Package Update Notification

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Course: CS 4850-03

Semester: Spring 2023

Professor Perry

Date: 2/8/23

Team: INDY2

Prepared by:

Jason Paek, Nicholas Ott, Erik Smith, Haris Yosufi

Table of Contents

1.0 Introduction

1.1 Overview

This document defines the requirements for the Package Update Notification application that is being developed for Kennesaw State University. The purpose of this document serves to illustrate our system in a readable format that adheres to and organizes the requirements for the stakeholders to verify correctness. Additionally, it allows for the developers to utilize this document as a way to understand the details set by the client stakeholders and implement/design a software from these requirements.

Topics such as schedule, costs, development methods/and or phases, deliverables, and testing protocols will not be addressed in this document, as those will be addressed in a separate project plan document.

The Package Update Notification software is a tool for optimizing and updating packages upon a provided repo.

1.2 Project Goals

These are the main goals of our application:

1. Should allow for end users to integrate their chosen repo website into our application, and then compare versions of those packages to versions of their own respective repos
2. The application should be configured to allow exclusion of certain versions and check how far packages have to go out of date before needing a reminder
3. It should save emails of users and send email notifications reminding them to update outdated packages and adjust frequency of these notifiers.

1.3 Definitions and Acronyms

Packages - A Package can be defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and namespace management.

Repository (REPO) - A repository is computer storage for maintaining data or software packages. This location contains files, databases, or information organized for quick access over a network or directly. A repo allows consolidating data with a version control system to store metadata for every file and log changes.

FrontEnd - This is all software and hardware that is part of the user interface, including data input, buttons, programs, websites, and etc…

BackEnd - This refers to the parts of a program that goes behind the user interface and includes operations non accessible by the user.

GIT - It is a version control system/tool used for file tracking and tracking changes in the source code.

GitHub - A distributed version control platform open to user collaboration on.

API - aka Application Programming Interface, refers to a set of functions and procedures that allow for the creation of applications. Additionally, they access data and features of other applications, services, or operating systems..

Notification - A message displayed by an operating system that provides the user with reminders, communications, and other relevant information.

Version - An assigned specific update of a software

Authentication - A verification of the identity of a user, process, or device to allow access to a certain resource in a system.

Third-party - This refers to any source that is independent of the supplier and customer of a major computer product.

1.4 Assumptions

These are the following assumptions of our project:

1. The Client has a GITHUB account and is familiar with its application
2. The program should have access to the user's repos and have the ability to read said repos
3. The program should have the ability to pull update information from package repos

2.0 Design Constraints

2.1 Environment

The Package Update Notification system will be a standalone application hosted on a dedicated server. It will interface with a Supabase database to store user details. The application will use OAuth to authenticate the connection to Github. A modern web browser is required to access the Package Update Notification system.

2.2 User Characteristics

Developers (Priority):

Developers are responsible for being vigilant about checking email notifications for updates. They should be familiar with the system's method of distributing notifications and what to do upon receiving said notifications. They will generally have a background in computer science, with many having at least an undergraduate degree in the field. Their robust technical knowledge allows them to quickly learn new systems and utilize them accordingly.

Project Managers:

Project Managers are responsible for ensuring steady progress on various projects. They should be aware of the system's general functionality as it will be an integral part of whatever project they are managing. They will have a background in project management with many having some kind of college degree. Their technical knowledge is likely less broad than that of the developers.

3.0 Requirements

(list the design, graphic, and operating system requirements and list any constraints here and assign them numbers like 2.1, 2.2, etc)

3.1 Website Requirements

3.1.1 Authentication with Git repository websites such as GitHub

3.1.1.1 Third party websites account info will have already been validated by the third party site

3.1.1.2 Should have access to read user's repositories

3.1.2 Data Access

3.1.2.1 Should be able to save user and repo settings to the connected Supabase database

3.1.3 Accessibility

3.1.3.1 Should be accessible from any major browser (Chrome, Firefox, Edge, etc.)

3.2 Back End Requirements

3.2.1 Access to Supabase database

3.2.1.1 Should be able to pull data from the Supabase database

3.2.2 Repo scanning

3.2.2.1 Should be able to parse package information for any accessible repos

3.2.2.2 Should be able to handle package info from multiple project types

3.2.3 Notification

3.2.3.1 Should be able to send users information about package out of date

3.2.3.2 Notification should be an email

3.2.3.3 The notification should be according to their specifications

3.2.4 Constraints:

3.2.4.1 In order to sync in with the website they have to have Oauth integration which will allow read access to the users repositories. While this is the case with websites such as Github and Bitbucket already, this may not be the case for all websites, which may mean those websites would go unsupported.

3.3 Security

Security of User accounts, user and repo preferences, repos the application has access to are accessed through GitHub and not stored by our application. This access is opt-in by the user, and this access can be opted out of at any time by the user through the website they signed into the application with. Some of the many security features include: Security Hardening, Encrypted Secrets, and Automatic token authentication which can all be further explained in their official website: https://docs.github.com/en/actions/security-guides/security-hardening-for-github-actions#reusing-third-party-workflows.

