Introduction

Purpose of document

This document aims to provide an architectural overview of the Crowd/tidbits system, and to describe the different aspects of the system with regard to architectural decisions made on the structure of the system.

Scope of project

The system needs to provide a mobile application allowing users to upload messages that are relevant to a specific location. To this end, an iOS/Android application, a back-end HTTP server, and a persistence unit.

Access channels

The system provides two points of access; both human access channels; namely the application proper and a web based CMS allowing administrators to interface with the database.

The application needs to run on mobile devices, and must have support for iOS version 7 and above, as well as Android version 3 and above. Although this is not strictly a requirement, adhering to the design guidelines of each operating system will be advantageous and is recommended.

The web front end, containing the CMS, will be a module that forms part of the back end server. This will run as a servlet, which is accessed through the server URL. Given that only system administrators will have access to this interface, having it run as part of the server will not have any negative impact.

Integration channels

Internal

Two internal integration interfaces are required. The server needs to integrate with the DBMS, as well as with the application.

The server-application integration will take place over HTTP POST calls, and therefore security is paramount to the implementation of the communication. Making secure calls with HTTPS, along with hashing of secret data such as passwords, are both requirements that must be taken into consideration.

The server and the DBMS are likely to run on the same server, so the communication channels between the two need not be extremely secure; however, to prepare for future scaling secure channels should be considered. Integration will take place using the Hibernate entity management framework for Java, which also protects against SQL injection attacks.

External

The application will require integration with a Maps API platform, in order to display the “tidbits” as nearby points on a map, as well as provide an overview map containing all tidbits on the screen area.

Architectural responsibilities

* Log all requests to a separate log file, for ease of searching
* Persist information received from users
* Communicate with Maps API
* Keep transferred data to a minimum
* Secure transfer of information
* Cater for intermittent signal of mobile data services

3 Quality requirements

The quality requirements are the requirements around the quality attributes of the systems and the

services it provides. This includes requirements like performance, scalability, security, auditabilty,

usability, and testability requirements.

3.1 Scalability

Specify quanti\_ed scalability requirements.

3.2 Performance requirements

Specify quanti\_ed performance requirements.

3.3 Maintainabilty

Specify the maintainability requirements including where change is expected and bounds to the

e\_ort required to apply changes to the system.

3.4 Reliability and Availability

Specify reliability and availability requirements.

3.5 Security

Specify security requirements

3.6 Monitorability and Auditability

Discuss monitorability and auditability requirements including what aspects and through which

channels the the system should be monitorable, which information should be captured for au-

diting purposes and how the audit logs are made accessible.

3.7 Testability

Specify testability requirements.

3.8 Usability

Specify usability requirements.

3

3.9 Integrability

Specify integration requirements including required integration channels, protocols and quality

requirements on integration channels

Architecture constraints

The technologies that will be used are various, and the following should not be considered an exhaustive list.

Application:

* Cordova
* AfriGIS Maps API

Server:

* Java
* Hibernate
* Spring
* Vaadin

Database:

* MySQL

Acronyms

API - Application Program Interface

CMS – Content Management Service

DBMS – Database Management Service

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

SQL – Structured Query Language

URL – Uniform Resource Locator