**Software Requirements**

**Specification**

**for**

**PetroPulse**

**Petrol Station**

**Management System**

**Prepared by**

***Anas Bin Rashid 22I-0907***

***Adan Malik 22I-1000***

***Hasnain Akhtar 22I-1241***

**April 20, 2025**

# **Product Backlog/User Stories**

| **ID** | **User Story** | **Acceptance Criteria** | **Estimation** | **Priority** |
| --- | --- | --- | --- | --- |
| 1 | **User Registration** - As a new user, I want to register on the system so that I can access PetroPulse services. | **Given**: A new user navigates to the registration page. **When**: They provide valid details (name, email, phone, password). **Then**: The system successfully creates an account and sends a confirmation email. | 3 | High |
| 2 | **User Login** - As a registered user, I want to log into the system so that I can access my dashboard. | **Given**: A user is on the login page. **When**: They enter valid credentials. **Then**: The system grants access to their dashboard. | 2 | High |
| 3 | **Password Reset** - As a user, I want to reset my password so that I can regain access to my account if I forget it. | **Given**: A user is on the login page. **When**: They click on "Forgot Password" and enter their email. **Then**: The system sends a password reset link to their email. | 3 | Medium |
| 4 | **Add Fuel Station** - As an owner, I want to add new fuel stations so that I can manage multiple stations in the system. | **Given**: An owner is logged in. **When**: They enter station details and submit the form. **Then**: The system registers the new station and updates the list. | 3 | High |
| 5 | **View Fuel Stations** - As an owner, I want to view a list of my fuel stations so that I can manage their operations efficiently. | **Given**: The owner is on the dashboard. **When**: They navigate to the "Manage Stations" section. **Then**: The system displays a list of owned stations. | 2 | High |
| 6 | **Update Fuel Station Details** - As an owner, I want to update my fuel station details so that I can keep the information accurate. | **Given**: The owner is on the "Manage Stations" page. **When**: They update station details and save changes. **Then**: The system reflects the updated details. | 3 | Medium |
| 7 | **Remove Fuel Station** - As an owner, I want to remove a fuel station so that I can stop managing a closed or sold station. | **Given**: The owner is on the "Manage Stations" page. **When**: They click delete and confirm the action. **Then**: The system removes the station from the database. | 3 | Medium |
| 8 | **Register Workers** - As a station owner, I want to register workers so that they can manage station activities. | **Given**: The owner is on the "Manage Workers" page. **When**: They enter the worker's details and submit. **Then**: The system registers the worker. | 3 | High |
| 9 | **Assign Workers to Stations** - As an owner, I want to assign workers to specific stations so that they can operate in their designated locations. | **Given**: An owner is on the worker management page. **When**: They select a worker and assign a station. **Then**: The system updates the worker's assigned station. | 3 | Medium |
| 10 | **Worker Login** - As a worker, I want to log into the system so that I can manage station operations. | **Given**: A worker is on the login page. **When**: They enter valid credentials. **Then**: The system grants access to the worker dashboard. | 2 | High |
| 11 | **Record Fuel Sales** - As a worker, I want to record fuel sales so that the system tracks all transactions. | **Given**: A worker is logged in. **When**: They enter fuel sale details. **Then**: The system stores the transaction. | 3 | High |
| 12 | **Generate Sales Reports** - As an owner, I want to generate sales reports so that I can analyze revenue and performance. | **Given**: The owner is on the reports page. **When**: They select a date range and generate a report. **Then**: The system displays the sales data. | 3 | High |
| 13 | **Track Fuel Inventory** - As a station owner, I want to track fuel levels so that I can ensure continuous supply. | **Given**: The owner is on the dashboard. **When**: They check the station inventory. **Then**: The system displays real-time fuel levels. | 4 | High |
| 14 | **Alert for Low Fuel Levels** - As a station owner, I want to receive alerts for low fuel levels so that I can order refills in time. | **Given**: A station's fuel level drops below a threshold. **When**: The system detects the low level. **Then**: It sends an alert to the owner. | 3 | High |
| 15 | **Order Fuel Supply** - As an owner, I want to order fuel refills so that my stations never run out of stock. | **Given**: The owner is on the fuel supply page. **When**: They place an order for fuel. **Then**: The system processes and confirms the order. | 4 | High |
| 16 | **View Loyalty Points** - As a customer, I want to view my loyalty points so that I can track my rewards. | **Given**: A customer is logged in. **When**: They navigate to the "Loyalty Points" page. **Then**: The system displays their current balance. | 2 | Medium |
| 17 | **Redeem Loyalty Points** - As a customer, I want to redeem my loyalty points so that I can get fuel discounts. | **Given**: A customer has sufficient points. **When**: They redeem them at checkout. **Then**: The system applies the discount. | 3 | Medium |
| 18 | **Customer Fuel Purchase** - As a customer, I want to purchase fuel from the station so that I can refuel my vehicle. | **Given**: A customer is at a station. **When**: They choose the fuel type, quantity, and payment method. **Then**: The system processes the payment and updates inventory. | 4 | High |
| 19 | **Manage Pricing for Fuel Types** - As a station owner, I want to update fuel prices so that I can adjust them based on market conditions. | **Given**: The owner is on the pricing page. **When**: They update the price of a fuel type. **Then**: The system reflects the new pricing in real time. | 3 | High |
| 20 | **Manage Maintenance Requests** - As a station owner, I want to log maintenance requests so that I can ensure station upkeep. | **Given**: A station has an issue. **When**: The owner submits a maintenance request. **Then**: The system records and tracks the request. | 3 | Medium |
| 21 | **View Transaction History -** As a customer, I want to view my transaction history, so that I can track my fuel purchases. | **Given:** A customer is logged in, **When:** They navigate to the transactions section, **Then:** The system displays all past fuel purchases. | 2 | Medium |
| 22 | **Payment via Cash -** As a customer, I want to pay for fuel using cash, so that I can complete my purchase without a card. | **Given:** A customer selects fuel, **When:** They choose cash as the payment method, **Then:** The system registers the payment and updates records. | 3 | High |
| 23 | **Payment via Card -** As a customer, I want to pay for fuel using my debit/credit card, so that I can make transactions easily. | **Given:** A customer selects fuel, **When:** They choose card as the payment method, **Then:** The system processes the payment and updates records. | 3 | High |
| 24 | **Manage Supplier Information -** As an owner, I want to manage my fuel suppliers, So that I can track orders and deliveries. | **Given:** The owner is on the supplier management page, **When:** They add, edit, or remove a supplier, **Then:** The system updates the supplier database. | 3 | Medium |
| 25 | **Generate Maintenance Reports -** As an owner, I want to track station maintenance, so that I can schedule repairs and ensure smooth operations. | **Given:** The owner is on the reports page, **When:** They request a maintenance report, **Then:** The system provides a list of past and upcoming maintenance tasks. | 3 | Medium |
| 26 | **Employee Attendance Tracking -** As a station owner, I want to track employee attendance, so that I can monitor working hours. | **Given:** A worker logs into the system, **When:** They mark attendance for the shift, **Then:** The system records their working hours. | 3 | Medium |
| 27 | **Secure Access -** As a user, I want to have secure access to system configurations, so that I can manage system settings safely. | **Given:** The user is on the login page, **When:** They enter valid admin credentials, **Then:** The system grants access to the admin panel. | 3 | High |
| 28 | **View Worker Profiles -** As an owner, I want to view worker profiles, so I can manage their information and roles. | **Given:** The owner is on the "Manage Workers" page. **When:** They select a worker profile. **Then:** The system displays the worker's details. | 2 | Medium |

