1/1/2024

Luthando Lincoln Jayiya

LJ Insurance

C.I.M.S SPECIFICATION DOCUMENT

CAR INSURANCE MANAGEMENT SYSTEM

Table of Contents

[**Overview** 6](#_Toc162000261)

[**Goal** 6](#_Toc162000262)

[**Description** 6](#_Toc162000263)

[**Company Description:** 6](#_Toc162000264)

[**What We Are Building:** 6](#_Toc162000265)

[**Validation Strategies:** 6](#_Toc162000266)

[**Users and Usage:** 6](#_Toc162000267)

[**UML DIAGRAMS** 8](#_Toc162000268)

[***USER STORIES*** 8](#_Toc162000269)

[***USE CASE*** 9](#_Toc162000270)

[9](#_Toc162000271)

[***ENTITY RELATION DIAGRAM*** 10](#_Toc162000272)

[10](#_Toc162000273)

[***CLASS DIAGRAM*** 11](#_Toc162000274)

[11](#_Toc162000275)

# **Overview**

## **Goal**

The goal of the LJ Insurance-developed Car Insurance Management System (CIMS) is to streamline claims handling procedures while offering clients complete insurance solutions. The system will provide a range of auto insurance policy options, such as Third Party, Fully Comprehensive, and Third Party, Fire and Theft. It will also have strong claims management capabilities that will enable drivers, body shops, surveyors, and insurers to communicate in real time.

## **Description**

LJ Insurance will be able to handle insurance policies, claims, risk analysis, and driver contact from a single, centralized platform thanks to the Car Insurance Management System. It will include a user-friendly interface for simple navigation and effective data administration.

## **Company Description:**

LJ Insurance is dedicated to providing its clients with trustworthy and cutting-edge insurance options. LJ Insurance works to improve services and optimize operations by utilizing technology, with an emphasis on risk mitigation and client happiness.

## **What We Are Building:**

To help LJ Insurance with policy issuance, claims processing, risk analysis, and driver communication, we are creating a full car insurance management system called CIMS. This system will provide a range of vehicle insurance coverage options and speed up the claims handling procedure to guarantee effective correspondence and resolution.

## **Validation Strategies:**

To validate the system throughout the development phase, extensive testing, including system, integration, and unit testing, will be carried out.

People or different representatives from LJ Insurance will participate in user acceptance testing, to make sure the system satisfies their needs and expectations.

## **Users and Usage:**

**Insurance Agents** who are in charge of handling client inquiries and issuing insurance coverage.

**Adjusters for claims** Will manage the processing of claims, evaluate damages, and work with repair businesses.

**Drivers:** Will use the system to buy insurance, file accident reports, and monitor the status of open claims.

**Surveyors:** Will use the system to valuate vehicles and submit reports to insurance companies.

**Administrators:** Will create reports, maintain user accounts, and supervise system operations.

Required Features:

*Includes but not limited to, the ideal features will perform CRUD operations:*

|  |  |
| --- | --- |
| Feature | Description |
| 1. Policy management | * Create, view, and modify insurance policies. * Offer different types of policies (Fully Comprehensive, Third Party, Third Party, Fire, and Theft). * Store policy details, including coverage, premiums, and expiration dates |
| 1. Risk Analysis | * Analyse accident causes and driver risk profiles. * Identify high-risk drivers and vehicles. * Implement measures to increase risk awareness and driver training |
| 1. Claims management | * Report accidents and file claims online. * Track claim status and updates in real-time. * Facilitate communication between drivers, repair shops, surveyors, and insurers |
| 1. Communication Tools: | * Provide chat or messaging features for real-time communication. * Send notifications and updates to relevant stakeholders. * Facilitate seamless communication throughout the claims process. |
|  |  |

# **UML DIAGRAMS**

## ***USER STORIES***

*As an Insurance agent*

*I Want to create new insurance policies*

*So that I can provide coverage options to customers efficiently and accurately*

*As a customer  
  
I want to view different types of insurance policies available (Fully Comprehensive, Third Party, Third Party, Fire, and Theft)*

