1. **Introduction** 
   1. **Purpose**

We want to build a Code Management and Version Control System, based on the concept of Git. This product, is inspired by the platform of Github, and will be having the necessary features to meet user requirements.

* 1. **Intended Audience**

This product will be targeting any person, who has a document, and wishes to use version control, to efficiently make & save changes while working. The code management aspect is targeting users, who are using this platform, to make changes to a code based file. The users can be anyone from a school or university students, to professional working software developers.

* 1. **Product Scope**

This product will be primarily focusing on 5 features for Version Control and Code Management, which are:

**Push, Pull, Commit, Staging and Checkout**

These features encompass all necessary requirements for making a good version control system, which can provide code management for our users. The product will be inspired by the current popular version control system of Github.

* 1. **References**

Our Product is using concepts of version control tool: Git and Github

**2. Overall Description**

**2.1 Product Perspective**

*<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>* ***Software Requirements Specification for <Project> Page 2***

**2.2 Product Functions**

*<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>*

**2.3 User Classes and Characteristics**

*<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>*

**2.4 Operating Environment**

*<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>*

**2.5 Design and Implementation Constraints**

*<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>*

**2.6 Assumptions and Dependencies**

*<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>*