Cheetah
Test Plan

Version 1.0

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

Date	Version	Description	Author
02/05/2017	1.0	Initial Version	Sara Olszewski

Table of Contents

Test Plan

1. Introduction

1.1 Purpose

The purpose of the Iteration Test Plan is to gather all of the information necessary to plan and control the test effort for a given iteration. It describes the approach to testing the software, and is the top-level plan generated and used by managers to direct the test effort.

This *Test Plan* for the Cheetah supports the following objectives:

- Models
- Views
- Controllers

1.2 Scope

Testing is done with Unit testing.

1.3 Intended Audience

The audience are the customers of Cheetah, also our teacher.

1.4 Document Terminology and Acronyms

N.a.

1.5 References

N.a.

1.6 Document Structure

N.a.

2. Evaluation Mission and Test Motivation

Testing is done to provide a stable software, that does, what it is intended to do!

2.1 Background

N.a.

2.2 Evaluation Mission

Testing is done to provide a stable software. And we will fulfill the goal by the following points.

- find as many bugs as possible
- find important problems
- certify a standard

2.3 Test Motivators

- technical risks
- functional requirements
- nonfunctional requirements

3. Target Test Items

The listing below identifies those test items—software, hardware, and supporting product elements —that have been identified as targets for testing. This list represents what items will be tested.

- UC Register (Password, Email)

- Login ()

4. Outline of Planned Tests

4.1 Outline of Test Inclusions

Unittests, Integrationtests

4.2 Outline of Other Candidates for Potential Inclusion

N.a.

4.3 Outline of Test Exclusions

Functional Test

5. Test Approach

Unittests
Integrationstests

5.1 Initial Test-Idea Catalogs and Other Reference Sources

5.2 Testing Techniques and Types

5.2.1 Data and Database Integrity Testing

Technique Objective:	Database and saving in it
Technique:	• Invoke each database access method and process, seeding each with valid and invalid data or requests for data. • Inspect the database to ensure the data has been populated as intended and all database events have occurred properly, or review the returned data to ensure that the correct data was retrieved for the correct reasons
Oracles:	Data is created and saved
Required Tools:	• Ruby on Rails • MySQL Database
Success Criteria:	Data has been correctly created and saved
Special Considerations:	• 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.

5.2.2 User Interface Testing

Technique Objective:	Navigation through the web application Window objects and characteristics can be exercised–such as menus, size, position, state, and focus.
Technique:	● Testing the API ● User ○ Create User ○ Login User ○ Is logged in ○ change password ● Tables of data
Oracles:	If test are correct it works correct
Required Tools:	Ruby on Rails, web-browser

Success Criteria:	All test run successfully
Special Considerations:	

5.2.3 Installation Testing

Technique Objective:	• new installation: a new machine, never installed previously with Cheetah web application
Technique:	• Develop automated or manual scripts to validate the condition of the target machine. o new: never installed o same or older version already installed • Launch or perform installation. • Using a predetermined subset of Function Test scripts, run the transactions.
Oracles:	Works correctly
Required Tools:	• base configuration imager and restorer • installation monitoring tools (registry, hard disk, CPU, memory, and so on) • webbrowser
Success Criteria:	Works correctly as Project planned
Special Considerations:	

6. Entry and Exit Criteria

6.1 Test Plan

6.1.1 Test Plan Entry Criteria

N.a.

6.1.2 Test Plan Exit Criteria

N.a.

6.1.3 Suspension and Resumption Criteria

N.a.

6.2 Test Cycles

6.2.1 Test Cycle Entry Criteria

N.a.

6.2.2 Test Cycle Exit Criteria

N.a.

6.2.3 Test Cycle Abnormal Termination

N.a.

7. Deliverables

N.a.

7.1 Test Evaluation Summaries

N.a.

7.2 Reporting on Test Coverage

N.a.

7.3 Perceived Quality Reports

N.a.

7.4 Incident Logs and Change Requests

N.a.

7.5 Smoke Test Suite and Supporting Test Scripts

N.a.

7.6 Additional Work Products

N.a.

7.6.1 Detailed Test Results

N.a.

7.6.2 Additional Automated Functional Test Scripts

N.a.

7.6.3 Test Guidelines

N.a.

7.6.4 Traceability Matrices

N.a.

