Iancu Paul
30432

CompSciPrep System
Supplementary Specification

Version 1.0

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	1.0	Initial Requirements Statement	Iancu Paul

Table of Contents

Supplementary Specification

1. Introduction

The Supplementary Specification captures the system requirements that are not readily captured in the use cases of the use-case model. Such requirements include:

- Legal and regulatory requirements, including application standards.
- Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.
- Other requirements such as operating systems and environments, compatibility requirements, and design constraints.

2. Non-functional Requirements

2.1 Availability

The system is not expected to be used in urgent scenarios so we can afford a SLA[1] of 99.5%. This translates into a yearly downtime of roughly 1 day and 19 hours, or a monthly downtime of 3 hours and 39 minutes. This time can be used to perform software updates, data compression and garbage collection.

2.2 Performance

Performance is not a key factor for our system. For this reason we can allow a response time of up to 30 seconds for uploading solutions in the worst case scenario. The average response time, depending on the load of the system, should be 1 second.

2.3 Security

The system will be secured using https encrypted connections. Also we will demand user authentication and will not keep passwords in plain text. Other user data will not be encrypted as we do not find it as being sensible information.

If the solution for a given problem was written in a compiled language:

- The source file of the solution will be compiled
- The executable will be generated in a different directory for each solution
- In the directory of the folder, input and output files will be generated (if they are required by the problem)
- The generated executable will have will be started with no rights to execute other processes, read rights only on the input file (if it exists), write rights only on the output file (if it exits).

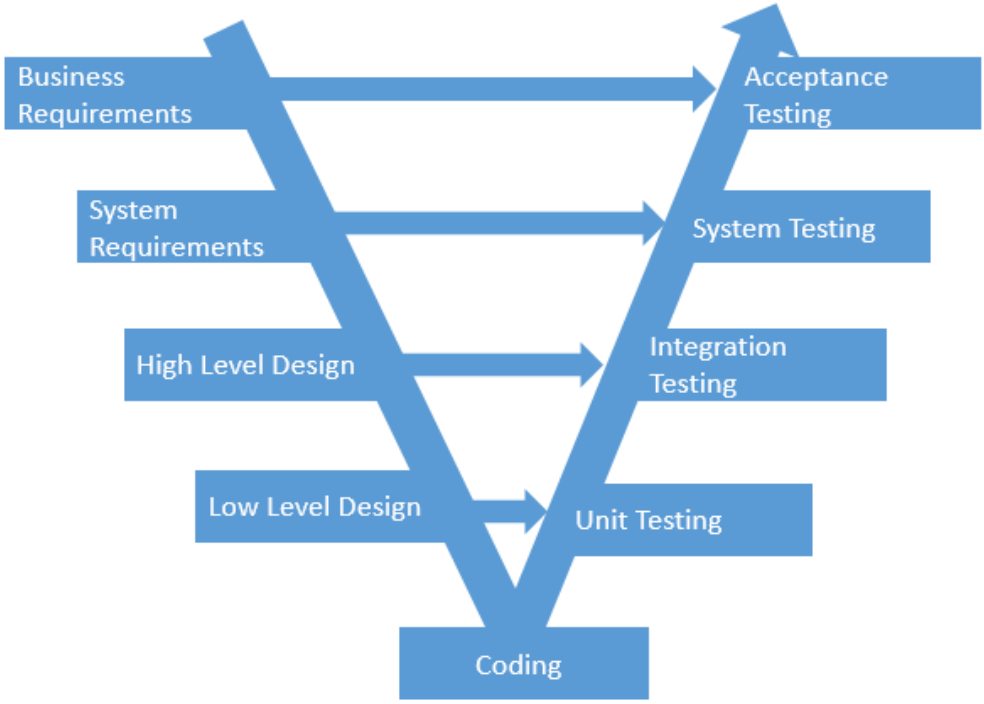
If the solution for a given problem was written in an interpreted language:

- The source file of the solution will be downloaded into a directory unique for that submission
- In the directory of the folder, input and output files will be generated (if they are required by the problem)
- The interpreter (which has as input the source file of the solution) will be started with no rights to execute other processes, read rights only on the input file (if it exists), write rights only on the output file (if it exits).

2.4 Testability

The business logic of the application must be tested independently from the user interface. We will employ V-Model testing as illustrated in **Error! Reference source not found.**. We aim to have over 90% test coverage, through unit and integration tests. With respect to manual testing, the system will log all information that is not displayed in the user interface, so that the system is fully observable and testable.

[1] SLA = Service Level Agreement = Availability



2.5 Usability

The user should be able to reach any desired goal in under 30 mouse clicks.

3. Design Constraints

The system is constrained to use Java 8 as implementation language. The software development process will be the Rational Unified Process (RUP), tailored to fit the team and the project. The conceptual architecture of the system will be a client server as illustrated in **Error! Reference source not found.**. The required development tools are either Eclipse IDE or IntelliJ IDEA. In terms of libraries we will use: JavaFX, Hibernate, JDBC and GSON.