# **Team Member Roles**

***Product Owner*** *– Hasnain Akhtar*

**Responsibilities**:

* Define and prioritize user stories for the Petroleum Management System.
* Ensure that features such as fuel request handling, inventory tracking, and payment processing align with stakeholder needs.
* Communicate business requirements clearly to the development team.
* Accept or reject completed user stories based on their adherence to acceptance criteria.
* Continuously refine the backlog based on feedback from stakeholders and the development team.

***Scrum Master*** *– Anas Bin Rashid*

**Responsibilities***:*

* Facilitate Agile ceremonies such as Sprint Planning, Daily Standups, Sprint Review, and Retrospective.
* Identify and remove blockers affecting the development of features like shift management, employee check-in, and real-time alerts.
* Ensure smooth collaboration between Hasnain (Product Owner) and Adan (Developer).
* Track sprint progress and ensure adherence to Agile principles.
* Guide the team in improving workflow efficiency and maintaining high productivity.

***Scrum Team*** *– Adan Malik, Anas Bin Rashid, Hasnain Akhtar*

**Responsibilities**:

* Develop and implement features such as fuel request processing, analytics dashboard, and customer profile management.
* Ensure code quality, security, and efficiency while integrating various modules.
* Conduct unit testing and debugging to ensure smooth system functionality.
* Collaborate with Hasnain (Product Owner) to clarify feature requirements and address feedback.
* Participate in sprint planning and daily standups to track development progress.

# **Sprint Details**

Total number of Sprints are 3, and their details are as follows.

## **Sprint 1**

**1. Sprint Overview**

* Objective : The goal of Sprint 1 was to lay the foundation for the PetroPulse system by implementing core user authentication features and ensuring secure access for all user roles.
* Scope of Work :
  + User registration and login functionality.
  + Password reset feature.
  + Worker login for managing station operations.
  + Secure access control for admin configurations.
* Duration : 2 weeks

**2. Sprint Backlog**

**Modules Worked On**

1. User Authentication (Owner, Worker, Customer roles)
2. Secure Access Control

**User Stories and Sub-User Stories**

**User Story 1 – User Registration**

Title : User Registration  
As a new user , I want to register on the system so that I can access PetroPulse services.  
Context / Note : This story ensures that new users can create accounts with valid details (name, email, phone, password).  
This story is related to Sprint #1.

Task :

* Create registration form.
* Validate input fields before submission.
* Store user data in the database.
* Send confirmation email upon successful registration.

Subtask List :

* [Create registration form UI]
* [Validate input fields (email, password, phone)]
* [Store user data in MongoDB]
* [Send confirmation email via SMTP]

Associated User Stories :

* User Story 2 : User Login
* Rule : A user cannot register with an existing email.
* ID : US-01

**User Story 2 – User Login**

Title : User Login  
As a registered user , I want to log into the system so that I can access my dashboard.  
Context / Note : This story ensures that registered users can securely log in using their credentials.  
This story is related to Sprint #1.

Task :

* Create login form.
* Authenticate credentials against the database.
* Redirect authenticated users to their respective dashboards.

Subtask List :

* [Create login form UI]
* [Authenticate credentials using bcrypt]
* [Redirect users to dashboard based on role]

Associated User Stories :

* User Story 3 : Password Reset
* Rule : Invalid credentials must display an error message.
* ID : US-02

**User Story 3 – Password Reset**

Title : Password Reset  
As a user , I want to reset my password so that I can regain access to my account if I forget it.  
Context / Note : This story provides a "Forgot Password" option for users to reset their passwords via email.  
This story is related to Sprint #1.

Task :

* Add "Forgot Password" option on the login page.
* Implement email verification for password reset requests.
* Allow users to set a new password using the reset link.

Subtask List :

* [Add "Forgot Password" button]
* [Send password reset link via email]
* [Allow users to set a new password]

Associated User Stories :

* User Story 2 : User Login
* Rule : Password reset links must expire after 24 hours.
* ID : US-03

**User Story 10 – Worker Login**

Title : Worker Login  
As a worker , I want to log into the system so that I can manage station operations.  
Context / Note : This story ensures that workers can securely log in and access their operational dashboard.  
This story is related to Sprint #1.

Task :

* Create a separate login form for workers.
* Authenticate worker credentials.
* Redirect workers to their operational dashboard.

