Test Plan

For Sprint 1 Dev Work (ID – 9, 10, 11, 12)

Project Name: TeamFactz

Revision History

Date	Version	Author	Description

Table of Contents

1. Overview

An outline of the test plan for TeamFactz, a web application that is designed to manage the activities and plans of clubs. The application provides users with access to and management of their club activities in accordance with their access permissions.

- Web
    - Django
    - Nust Js/Next JS
- Database
    - PostgreSQL
    - Multi Tenants
- Server
    - Heroku
    - Heroku 3 dynos (web, worker, beat) (7+7+7) (25 + 25 + 25) + 9 DB

1.1. Purpose

The purpose of this document is to define:

- The test scope, focus areas and objectives
- The test responsibilities
- The test strategy for the levels and types of tests for this release
- The entry and exit criteria
- The basis of the test estimates
- Any risks, issues, assumptions and test dependencies
- The test schedule and major milestones
- The test deliverables

1.2. Scope

This document details the testing that will be performed by QA for Sprint 1 Dev works of TeamFactz project. It defines the overall testing requirements and provides an integrated view of the project test activities for sprint 1 Dev works. Its purpose is to document:

- What will be tested;
- How testing will be performed;
- What resources are needed, and when

2. Testing Summary

2.1. Scope of Testing

2.1.1. In scope

The scope of this test plan covers the following features and functionality of the TeamFactz application (Sprint 1 Dev works, ID – 9,10,11,12):

- login
- sign-up
- Profile
- Import Users
- Change Password
- Forget/Reset Password
- Profile Update logs
- Email Verifications
- Export user in CSV, Excel, PDF.

2.1.2. Out of scope

The scope of this test plan not covers the following features and functionality of the TeamFactz application (Sprint 1 Dev works, ID – 9,10,11,12):

- Contact vCard Export.
- Upcoming Birthdays
- Groups
- Add
- Update
- Delete
- Group Members
- Assign groups to members

3. Test Methodologies

We will use an Agile testing methodology for the TeamFactz application. The approach is ideally suited to the iterative, collaborative nature of app development, as well as the flexibility it provides for testing and integrating applications.

3.1. Test Types.

- **Functional Testing**: Verify each function of the application operates in conformance with the requirement specification.
- **Integration Testing**: Ensure that different modules or services work together correctly.
- **Regression Testing**: Ensure that new code changes do not adversely affect the existing functionality.

Firstly, will conduct the testing manually and then automate the main functionalities

3.2. Testing Environment

The following test environments will be used:

- Windows 10 – Chrome, Firefox and Brave
- Device Types - Desktop computers, Laptops
- Network - Wi-Fi, Broadband
- Hardware - i3 10th gen/Ryzen 3 or above, Ram 8GB, Storage 128 SSD + 1TB HDD

4. Test Strategy

4.1. Test level responsibility

Testing levels expected to be applied and who has primary (P) and secondary (S) responsibility for performing this testing.

Test Level	Developer	QA	PM
Unit Testing	P	S	
Integration Testing	P	S	
Functional Testing		P	S
Production Verification Testing		S	P

4.2. Test Type & Approach

The types of testing covered by the project team and their standard objectives:

Component	Objectives
Progression Requirements	The objectives are to verify that the application: • Meets the defined requirements; • Performs and functions accurately; • Correctly handles error conditions; • Interfaces function correctly; • Data load is successful. Functional testing will occur in an iterative and controlled manner, ensuring the solution matches the defined requirements.
Test Techniques	Following Test techniques will be used: • Black-Box Testing • Manual Testing • Automation Testing
Test Deliverables	• **Test Plan Document**: This document outlining the test strategy. • **Test Cases and Test Scripts**: Detailed test cases for all functionalities. • **Test Data**: Data sets prepared for testing purposes. • **Test Summary Report**: Summary of the testing activities and outcomes. • **Defect Report**: List of identified defects with their status and severity. • **Test Metrics**: Key metrics and KPIs gathered during testing (e.g., number of tests executed, passed, failed).

Step 1: The first step is to create test scenarios and test cases for the various features in Scope.

We will use a variety of test design techniques when developing test cases. (If needed)

- Equivalence Class Partition
- Boundary Value Analysis
- Decision Table Testing
- State Transition Testing
- Use Case Testing

In addition to applying our expertise in creating Test Cases, we also apply the following methods:

- Error Guessing
- Exploratory Testing
- We prioritize the Test Cases

Step 2: Our testing procedure when we receive a request for testing:

- First, we'll conduct smoke testing to see if the various and important functionalities of the application are working.
- We reject the build, if the Smoke Testing fails and will wait for the stable build before performing in depth testing of the application functionalities.
- Once we receive a stable build, which passes Smoke Testing, we perform in depth testing using the Test Cases created.
- Multiple Test Resources will be testing the same Application on Multiple Supported Environments simultaneously.

We then report the bugs in the bug tracking tool and send dev. management the defect found on that day in a status.

As part of the testing, we will perform the below types of testing:

- Smoke Testing and Sanity Testing
- Regression testing and Retesting
- Usability Testing, Functionality & UI Testing

We repeat Test Cycles until we get a quality product.

