<Company Name>
<Company Name>

<College Portal> Supplementary Specification

Version <1.0>

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

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 Aqwer :*

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

Availability of a system is typically measured as a factor of its reliability – as reliability increases, so does availability. Availability of a system may also be increased by the strategy of focusing on increasing testability, diagnostics and maintainability and not on reliability. Improving maintainability during the early design phase is generally easier than reliability (and Testability & diagnostics). Maintainability estimates (item Repair [by replacement] rates) are also generally more accurate. However, because the uncertainties in the reliability estimates (and also in diagnostic times) are in most cases very large, it is likely to dominate the availability (and the prediction uncertainty) problem, even while maintainability levels are very high. If failures are prevented, none of the others are of any importance and therefore reliability is generally regarded as the most important part of availability!

2.2 Performance

Computer performance is the amount of work accomplished by a computer system. Depending on the context, high computer performance may involve one or more of the following:

Short response time for a given piece of work
High throughput (rate of processing work)
Low utilization of computing resource(s)
High availability of the computing system or application
Fast (or highly compact) data compression and decompression
High bandwidth
Short data transmission time

2.3 Security

Security is freedom from, or resilience against, potential harm (or other unwanted coercive change) from external forces. Security mostly refers to protection from hostile forces, but it has a wide range of other senses: for example, as the absence of harm (e.g. freedom from want); as the presence of an essential good ; as resilience against potential damage or harm.

2.4 Testability

Testability is not an intrinsic property of a software artifact and can not be measured directly (such as software size). Instead testability is an extrinsic property which results from interdependency of the software to be tested and the test goals, test methods used, and test resources.
If the testability of the software artifact is high, then finding faults in the system (if it has any) by means of testing is easier.

2.5 Usability

Usability is the ease of use and learnability of a human-made object such as a tool or device. In software engineering, usability is the degree to which a software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use. Usability includes methods of measuring usability, such as needs analysis and the study of the principles behind an object's perceived efficiency or elegance.

3. Design Constraints

[This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]