Subtask List :

* [Create worker login form UI]
* [Authenticate worker credentials]
* [Redirect to worker dashboard]

Associated User Stories :

* User Story 27 : Secure Access
* Rule : Workers must have role-based access to specific features.
* ID : US-10

**User Story 27 – Secure Access**

Title : Secure Access  
As a user , I want to have secure access to system configurations so that I can manage system settings safely.  
Context / Note : This story ensures that only authorized users can access admin panels and sensitive configurations.  
This story is related to Sprint #1.

Task :

* Implement role-based access control (RBAC).
* Restrict admin panel access to authorized users only.
* Log all access attempts for auditing purposes.

Subtask List :

* [Implement RBAC middleware]
* [Restrict access to admin routes]
* [Log access attempts in the database]

Associated User Stories :

* User Story 10 : Worker Login
* Rule : Unauthorized access attempts must trigger alerts.
* ID : US-27

**3. Software Testing**

**Login Page: Test Design**

**Fields:**

* Email
* Password

**Field-Wise Analysis and Test Design (Login)**

| **Field** | **Input Criteria** | **Valid Equivalence Class** | **Invalid Equivalence Class** | **Boundary Values** |
| --- | --- | --- | --- | --- |
| Email | Valid email format | user@example.com | Missing @, invalid domain, etc. | Structure-based, no length boundary |
| Password | 8–20 characters | Length 8–20, any characters | Less than 8 or more than 20 characters | 7 (invalid), 8 (valid), 20 (valid), 21 (invalid) |

**Weak Equivalence Class Testing (Login)**

| **Field** | **Test Case ID** | **Test Input** | **Expected Result** | **Remarks** |
| --- | --- | --- | --- | --- |
| Email | LOGIN-ECT-WEAK-01 | john.doe@mail.com | Accepted | Valid email format |
| Email | LOGIN-ECT-WEAK-02 | johnmail.com | Rejected | Missing @ symbol |
| Password | LOGIN-PWD-WEAK-01 | LoginPass1! | Accepted | Valid password length |
| Password | LOGIN-PWD-WEAK-02 | short | Rejected | Too short |

**Strong Equivalence Class Testing (Login)**

| **Field** | **Test Case ID** | **Test Input** | **Expected Result** | **Remarks** |
| --- | --- | --- | --- | --- |
| Email | LOGIN-ECT-STRONG-01 | valid@example.com | Accepted | Correct format |
| Email | LOGIN-ECT-STRONG-02 | invalidemail.com | Rejected | Missing @ |
| Email | LOGIN-ECT-STRONG-03 | @domain.com | Rejected | Missing local part |
| Email | LOGIN-ECT-STRONG-04 | user@.com | Rejected | Incomplete domain |
| Password | LOGIN-PWD-STRONG-01 | Pass1234 | Accepted | Exactly 8 characters |
| Password | LOGIN-PWD-STRONG-02 | StrongPassword12345678 | Accepted | Exactly 20 characters |
| Password | LOGIN-PWD-STRONG-03 | Short1 | Rejected | 7 characters |
| Password | LOGIN-PWD-STRONG-04 | TooLongPasswordInputText | Rejected | >20 characters |

**Boundary Value Analysis (Login)**

**Password Length (8–20)**

| **Test Case ID** | **Input Length** | **Test Input** | **Expected Result** | **Remarks** |
| --- | --- | --- | --- | --- |
| LOGIN-BVA-PWD-01 | 7 | Pass123 | Rejected | Below valid range |
| LOGIN-BVA-PWD-02 | 8 | Pass1234 | Accepted | Lower boundary |
| LOGIN-BVA-PWD-03 | 20 | PasswordLenExactly20!! | Accepted | Upper boundary |
| LOGIN-BVA-PWD-04 | 21 | TooLongPassword21Text | Rejected | Just above upper limit |

**Summary (Login)**

* Weak EC Tests: 4
* Strong EC Tests: 8
* BVA Tests: 4
* **Total Test Cases (Login): 16**

**Signup Page: Test Design**

**Fields:**

* Email
* Password
* Phone Number
* Name
* Username

**Field-Wise Analysis and Test Design (Signup)**

| **Field** | **Input Criteria** | **Valid Equivalence Class** | **Invalid Equivalence Class** | **Boundary Values** |
| --- | --- | --- | --- | --- |
| Email | Valid email format | user@example.com | Missing @, invalid domain | Structure-based |
| Password | 8–20 characters | Valid length, any characters | <8 or >20 characters | 7 (invalid), 8 (valid), 20 (valid), 21 (invalid) |
| Phone Number | Exactly 10 digits, numeric only | 1234567890 | <10, >10 digits, letters | 9 (invalid), 10 (valid), 11 (invalid) |
| Name | Alphabetic only, 1–50 characters | John Doe | Contains numbers/symbols, empty, >50 chars | 0 (invalid), 1 (valid), 50 (valid), 51 (invalid) |
| Username | Alphanumeric only, 4–15 characters | user123, Alpha | <4, >15 characters, special characters | 3 (invalid), 4 (valid), 15 (valid), 16 (invalid) |

**Weak Equivalence Class Testing (Signup)**

| **Field** | **Test Case ID** | **Test Input** | **Expected Result** | **Remarks** |
| --- | --- | --- | --- | --- |
| Email | SIGNUP-ECT-WEAK-01 | new.user@mail.com | Accepted | Valid format |
| Email | SIGNUP-ECT-WEAK-02 | useratmail.com | Rejected | Missing @ |
| Password | SIGNUP-PWD-WEAK-01 | SignupPass1 | Accepted | Valid |
| Password | SIGNUP-PWD-WEAK-02 | 123 | Rejected | Too short |
| Phone Number | SIGNUP-PHN-WEAK-01 | 9876543210 | Accepted | Valid 10-digit number |
| Phone Number | SIGNUP-PHN-WEAK-02 | 12345abcde | Rejected | Contains letters |
| Name | SIGNUP-NAM-WEAK-01 | Alice | Accepted | Valid name |
| Name | SIGNUP-NAM-WEAK-02 | John123 | Rejected | Contains digits |
| Username | SIGNUP-USR-WEAK-01 | user01 | Accepted | Valid |
| Username | SIGNUP-USR-WEAK-02 | us! | Rejected | Too short, invalid char |

