SOFTWARE REQUIREMENTS SPECIFICATION

"The Five Minute Mushroom Farm" – A Backyard Mushroom Culture and Cooking Blog

Project Version <1.0>

SRS Version Author: Nikko Gabriel J. Hismaña

1. INTRODUCTION

The introduction will provide a brief overview of the SRS and the blog.

1.1 Purpose

1.1.a Purpose of the SRS

The purpose of this SRS is to provide details of the requirements determined during the requirements elicitation phase of the blog's development.

1.1.b Purpose of the blog

The purpose of *"The Five Minute Mushroom Farm" – A Backyard Mushroom Farming and Cooking Blog*, referred to as "*the blog*" for brevity, is to have an online platform to which the blog author can share the documentation of their backyard mushroom farming experience, the dishes they prepare using mushrooms they have harvested, and also to provide a way for blog visitors to communicate with the author for inquiries and comments.

1.2 Intended Audience

1.2.a Intended Audience of the SRS:

The intended audience of this SRS are: (1) the blog author, (2) the blog's fullstack web developer --- who is currently the blog author, (3) any third party who wishes to evaluate the blog.

1.2b Intended Audience of the blog:

Since the blog serves more of a platform to share the author's experience rather than providing expert advice or tips in mushroom farming and cooking, its intended audience are: (a) the author (as a means to digitally 'save' their experience), (b) mushroom farming enthusiasts, (c) people who are looking for mushroom cooking recipes.

1.3 Intended Use

1.3.a. Intended Use for SRS:

The intended use of the SRS is to provide precise descriptions of the blog's requirements to serve as a guide for its development.

1.3.a. Intended Use for the blog:

The blog serves two main purposes: (1) as a platform for the author to store and share their mushroom farming and cooking experience, (2) as a platform where visitors of the blog can communicate with the author.

1.4 Product Scope and Limitations

1.4.a Product Content and Feature:

The blog will be an online published website that contains text, pictures, and embedded videos documenting mushroom farming and cooking, that also allows visitors to leave a message to the author via a "Contact Me" form.

Communication sent by visitors can be accessed by the admin (the author) through Django's built-in admin interface. Communication via the blog is one-way --- the visitor can send a message containing their email address through a contact me page, whereas the admin will use a third-party email platform to send an email replying to the visitor's message.

1.4.b Product Modification and Internal Access:

Only the author and/or developer can make modifications to the webpages (including content and updates). Only the admin (the author) has access to communication sent by visitors via a "Contact Me" form on the website. Only the developer and author have access to the backend source code, whereas visitors only have access to front end source code (via "View Source code in their browsers").

1.5 Definitions and Acronyms

The technical terms used in this SRS are as follows:

Bootstrap – a CSS framework for responsive front-end web development.

Django – a Python-based web framework used for developing and maintaining websites

Django ORM (Object-relational Mapping) – Django's built-in database management feature

Relational Database – collection of data on the website with pre-defined relationships

2. OVERALL DESCRIPTION

"The Five Minute Mushroom Farm" is a Django-developed blog that features the author's experiences in farming and cooking mushrooms through text, photos, and embedded videos, and features a "Contact Me" webpage that allows visitors to leave an inquiry or comment to the author which only the author can access.

2.1 User Needs

USER	USER DESCRIPTION	USER NEEDS
*Author**	Refers to the owner and content-creator of the blog	- The author must be able to post and edit content (text, images, videos, links) on the website
Visitor	Also called 'reader'; refers to any person who visits and views the webpage	- The visitor must be able to seamlessly view and navigate all the webpages of the website regardless of browser or device used - The visitor should be able to contact the author for inquiries or comments through a "Contact Me" form
*Website Admin**	Also called 'admin'; refers to the person with full access of the website's admin interface.	- The admin should be able to add/remove registered website superusers/admins - The admin needs to have access to the messages left by the visitors through the "Contact Me" form.

**In the blog's current build, the author is also the sole website admin.*

Moreover, in the development side, the author is also the fullstack developer of the website --- in charge of the website's frontend and backend development and maintenance.

2.2 Assumptions and Dependencies

2.2.a Assumptions:

a) *Django Development*. The backend code of the blog will be developed and maintained using the Python-based Django framework. The system will utilize a relational database to store and manage its data via the built-in Django ORM. At the time of development, the Django framework version to be used is Django 4.0 which features an admin interface

b) *Google Fonts and Alternate Font Settings.* All the fonts used in the website will utilize the open-sourced Google Fonts library so that any machine, regardless of OS and font cache, can view the website as intended. Alternate font family settings will also be declared in the CSS stylesheets in case Google Fonts cannot be retrieved (for example “cursive” font family for the header and navigation bar, “sans-serif” font family for the content).

c) *Bootstrap Framework.* Bootstrap framework will be utilized for the styling and dynamic features of the website in order for it to be more responsive and also mobile-compatible. At the time of development, Bootstrap 4 or higher will be utilized for the front-end development of the website.

d) *Browser Compatibility*. We are assuming that the visitor is using a bootstrap-compatible desktop or mobile browser which allows them to view and navigate the blog as it is designed. Furthermore, it is assumed that the “Contact Me” form can be used without any issues, regardless of browser and device.