Step 3: We will follow the below best practices to make our testing better:

• **Context Driven Testing** – We will be performing testing as per the context of the given application.

• **Shift Left Testing** – We will start testing at the beginning stages of the Development itself, instead of waiting for the stable build.

• **Exploratory Testing** – Using our expertise we will perform Exploratory Testing, apart from the normal execution of the Test cases.

• **End to End Flow Testing** – We will test the end-to-end scenario which involve multiple functionalities to simulate the end user flows.

4.3. Test Execution Schedule

Following is the test schedule planned for the project – Task Time Duration

Task	Dates
▪ Creating Test Plan	
▪ Test Case Creation	
▪ Test Case Execution	
▪ Summary Reports Submission Date	

4.4. Facility, data, and resource provision plan

4.4.1. Data Requirements

4.5. Testing Tools

The following tools will be used for testing:

Process	Tool
Test case creation	Microsoft Word
Test case tracking	Microsoft Excel
Test case execution	Manual + Automation
Test case management	Microsoft Excel
Defect management	Microsoft Excel + Snipping tool + Azure DevOps

5. Assumptions and Dependencies

5.1. Assumptions

- The scope of testing will be sufficient to cover all critical functionalities, including new features and existing ones. There will be enough time allocated for regression testing to ensure that existing functionalities are not adversely affected by new changes.

- Contingency plans will be in place to address potential risks that could impact the testing process. This includes having backup test environments, additional resources, and clear communication channels for resolving issues quickly.

5.2. Dependencies

- Business analysts will be available throughout the testing phase to provide clarifications on requirements, assist in defining acceptance criteria, and support the test team in understanding complex business logic.

- Developers will be accessible to address any defects reported by the test team, provide necessary support during integration testing, and assist with technical issues encountered during the testing process.

- Test environments will be set up and configured correctly before the testing phase begins. These environments will remain stable and accessible to the test team throughout the testing period.

- Necessary tools and resources for testing (e.g., testing software, hardware, and network access) will be available and operational.
- Requirements will be finalized and approved by all stakeholders before the testing phase begins. Any changes to requirements after testing has started will be managed through a formal change control process.
- Test data will be prepared and made available prior to the start of testing. This includes data required for various testing phases such as functional, performance, security, and usability testing.
- External systems or third-party integrations required for testing will be available and stable. Any dependencies on third-party systems will be managed to avoid disruptions in the testing process.

6. Entry and Exit Criteria

The below are entry and exit criteria for every phase of the Software Testing Life Cycle:

6.1. Requirement Analysis

6.1.1. Entry Criteria

- Once the testing team receives the Requirements Documents or details
- about the Project

6.1.2. Exit Criteria

- List of Requirements are explored and understood by the Testing team
- Doubts are cleared

6.2. Test Execution

6.2.1. Entry Criteria

- Test Scenarios and Test Cases Documents are signed-off by the Client
- Application is ready for Testing

6.2.2. Exit Criteria

- Test Case Reports, Defect Reports are ready

6.3. Test Closure

6.3.1. Entry Criteria

- Test Case Reports, Defect Reports are ready

6.3.2. Exit Criteria

- Test Summary Reports submitted

7. Administrative Plan

7.1. Approvals

The following persons are responsible for the critical aspects of testing:

Task	Responsible Person	Escalation/ Approver
Functional testing Signoff	Md Aminul Islam	
UI Testing Signoff	Md Aminul Islam	
Production Verification Testing Signoff		

7.2. Test Milestones and Schedule

Detailed below are the high-level testing milestones.

Milestone	Planned End Date	Actual End Date	Resource

7.3. Training

The following training requirements have been identified to ensure testing can commence:

Training Requirement	Staff	Date
-	-	-
-	-	-

7.4. Defect Management

The criteria for identifying a defect, such as deviation from specified requirements, user experience issues, or technical errors.

- This document describes how to report a defect, including the use of a template, providing detailed reproduction steps, and attaching screenshots or logs.
- A procedure for triaging and prioritizing defects, such as assigning severity levels and priority levels, and identifying the appropriate team members to investigate and resolve the issue.
- Defect tracking software or a project management tool will be used to track and manage defects.
- Team members who are involved in the defect reporting process, such as testers, developers, and the test lead, should understand their roles and responsibilities.
- Information about the progress of defects and the status of the defects is communicated to stakeholders through communication channels and at regular intervals.
- In order to measure the effectiveness of the defect reporting process, metrics such as number of defects found, time taken to resolve them, and percentage of defects resolved successfully must be identified.

✓ **Bug tracking tool: Azure DevOps**

8. Definitions

The following acronyms and terms have been used through out this document

Term/Acronym	Definition
UI	User Interface
KPI	Key performance Indicator
SDLC	Software Development Life Cycle

9. References

The following documents have been used to assist in creation of this document.

#	Document name	Version	Comments
1	TeamFactz Feature lists	-	N/A

10. Points of Contact

The following people can be contacted in reference to this document

Primary Contact	
Name	
Title/Organisation	
Phone	
Email	
Secondary Contact	
Name	
Title/Organisation	
Phone	
Email	