**Strong Equivalence Class Testing (Signup)**

| **Field** | **Test Case ID** | **Test Input** | **Expected Result** | **Remarks** |
| --- | --- | --- | --- | --- |
| Email | SIGNUP-ECT-STRONG-01 | user@domain.com | Accepted | Valid email |
| Email | SIGNUP-ECT-STRONG-02 | nodomain@ | Rejected | Invalid domain |
| Email | SIGNUP-ECT-STRONG-03 | @nodomain.com | Rejected | Missing local part |
| Email | SIGNUP-ECT-STRONG-04 | user@.com | Rejected | Invalid domain format |
| Password | SIGNUP-PWD-STRONG-01 | Pass1234 | Accepted | Exactly 8 characters |
| Password | SIGNUP-PWD-STRONG-02 | Pass123456789012345! | Accepted | Exactly 20 characters |
| Password | SIGNUP-PWD-STRONG-03 | Short1 | Rejected | Only 7 characters |
| Password | SIGNUP-PWD-STRONG-04 | ThisPasswordIsTooLongToBeValid | Rejected | Over 20 characters |
| Phone Number | SIGNUP-PHN-STRONG-01 | 0123456789 | Accepted | 10-digit valid |
| Phone Number | SIGNUP-PHN-STRONG-02 | 987654321 | Rejected | 9 digits |
| Phone Number | SIGNUP-PHN-STRONG-03 | 12345678901 | Rejected | 11 digits |
| Phone Number | SIGNUP-PHN-STRONG-04 | 98abc12345 | Rejected | Contains characters |
| Name | SIGNUP-NAM-STRONG-01 | Emily | Accepted | Valid name |
| Name | SIGNUP-NAM-STRONG-02 | Tom123 | Rejected | Contains numbers |
| Name | SIGNUP-NAM-STRONG-03 | `` | Rejected | Empty |
| Name | SIGNUP-NAM-STRONG-04 | A repeated 51 times | Rejected | >50 characters |
| Username | SIGNUP-USR-STRONG-01 | user123 | Accepted | Valid |
| Username | SIGNUP-USR-STRONG-02 | u1 | Rejected | Too short |
| Username | SIGNUP-USR-STRONG-03 | user!name | Rejected | Contains special char |
| Username | SIGNUP-USR-STRONG-04 | averylongusernamethatisbad | Rejected | >15 characters |

**Boundary Value Analysis (Signup)**

**Password Length (8–20)**

| **Test Case ID** | **Input Length** | **Test Input** | **Expected Result** | **Remarks** |
| --- | --- | --- | --- | --- |
| SIGNUP-BVA-PWD-01 | 7 | Sign123 | Rejected | Below lower boundary |
| SIGNUP-BVA-PWD-02 | 8 | Sign1234 | Accepted | Lower boundary |
| SIGNUP-BVA-PWD-03 | 20 | ValidPasswordLen20!! | Accepted | Upper boundary |
| SIGNUP-BVA-PWD-04 | 21 | TooLongPasswordText123 | Rejected | Just above max |

**Phone Number (10 digits)**

| **Test Case ID** | **Input Length** | **Test Input** | **Expected Result** | **Remarks** |
| --- | --- | --- | --- | --- |
| SIGNUP-BVA-PHN-01 | 9 | 123456789 | Rejected | Below valid range |
| SIGNUP-BVA-PHN-02 | 10 | 1234567890 | Accepted | Valid boundary |
| SIGNUP-BVA-PHN-03 | 11 | 12345678901 | Rejected | Above valid range |

**Name Length (1–50)**

| **Test Case ID** | **Input Length** | **Test Input** | **Expected Result** | **Remarks** |
| --- | --- | --- | --- | --- |
| SIGNUP-BVA-NAM-01 | 0 | `` | Rejected | Empty name |
| SIGNUP-BVA-NAM-02 | 1 | A | Accepted | Lower boundary |
| SIGNUP-BVA-NAM-03 | 50 | A repeated 50 times | Accepted | Upper boundary |
| SIGNUP-BVA-NAM-04 | 51 | A repeated 51 times | Rejected | Just above max |

**Username Length (4–15)**

| **Test Case ID** | **Input Length** | **Test Input** | **Expected Result** | **Remarks** |
| --- | --- | --- | --- | --- |
| SIGNUP-BVA-USR-01 | 3 | usr | Rejected | Below valid range |
| SIGNUP-BVA-USR-02 | 4 | user | Accepted | Lower boundary |
| SIGNUP-BVA-USR-03 | 15 | username1234567 | Accepted | Upper boundary |
| SIGNUP-BVA-USR-04 | 16 | username12345678 | Rejected | Above valid range |

**Summary (Signup)**

* Weak EC Tests: 10
* Strong EC Tests: 20
* BVA Tests: 15
* **Total Test Cases (Signup): 45**

**4. Meetings Conducted**

**Sprint Planning Meeting**

Overview :

a. Date Time Stamp : 1 March 2025  
b. Attendees :

| **Present** | **Absent** |
| --- | --- |
| Anas Bin Rashid | - |
| Adan Malik | - |
| Hasnain Akhtar | - |

Topics Discussed and Conclusions Reached :

* Defined sprint goals: Implement user authentication and secure access features.
* Prioritized user stories: US-1, US-2, US-3, US-10, US-27.
* Assigned tasks to team members.

Conclusions :

* Sprint backlog finalized.
* Daily scrum schedule set for 9:00 AM daily.