2.2.b Dependencies:

a) *Old Project Reuse.* The website’s design and contact form is reused from an old project (CMSC 126 web development class project) which also utilizes bootstrap for its frontend, and Django for its backend. The old project from which the blog reuses code from features a fully functional “Contact Me” form in a dedicated “Contact Me” webpage. The code used for this feature will either be reused as it is or refactored for the current iteration of the blog.

3. SYSTEM FEATURES AND REQUIREMENTS

3.1 Functional Requirements

The primary functional requirements of the blog, which are based on the user needs, are as follows:

a) *Viewing and Navigating the Site*. Visitors should be able to view the site as intended through a mobile or desktop browser
b) *Contacting the Author*. Visitors should be able to contact the author through a contact form.
c) *Adding, Editing, Removing Posts*. The author should be able to add and/or edit blog posts, specifically on the Home, Farm, and Kitchen webpages.

3.2 External Interface Requirements

3.2.a *User Interface Requirements.* These will focus on the website pages and design which visitors will have direct interaction with.

a) *Blog Pages.* The website will feature three main blog pages:

a.i) *'Home' page.* The landing/welcome page which also serves as an "about me" and "about the site" page for the website.

a.ii) *'The Farm' page.* Features posts on the progress and updates of the mushroom farm.

a.iii) *'The Kitchen' page*. Features posts about recipes and dishes using the mushroom harvested from the farm.

b) *'Contact Me*' *Form and Webpage*. The website allows visitors to contact the author through a dedicated 'Contact Me' webpage. The contact form asks for (and validates) the following:

b.i) *Name of visitor*. A required field; must not be blank. Validates an alpha-numeric string.

b.ii) *Email*. A required field; must not be blank. Validates if the email has the correct format (email_prefix@email_domain).

b.iii) *Message*. A required text field; must not be blank. Validates if the text field is empty.

The form will have a 'Send Message' button that will trigger the validation of visitor's input (or lack thereof) in the required fields, then upload the response to the website database once if valid.

c) *Navigation Bar.* The website will consistently feature a navigation bar throughout its webpages which will link to all available webpage for the visitor without having to type in the address of each webpage in order to explore the site.

3.2.b Hardware Interface Requirements. These will refer to the hardware requirements of the blog, particularly the supported device types and control interactions.

a) *Screen requirements*. The blog supports any screen resolution for desktop, tab, and smartphones.

b) *Mouse and Keyboard.* The blog supports navigation and input using mouse and keyboard.

c) *Touchscreen and on-screen keyboard*. The blog supports navigation and input using touch screen and on-screen keyboard.

d) *Network Hardware and Internet Connection*. Since the blog is fully online, the device needs to have a network hardware to connect to the internet and an existing internet connection.

3.2.c Software Interface Requirements. Will focus on the software requirements of the blog.

a) *Web Browser.* The blog can only be accessed through a web browser that supports bootstrap and javascript.

3.3 System Features

a) *Database.* In order for the blog to properly function (add/edit/remove posts, contact form), it needs to have a working database that stores and organizes data. The database to be used for this build will be a relational database using Django's ORM and built-in SQLite.

b) *Admin Interface*. The author can access messages from visitors through an admin interface which is built-in with Django. The admin interface also allows the web admin to add and remove admins/superusers.

c) *Validators.* The blog should have a validator for the 'Contact Me' form to ensure that the user properly fills in all the required fields in order to send their message to the author.

3.4 Nonfunctional Requirements

a) *Low data usage*. Compressing images, minimizing animations, and utilizing bootstrap (which are external) instead of embedded formatting

b) *Low memory and CPU usage.* Opening the website should not lag the device by taking up too much ram. Optimizing the website's code and database tables, as well as removing unnecessary caches and plugins should ensure that the website would not take up too much of the visitors' device's resource.

c) *Minimal page load times*. This can be achieved when low data, memory, CPU usage, and also by minifying CSS, JS, HTML, as well as leveraging browser caching, image optimization, and using compression (among other load time reduction techniques).

d) *Screen Scalability.* The blog should scale (and adjust its elements) properly depending on the resolution of the device it's being accessed on, and the size of the window (when accessed via desktop).

e) *Data and Service Security.* Visitors should rest assured that the data they inputted (especially their email) could only be seen by the author/web admin. The blog should also be secured such that data stored in it (i.e., the posts and the visitor message) won't be tampered by unauthorized users. The blog should also be secured from DDoS and similar attacks.

f) *Capacity*. The blog needs to function with minimal probability of crashing when accommodating high visitor traffic.

g) *Reliability and Availability*. The blog has to have constant uptime (24 hours a day, 7 days a week).