SRD

# Team

*Team Name*

*Team Members*

# Proposal Name

*The Name of your proposal.*

# Vision

*Explain the vision for your proposal. Make it an opening sentence or two like an advertisement. From there provide some details explaining the motivation and what will be achieved with the proposal.*

# In Scope

*Bulleted list of the key features of your proposal*



# Out of Scope

*Bulleted list of the features your system will NOT support*



# Key Features

*Key feature, and brief description. These are the features that you will use to “sell” the solution.*

|  |  |  |
| --- | --- | --- |
| ID | Feature | Description |
| *FR1* |  |  |
| *FR2* |  |  |
| *FR3* |  |  |

# Personas

*Persona writeup, include Picture, background, motivation, desires, pain points. Must have minimum of 2 personas. You can add more if the persona is distinct, and no more than 4 altogether.*

## Student Persona

## Administrator Persona

# Scenarios

*Scenario for each persona showing them using the solution to solve the problem. Consider a scenario a TV commercial. You need to show how the persona uses the system, not just state that they do. For example: Good: “After clicking the login button, Mike navigates to Store area to view what is available, he is excited to see the new Lamp for only 5 eggs. He has 10 eggs saved up, and decides to splurge for the cool decoration.” Bad: “Login, Open Store Feature, buy lamp”*

## Student Scenarios

### SS1

### SS2

## Administrator Scenarios

### SA1

### SA2

# Site Map Student

*Overall map of the Student Portal and how the pages connect to each other*

# Wireframes Student

*Wireframes for the section, must have major functionality called out on the wireframes. Everything in the scenario, and Key features must have wireframes.*

## WS1

## WS2

## WS3

## WS4

# Functional Requirements Student

*List all functional requirements the system must support, provide the ID, requirement and description*

## FRS1

## FRS2

## FRS3

## FRS4

# Site Map Administrator

*Overall map of the Student Portal and how the pages connect to each other*

# Wireframes Administrator

*Wireframes for the section, must have major functionality called out on the wireframes. Everything in the scenario, and Key features must have wireframes.*

## WA1

## WA2

## WA3

## WA4

# Functional Requirements Administrator

*List all functional requirements the system must support, provide the ID, requirement and description*

*Comment on Requirements:*

*This section contains all the software requirements at a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output. The following principles apply:*

1. *Specific requirements should be stated with all the characteristics of a good SRS*
2. *correct*
3. *unambiguous*
4. *complete*
5. *consistent*
6. *ranked for importance and/or stability*
7. *verifiable*
8. *modifiable*
9. *traceable*
10. *Specific requirements should be cross-referenced to earlier documents that relate*
11. *All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)*
12. *Careful attention should be given to organizing the requirements to maximize readability (Several alternative organizations are given at end of document)*

*Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in the following subclasses. This section reiterates section 2, but is for developers not the customer. The customer buys in with section 2, the designers use section 3 to design and build the actual application.*

*Remember this is not design. Do not require specific software packages, etc unless the customer specifically requires them. Avoid over-constraining your design. Use proper terminology:*

*The system shall… A required, must have feature*

*The system should… A desired feature, but may be deferred til later*

*The system may… An optional, nice-to-have feature that may never make it to implementation.*

*Each requirement should be uniquely identified for traceability. Usually, they are numbered 3.1, 3.1.1, 3.1.2.1 etc. Each requirement should also be testable. Avoid imprecise statements like, “The system shall be easy to use” Well no kidding, what does that mean? Avoid “motherhood and apple pie” type statements, “The system shall be developed using good software engineering practice”*

*Avoid examples, This is a specification, a designer should be able to read this spec and build the system without bothering the customer again. Don’t say things like, “The system shall accept configuration information such as name and address.” The designer doesn’t know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables.*

## FRA1

## FRA2

## FRA3

## FRA4

# Non Functional Requirements

*List all non functional requirements that the system must support, provide the ID, requirement and description*

## NF1

## NF2

## NF3

## NF4

# Technologies utilized

*Describe in detail the technologies used in the system. For example: HTML 5, CSS, Bootstrap, JavaScript, ASP.NET MVC, Document DB, Azure etc… Include any nuget packages or libraries used.*

# System Architecture

*Diagram of the system architecture and how the system components fit together.*

# Physical Architecture

*Diagram of the physical architecture*

# Class Design

*Make a class diagram including the properties and methods for each major component. All data elements must be on the class diagram*