Overall Duration : 1 hour

**Daily Scrum Meetings**

The meeting focused on different topics each day and they are reflected in the burndown chart below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Day | Story Points Remaining (Planned) | Story Points Remaining (Actual) | Tasks Completed | Notes |
| Day 1 | 30 | 30 | 0 | Sprint starts; no tasks done. |
| Day 2 | 27 | 28 | 2 | Minor delays in setup. |
| Day 3 | 24 | 26 | 2 | UI design started. |
| Day 4 | 21 | 23 | 3 | Authentication UI completed. |
| Day 5 | 18 | 20 | 3 | Dashboard UI progress. |
| Day 6 | 15 | 17 | 3 | Minor bug fixes. |
| Day 7 | 12 | 14 | 3 | Inventory UI completed. |
| Day 8 | 9 | 11 | 3 | Testing ongoing. |
| Day 9 | 6 | 8 | 3 | Leftover tasks identified. |
| Day 10 | 0 | 3 | 5 | Sprint ends; some tasks left. |

A graph with blue and orange lines

AI-generated content may be incorrect.

**Sprint Review Meeting**

Overview :

a. Date Time Stamp : 10 March 2025  
b. Attendees :

| **Present** | **Absent** |
| --- | --- |
| Anas Bin Rashid | - |
| Adan Malik | - |
| Hasnain Akhtar | - |

Topics Discussed and Conclusions Reached :

* Reviewed completed user stories: US-1, US-2, US-3, US-10, US-27.
* Demonstrated working features: Registration, Login, Password Reset, Worker Login, Secure Access.

Conclusions :

* All high-priority user stories completed successfully.
* Minor bugs identified during testing.

Overall Duration : 1 hour

**Sprint Retrospective Meeting**

Retrospective Meeting : Iteration/Sprint #1  
Overview :

c. Date Time Stamp : 11 March 2025

d. Attendees :

| **Present** | **Absent** |
| --- | --- |
| Anas Bin Rashid | - |
| Adan Malik | - |
| Hasnain Akhtar | - |

Retrospective Content :

| **Successes** | **To Improve** | **Actions and Suggestions** |
| --- | --- | --- |
| Completed all high-priority user stories. | Minor delays in integrating password reset functionality. | Allocate more time for testing in future sprints. |
| Strong collaboration among team members. | Communication gaps during task handovers. | Use Trello for better task tracking. |
| Well-documented codebase. | Limited focus on UI/UX. | Involve a designer for future sprints. |

**5. Definition of Done (DoD)**

* All unit tests are written.
* The written code is documented.
* Code reviewed and approved by peers.
* Features tested in staging environment.
* No critical bugs reported.

**6. Bug Report**

Name About : Bug Report: Duplicate Email Registration

Expected Behavior : Users should not be able to register with an existing email.

Actual Behavior : The system allows duplicate email registrations.

Steps to Reproduce :

1. Navigate to the registration page.
2. Enter an email that already exists in the database.
3. Submit the form.

Context / Environment : This issue is related to Sprint #1, User Story #1.

Possible Solutions :

* Add backend validation to check for duplicate emails.
* Display an error message on the frontend if the email exists.

Definition of Done (DoD) :

* Unit test written for email validation.
* Code documented.
* Issue resolved and tested in staging.

Date Time Stamp : 5 March 2025

Reported By : Adan Malik

**7. Output**

**Tested Code Implementation**

***API Endpoint for User Registration (Node.js + Express.js)***

*app.post('/api/register', async (req, res) => {*

*try {*

*const { name, email, phone, password } = req.body;*

*// Validation*

*if (!name || !email || !phone || !password) {*

*return res.status(400).json({ error: 'All fields are required.' });*

*}*

*// Save to Database*

*const newUser = new UserModel({ name, email, phone, password });*

*await newUser.save();*

*res.status(201).json({ message: 'User registered successfully.' });*

*} catch (error) {*

*res.status(500).json({ error: 'Server error.' });*

*}*

*});*

*import React, { useState } from 'react';*

*import axios from 'axios';*

*const RegisterForm = () => {*

*const [user, setUser] = useState({ name: '', email: '', phone: '', password: '' });*

*const handleSubmit = async (e) => {*

*e.preventDefault();*

*try {*

*const response = await axios.post('/api/register', user);*

*alert(response.data.message);*

*} catch (error) {*

*console.error(error);*

*}*

*};*

*return (*

*<form onSubmit={handleSubmit}>*

*<input*

*type="text"*

*placeholder="Name"*

*value={user.name}*

*onChange={(e) => setUser({ ...user, name: e.target.value })}*

*/>*

*<input*

*type="email"*

*placeholder="Email"*

*value={user.email}*

*onChange={(e) => setUser({ ...user, email: e.target.value })}*

*/>*

*<input*

*type="text"*

*placeholder="Phone"*

*value={user.phone}*

*onChange={(e) => setUser({ ...user, phone: e.target.value })}*

*/>*

*<input*

*type="password"*

*placeholder="Password"*

*value={user.password}*

*onChange={(e) => setUser({ ...user, password: e.target.value })}*

*/>*

*<button type="submit">Register</button>*

*</form>*

*);*

*};*

*export default RegisterForm;*

***Katalon Studio Code***

*WebUI.openBrowser('')*

*WebUI.navigateToUrl('http://localhost:3000/register')*

*WebUI.setText(findTestObject('Object Repository/input\_Name'), 'John Doe')*

*WebUI.setText(findTestObject('Object Repository/input\_Email'), 'john.doe@example.com')*

*WebUI.setText(findTestObject('Object Repository/input\_Phone'), '1234567890')*

*WebUI.setText(findTestObject('Object Repository/input\_Password'), 'password123')*