8. Testing Workflow

[Provide an outline of the workflow to be followed by the Test team in the development and execution of this ***Test Plan****.]*

The specific testing workflow that you will use should be documented separately in the project's Development Case. It should explain how the project has customized the base RUP test workflow (typically on a phase-by-phase basis). In most cases, we recommend you place a reference in this section of the ***Test Plan*** *to the relevant section of the Development Case. It might be both useful and sufficient to simply include a diagram or image depicting your test workflow.*

More specific details of the individual testing tasks are defined in a number of different ways, depending on project culture; for example:

- *defined as a list of tasks in this section of the* ***Test Plan****, or in an accompanying appendix*
- *defined in a central project schedule (often in a scheduling tool such as Microsoft Project)*
- *documented in individual, "dynamic" to-do lists for each team member, which are usually too detailed to be placed in the* ***Test Plan***
- *documented on a centrally located whiteboard and updated dynamically*
- *not formally documented at all*

Based on your project culture, you should either list your specific testing tasks here or provide some descriptive text explaining the process your team uses to handle detailed task planning and provide a reference to where *the details are stored, if appropriate.*

For Master Test Plans, we recommend avoiding detailed task planning, which is often an unproductive effort if done as a front-loaded activity at the beginning of the project. A Master Test Plan might usefully describe the phases and the number of iterations, and give an indication of what types of testing are generally planned for each Phase or Iteration.

***Note**: Where process and detailed planning information is recorded centrally and separately from this Test Plan, you will have to manage the issues that will arise from having duplicate copies of the same information. To avoid team members referencing out-of-date information, we suggest that in this situation you place the minimum amount of process and planning information within the Test Plan to make ongoing maintenance easier and simply reference the "Master" source material.]*

9. Environmental Needs

*[This section presents the non-human resources required for the **Test Plan**.]*

9.1 Base System Hardware

The following table sets forth the system resources for the test effort presented in this *Test Plan*.

[The specific elements of the test system may not be fully understood in early iterations, so expect this section to be completed over time. We recommend that the system simulates the production environment, scaling down the concurrent access and database size, and so forth, if and where appropriate.]

*[**Note**: Add or delete items as appropriate.]*

System Resources		
Resource	**Quantity**	**Name and Type**
Database Server		
Network or Subnet		TBD
Server Name		TBD
Database Name		TBD
Client Test PCs		
Include special configuration requirements		TBD
Test Repository		
Network or Subnet		TBD
Server Name		TBD
Test Development PCs		TBD

9.2 Base Software Elements in the Test Environment

The following base software elements are required in the test environment for this *Test Plan*.

[Note: Add or delete items as appropriate.]

Software Element Name	Version	Type and Other Notes
NT Workstation		Operating System
Windows 2000		Operating System

Software Element Name	Version	Type and Other Notes
Internet Explorer		Internet Browser
Netscape Navigator		Internet Browser
MS Outlook		eMail Client software
Network Associates McAfee Virus Checker		Virus Detection and Recovery Software

9.3 Productivity and Support Tools

The following tools will be employed to support the test process for this *Test Plan*.

[Note: Add or delete items as appropriate.]

Tool Category or Type	Tool Brand Name	Vendor or In-house	Version
Test Management			
Defect Tracking			
ASQ Tool for functional testing			
ASQ Tool for performance testing			
Test Coverage Monitor or Profiler			
Project Management			
DBMS tools			

9.4 Test Environment Configurations

The following Test Environment Configurations needs to be provided and supported for this project.

Configuration Name	Description	Implemented in Physical Configuration
Average user configuration		
Minimal configuration supported		
Visually and mobility challenged		
International Double Byte OS		
Network installation (not client)		

10. Responsibilities, Staffing, and Training Needs

[This section presents the required resources to address the test effort outlined in the ***Test Plan****—the main responsibilities, and the knowledge or skill sets required of those resources.]*

10.1 People and Roles

This table shows the staffing assumptions for the test effort.

