Software Design Specification

for

Notelee

Version 1.0 approved

Prepared by Deja Hintzen, Josh Warren, Christian King

Kennesaw State University

Fall 2023

9/24/2023

Table of Contents

1. Revision History

Name	Date	Reason For Changes	Version
Christian King	9/24/23	Document Revision	1.0

2. Introduction

The System Design Document (SDD) discusses the components of Notelee, a note-taking Google Chrome plugin, and its low and high-level system architecture. Using the requirements document, the SDD provides an overview of the functional and nonfunctional software components of the system:

- Revision history.
- What was already assumed before and during the design process.
- The software and methods Notelee is dependent on to help with functionality.
- Constraints that restrict some design aspects of Notelee.
- Goals we want to achieve with creating and managing Notelee.
- Methods used for the frontend and backend development process.

- Design decisions that affect the overall structure of the system.
- Notelee's system architecture.
- Policies to follow when designing and developing Notelee.

2.1.1. System Overview

Notelee is a Chrome extension designed to improve Chrome's functionality by enabling users to take notes through our user interface, either for/on a webpage or as generic notes. The desired functionalities include creating/editing/deleting notes, highlighting and making notes on any webpage, voice notes and drawings, grouping notes into folders, color customization, and font style adjustment. The core design strategy is to use HTML with stylesheets as the frontend and Javascript as the backend, together with a database system powered by Firebase.

3. Design Considerations

3.1. Assumptions and Dependencies

This section lists any dependencies the software system has and explains any assumptions made during the design process. Assumptions include:

- Internet connectivity: It is assumed that the user has internet connectivity because the plugin will use various APIs to change the webpage as well as fetch and save notes to Firebase.
- Notelee was created primarily to be a Google Chrome plugin. Chrome must be installed and updated on a regular basis because it is incompatible with any other browser.
- Computer Knowledge: Users and developers must have basic computer knowledge, such as how to install and manage browser extensions, navigate web pages, and interact with online platforms.
- Javascript and HTML support: Javascript and HTML support are required for the plugin to work properly.

Dependencies include:

- Firebase access is required for the plugin to connect to and interact with the database in order to save and retrieve data from Notelee.
- Software Developers: Developers with experience in Javascript and web page construction are critical to Notelee's success because they will be in charge of connecting the plugin with Firebase and guaranteeing smooth functionality.
- Testing: All webpages, programs, and plugins must go through testing to help developers find any problems or defects that were previously undetectable. This guarantees that the note-taking plugin is functional.

3.2. General Constraints

- Compatibility with web browsers: Notelee is only designed to work with Google Chrome. Other browsers, such as Opera and Microsoft Edge, are not being examined, therefore the plugin may not function properly.
- Programming languages and web development: Other than javascript for logic coding and event handling for html5 user interface, no other programming language can be utilized. In order for data to be transported through Chrome, manifest.json is required.
- APIs: Web page modification is part of our plan for Notelee.
- Performance: It is preferred that the plugin works fluidly and without latency as data and usage grow.
- Network: If there is no or slow internet access, Notelee must still work on creating notes and folders. If not, error handling is required.

3.3. Goals

Our primary goal with Notelee is to provide a fully functional, error-free plugin by December. Its capabilities should include, but are not limited to, the following:

- Using web page features such as buttons and text spaces, you may create, delete, and edit notes and folders.
- Color theme modification.
- A simple and basic user interface.
- Webpage modification by highlighting text regions and adding/loading notes in a sidebar for web pages.
- Interaction with a database system to load and save data on multiple devices.

3.4. Development Methods

Methods and approaches for Notelee system and software design include web development with object-oriented javascript, html, and css programming. Additionally, every plugin must include a manifest.json file that will transfer data into Chrome via our plugin.

Frontend:

- HTML for the user interface using our preferred IDE.
- CSS dynamic visuals using our preferred IDE.

Backend:

- Firebase is used to save user-transmitted data into Notelee.
- Logic coding, APIs, and interactions in Javascript utilizing our chosen IDE.

- Tools like Github are used to share code and track developed versions.

Furthermore, there will be unit tests for each Notelee version generated, with tests conducted using Chrome and continuous and regular team meetings.

4. Architectural Strategies

Design decisions that affect the system include:

- Creating a Google Chrome extension. This implies we must adhere to the programming requirements for Chrome plugins, such as only using specific programming languages as well as how we should code in Java script, such as no inline event handlers, resulting in us to code differently than normal. Chrome also requires us to utilize only HTML and CSS for the plugin's user interface.
- Using HTML, we can create a popup window for Notelee which enables the user to interact with it by including various buttons and text areas.
- Using CSS, we can visually style Notelee by changing the color, margins, font styles, alignments, popup size, and more.
- Using a database affects the system by requiring developers to code in ways to add and retrieve information from the database.
- The decision to engage with and update any webpage that the user views has an impact on our system because we must now use APIs.

5. System Architecture

Using a manifest.json file, the developers can load the Notelee folder in Chrome and add it as an extension. The frontend then enables a popup with which the user can engage in a variety of ways, all utilizing HTML and CSS through the user's browser. Our user interface allows the user to create, edit, and delete items, while our backend database, which uses Firebase, continuously saves the changes made. Additionally, the backend is running javascript while using event listeners and handlers. Using the web server, the plugin can utilize Chrome Extension APIs for webpages to enable for page manipulation such as text highlighting and note taking. Our frontend user interface and our backend logic coding can communicate with each other through HTTP requests.

6. Policies and Tactics

- Coding tasks are evenly distributed among the three of us while we collaborate on Github. It is important to us that the code is easy to understand with documentation and comments throughout the code.
- Notelee should not directly plagiarize from already existing note taking applications. We should document where our ideas come from.

- Notelee should be tested after every new version to find bugs. This requires the software test plan STP document where we will use a table to describe the function, what it should do, and whether it passed or failed the test.
- As a team, we must communicate with each other with ideas, questions, or concerns.

7. Detailed System Design

7.1. Classification

HTML: Subsystem
CSS: Subsystem
Javascript: Subsystem/Programming Language
Firebase: Database System

7.2. Definition

HTML: Markup language to create webpages.
CSS: HTML styling language for visual manipulation.
Javascript: Programming language for logic coding and event handling.
Firebase: Backend cloud computing service to save Notelee data for every user.

7.3. Responsibilities

HTML: Shows the main menu of Notelee as a popup webpage window. Acts as the user interface for the user to interact with.
CSS: Decorates the HTML Notelee menu.
Javascript: Responsible for API services and logic coding for HTML properties
Firebase: Notes made need to be saved into a database so notes won't be deleted every time the chrome browser is closed.

Appendix A: Glossary

1. Backend: Server-side logic of an application that a user cannot see.
2. CSS: Cascading stylesheet. Used to design HTML documents.
3. Extension: software added to a different piece of software to enhance its features.
4. Frontend: The user interface of an application.
5. Functionality: Operations done by a software system.
6. HTML: Most used markup language to create web pages.
7. Javascript: Programming language used for websites.
8. Notelee: Note Taking plugin for Chrome created by Christian King, Deja Hintzen and Josh Warren.
9. Plugin: software added to a different piece of software to enhance its features.