*So, I can choose the one that best suits my needs and budget*

*As a driver*

*I want to report accidents and file claims online*

*So, I can initiate the claims process quickly and conveniently.*

*As a surveyor*

*I want to analyse accident causes and driver risk profiles*

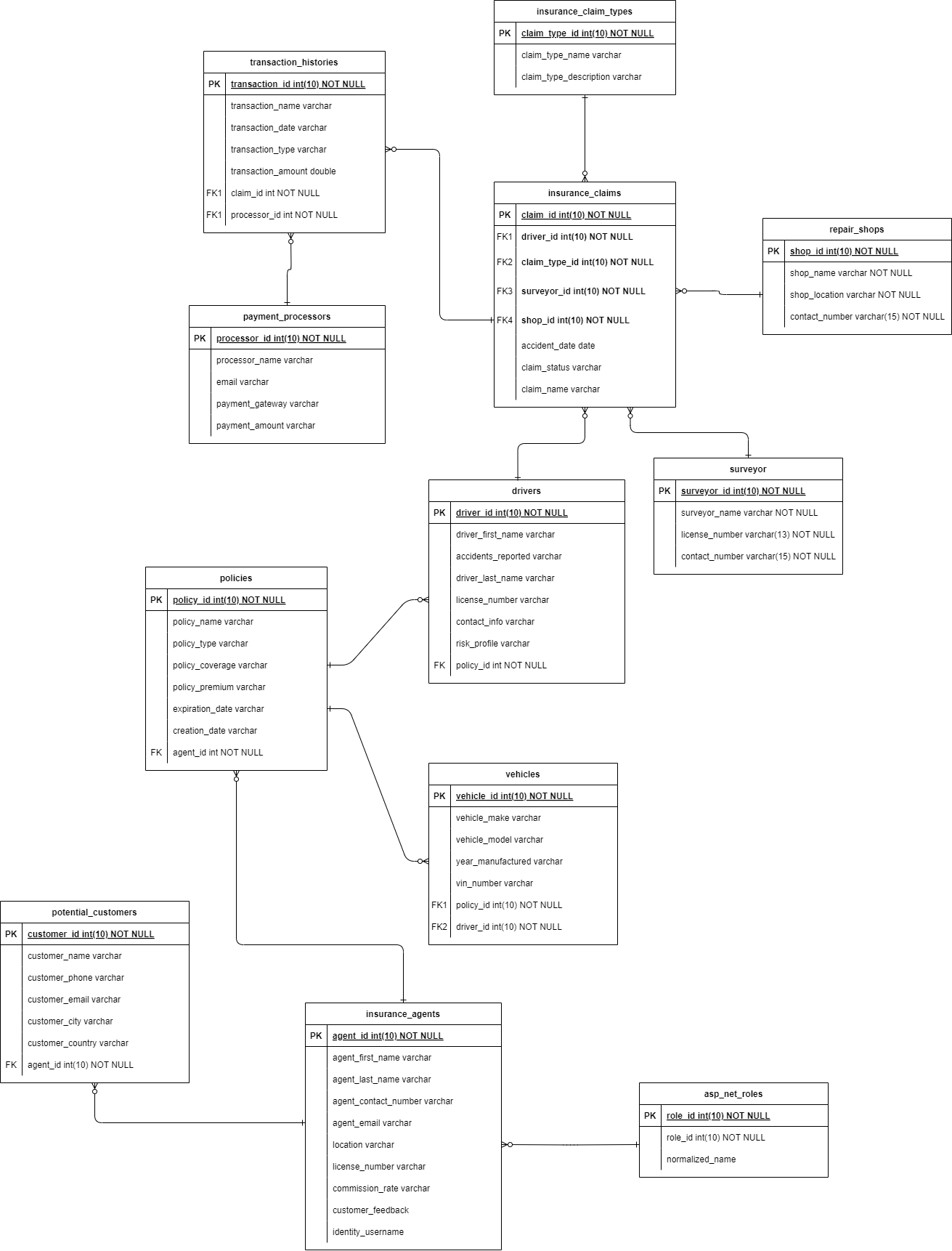
*So, I can identify potential areas of improvement and implement proactive measures to mitigate risks.*

