<Hillel Garage Group 1>
Test Plan

Version <1.0>

Revision History

Date	Version	Description	Author
15.11.2022	1.0	Stage 1. 1.1-1.4	Олексій
04.12.2022	1.0	Stage 2. Stage 3	Марія

Table of Contents

Test Plan

Introduction

1.1 Purpose

This Test Plan document for the “Hillel Garage Group 1” supports the following objectives:

1.We will conduct regression testing.Make a decision to assess the feasibility, risks, time.Performance testing.Testing version installation. Localization testing(which languages must be supported,whether or not additional localizations are to be added during the operation of the solution,what countries our users will be working in,whether there will be sales and in what currencies,whether there is a need to work with time zones.

2.We will define such requirements as correctness; unambiguity; completeness; consistency, verifiability; modifiability; traceability

3.Hillel Garage project components - Garage, Fuel expenses, Instruction should be tested

1.2 Background

You need to create your profile. The Hillel Garage object includes such components as "Garage" "Fuel expenses" "Instruction". Version © 2021 Hillel IT school. On this site you can create cars, add car mileage, search for car parts

1.3 Scope

We conduct System testing. We check the functional requirements in general in the system. We identify defects, incompatibility with the environment, unexpected use cases, missing or incorrect functionality, inconvenience of use. We will use the environment as close as possible to the project.Functional type of testing.

There may still be risks of product testing - system failure with the maximum number of simultaneous connections, working with hard-to-reproduce cases that cannot be found a priori in the environment under test.

We will test requirements, business processes, security, interaction, performance

1.4 Project Identification

The table below identifies the documentation and availability used for developing the *test plan*:

Document (and version / date)	Created or Available	Received or Reviewed	Author or Resource	Notes
Requirements Specification	☐ Yes	☐ Yes		
Functional Specification	☐ Yes	☐ Yes		
Use-Case Reports	☐ Yes	☐ Yes		
Project Plan	☐ Yes	☐ Yes		
Design Specifications	☐ Yes	☐ Yes		
User's Manuals	☐ Yes	☐ Yes		
Business Model or Flow	☐ Yes	☐ Yes		
Business Functions and Rules	☐ Yes	☐ Yes		
Project or Business Risk Assessment	☐ Yes	☐ Yes		

2. Requirements for Test

The listing below identifies those items —use cases, functional requirements, and non-functional requirements —that have been identified as targets for testing. This list represents what will be tested.

Functional requirements:

1.Garage: User can add a car from the drop down menu

2.Fuel expenses: User can count fuel expenses

3.Instructions: User can see information about cars

4.Profile:User can enter data and edit it

5.Settings:User can change currency,unit of distance,mail or password

Non-functional requirements:

1. Quick response to the request no more than 200 ms
2. Additional features should be available only for registered users(profile,settings)
3. The system should support 10000 users at the same time
4. The system should be able support registered 10000 cars per 1 sek
5. The website is available in a browser on the personal computer

3. **Test Strategy**

Garage,Fuel expenses,Instructions,Profile,Settings will be testing by:

Function Testing
User Interface Testing

Quick response,Additional features,system should support 10000 users,system should be able support, registered 10000 cars per 1 sek,website is available in a browser will be testing by:

Performance Profiling
Load Testing
Stress Testing
Volume Testing
Security and Access Control Testing
Failover and Recovery Testing
Configuration Testing

This test will not be implemented or executed. This test is not appropriate:
Data and Database Integrity Testing
Business Cycle Testing
Installation Testing

1.1 Testing Types

1.1.1 Data and Database Integrity Testing

This testing is not available

1.1.1 Function Testing

Function testing of the target-of-test should focus on any requirements for test that can be traced directly to use cases or business functions and business rules. The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules. This type of testing is based upon black box techniques; that is verifying the application and its internal processes by interacting with the application via the Graphical User Interface (GUI) and analyzing the output or results. Identified below is an outline of the testing recommended for each application:

Test Objective:	Ensure proper target-of-test functionality, including navigation, data entry, processing, and retrieval.
Technique:	Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following: · The expected results occur when valid data is used. · The appropriate error or warning messages are displayed when invalid data is used. · Each business rule is properly applied.
Completion Criteria:	· All planned tests have been executed. · All identified defects have been addressed.
Special Considerations:	All features should be implemented completely before the execution functional test

1.1.1 Business Cycle Testing

These tests are not performed because it is not applicable for Hillel Garage project because it doesn't have date sensitive or repetitive features.

User Interface Testing

User Interface (UI) testing verifies a user's interaction with the software. The goal of UI testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards.

