



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

July 29, 2016

Contents

1	Introduction	2
2	Vision	2
3	Background	2
4	Architecture Requirements	3
4.1	Access channel requirements	3
4.2	Quality requirements	3
4.3	Architecture styles	4
4.4	Integration requirements	4
4.5	Architecture constraints	4
4.5.1	Technologies	4
4.5.2	Architectural patterns/frameworks	4

1 Introduction

The project is called *Arcane Arcade*, which references the esoteric language users will have to use, as well as the gamification approach to try and make it as fun as possible.

2 Vision

The vision is to provide a fun and easily accessible platform that can be used to gauge the programming aptitude and capabilities of existing or future software developers using a custom esoteric programming language.

3 Background

There are many models in which to ascertain whether a prospective employee has the required skills or aptitude to be a valuable software developer. On-line assessments are easy to execute, but lack the insight provided by manual and proprietary testing of how the candidate reasons. Manual and proprietary testing on the other hand is time-consuming and expensive to execute. In addition, company proprietary tests become stale and are leaked into the industry reducing the value tests may have.

BBD has thus opted to use an esoteric programming language in order to test the skills and aptitude of prospective employees, regardless of their programming experience or preferred programming language. A mobile and online platform is thus required to test prospective employees while keeping the tests fun by using gamification principles. The tests should also indicate in which area the user is likely to be better suited, by looking at how the user completed the challenges, such as whether the user used any hints.

4 Architecture Requirements

4.1 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

4.2 Quality requirements

The following quality aspects needs to be addressed:

- **Performance:** 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.

4.3 Architecture styles

- REST

4.4 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.

4.5 Architecture constraints

4.5.1 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.

4.5.2 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 the virtual DOM is also less efficient in Angular.js.