*[**Note**: Add or delete items as appropriate.]*

Human Resources		
Role	**Minimum Resources Recommended** **(number of full-time roles allocated)**	**Specific Responsibilities or Comments**
Test Manager		Provides management oversight. Responsibilities include: • planning and logistics • agree mission • identify motivators • acquire appropriate resources • present management reporting • advocate the interests of test • evaluate effectiveness of test effort
Test Analyst		Identifies and defines the specific tests to be conducted. Responsibilities include: • identify test ideas • define test details • determine test results • document change requests • evaluate product quality
Test Designer		Defines the technical approach to the implementation of the test effort. Responsibilities include: • define test approach • define test automation architecture • verify test techniques • define testability elements • structure test implementation
Tester		Implements and executes the tests. Responsibilities include: • implement tests and test suites • execute test suites • log results • analyze and recover from test failures • document incidents

Human Resources		
Role	**Minimum Resources Recommended** **(number of full-time roles allocated)**	**Specific Responsibilities or Comments**
Test System Administrator		Ensures test environment and assets are managed and maintained. Responsibilities include: • administer test management system • install and support access to, and recovery of, test environment configurations and test labs
Database Administrator, Database Manager		Ensures test data (database) environment and assets are managed and maintained. Responsibilities include: • support the administration of test data and test beds (database).
Designer		Identifies and defines the operations, attributes, and associations of the test classes. Responsibilities include: • defines the test classes required to support testability requirements as defined by the test team
Implementer		Implements and unit tests the test classes and test packages. Responsibilities include: • creates the test components required to support testability requirements as defined by the designer

10.2 Staffing and Training Needs

This section outlines how to approach staffing and training the test roles for the project.

N.a.

11. Iteration Milestones

Milestone	Planned Start Date	Actual Start Date	Planned End Date	Actual End Date
Iteration Plan agreed				
Iteration starts				
Requirements baselined				
Architecture baselined				
User Interface baselined				

Milestone	Planned Start Date	Actual Start Date	Planned End Date	Actual End Date
First Build delivered to test				
First Build accepted into test				
First Build test cycle finishes				
[Build Two will not be tested]				
Third Build delivered to test				
Third Build accepted into test				
Third Build test cycle finishes				
Fourth Build delivered to test				
Fourth Build accepted into test				
Iteration Assessment review				
Iteration ends				

12. Risks, Dependencies, Assumptions, and Constraints

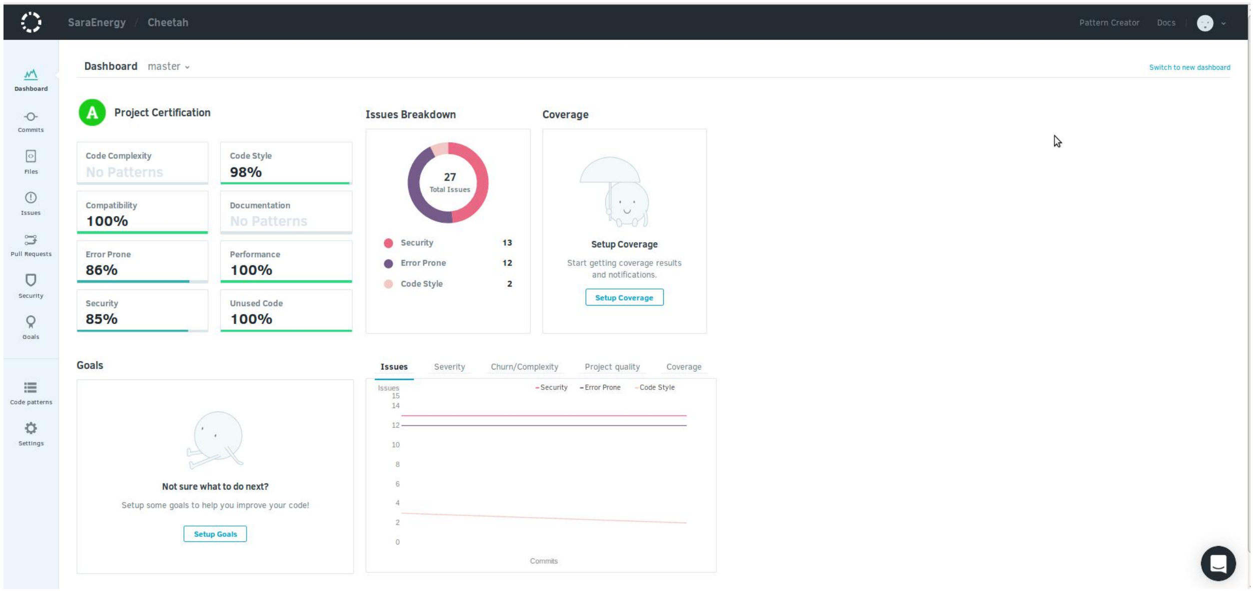
Risk	Mitigation Strategy	Contingency (Risk is realized)
Prerequisite entry criteria is not met.	<Tester> will define the prerequisites that must be met before Load Testing can start. <Customer> will endeavor to meet prerequisites indicated by <Tester>.	• Meet outstanding prerequisites • Consider Load Test Failure
Test data proves to be inadequate.	<Customer> will ensure a full set of suitable and protected test data is available. <Tester> will indicate what is required and will verify the suitability of test data.	• Redefine test data • Review Test Plan and modify • components (that is, scripts) • Consider Load Test Failure
Database requires refresh.	<System Admin> will endeavor to ensure the Database is regularly refreshed as required by <Tester>.	• Restore data and restart • Clear Database

