# 

# 

# 

# 

Test Plan for the Truck Pup App

Version 1.0

# 

# 

# Team Intereus

## 

## 

# Table of Contents

[**Title Page**](#_cbikzuj70dle) **0**

[**Table of Contents**](#_djoepqdapw0u) **1**

[**1. Objectives**](#_cahcdelbbopr) **3**

[1.1 Purpose](#_5jj75ttro15y) 3

[1.2 Scope](#_xwzbxpt96y75) 3

[1.3 References](#_pv4703a243qv) 4

[**2. Requirements for Test**](#_2w7cpw65wxrr) **5**

[2.1 Data and Database Integrity Testing](#_vxef01jayjll) 5

[2.2 Function Testing](#_wayixwh5jda5) 5

[2.3 Business Cycle Testing](#_3vzuirjxilf) 5

[2.4 User Interface Testing](#_97wujmv6v4d0) 5

[2.5 Performance Testing](#_wex61o1gr0nj) 6

[2.6 Load Testing](#_lstb4x6mahx0) 6

[2.7 Configuration Testing](#_t4zpbjdboz11) 6

[2.8 Installation Testing](#_o1gsmzgf9vwz) 6

[**3. Test Strategy**](#_2r5p83qdmuzn) **7**

[3.1 Testing Types](#_tc4zfz688gqc) 7

[3.1.1 Data and Database Integrity Testing](#_z7uwcw396log) 7

[3.1.2 Function Testing](#_8a0tl3dg67ww) 8

[3.1.3 Business Cycle Testing](#_hlnpo91g077l) 9

[3.1.4 User Interface Testing](#_68yips2aexpr) 9

[3.1.5 Performance Profiling](#_c7tb2sdf0rnb) 10

[3.1.6 Load Testing](#_fbg9mjbglyha) 12

[3.1.7 Stress Testing](#_ufeb2m6tqad2) 13

[3.1.8 Volume Testing](#_1vtf69u8nhuz) 13

[3.1.9 Security and Access Control Testing](#_3sru2561wjdt) 13

[3.1.10 Failover and Recovery Testing](#_9n1g42l5ceub) 14

[3.1.11 Configuration Testing](#_dkqfoyh7sxw5) 14

[3.1.12 Installation Testing](#_bw6eyfj5ntg2) 16

[3.2 Tools](#_cu5bczvnvcgm) 16

[**4. Resources**](#_ea9qgofyizf) **17**

[4.1 Roles](#_cxakgmrsij7r) 17

[4.2 System](#_1xqe9kasjscd) 17

[**5. Project Milestones**](#_1r4lf0sgfijx) **19**

[**6. Deliverables**](#_1vgfe9cpz3zo) **20**

[6.1 Test Suite](#_j0w8tiwn7lcb) 20

[6.2 Test Logs](#_k9hfxdhz9tp) 20

[**7. Project Tasks**](#_4tzf4wqdtig5) **21**

## 

## 

# 1. Objectives

#### 1.1 Purpose

This document describes the plan for testing the architectural prototype of the Truck Pup App.

This Test Plan document supports the following objectives:  
 Identify existing project information and the software that should be tested.  
 List the recommended test requirements (high level).  
 Recommend and describe the testing strategies to be employed.  
 Identify the required resources and provide an estimate of the test efforts.  
 List the deliverable elements of the test activities.

#### 1.2 Scope

This Test Plan describes the integration and system test that will be conducted on the architectural prototype following integration of the subsystem and components identified in the Integration Build Plan for the Prototype.   
 It is assumed that unit testing already provided through black box testing, extensive coverage of source code, and testing of all module interfaces.   
 The purpose of assembling the architectural prototype was to test feasibility and performance of the selected architecture. It is critical that all systems and subsystem interfaces be tested as well as system performance at this early stage. Testing of system functionality and features will not be conducted on the prototype.  
   
