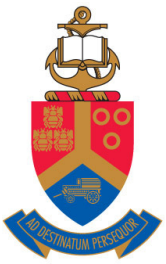

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

COS301 Mini Project Functional Testing - Space module

Group Members:

Xolieswa Ntshingila 13410378
Godfrey Mathe 13103394
Renette Ros 13007557
Byron Dinkelmann 11057638
Elzahn Botha 13033922
Name Surname 13070305
Lelethu Zazaza 13028023
Name Surname 12026442
Maria Qumayo 29461775

Git repository link:

`https://github.com/u12026442/cos301_testing`

Version One

April 22, 2015

Contents

1 Functional Requirements

1.1 Login and Admnistartive user

Space A

Functional Testing - for each use case tested * either success or a list of violations of the contract requirements (pre- and post-condition violations or data structure requirements) * a test coverage analysis reporting which percentage of the use cases have been covered by the testing

2. Non-functional testing/assessment - any performance, scalability, maintainability, reliability, usability, ... problems identified with evidence for the identified problem.

- The Buzz Space system has to accommodate and host a multitude of users concurrently, thus making it prone to various malfunctions and glitches
- The Space needs to be monitored in real time at all times to ensure relevance of topics and subject matters.
- The rating and tagging functionality need to be fair and accurate
- Sufficient feedback and updates of the Buzz Space state must be provided to the users. Users that create threads or just comment on one.
- General software control and application usage..

Space B

Functional Testing - for each use case tested * either success or a list of violations of the contract requirements (pre- and post-condition violations or data structure requirements) * a test coverage analysis reporting which percentage of the use cases have been covered by the testing

2. Non-functional testing/assessment - any performance, scalability, maintainability, reliability, usability, ... problems identified with evidence for the identified problem.

1.2 getProfileForUser

The getProfileForUser is a critical use case in the Spaces module as it used to return the profile a user has on the Buzz Space.
The implementation of this use case is inclusive of a query service which returns the relevant profile.
Before the query is made there is a requirement that a Buzz Space exists and that a user is associated with a Buzz Space.
Thereafter the result from the query is returned with no other modifications to the Buzz Space.
Detailed below is an evaluation of Buzz Space A and Buzz Space B to determine whether or not they conform to the pre and post conditions required by the getProfileForUser service contract.

1.2.1 Spaces A

Pre-condition: BuzzSpace is active
Result: Failure
Discussion: There is no apparent evidence present in the getProfileForUser use case that indicates that a Buzz Space is tested for activness/inactivess. The use case is conducted under the assumption that the Buzz Space is active, this is in the violation of the service contract pre-condition.

Pre-condition: User is associated to the relevant Buzz Space
Result: Success
Discussion: The getProfileForUser use case conducts the query service by ensuring that the current user being searched is associated with the respective Buzz Space.

1.2.2 Spaces B

Pre-condition: Buzz Space is active
Result: Failure
Discussion: There is no apparent evidence present in the getProfileForUser use case that indicates that a Buzz Space is tested for activness/inactivess. The use case is conducted under the assumption that the Buzz Space is active, this is in the violation of the service contract pre-condition.

Pre-condition: User is associated to the relevant Buzz Space
Result: Success
Discussion: The getProfileForUser use case conducts the query service by ensuring that the current user being searched is associated with the respective Buzz Space.

1.3 Close Buzz Space

Close BuzzSpace use case is important across the Buzz System. It simply sets the buzzSpace to inactive.

1.3.1 Spaces A

Pre-condition: BuzzSpace for current year and module exists
Result: Success
Comments: There exists proof that shows that a Space is tested for being active or not in the Moongoose Database.
Pre-condition: -User is authorised to create BuzzSpace

Result: Success
Comments: There exists proof that shows that a User is tested for being authorised to create a space.
Post-condition: BuzzSpace is not active
Result: Success
Comments: There exists proof that the isOpen field is set to false.

1.3.2 Spaces B

Pre-condition: BuzzSpace for current year and module exists
Result: Success
Comments: There exists proof that shows that a Space is tested for being active or not in the Database.
Pre-condition: User is authorised to create BuzzSpace
Result: Success
Comments: There exists proof that shows that a User is tested for being authorised to create a space.
Post-condition: BuzzSpace is not active
Result: Success
Comments: There exists proof that the isOpen field is set to false.

2 Non-Functional Requirements

We identified problems with the following non-functional requirements:

3 Conclusion