TEST PLAN FOR RESTROSYNC

Note that you can refine your testing plan as the project development goes. Keep the change log as follow:

ChangeLog

Version	Change Date	By	Description
version number	Date of Change	Name of person who made changes	Description of the changes made
1.0	2025-02-28	Divy Patel	Created a test plan.
2.0	2025-03-21	Aswin Manoj	Moved acceptance tests to Sprint 4

1 Introduction

1.1 Scope

Scope defines the features, functional or non-functional requirements of the software that **will be** tested.
RestroSync is a restaurant management system designed to streamline daily operations, enhance customer experience, and optimize business performance. This test plan outlines the strategies and tools for verifying the system's functionality, performance, and reliability across various features, including:

- User Account Management
- Table Management
- Menu Management
- Order Processing
- Sales Analysis

Non Functional Requirements:

- The web application can handle 100 concurrent servers, processing up to 1000 orders per minute

1.2 Roles and Responsibilities

Name	**Net ID**	**GitHub username**	**Role**
Divy Patel	patelda2@myumanitoba.ca	Divy63	Test Manager
Het Patel	Patelh22@myumanitoba.ca	hetu2344	Automation Tester
Aswin Manoj	manoja@myumanitoba.ca	Aswin-Manoj	Developer
Rishamdeep Singh	singhr50@myumanitoba.ca	rishamD	QA Analyst
Seyi Asoga	asogao@myumanitoba.ca	asogaseyi1	Performance Testing
Aidan Labossiere	laboss42@myumanitoba.ca	AidanLabs	Developer

- **Test Manager**: Oversees test execution and ensures alignment with project requirements.
- **QA Analyst**: Creates and executes test cases.
- **Automation Tester**: Implements automated testing for frontend and backend.
- **Performance Tester**: Conducts load and stress testing.
- **Developer:** Develops features to test.

2 Test Methodology

2.1 Test Levels

Testing will be conducted at different levels to ensure system stability and functionality.

- **Unit Testing**: Verify individual functions and modules (e.g., Jest for backend tests).
- **Integration Testing**: Test interactions between services using Postman or Supertest.
- **Functional Testing**: Validate core features such as User Account Management, Menu Management, Table Management, Order Processing, Sales Analysis.
- **Acceptance Testing**: Evaluate the system from the end-user's perspective.

  (Note: Acceptance testing has been moved to Sprint 4 due to unforeseen factors)
- **Regression Testing**: Ensure updates do not break existing functionality.
- **Load Testing**: Measure system response time and scalability using JMeter.

Requirements:

- At least 10 unit tests per core feature.
- 10 integration tests covering major workflows.
- Acceptance tests for each core feature through user story validation.
- Regression tests executed on every commit in CI/CD pipeline.
- Load testing with at least one requests per core feature in test load.

2.2 Test Completeness

Testing will be considered complete when:

- 100% of backend code is covered by Jest tests.
- All critical and high-priority test cases pass.
- No severe defects remain unresolved.
- Load testing confirms the system can handle 100 concurrent users and 1000 orders per minute.

3 Resource & Environment Needs

3.1 Testing Tools

Testing Tool	Purpose

Jest	Backend Testing
GitHub Actions	CI/CD pipeline for automated testing
Cypress	Frontend Testing (May change as per requirements.)
Postman	API request validation
JMeter	Load testing (May change as per requirements.)
GitHub Issues	Bug Tracking

3.2 Test Environment

- **Operating Systems**: Windows 10,Windows 11, macOS
- **Browsers**: Chrome, Firefox, Brave
- **Backend**: Node.js, Express.js, PostgreSQL
- **Frontend**: React.js
- **Infrastructure**: Docker, GitHub Actions for CI/CD

4 Terms/Acronyms

Make a mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
API	Application Program Interface
AUT	Application Under Test