 The interfaces between the following subsystems will be tested:  
   
 1. Truck Registration  
 2. Adding Menu Items  
 3. View Trucks on map (Including Trucks that are close to each other)  
   
 The external interfaces to the following devices will be tested:  
   
 1. Local PCs  
 2. Remote PCs  
 3. Mobile Devices  
   
 The most critical performance measures to test are:  
   
 1. Response time for large number of trucks on record  
 2. Response time for large number of food items entered  
 3. Possible bypass of user authentication  
 4. How system handles symbols in truck names and/or menu items

#### 1.3 References

1. [http://pusher.com](http://pusher.com/)
2. [https://firebase.google.com](https://firebase.google.com/)
3. [https://ionicframework.com](http://ionicframework.com/)
4. <https://asp.net/mvc>
5. [https://mdbootstrap.com](https://mdbootstrap.com/)

### 

# 2. Requirements for Test

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

#### 2.1 Data and Database Integrity Testing

* Verify access to Truck Listing/Availability Database.
* Verify simultaneous Truck Listing/Availability Accesses.
* Verify access to Truck Menu Database.
* Verify simultaneous Truck Menu Accesses.

#### 2.2 Function Testing

The system shall interface with the existing Truck database and shall support the data format.  
  
The server component of the system shall operate on the Server and shall run under the Windows and Android Systems.  
  
The client component of the system shall operate on any personal computer with a 486 Microprocessor or better.

#### 2.3 Business Cycle Testing

None

#### 2.4 User Interface Testing

* Verify ease of navigation through a sample set of screens.
* Verify sample screens conform to GUI standards.

#### 2.5 Performance Testing

* Verify response time to login to system
* Verify response time to access Truck Database
* Verify response time to access Truck Menu Database

#### 2.6 Load Testing

* Verify system response when loaded with large number of trucks on system
* Verify login from a mobile device
* Verify login security through username and password mechanisms

#### 2.7 Configuration Testing

* The client component on the system shall run on Windows 8.0, 8.1, 10.0, and Android Devices
* The web-based interface for the Truck Pup App shall run in Internet Explorer 11 and Google Chrome 57
* The web-based interface shall be compatible with the Java 8 VM runtime environment

#### 2.8 Installation Testing

None

### 

### 

# 3. Test Strategy

The Test Strategy presents the recommended approach to the testing of the software applications. The previous section on Test Requirements described what will be tested; this describes how it will be tested.  
  
The main consideration for the testing strategy are the techniques to be used and the criterion for knowing when the testing is completed. In addition to the consideration provided for each test below, testing should only be executed using known, controlled databases, in secured environments.  
  
The following test strategy is generic in nature and is meant to apply to the requirements listed in Section 4 of this document.

#### 3.1 Testing Types

##### 3.1.1 Data and Database Integrity Testing

The databases and the database processes should be tested as separate systems. These systems should be tested without the applications (as the interface to the data). Additional research into the SQL Database needs to be performed to identify the tools/techniques that may exist to support the testing identified below.   
  
Test Objective:

Ensure Database access methods and processes function properly and without data corruption  
   
Technique:

1. Invoke each database access method and process, seeding each with valid and invalid data (or requests for data)
2. Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved (for the correct reason).

Completion Criteria: All database access methods and processes function as designed and return without errors. Database access methods and processes do not accept invalid input.  
   
Special Considerations :

* Testing may require an SQL development environment or drivers to enter or modify data directly in the database.
* Processes should be invoked manually
* Small or minimally sized databases (limited number of records) should be used to increase the visibility of any non-acceptable events.

##### 3.1.2 Function Testing

Testing of the application should focus on any target requirements 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 GUI and analyzing the output (results). Identified below is an outline of the testing recommended for each application.  
  
Test Objective: Ensure proper application 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:

1. The expected results occur when valid data is used.
2. The appropriate error / warning messages are displayed when invalid data is used.
3. Each business rule is properly applied.

Completion Criteria:

* All planned tests have been executed
* All identified defects have been addressed.