## ***USE CASE***

*Below shows the use case diagram*

## 

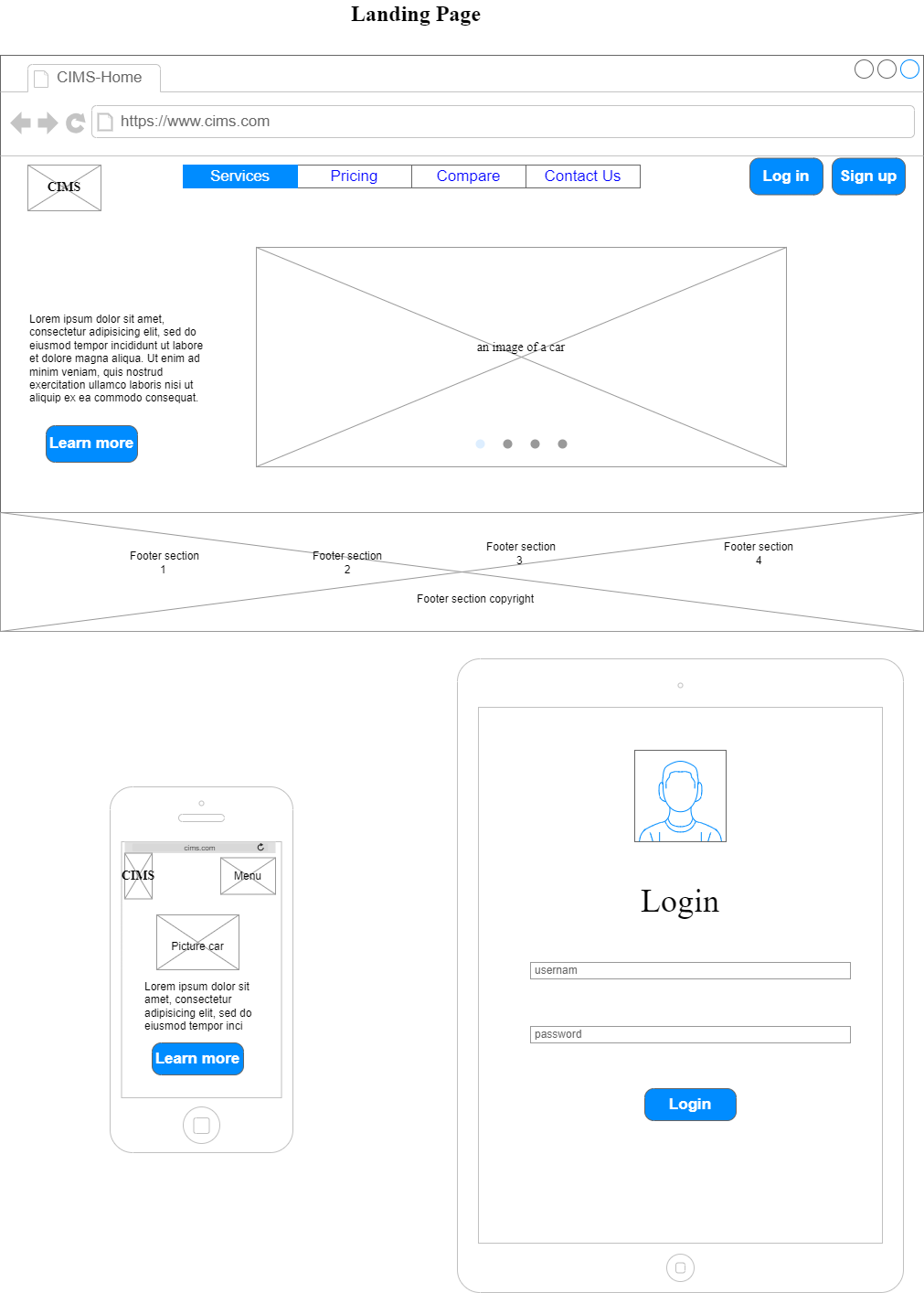
## ***ENTITY RELATION DIAGRAM***



## ***CLASS DIAGRAM***

## 

## **Mockups**



**Leason learned**

On Week 3 Web Apps I have learnt how to create website using HTML, CSS and Javascript. I have learnt how to connect the front-end to the backend using axios and how to fetch and put data using axios. I have learnt about UI And UX design as well as user experience

On Week 2 Rest Api’s i have learnt how to create rest API’S and i have learnt about HTTP and http protocols, i have learnt about rest api crud operations the three A’s (Audit,Authenticate,Authorize), i have learnt how to test rest api’s using in memory database, i have learnt about swagger and postman to test api endpoints.

