria	Testcase Steps	Expected Result	Actual Result	Testcase Result
TC_001	Mobile app is available for the user to test	Check if the controls are as per the user specifications	1. Launch mobile app. 2. Check for the controls	The controls should be as per the user specification(Checking for the font, font style,colour, logo)	The controls are as per the user specification	PASS
TC_002	Mobile app is available for the user to test	Check if the first name accepts only alphabets	1. Launch mobile app. 2. Enter the values other than alphabets in the first name(numbers /special characters)	The user should be alerted to provide proper values in the first name	The user is alerted in case the value other than alphabets is added	PASS
		Check if the last name accepts only alphabets	1. Launch mobile app. 2. Enter the			

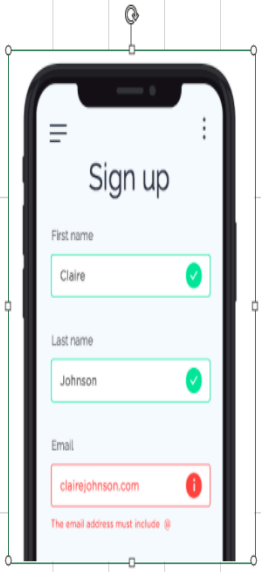

Figure 2.4.5 Test Case Sample 4

Testcase No	Assumptions	Testcase Criteria	Testcase Steps	Expected Result	TestData	Actual Result	Testcase Result
TC_001	Anti Virus software is available for user to test	Check if the controls on the screen are as per the user specification/requirement document	z	The logo/images/the font/font stylle/colour are as per the user specification.			
TC_002	Anti Virus software is available for user to test	Check if the dashboard is accesible if the dashboard link is clicked	1. Launch/open anti virus software. 2. Click on the Dashboard	The dashboard should be accesible by the end user.			
TC_003	Anti Virus software is available for user to test	Check if the Protection link is accesible if the Protection link is clicked	1. Launch/open antivirus software. 2. Click on protection	The protection page should open for the end user			
TC_004	Anti Virus software is available for user to test	Check if the privacy link is accesible if the privacy link is clicked	1. Launch/open antivirus software. 2. Click on privacy link	The privacy page should open for the end user.			
TC_005	Anti Virus software is available for user	Check if the Utilities link is accesible if the Utilities link is clicked	1. Launch/open antivirus software. 2. Click on utilites	The utilities page should open for the end user.			

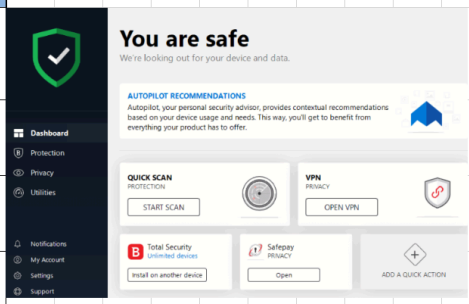

Figure 2.4.6 Test Case Sample 5

CHAPTER 3

SOURCE CODE

```
package dev.selenium.getting_started;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import java.time.Duration;

public class FirstScript {

    public static void main(String[] args) {

        WebDriver driver = new ChromeDriver();



driver.get("https://www.selenium.dev/selenium/web/web-form.html"
);

        driver.getTitle();


driver.manage().timeouts().implicitlyWait(Duration.ofMillis(500)
);

         WebElement textBox =
driver.findElement(By.name("my-text"));
```

```
        WebElement submitButton =
driver.findElement(By.cssSelector("button"));

         textBox.sendKeys("Selenium");

        submitButton.click();

         WebElement message =
driver.findElement(By.id("message"));

        message.getText();



        driver.quit();

    }
```

CHAPTER 4

CONCLUSION AND RECOMMENDATIONS

Efficient management of test cases is vital for ensuring software quality. Leveraging tools like JIRA and Bugzilla streamline tracking and resolving issues during testing. Transitioning from Excel to file-based test cases enhances organization and collaboration. Regularly review test cases to ensure alignment with project requirements and cover all scenarios. Continuously monitor and evaluate the test case management process to make necessary adjustments for improved efficiency and quality assurance.

CHAPTER 5

REFERENCES

Manual Testing: https://www.geeksforgeeks.org/software-testing-manual-testing/

Postman API: https://learning.postman.com/docs/introduction/overview/

Test Cases: https://www.javatpoint.com/test-case

Bugzilla: https://www.javatpoint.com/bugzilla

JIRA: https://www.atlassian.com/software/jira/guides/getting-started/basics#step-7-move-work-forward