UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Architecture specification

Project name: Arcane Arcade

Client: Tony vd Linden

Team name: Terabites

Team members

NG Maluleke	D Mulugisi	C Nel	LE Tom
13229908	13071603	14029368	13325095

August 20, 2016

Contents

1 Introduction

Our project *Arcane Arcade* has a vision to provide a profoundly manner of testing the abilities of developers, who are potential employees, and using the results of the captured data from these tests to be able to assign them accordingly to a specific postion within the company which is best suited for their skills.

Since the core focus is on software developers, *Arcane Arcade* does testing on the programming abilities of the candidates using an esoteric language (esolang). Candidates are provided with a series of programming challenges and questions which they have to complete using the esoteric language.

2 Architecture Requirements

2.1 Access channel requirements

The system requires two interfaces:

- Web interface(Maintenance portion)
- It is preferred that the game portion be built to run on a tablet device

2.2 Quality requirements

The following quality aspects needs to be addressed:

- **Perfomance:** Performance is the most important quality requirement for this application. This requirement pertains to how fast the application responds to certain actions within a set time interval. The application will use a compiler instead of an interpreter which will translate the esoteric language into object code, which performs better than an interpreter.
- **Reliability:** Reliability is the second most important quality requirement. Care should be given to make the application as reliable as possible without sacrificing performance. The application should have maximum uptime and proper fault tolerance. The application should help the user in avoiding errors, such as submitting incompatible values.
- **Maintainability:** Maintainability is the third most important quality requirement for the application. Since the future of the application may require total change of the esolang, developers should be able to easily and relatively quickly change aspects of the functionality the system provides or be able to add new functionality to the system. This means that care should be given to modularity and in making the system clear to understand for future developers. Technologies to be used should also be expected to be available for long.

- **Scalability:** The system should be able to service at least one administrator and a few users (potential employees).
- **Security:** The system should be secure enough so that no unauthorized users will be able to access it and data integrity should be maintained.
- **Cost:** Since some users will be accessing the application via mobile internet, care should be given to keep the required bandwidth to a minimum.

2.3 Architecture styles

- REST

2.4 Integration requirements

The system will be primarily used via a web interface which will interact with the server through a RESTful API, decoupling the client from the server as best as possible.

2.5 Architecture constraints

2.5.1 Technologies

- HTML and JavaScript for the front end functionality for both the browser and game application
- Java Standard Edition (J2SE) will be used for backend functionality.
- PostgreSQL for persistence because it is a mature, efficient and reliable relational database implementation which is available across platforms and for which there is a large and very competent support community.

2.5.2 Architectural patterns/frameworks

Three-tier Architecture The main reason for using three-tier architecture is to allow any of the three tiers to be upgraded or replaced independently, when changes in requirements or technology require such upgrades or replacements. Thus addressing **maintainability** of the application. By having a dedicated application-tier which can run on a capable server, **performance** is also addressed since the client side is freed from the more demanding computations such as the compilation of the user code.

The three tiers are:

- **Presentation-tier:** This will be represented the front-end of the management or maintenance portal which will communicate with other tiers by sending results to the browser and other tiers in the network.

- **Application-tier:** This is the logical tier which provides the application's functionality.
- **Data tier:** All the data persistence mechanisms which will be used. Data in this tier is kept independent of application servers or business logic.

Frameworks
AngularJS

- Which is a powerful flexible framework with good modularization and extensive module repositories. With Angular.js there is, however, a higher risk for the application to degenerate into an inconsistent and difficult to maintain web application as it does not enforce a consistent way of developing the application. The two-way binding and the updating of athe virtual DOM is also less efficient in Angular.js.