3.4 Capacity

At a bare minimum:

- The system should be able to handle 50 concurrent users with further scalability.
- Other system requirements and data capacities are handled and managed by GitHub

3.5 Usability

User satisfaction surveys will be offered to testers during development and to end users post development. Quantifiable data should be gathered in order to evaluate system efficiency and quality of life to the program created. More data should be gathered upon the closer completion of the software, surveys can be created using google survey.

4.0 External Interface Requirements

4.1 User Interface Requirements

Users should be able to access the website on any major web-browser (Chrome, Firefox, Edge, etc…)

4.2 Hardware Interface Requirements

An external server is required to host both the frontend and backend for the application. Software developed for any operating system capable of opening a web browser. The server should run on Linux.

4.3 Software Interface Requirements

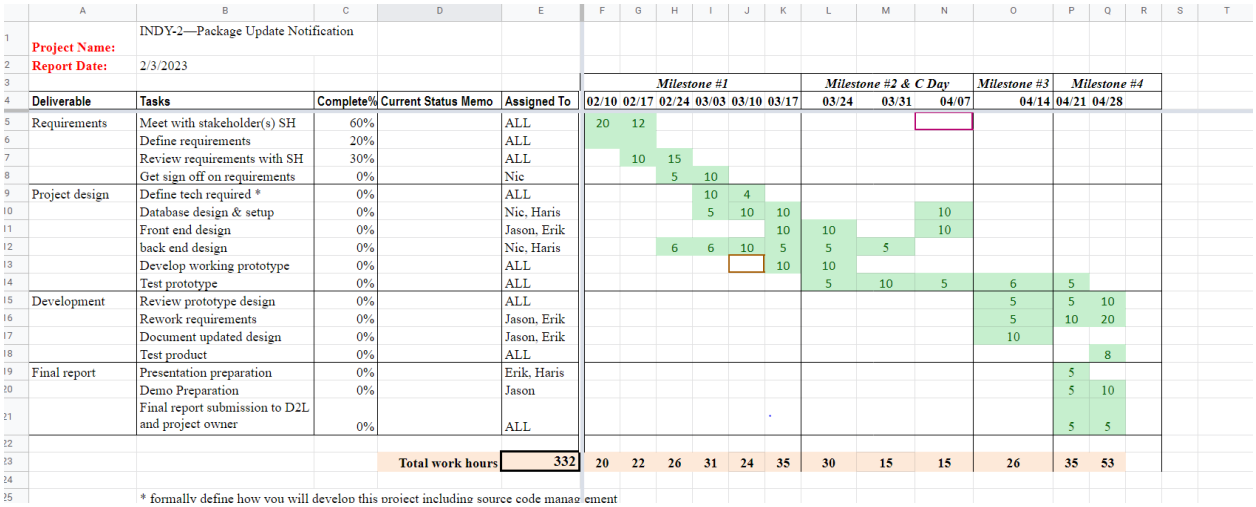
Building website with svelte kit, bootstrap for user interface, and supabase for authenticating users with their account in the repo website and to store and save user settings. C# libraries are being utilized for supabase which can be cross referenced to this website: https://supabase.com/docs/reference/csharp/v1/select . All SupaBase-Csharp objects and methods will be inherited into our project.

4.4 Communication Interface Requirements

An active internet connection is required for users to access the website application. The server used to host the application must be online to allow users to connect and to guarantee successful communication between the frontend and backend of the application.

APPENDIX A: Project Schedule

The following Gantt chart showcases the current schedule for the project along with the required work listed under “tasks”.



	Project Name:	INDY-2—Package Update Notification																		
	Report Date:	2/3/2023																		
								Milestone #1						*Milestone #2 & C Day*			*Milestone #3*		*Milestone #4*	
	Deliverable	**Tasks**	**Complete%**	**Current Status Memo**	**Assigned To**	**02/10**	**02/17**	**02/24**	**03/03**	**03/10**	**03/17**	**03/24**	**03/31**	**04/07**	**04/14**		**04/21**	**04/28**		
	Requirements	Meet with stakeholder(s) SH	60%		ALL	20	12													
		Define requirements	20%		ALL															
		Review requirements with SH	30%		ALL		10	15												
		Get sign off on requirements	0%		Nic			5	10											
	Project design	Define tech required *	0%		ALL				10	4										
		Database design & setup	0%		Nic, Haris				5	10	10			10						
		Front end design	0%		Jason, Erik						10	10		10						
		back end design	0%		Nic, Haris			6	6	10	5	5	5							
		Develop working prototype	0%		ALL						10	10								
		Test prototype	0%		ALL							5	10	5	6		5			
	Development	Review prototype design	0%		ALL										5		5	10		
		Rework requirements	0%		Jason, Erik										5		10	20		
		Document updated design	0%		Jason, Erik										10					
		Test product	0%		ALL													8		
	Final report	Presentation preparation	0%		Erik, Haris												5			
		Demo Preparation	0%		Jason												5	10		
		Final report submission to D2L and project owner	0%		ALL												5	5		
				Total work hours	**332**	**20**	**22**	**26**	**31**	**24**	**35**	**30**	**15**	**15**	**26**		**35**	**53**		
		* formally define how you will develop this project including source code manag ement																		