Test Objective:	Verify the following: · Navigation through the target-of-test properly reflects business functions and requirements, including window-to-window, field-to- field, and use of access methods (tab keys, mouse movements, accelerator keys) · Window objects and characteristics, such as menus, size, position, state, and focus conform to standards.
Technique:	Create or modify tests for each window to verify proper navigation and object states for each application window and objects.
Completion Criteria:	Each window successfully verified to remain consistent with benchmark version or within acceptable standard
Special Considerations:	Not all properties for custom and third party objects can be accessed.

1.1.2 Performance Profiling

Performance profiling is a performance test in which response times, transaction rates, and other time-sensitive requirements are measured and evaluated. The goal of Performance Profiling is to verify performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune a target-of-test's performance behaviors as a function of conditions such as workload or hardware configurations.
Note: Transactions below refer to "logical business transactions". These transactions are defined as specific use cases that an actor of the system is expected to perform using the target-of-test, such as add or modify a given contract.

Test Objective:	Verify performance behaviors for designated transactions or business functions under the following conditions: · normal anticipated workload · anticipated worst case workload
Technique:	· Modify data files to increase the number of transactions or the scripts to increase the number of iterations each transaction occurs. · Scripts should be run on one machine (best case to benchmark single user, single transaction) and be repeated with multiple clients (virtual or actual, see Special Considerations below).
Completion Criteria:	· Single Transaction or single user: Successful completion of the test scripts without any failures and within the expected or required time allocation per transaction. · Multiple transactions or multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation.
Special Considerations:	Comprehensive performance testing includes having a background workload on the server. There are several methods that can be used to perform this, including: · "Drive transactions" directly to the server, usually in the form of Structured Query Language (SQL) calls. · Create "virtual" user load to simulate many clients, usually several hundred. Remote Terminal Emulation tools are used to accomplish this load. This technique can also be used to load the network with "traffic". · Use multiple physical clients, each running test scripts to place a load on the system. Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. The databases used for Performance Testing should be either actual size or scaled equally.

1.1.3 Load Testing

Load testing is a performance test which subjects the target-of-test to varying workloads to measure and evaluate the performance behaviors and ability of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times, transaction rates, and other time sensitive issues).

Test Objective:	Verify performance behavior time for designated transactions or business cases under varying workload conditions.
Technique:	Modify data files to increase the number of transactions or the tests to increase the number of times each transaction occurs.
Completion Criteria:	Multiple transactions or multiple users: Successful completion of the tests without any failures and within acceptable time allocation.

Special Considerations:	· Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. · The databases used for load testing should be either actual size or scaled equally.

1.1.4 Stress Testing

Stress testing is a type of performance test implemented and executed to find errors due to low resources or competition for resources. Low memory or disk space may reveal defects in the target-of-test that aren't apparent under normal conditions. Other defects might result from competition for shared resources like database locks or network bandwidth. Stress testing can also be used to identify the peak workload the target-of-test can handle.

Test Objective:	Verify that the target-of-test functions properly and without error under the following stress conditions: · little or no memory available on the server (RAM and DASD) · maximum actual or physically capable number of clients connected or simulated · multiple users performing the same transactions against the same data or accounts · worst case transaction volume or mix (see Performance Testing above). Notes: The goal of Stress Testing might also be stated as identify and document the conditions under which the system FAILS to continue functioning properly. Stress Testing of the client is described under section 3.1.11, Configuration Testing.
Technique:	· Use tests developed for Performance Profiling or Load Testing. · To test limited resources, tests should be run on a single machine, and RAM and DASD on server should be reduced or limited. · For remaining stress tests, multiple clients should be used, either running the same tests or complementary tests to produce the worst-case transaction volume or mix.
Completion Criteria:	All planned tests are executed and specified system limits are reached or exceeded without the software failing or conditions under which system failure occurs is outside of the specified conditions
Special Considerations:	· Stressing the network may require network tools to load the network with messages or packets. · The DASD used for the system should temporarily be reduced to restrict the available space for the database to grow. · Synchronization of the simultaneous clients accessing of the same records or data accounts.

1.1.5 Volume Testing

Volume Testing subjects the target-of-test to large amounts of data to determine if limits are reached that cause the software to fail. Volume Testing also identifies the continuous maximum load or volume the target-of-test can handle for a given period. For example, if the target-of-test is processing a set of database records to generate a report, a Volume Test would use a large test database and check that the software behaved normally and produced the correct report.