Special Considerations: N/A

##### 3.1.3 Business Cycle Testing

This section is not applicable to test of the architectural prototype.

##### 3.1.4 User Interface Testing

User Interface 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 applications. 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:

1. Navigation through the application properly reflects business functions and requirements, including window to window, field to field, and use of access methods (tab keys, mouse movements, accelerator keys, and finger dragging for mobile version)
2. Window objects and characteristics, such as menus, size, position, state, and focus conform to standards.

Technique:

Create / 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: N/A

##### 3.1.5 Performance Profiling

Performance testing measures response times, transaction rates, and other time sensitive requirements. The goal of performance testing is to verify and validate the performance requirements have been achieved. Performance testing is usually executed several times, each using a different "background load" on the system. The initial test should be performed with a "nominal" load, similar to the normal load experienced (or anticipated) on the target system. A second performance test is run using a peak load.  
   
Additionally, Performance tests can be used to profile and tune a system's performance as a function of conditions such as workload or hardware configuration.  
  
NOTE: Transactions below refer to "logical business transactions." These transactions are defined as specific functions that an end user of the system is expected to perform using the application, such as add or modify a given contract (or truck in this case).  
  
Test Objective: Validate System Response time for designed transactions or business functions under the following two conditions:

1. normal anticipated volume
2. anticipated worst case volume

Technique:

* Use Test scripts developed for Business Model Testing (System Testing).
* Modify Data files (to increase the number of transactions) or modify 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 / Single User: Successful completion of the test scripts without any failures and within the expected / required time allocations (per transaction)
* Multiple Transactions / 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" load 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 SQL calls
* Create "Virtual" user load to simulate many (usually several hundred) clients. 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 database used for performance testing should be either actual size, or scaled equally.

##### 3.1.6 Load Testing

Load Testing subjects the system-under-test to varying workloads to evaluate the system's ability 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 (response time, transaction rates, and other time sensitive issues).  
  
NOTE: Transactions below refer to "logical business transactions." These transactions are defined as specific functions than an end user of the system is expected to perform using the application, such as add or modify a given contract.  
  
Test Objective: Verify System response time for designated transactions or business cases under varying workload conditions.  
  
Technique:

* Use tests developed for Business Cycle Testing.
* 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 / Multiple Users: Successful completion of the tests without any failures and within acceptable time allocations.  
  
Special Considerations:

Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.

The database used for load testing should be either actual size, or scaled equally

##### 3.1.7 Stress Testing

This section is not applicable to test of the architectural prototype as there is only one server planned.

##### 3.1.8 Volume Testing

This section is not applicable to test of the architectural prototype since it is impractical to fill up server.

##### 3.1.9 Security and Access Control Testing

Security and Access Control Testing focus on two key areas of security:  
  
Application security, including access to the Data or Business Functions, and  
System security, including logging into / remote access to the system.  
  
Test Objectives:

Function / Data Security: Verify that user can access only those functions / data for which their user type is provided permissions.  
System Security: Verify that only those users with access to the system and application are permitted to access them.  
  
Technique:

Function / Data Security: Identify and list each user type and the functions / data each type has permissions for.

* Create tests for each user type and verify 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 / data are correctly available or denied.

System Access (see special considerations below)   
  
Completion Criteria:

For each known user type, the appropriate function / data are available and all  
transactions function as expected and run in prior Application Function tests. Functions / data are not available to unknown user.  
  
Special Consideration:

Access to the system must be reviewed / 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.

##### 3.1.10 Failover and Recovery Testing

This section is not applicable to test of the architectural prototype.

##### 3.1.11 Configuration Testing

Configuration testing verifies operation of the 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 (e.g. applications, drivers, etc.) and at any one time many different combinations may be active and using different resources.  
  
Test Objective:

Validate and verify that the client Application functions properly on the prescribed client  
workstations.  
  
Technique:

Open / close various PC applications, either as part of the test or prior to the start of the test.