HTTP Protocol And Microservices  
In a RESTful API, HTTP is the protocol used for communication between clients (browsers etc) and servers. HTTP protocols commonly used in RESTful APIs are:

GET: Used to retrieve a representation of a resource from the server. It should not modify the server's state.

POST: Used to create a new resource on the server.

PUT: Used to update an existing resource on the server.

PATCH: Like PUT but used for partial updates.

DELETE: Used to remove a resource from the server.

HEAD: Like GET but returns only the headers of the response, without the body.

OPTIONS: Returns the HTTP methods that the server supports for a specific URL. It's often used for CORS (Cross-Origin Resource Sharing) preflight requests. These HTTP protocols are mapped to CRUD (Create, Read, Update, Delete) operations in a RESTful API.

Microservices is an architectural approach where an application is composed of small, independently deployable services, each focused on a specific business capability

On Week1 of Data Access I have learnt the difference between ADO.NET Disconect Model and Connected model and SQL Injection and how to prevent it, I've also learnt to use stored procedures with ADO Disconnect. I’ve learnt entity framework and how to query the database using entity framework. I’ve learnt effort and fluent assertion testing

On week 2 of OOD, I have about different types of data types, I have learned about abstract classes and how to use them, I have learned about 4 pillars, solid principles, generics, collections and memory handling. I have applied what I learned to my project (Car Insurance Management System)

On week 3 of OOD, I have learned code documentation clean coding and Test-Driven Development. I have learned how to write test before writing code then gradually add the code to make the tests pass, I have learned to write Mock tests using MOQ and Unit tests using NUnit. By writing tests first i was able iteratively develop features, detect problems early and maintain high level confidence in my code base.

Technical Architecture

FDM Git Lab Project Name. LLJ\_CarInsuranceMS\_WK2.NETREST.

Git and Gitlab

Logging framework: log4net ; default path (bin/degub)

Design Patterns: MVC pattern, Repository patttern, Dependency injection,RESTful Design Principles

## Sample data for User Login

UserAdmin

{

"userName": "HighLordCIMSAdmin",

"password": "@#Secret1234"

}

InsuranceAgent{

"userName": "AgentNeil",

"password": "AgentNeil@#1234"

}

Surveyor{

"userName": "SLani",

"password": "LaniS4321@#!"

}

Driver{

"userName": "DSeth",

"password": "@123\*#DSeth"

}

PotentialCustomer{

"userName": "CJohn",

"password": "CJohn@#1987"

}

RepairShop{

"userName": "TJRepairs",

"password": "TJauto@#1987"

}

## Sample data for User Registration

Register PotentialCustomer

{

"userName": "JohnC",

"password": "WickJohn@4546!",

"email": "john@gmail.com",

"fullName": "John",

"role": "PotentialCustomer",

"phoneNumber": "025-215-7894",

"licenseNumber": "CA8790"

}

Register Administrator

{

"userName": "NateAdmin",

"password": "NateA@a#F!Yr]ZyB6",

"email": "nateA@gmail.com",

"fullName": "Nate",

"role": "Administrator",

"phoneNumber": "025-215-5587",

"licenseNumber": "GP45657"

}

Register Surveyor

{

"userName": "TinaS",

"password": "pg4NB'9!66Nk",

"email": "tinaS@gmail.com",

"fullName": "Tina Jaxa",

"role": "ClaimSurveyor",

"phoneNumber": "025-455-7896",

"licenseNumber": "EC3621"