Test Objective:	Verify that the target-of-test successfully functions under the following high volume scenarios: · Maximum (actual or physically- capable) number of clients connected, or simulated, all performing the same, worst case (performance) business function for an extended period. · Maximum database size has been reached (actual or scaled) and multiple queries or report transactions are executed simultaneously.
Technique:	· Use tests developed for Performance Profiling or Load Testing. · Multiple clients should be used, either running the same tests or complementary tests to produce the worst-case transaction volume or mix (see Stress Testing above) for an extended period. · Maximum database size is created (actual, scaled, or filled with representative data) and multiple clients used to run queries and report transactions simultaneously for extended periods.
Completion Criteria:	· All planned tests have been executed and specified system limits are reached or exceeded without the software or software failing.
Special Considerations:	What period of time would be considered an acceptable time for high volume conditions, as noted above?

1.1.6 Security and Access Control Testing

Security and Access Control Testing focus on two key areas of security:
· Application-level security, including access to the Data or Business Functions
· System-level Security, including logging into or remote access to the system.
Application-level security ensures that, based upon the desired security, actors are restricted to specific functions or use cases, or are limited in the data that is available to them. For example, everyone may be permitted to enter data and create new accounts, but only managers can delete them. If there is security at the data level, testing ensures that" user type one" can see all customer information, including financial data, however," user two" only sees the demographic data for the same client.
System-level security ensures that only those users granted access to the system are capable of accessing the applications and only through the appropriate gateways.

Test Objective:	● Application-level Security: Verify that an actor can access only those functions or data for which their user type is provided permissions. ● System-level Security: Verify that only those actors with access to the system and applications are permitted to access them.
Technique:	● Application-level Security: Identify and list each user type and the functions or data each type has permissions for. · Create tests for each user type and verify each permission by creating transactions specific to each user type. · Modify user type and re-run tests for same users. In each case, verify those additional functions or data are correctly available or denied. ● System-level Access: See Special Considerations below
Completion Criteria:	For each known actor type the appropriate function or data are available, and all transactions function as expected and run in prior Application Function tests.

Special Considerations:	Access to the system must be reviewed or discussed with the appropriate network or systems administrator. This testing may not be required as it may be a function of network or systems administration.

1.1.7 Failover and Recovery Testing

Failover and Recovery Testing ensures that the target-of-test can successfully failover and recover from a variety of hardware, software or network malfunctions with undue loss of data or data integrity.
Failover testing ensures that, for those systems that must be kept running, when a failover condition occurs, the alternate or backup systems properly "take over" for the failed system without loss of data or transactions.
Recovery testing is an antagonistic test process in which the application or system is exposed to extreme conditions, or simulated conditions, to cause a failure, such as device Input/Output (I/O) failures or invalid database pointers and keys. Recovery processes are invoked and the application or system is monitored and inspected to verify proper application, or system, and data recovery has been achieved

Test Objective:	Verify that recovery processes (manual or automated) properly restore the database, applications, and system to a desired, known, state. The following types of conditions are to be included in the testing: · power interruption to the client · power interruption to the server · communication interruption via network servers · interruption, communication, or power loss to DASD and or DASD controllers · incomplete cycles (data filter processes interrupted, data synchronization processes interrupted). · invalid database pointer or keys · invalid or corrupted data element in database

Technique:	Tests created for Function and Business Cycle testing should be used to create a series of transactions. Once the desired starting test point is reached, the following actions should be performed, or simulated, individually: · Power interruption to the client: power the PC down. · Power interruption to the server: simulate or initiate power down procedures for the server. · Interruption via network servers: simulate or initiate communication loss with the network (physically disconnect communication wires or power down network servers or routers. · Interruption, communication, or power loss to DASD and DASD controllers: simulate or physically eliminate communication with one or more DASD controllers or devices. Once the above conditions or simulated conditions are achieved, additional transactions should be executed and upon reaching this second test point state, recovery procedures should be invoked. Testing for incomplete cycles utilizes the same technique as described above except that the database processes themselves should be aborted or prematurely terminated. Testing for the following conditions requires that a known database state be achieved. Several database fields, pointers, and keys should be corrupted manually and directly within the database (via database tools). Additional transactions should be executed using the tests from Application Function and Business Cycle Testing and full cycles executed.
Completion Criteria:	In all cases above, the application, database, and system should, upon completion of recovery procedures, return to a known, desirable state. This state includes data corruption limited to the known corrupted fields, pointers or keys, and reports indicating the processes or transactions that were not completed due to interruptions.
Special Considerations:	· Recovery testing is highly intrusive. Procedures to disconnect cabling (simulating power or communication loss) may not be desirable or feasible. Alternative methods, such as diagnostic software tools may be required. · Resources from the Systems (or Computer Operations), Database, and Networking groups are required. · These tests should be run after hours or on an isolated machine.