Execute selected transactions to simulate user activities into and out of various PC applications.

Repeat the above process, minimizing the available conventional memory on the client.  
  
Completion Criteria:

For each combination of the Prototype and PC application, transactions are successfully completed without failure.  
  
Special Considerations:

What PC Applications are available, accessible on the client?  
What applications are typically used?  
What data are the applications running? (i.e. large spreadsheet opened in Excel, 100 page document in Word)

The entire systems, network servers, databases, etc. should also be documented as part of this test.

##### 3.1.12 Installation Testing

This section is not applicable to test of the Truck Pup App prototype.

#### 3.2 Tools

The following tools will be employed for the testing of the architectural prototype:

|  |  |
| --- | --- |
| **Tool** | **Version** |
| Live HTTP Headers | 0.17.1-signed.1-signed |
| Visual Studio 2015 | 14.0.25431.01  Update 3 |

### 

### 

# 4. Resources

This section presents the recommended resources for testing the Truck Pup App prototype, their main responsibilities, and their knowledge or skill set.

#### 4.1 Roles

This table shows the staffing assumptions for the test of the prototype:

|  |  |  |
| --- | --- | --- |
| **Role** | **Minimum Resources Recommended** | **Specific Responsibilities/Comments** |
| Team Leader | Clayton Mcguire | Backend Programming |
| Team Member | Branden Wagner | Mobile App Programming |
| Team Member | Todd Ayers | Mobile App Programming |
| Team Member | Jason Lytle | Primary Tester |

#### 4.2 System

The following table sets forth the system resources for testing the Truck Pup App prototype:

System Resources

|  |  |
| --- | --- |
| **Resource** | **Description** |
| Server | Dedicated Virtual Server Hosted by Third Party |
| Mobile Devices | Various Android and iOS mobile devices |
| Workstations | Various Mac OS and Windows terminals |

## 

## 

# 5. Project Milestones

Testing of the Truck Pup App Prototype incorporates test activities for each of the test efforts identified in the previous sections. Separate project milestones are identified to communicate project status and accomplishments.

|  |  |  |
| --- | --- | --- |
| **Milestone Task** | **Start Date** | **End Date** |
| Prototype Test Planning | 03/13 | 03/18 |
| Prototype Test Design | 03/22 | 03/27 |
| Prototype Test Development | 03/29 | 04/13 |
| Prototype Test Execution | 03/29 | 04/19 |
| Prototype Test Evaluation | 04/20 | 05/01 |

### 

### 

# 6. Deliverables

The deliverables of the test activities as defined in this Test Plan are outlined in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Deliverable** | **Owner** | **Review/**  **Distribution** | **Due Date** |
| Close HTML Injection Attack Vector | Jason Lytle | Jason Lytle | 03/31 |
| Fix for Calendar Render bug | Clayton McGuire | Clayton McGuire | 4/14 |

#### 

#### 6.1 Test Suite

The Test Suite will define all the test cases and the test scripts which are associated with each test case.

#### 6.2 Test Logs

Github repository and issues sections are used to show test executions and discussions.

### 

### 

# 7. Project Tasks

Below are the test related tasks for testing the Truck Pup App prototype:

**Plan Test**

Identify Requirements for Test  
  
 Assess Risk  
  
 Develop Test Strategy  
  
 Identify Test Resources  
  
 Create Schedule  
  
 Generate Test Plan

**Design Test**

Workload Analysis (not applicable for prototype)  
  
 Develop Test Suite  
  
 Identify and Structure Test Scripts (where applicable)  
  
 Review and Assess Test Coverage

**Implement Test**

Setup Test Environment  
  
 Record or Program Test Scripts (where applicable)  
  
 Develop Test Stubs and Drivers  
  
 Identify Test-Specific functionality in the design and implementation model  
  
 Establish External Data sets

**Execute Test**

Execute Test Scripts (where applicable)  
  
 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  
  
 Create Test Evaluation Report