



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS 301 MINI PROJECT

Top Level Integration Testing

Elana Kuun u12029522
Hlavutelo Maluleke u12318109
Estian Rosslee u12223426
David Breetzke u12056503
Sylvester Mpungane u11241617
Phethile Mkhabela u12097561
Renaldo van Dyk u12204359
Antonia Michael u13014171
Herman Keuris u13037618
Jaco-Louis Kruger u13025105

Github Repository

April 2015

Contents

1	Introduction	3
2	Testing Results	3
2.1	Top Level A Testing Results	3
2.1.1	Functional Testing	3
2.1.2	Non-functional Testing	3
2.2	Top Level B Testing Results	5
2.2.1	Functional Testing	5
2.2.2	Non-functional Testing	5

1 Introduction

This document contains the findings of the various activities performed as per testing of the Buzz Space system top level integration.

2 Testing Results

2.1 Top Level A Testing Results

2.1.1 Functional Testing

2.1.2 Non-functional Testing

1. Usability:

The system is usable because firstly the necessary actions that a user can take appear in a navigation bar at the top of the screen. Thus it is easy for a novice user to be able to know what to click on and navigate the website.

The interface is not cluttered, and only basic functionality is displayed on the home screen, making the system more learnable. The buttons are labelled with text rather than with graphical icons, and the text on the button is quite explanatory, which makes their purpose more clear.

Larger headings are used to label the different sections, for example under the Manage Constraints tab there are large headings to indicate the Existing constraints section and the Add new constraint section. This again contributes to ease of use for the novice first year user.

Through these mechanisms the system is memorable hence it is also be understandable.

2. Integrability:

The system is able to address future integration requirements by providing access to its services using widely adopted public standards such as firstly having separate npm packages for all the different modules. The packages are stored on Synopia. Also, electrolyte is being used in the server to provide a dependency injection. The HandleBars server is the main server that needs to be used to test and integrate all the modules on. The routes/index.js, routes/infrastructure.js and routes/content.js

files use express to route the different hbs files for the different modules in order to integrate the infrastructure and content subsystems into the main system.

A separate file is used to establish the connection to the database to avoid having this done in all the separate files. Also, global variables are now used such as the global password and username for example.

A document has been provided via email and a README file has been provided to specify important standards and regulations that must be followed.

The functional code in the separate packages must be placed in an exportable function taking parameters such as the database or settings, and this is done as part of the electrolyte dependency injection.

Exports are also used in the different files to make the code accessible to the other files.

3. Deployability:

The system is deployable on Linux servers as we have run it using Ubuntu 14.04 Linux and the system was able to run. The following screenshot shows the system running on a Linux server:

The system is deployable on an environment using different databases for persistence of the Buzz database because the Handlebars server contains a folder called `node_modules`, and it contains the `buzz_database` package. This package can easily be swapped out and an alternative database package can be plugged in, that the system can use due to the flexibility of this server. As long as the new package has the same name so that the files that require the database do not need to be changed, no major changes will need to be made. //screenshot

The system is deployable in environments where the user authentication credentials and roles are sourced from different repositories. The following screenshot indicates that the Handlebars server contains a folder called `node_modules`, and it contains the `buzz_csds` package. //screenshot

This package can easily be swapped out and an alternative data source package with different credentials and roles can be used, due to the pluggability of the system and dependency injection employed by the

top level team. The new package must just be named the same as the old one to avoid errors where the file is required from the package in the code.

2.2 Top Level B Testing Results

2.2.1 Functional Testing

2.2.2 Non-functional Testing

1. Auditability:

The Buzz Space system is partially auditable, only the requests for the user serves of the system are logged. The logged request only contains:

- User serves requested.
- Request object stringified as JSON

There are no log responses that are provided by the system, and the serves required to extract information from the audit logs is also not available not provided.

2. Testability:

The Buzz system is divided into manageable components (modules) for each use case. Breaking the system into these components makes make the system testable through unit testing using mock objects. The Buzz system is a modular system, the system components are pluggable and makes automated integration testing simple as the component is integrated in the entire system.