}

## **List of REST API endpoints**

|  |  |  |
| --- | --- | --- |
| ***Endpoint: Potential Customer*** | ***Description*** | ***Role*** |
| https://localhost:7274/api/Customer | Gets a list of customers | Potential Customer |
| https://localhost:7274/api/Customers/PostCustomers | Adds a new customer | Potential Customer |
| https://localhost:7274/api/Customers/5 | Gets a customer by an ID | Potential Customer |
| https://localhost:7274/api/Customers/5 | Updates an existing customer | Potential Customer |
| https://localhost:7274/api/Customers/5 | Deletes an existing customer | Potential Customer |

|  |  |  |
| --- | --- | --- |
| ***Endpoint: Insurance Agents*** | ***Description*** | ***Role*** |
| https://localhost:7274/api/InsuranceAgents | Gets a list of Insurance Agents | InsuranceAgents |
| https://localhost:7274/api/InsuranceAgents/PostInsuranceAgent | Adds a new Insurance Agent | InsuranceAgents |
| https://localhost:7274/api/InsuranceAgents /5 | Gets an Insurance Agent by an ID | InsuranceAgents |
| https://localhost:7274/api/InsuranceAgents/5 | Updates an Insurance Agent | InsuranceAgents |
| https://localhost:7274/api/InsuranceAgents/5 | Deletes an existing Insurance Agent | InsuranceAgents |

|  |  |  |
| --- | --- | --- |
| ***Endpoint: InsuranceClaims*** | ***Description*** | ***Role*** |
| https://localhost:7274/api/InsuranceClaims | Gets a list of Insurance Claims | Administrator |
| https://localhost:7274/api/InsuranceClaims/PostInsuranceClaim | Adds a new Insurance Claim | Administrator |
| https://localhost:7274/api/InsuranceClaims/5 | Gets an Insurance Claim by an ID | Administrator |
| https://localhost:7274/api/InsuranceClaims/5 | Updates an Insurance Claim | Administrator |
| https://localhost:7274/api/InsuranceClaims/5 | Deletes an existing Insurance Claim | Administrator |
| https://localhost:7274/api/InsuranceClaims/InsuranceClaimsDrivers | Gets a list of insurance claims with drivers | Administrator |
| https://localhost:7274/api/InsuranceClaims/InsuranceClaimsDriver/5 | Gets an insurance claim with a driver ID | Administrator |
| https://localhost:7274/api/InsuranceClaims/InsuranceClaimsSurveyors | Gets a list of Insurance claims with Surveyors | Administrator |
| https://localhost:7274/api/InsuranceClaims/InsuranceClaimsSurveyor/1 | Gets a list of Insurance claims with Surveyor ID | Administrator |
| https://localhost:7274/api/InsuranceClaims/InsuranceClaimsRepairShops | Gets a list of Insurance claims with Repair Shops | Administrator |
| https://localhost:7274/api/InsuranceClaims/InsuranceClaimsRepairShop | Gets a list of Insurance claims with Repair Shop ID | Administrator |

|  |  |  |
| --- | --- | --- |
| ***Endpoint: InsuranceClaimType*** | ***Description*** | ***Role*** |
| https://localhost:7274/api/InsuranceClaimTypes | Gets a list of Insurance Claim Types | Administrator |
| https://localhost:7274/api/InsuranceClaimTypes/PostInsuranceClaimType | Adds a new Insurance Claim Type | Administrator |
| https://localhost:7274/api/ InsuranceClaimTypes/5 | Gets an Insurance Claim Type by an ID | Administrator |
| https://localhost:7274/api/InsuranceClaimTypes/5 | Updates an Insurance Claim Type | Administrator |
| https://localhost:7274/api/ InsuranceClaimTypes/5 | Deletes an existing Insurance Claim Type | Administrator |

|  |  |  |
| --- | --- | --- |
| ***Endpoint: Policies*** | ***Description*** | ***Role*** |
| https://localhost:7274/api/Policies | Gets a list of Policies | PotentialCustomer |
| https://localhost:7274/api/Policies/1 | Gets a Policy by an ID | PotentialCustomer |

# **DESIGN CONSIDERATIONS BACKLOG**

## *USER EXPERIENCE CONSIDERATIONS*

Ease of use: to make sure that the interface is easy to use and navigate.

Consistency: Maintain consistent design layout, colours and design elements

Feedback it must provide clear feedback to users about their actions and the system’s status

## *Interaction Design Laws*

Hick’s Law: to keep the number of options manageable to reduce decision time for users.

Miller’s Law: Limit the amount of information presented at once to prevent overwhelm and aid comprehension

Tesler’s Law: it must not be complex.