*WebUI.click(findTestObject('Object Repository/button\_Register'))*

*WebUI.verifyTextPresent('User registered successfully.', false)*

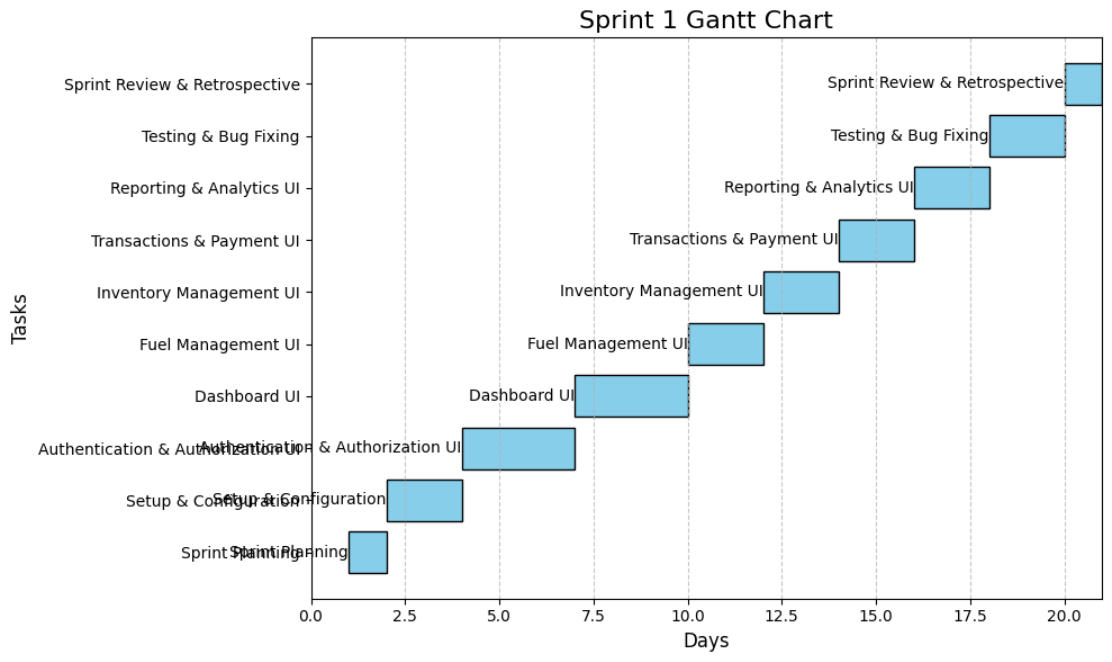
*WebUI.closeBrowser()*

**8. Project Planner and Gantt Chart**

**Project Planner**

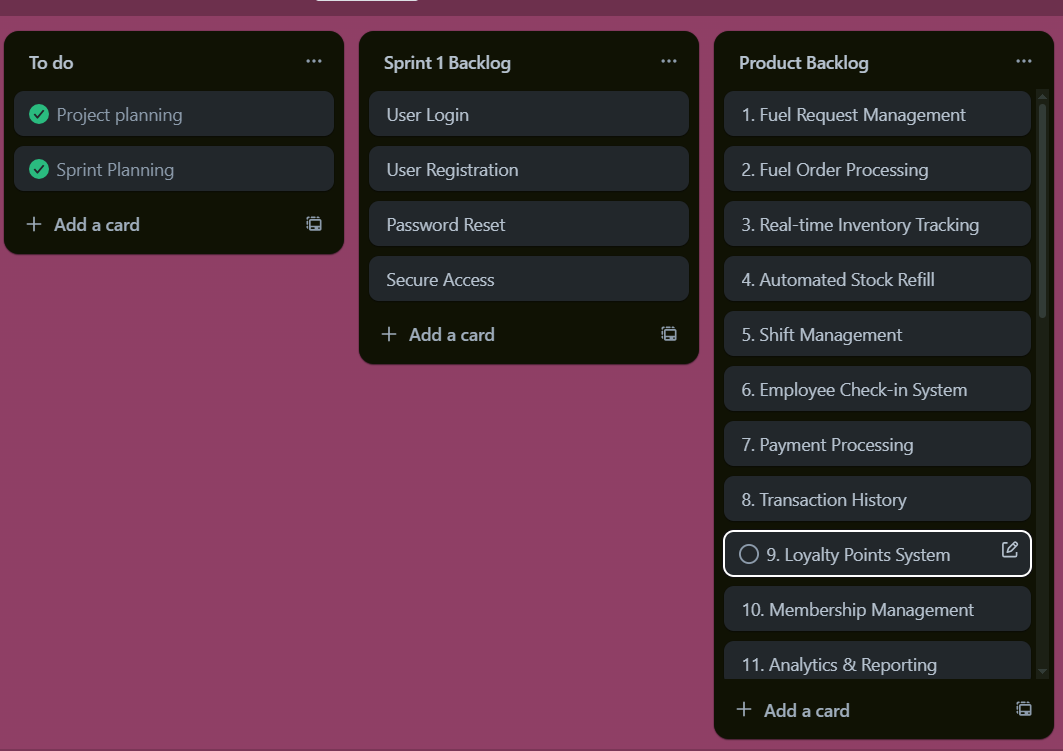
* Task 1: Set up project infrastructure (Day 1–2).
* Task 2: Implement user registration (Day 3–4).
* Task 3: Implement user login and password reset (Day 5–7).
* Task 4: Implement worker login and secure access (Day 8–10)

**Gantt Chart**



**9. Trello Screenshots**

**Start of Sprint**

****

**Middle of Sprint**

**Screens screenshot of a computer screen

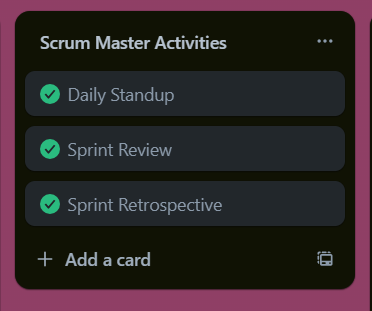
AI-generated content may be incorrect.**