1.1.8 Configuration Testing

Configuration testing verifies the operation of the target-of-test on different software and hardware configurations. In most production environments, the particular hardware specifications for the client workstations, network connections and database servers vary. Client workstations may have different software loaded_for example, applications, drivers, and so on_and at any one time, many different combinations may be active using different resources.

Test Objective:	Verify that the target-of-test functions properly on the required software browsers
Technique:	· Use Function Test scripts. · Execute selected transactions to simulate actor's interacting with the target-of-test and the non-target-of-test software.
Completion Criteria:	For each combination of the target-of-test and non-target-of-test software, all transactions are successfully completed without failure.
Special Considerations:	· The entire systems, netware, network servers, databases, and so on also needs to be documented as part of this test.

1.1.9 Installation Testing

These tests are not performed because there is nothing to install on Hillel Garage project.

1.2 Tools

The following tools will be employed for this project:

	Tool	Vendor/In-house	Version
Test Management	TasteLab		
Defect Tracking	Jira		
ASQ Tool for functional testing	postman		
ASQ Tool for performance testing	postman		
Project Management	jira		

Resources

[This section presents the recommended resources for the <Project Name> project, their main responsibilities, and their knowledge or skill set.]

1.3 Roles

This table shows the staffing assumptions for the project.

[NOTE: Delete or add items as appropriate.]

Human Resources		
Worker	Minimum Resources Recommended (number of full-time roles allocated)	Specific Responsibilities or Comments
Test Manager, Test Project Manager		Provides management oversight. Responsibilities: ● provide technical direction ● acquire appropriate resources ● provide management reporting
Test Designer		Identifies, prioritizes, and implements test cases. Responsibilities: ● generate test plan ● generate test model ● evaluate effectiveness of test effort
Tester		Executes the tests. Responsibilities: ● execute tests ● log results ● recover from errors ● document change requests
Test System Administrator		Ensures test environment and assets are managed and maintained. Responsibilities: ● administer test management system ● install and manage access to test systems

Database Administrator, Database Manager		Ensures test data (database) environment and assets are managed and maintained. Responsibilities: ● administer test data (database)
Designer		Identifies and defines the operations, attributes, and associations of the test classes. Responsibilities: ● identifies and defines the test classes ● identifies and defines the test packages
Implementer		Implements and unit tests the test classes and test packages. Responsibilities: ● creates the test classes and packages implemented in the test model

1.4 System

The following table sets forth the system resources for the testing project.

[The specific elements of the test system are not fully known at this time. It is recommended that the system simulate the production environment, scaling down the accesses and database sizes if and where appropriate.]

[Note: Delete or add items as appropriate.]

System Resources	
Resource	Name / Type
Database Server	
Network or Subnet	TBD
Server Name	TBD
Database Name	TBD
Client Test PC's	
Include special configuration requirements	TBD
Test Repository	
Network or Subnet	TBD
Server Name	TBD
Test Development PC's	TBD

Project Milestones

[Testing of <Project Name> should incorporate test activities for each of the test efforts identified in the previous sections. Separate project milestones should be identified to communicate project status accomplishments.]

Milestone Task	Effort	Start Date	End Date
Plan Test			
Design Test			
Implement Test			
Execute Test			
Evaluate Test			

Deliverables

[In this section, list the various documents, tools, and reports that will be created, by whom, delivered to who, and when delivered.]

1.5 Test Model

[This section identifies the reports that will be created and distributed from the test model. These artifacts in the test model need to be created or referenced in the ASQ tools.]

1.1 Test Logs

[Describe the method and tools used to record and report on the test results and testing status.]

1.2 Defect Reports

[In this section, identify the method and tools used to record, track, and report on test incidents and their status.]

● Appendix A Project Tasks

Below are the test-related tasks:

- Plan Test
  - identify requirements for test
  - assess risk
  - develop test strategy
  - identify test resources
  - create schedule
  - generate Test Plan
- Design Test
  - prepare workload analysis
  - identify and describe test cases
  - identify and structure test procedures
  - review and assess test coverage
- Implement Test
  - record or program test scripts
  - identify test-specific functionality in the Design and Implementation Model
  - establish external data sets
- Execute Test
  - execute Test procedures
  - evaluate execution of Test
  - recover from halted Test
  - verify the results
  - investigate unexpected results
  - log defects
- Evaluate Test
  - evaluate Test-case coverage
  - evaluate code coverage
  - analyze defects
  - determine if Test Completion Criteria and Success Criteria have been achieved