## *Site Maps*

Hierarchical Site Map: to organize content in a deep hierarchy for this system that has multiple levels

## *Patterns Of Visualization*

Will consider the F and the Z pattern so that users can easily scan and find what they want quickly

# **KANBAN PRODUCT BACKLOG**

|  |  |  |  |
| --- | --- | --- | --- |
| **.NET Project Backlog** | | | |
| **Sprint** | **Product backlog items** | **Priority** | **Status** |
| 1 | OOD Week1 – Research Project | 1 | Completed |
| 2 | OOD Week2 – Code-based Solo Project | 2 | Completed |
| 3 | OOD Week3 – TDD Solo Project | 3 | Completed |
| 4 | 1 .NET Data Access | 4 | Completed |
| 5 | 2 .NET REST API | 5 | Completed |
| 6 | 3 .NET Web Apps | 6 | Completed |
| 7 | 4 .NET ASP.NET | 7 | In Progress |
| 8 | 5 .NET Angular/React | 8 | Not Started |

**SPRINT 7 BACKLOG – .NET ASP.NET**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ID* | *USER* | *TASKS* | *PRIORITY* | *STATUS* |
| 1 | Luthando | Create Landing page | 1 | Done |
| 2 | Luthando | Create a Login page | 2 | Done |
| 3 | Luthando | Create a Register page | 3 | Done |
| 3 | Luthando | Create an Admin page | 4 | Done |
| 4 | Luthando | Configure Log4net | 5 | Done |
| 5 | Luthando | Create Controllers for CRUD operations for admin | 6 | In progress |
| 6 | Luthando | Update Requirement specification documentation | 7 | In progress |
| 7 | Luthando | Add all necessary packages | 8 | Done |
| 8 | Luthando | Do CRUD operation for customer | 9 | Not Started |
| 9 | Luthando | Do CRUD operations for Insurance Agent | 10 | Not started |
| 10 | Luthando | Add xUnit project | 11 | Not started |
| 11 | Luthando | Do xunit Tests for my controllers | 12 | Not Started |

**SPRINT 6 BACKLOG – WebApp**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ID* | *USER* | *TASKS* | *PRIORITY* | *STATUS* |
| 1 | Luthando | Create a HTML Index file | 1 | Done |
| 2 | Luthando | Create a CSS for index file | 2 | Done |
| 3 | Luthando | Create a javascript file | 3 | Done |
| 3 | Luthando | Connect front-end with backend Rest API | 4 | Done |
| 4 | Luthando | Configure axios | 5 | Done |
| 5 | Luthando | Create Landing page | 6 | Done |
| 6 | Luthando | Create Login page | 7 | Done |
| 7 | Luthando | Fetch from the API using axios | 8 | Done |
| 8 | Luthando | Create a register page | 9 | Done |
| 9 | Luthando | Connect register with backend using axios | 10 | Done |
| 10 | Luthando | Create customer page and fetch data from API | 11 | Done |
| 11 | Luthando | Add Coming soon page for upcoming features | 12 | Done |

**SPRINT 5 BACKLOG – .NET REST API**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ID* | *USER* | *TASKS* | *PRIORITY* | *STATUS* |
| 1 | Luthando | Create REST API App | 1 | Done |
| 2 | Luthando | Add all necessary packages | 2 | Done |
| 3 | Luthando | Configure Identity Management | 3 | Done |
| 3 | Luthando | Add log4net and configure log4net.config file | 4 | Done |
| 4 | Luthando | Add Additional Roles | 5 | Done |
| 5 | Luthando | Configure JSON Web Tokens | 6 | Done |
| 6 | Luthando | Create Identity Database | 7 | Done |
| 7 | Luthando | Configure the REST API project  Add end point for user Registration, user Login, and user Profile | 8 | Done |
| 8 | Luthando | Configure the REST API project to do  API Testing using Swagger UI  API Testing using Postman | 9 | Done |
| 9 | Luthando | Use logging framework to add Accounting/Auditing | 10 | Done |
| 10 | Luthando | Add end point for CRUD operations | 11 | Done |
| 11 | Luthando | Configure project to use role-base Authorization | 12 | Done |