**End of Sprint**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Scrum Master Activities and Leanings**

****

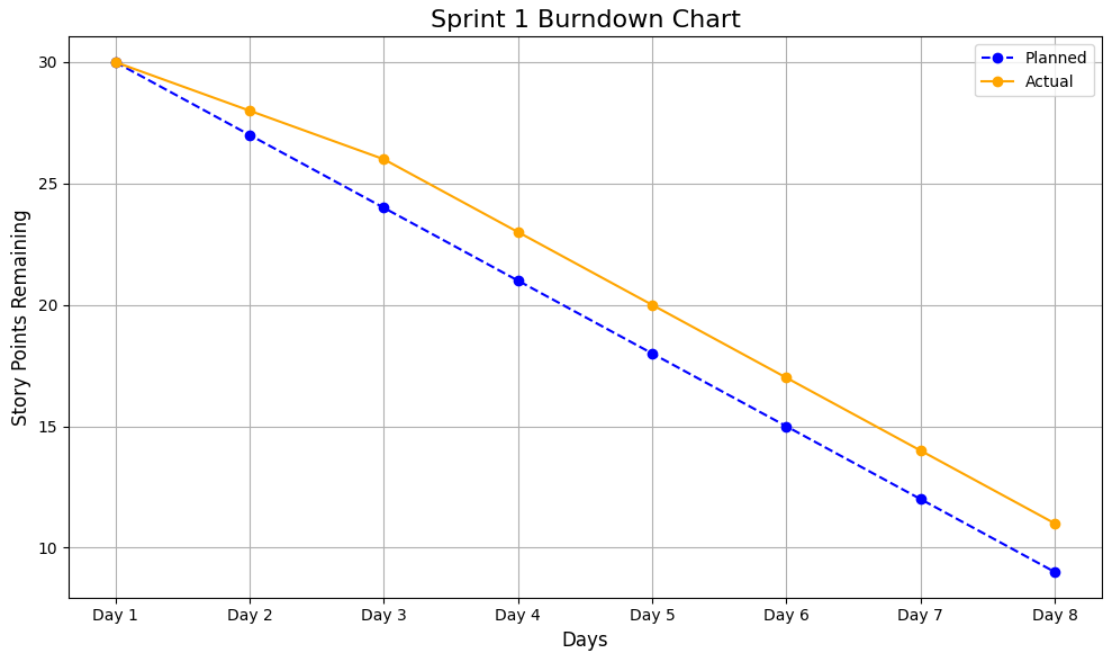
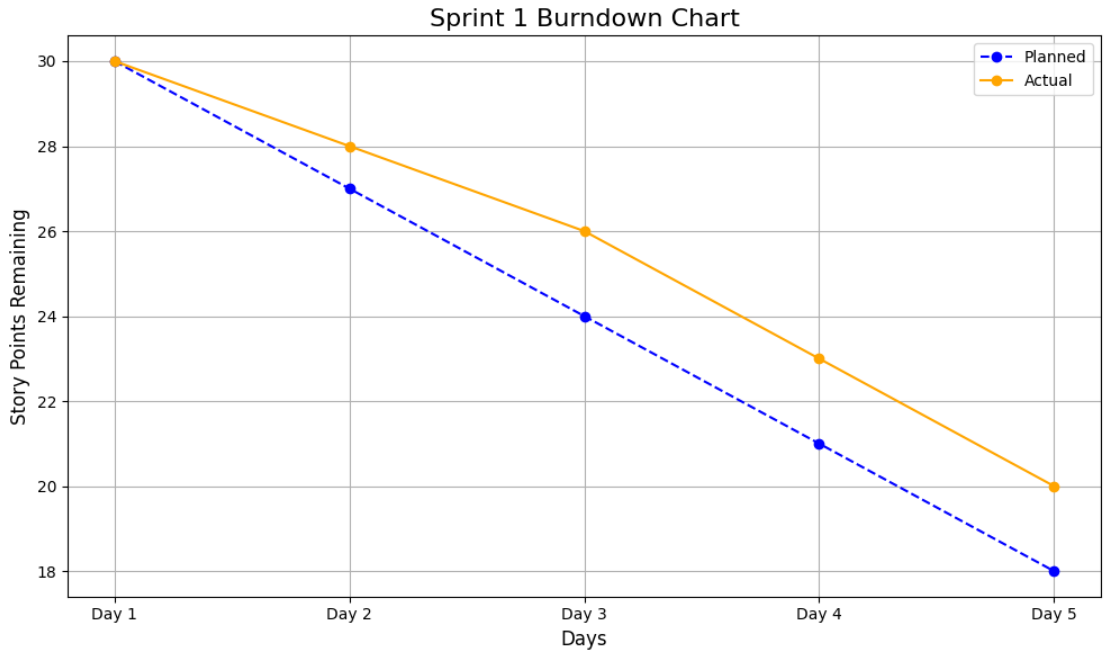
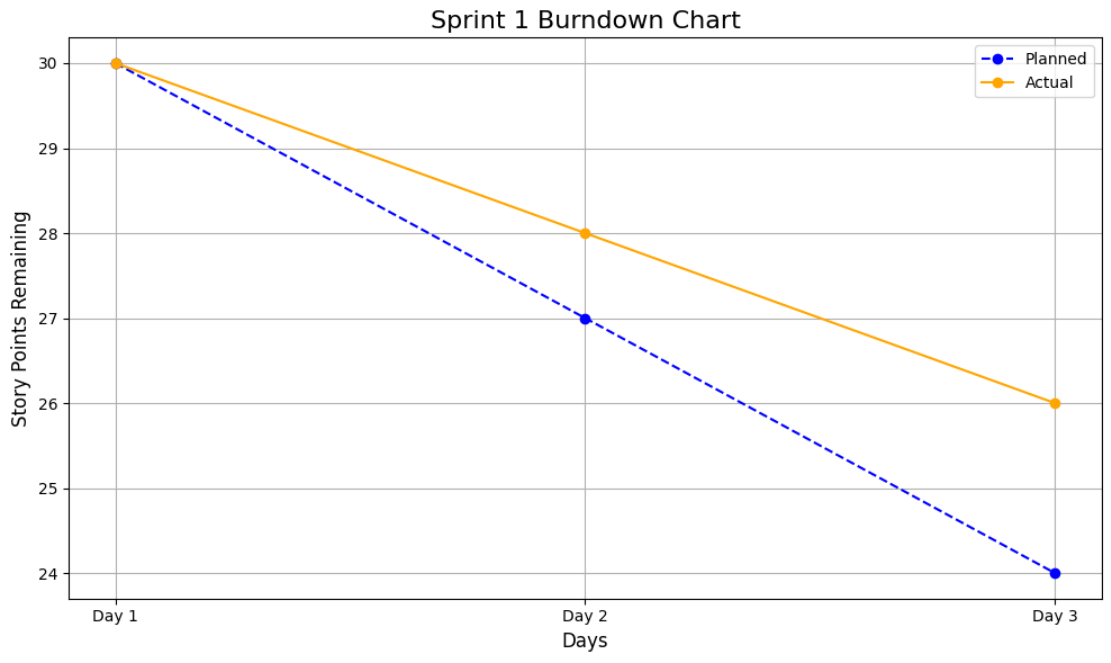
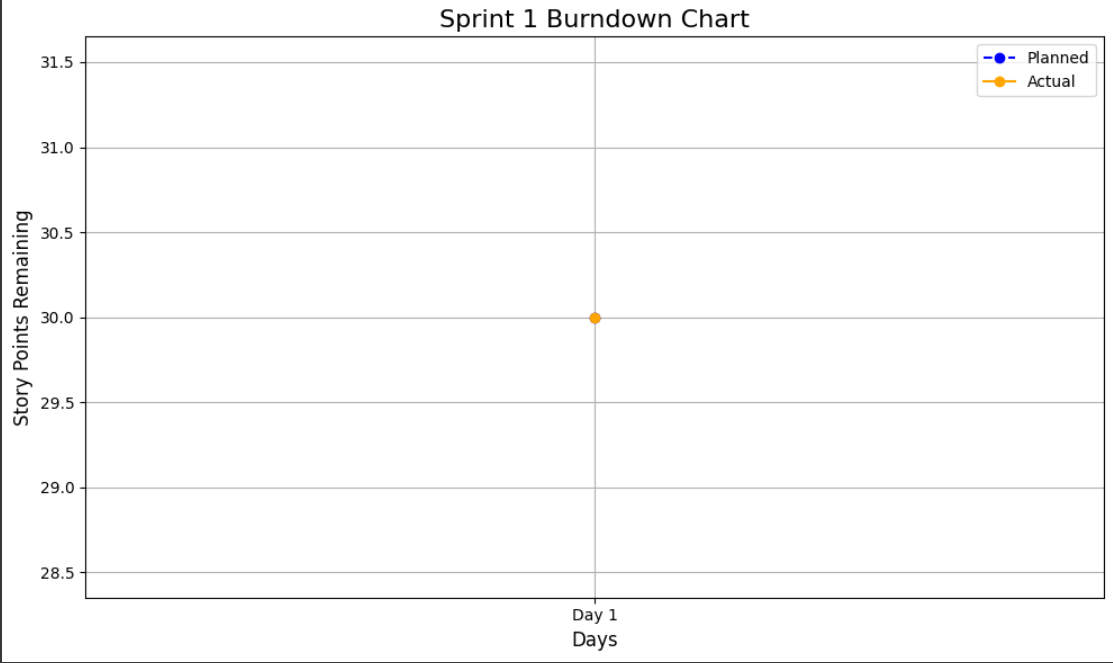
Topics Discussed and Conclusions Reached :