Dependency between	Potential Impact of Dependency	Owners

Assumption to be proven	Impact of Assumption being incorrect	Owners

Constraint on	Impact Constraint has on test effort	Owners

Before:



Dashboard master

A Project Certification

Code Complexity No Patterns
Code Style 98%
Compatibility 100%
Documentation No Patterns
Error Prone 86%
Performance 100%
Security 85%
Unused Code 100%

Issues Breakdown

27 Total Issues

Security	13
Error Prone	12
Code Style	2

Coverage

Setup Coverage

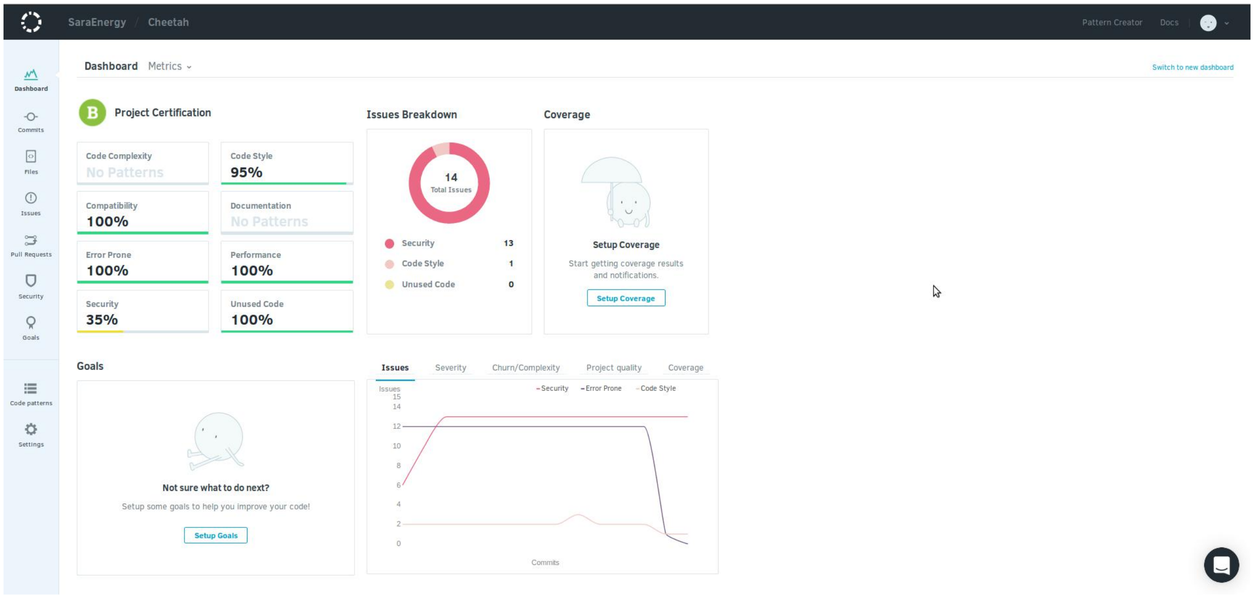
Start getting coverage results and notifications.

Goals

Not sure what to do next?

Setup some goals to help you improve your code!

After:



Dashboard Metrics

B Project Certification

Code Complexity No Patterns
Code Style 95%
Compatibility 100%
Documentation No Patterns
Error Prone 100%
Performance 100%
Security 35%
Unused Code 100%

Issues Breakdown

14 Total Issues

Security	13
Code Style	1
Unused Code	0

Coverage

Setup Coverage

Start getting coverage results and notifications.

Goals

Not sure what to do next?

Setup some goals to help you improve your code!