**SPRINT 4 BACKLOG – DATA ACCESS WEEK1 ENTITY FRAMEWORK**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ID* | *USER* | *TASKS* | *PRIORITY* | *STATUS* |
| 1 | Luthando | Write Effort Tests for customer repository | 1 | Done |
| 2 | Luthando | Write MOQ Tests for customer repository | 3 | Done |
| 3 | Luthando | Write Fluent Tests for customer repository | 2 | Done |
| 4 | Luthando | Write Effort Tests for Insurance Agent repository | 4 | Done |
| 5 | Luthando | Write Fluent Tests for Insurance Agent repository | 5 | Done |
| 6 | Luthando | Write Effort Tests for Payment processor repository | 6 | Done |
| 7 | Luthando | Write Fluent Tests for Payment processor repository | 7 | Done |
| 8 | Luthando | Write Fluent Tests for Repair Shop repository | 1 | Done |
| 9 | Luthando | Write Effort Tests for Repair Shop repository | 2 | Done |
| 10 | Luthando | Write Effort Tests for Surveyor repository | 3 | Done |
| 11 | Luthando | Write Fluent Tests for Surveyor repository | 1 | Done |
| 12 | Luthando | Use ADO.NET Connect to perform CRUD operations | 1 | Done |
| 13 | Luthando | Use ADO.NET Disconnect to perform CRUD operations | 1 | Done |
| 14 | Luthando | Use ADO.NET Disconnect with stored procedures to perform CRUD operations | 1 | Done |
| 15 | Luthando | Create data entities and migrations using Entity framework | 1 | Done |
| 16 | Luthando | Create Repo Crud project with entity framework | 1 | Done |
| 17 | Luthando | Update entity relational diagram | 1 | Done |
| 18 | Luthando | Create a script file | 1 | Done |

**ODD1 WEEK1 SOLID CONSOLE APP – SPRINT BACKLOG**

**SPRINT 2 BACKLOG – 00D WEEK2 SOLID CONSOLE APP**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ID* | *USER* | *TASKS* | *PRIORITY* | *STATUS* |
| 1 | Driver | File a claim | 1 | done |
| 2 | Driver | Track claim status | 2 | done |
| 3 | Customer | See all policies | 2 | done |
| 4 | Customer | Apply for a policy | 2 | done |
| 5 | Surveyor | Get a list of claims | 2 | done |
| 6 | Surveyor | Do an investigation on claims made | 2 | done |
| 7 | Surveyor | Update claim status (and the nature of the accident) | 2 | done |
| 8 | Insurance Agent | Get All customer applications | 1 | done |
| 9 | Insurance Agent | Approve or reject applications (if approve add to driver list) | 2 | done |
| 10 | Insurance Agent | Get list of claims | 1 | done |
| 11 | Insurance Agent | Approve claims (send it to surveyor) | 2 | done |
| 11 | Insurance Agent | Get list of repair shops showing details | 2 | done |
| 12 | Insurance Agent | Get a list of invoices from repair shop | 2 | done |
| 13 | Insurance Agent | Get repair status then update claim status | 2 | done |
| 14 | Repair Shop | Get list of claims and details | 2 | done |
| 15 | Repair Shop | Send invoice | 2 | done |
| 16 | Repair Shop | Do repair update status | 2 | done |

**SPRINT 3 BACKLOG – 00D WEEK3 TDD CONSOLE APP**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ID* | *USER* | *TASKS* | *PRIORITY* | *STATUS* |
| 1 | Insurance Agent | Add a claim Test | 1 | done |
| 2 | Insurance Agent | Get all claims test | 1 | done |
| 3 | Insurance Agent | Remove claim test | 1 | done |
| 4 | Insurance Agent | update claim test | 1 | done |
| 5 | Insurance Agent | Get a claim by id test | 1 | done |
| 6 | Insurance Agent | Add a claim repository | 2 | done |
| 7 | Insurance Agent | Add unit tests | 1 | done |
| 8 | Insurance Agent | Add iclaim interface | 3 | done |
| 9 | Insurance Agent | Add claim queue | 3 | done |
| 10 | Insurance Agent | Icustomer interface | 1 | done |
| 11 | Insurance Agent | Add potential customer and its repository | 4 | done |
| 11 | Insurance Agent | Add a policy and policy repository | 4 | done |
| 12 | Insurance Agent | Add CRUD methods to claim repository | 2 | done |
| 13 | Insurance Agent | Add PolicyRepoImplementation | 5 | done |