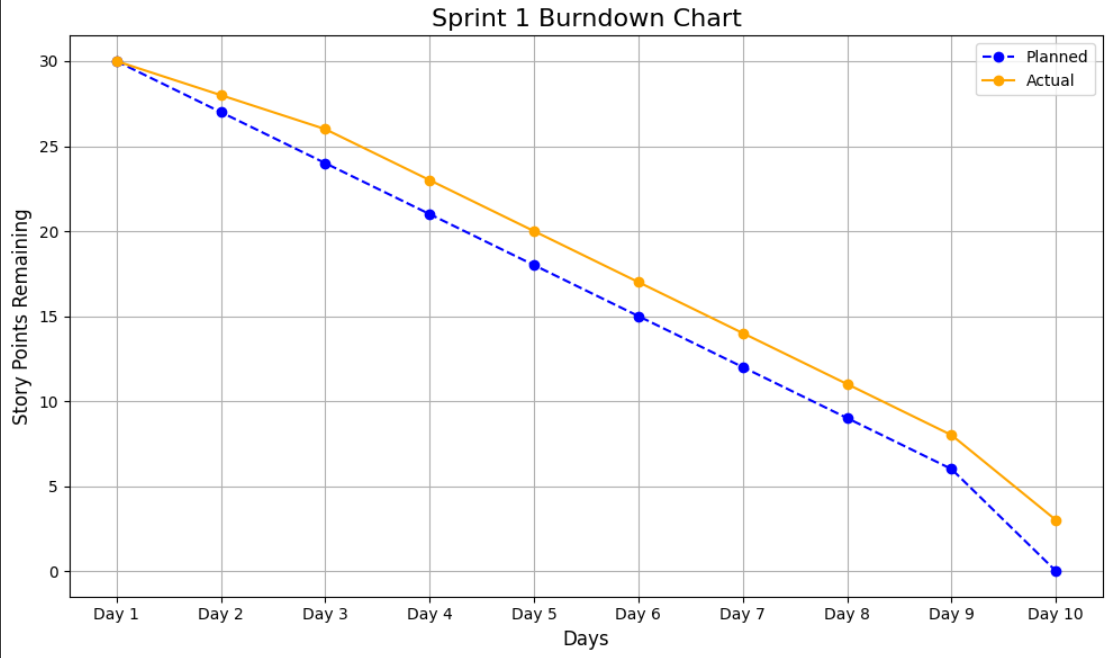
* Reviewed completed user stories: US-1, US-2, US-3, US-10, US-27.
* Demonstrated working features: Registration, Login, Password Reset, Worker Login, Secure Access.

Conclusions :

* All high-priority user stories completed successfully.
* Minor bugs identified during testing.

**10. Burndown Charts**

****

****