Functional Specification - Sonic

Jason Henderson: 19309916
Conor Joyce: 19425804
Date: 10/12/21

Table of Contents

1. Introduction

1.1 Overview

Sonic is an NFC-enabled Covid contact tracing solution for businesses, venues and other busy locations. Sonic is an alternative to the HSE's Covid Tracker app or the NHS's equivalent, but instead of using each user's smartphone via the Bluetooth protocol for contact tracing, an NFC scanner at the door to a building and cards owned by users will be used.

The problem with current solutions is the dependency on having a smartphone and actively searching for other nearby users to perform contact tracing against via Bluetooth. This works fine in a lot of situations, but is not ideal as it drains the user's phone battery and is not as flexible. Many buildings also manually note each customer coming into a given building with their phone number for contact tracing purposes. Sonic improves the ease of use of a system like this and provides more flexibility to the business/building owners.

Sonic plans to do this via a backend server that handles and processes all the contact tracing data and user info, alongside an app to be used by customers where they can tell the backend server that they have Covid for example. When this occurs, all other users who Sonic believes may be a close contact will be notified of such via the app.

Sonic will be primarily written in Go and Python. The client app will be a PWA (Progressive Web App) and will be written in JavaScript. The NFC scanner for demonstration and ease of development purposes will be a Raspberry Pi with an NFC-enabled addon card.

The name Sonic is based on the video game character of the same name, whose main unique feature is his speed, which is also a main and prominent feature of our project.

1.2 Business Context

Sonic can potentially be used in a commercial context, for example at a pub or restaurant that wants to handle contact tracing easily. As we want to keep our project small for the demo aspect we will be implementing our database as if we were to run it for the companies, meaning we will store all the user data on our own server. If this project was to expand we would need massive servers or would need each business/company to have their own server to store and host their own databases.

1.3 Glossary

Nfc
Near-field communication is a set of communication protocols for communication between two electronic devices over a distance of 4 cm or less.
Backend Server
In simple terms a backend server is a server that we don't see that is a running server.
Frontend
The frontend refers to the clients side of things, e.g the index page of an online shop
Database
A database is used to store information.
Raspberry Pi
Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom.
Go
Go is a statically typed, compiled programming language.
RESTful API
A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer.
HTTP
The Hypertext Transfer Protocol is an application layer protocol in the Internet protocol suite model for distributed, collaborative, hypermedia information systems.
POST request
POST is a request method supported by HTTP used by the World Wide Web. By design, the POST request method requests that a web server accept the data enclosed in the body of the request message, most likely for storing it. It is often used when uploading a file or when submitting a completed web form.
JSON
JSON is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays. It is a common data format with diverse uses in electronic data interchange, including that of web applications with servers.

2. General Description

2.1 Product / System Functions

- **Scanning a user's NFC card**
  A user upon entry to a building/business can scan their NFC card on the scanner at the entrance. The scanner will send a POST request to the backend server which will log the unique user id linked to the NFC card that was scanned.

- **Sending data to backend server**
  The scanner (a raspberry pi) can send POST requests containing relevant JSON data to the backend server's API endpoints which will accept this data.

- **Button in app for user to press if they have Covid**
  On our app there will be a button for users to press if they have Covid which will send that data then to the backend server. The back end server after identifying a covid case will then call the push notifications function.

- **Push notifications sent to users**
  Once a covid case has been identified the back end sends a notification to the client front end side and they are informed of the Covid case.

- **View map of visited locations in app**
  This Function will act as an in-built display on our app which maps the locations of where a user has scanned their NFC tag and displays a historical log of their visited locations on the app.

2.2 User Characteristics and Objectives

Sonic is designed to work well for commercial businesses and other venues that wish to perform Covid contact tracing easily and efficiently. The use of NFC cards means the user base does not need any extra knowledge to use the service, as it can be compared to tapping onto a Luas for example or a bus.

The app will be designed in a way that is simple, easy to use and with efficiency in mind. The settings page will be minimal and setup quick and painless. Because of this, Sonic will be more approachable to lots of users, even ones that are not as comfortable with smartphones.

2.3 Operational Scenarios

- **The User enters a building that has an NFC scanner**
  The User will simply enter the building and hold their NFC card to the NFC scanner. The scanner device will send the user data to the backend server via a POST request.

- **User has Covid and presses the in-app button**
  The User presses the in-app button saying that they have Covid and the System will notify all other potential close contacts via a push notification.

- **The User is notified of being a potential close contact**
  The User receives a notification as someone who was in a previous location with them has entered into the app that they have Covid, therefore all close contacts were notified.

- **The User views their map of visited locations**
  The User can view an in-app interactive map of their previously visited locations where they have scanned their NFC card.

2.4 Constraints

- **Speed**
  A lot of the backend logic will require getting information from the database and therefore will not be too time complex. The front end app must be responsive and quick to perform user actions.

- **Hardware**
  The scanner hardware requires an NFC-enabled board that can read from a card and also the ability to send HTTP requests. Therefore for our demonstration of the project we will use a Raspberry Pi as this can perform both tasks easily.

- **Connectivity**
  Internet connectivity is required from the scanner to send data to the server, and also for the app to receive data from the server.

3. Functional Requirements

3.1 NFC card reading

Description	In order for Sonic to work, it must first of all have the ability to read NFC cards. This can be done through software as long as capable hardware is present to do the reading.
Criticality	This is the most critical part of the project, as the functionality of Sonic as a service depends on this function.
Technical Issues	● Hardware: The scanner will require capable hardware that has the ability to read NFC cards. ● Performance: the scanner must be performant to ensure a good user experience when interacting with it.
Dependencies	n/a

3.2 User Accounts

Description	Simple user accounts must be provided where a given NFC card can be linked to a user's account within the app, so that they can be notified given a close contact confirmation.
Criticality	Somewhat critical as other methods of user differentiation could be used, but this is the most flexible and secure option if implemented correctly.
Technical Issues	● Security: User accounts must be created securely and malicious users must not be able to tamper with data.
Dependencies	n/a

3.3 Card Data Upload

Description	The NFC scanner must upload card data to the backend server via a POST request or via other means of network transfer. This is so the backend server can store and use the data for contact tracing purposes.
Criticality	This function is also very critical as it is required so that the backend server actually receives the data to perform contact tracing with.
Technical Issues	● Connectivity: The scanner will require network connectivity to send the data to the backend server.
Dependencies	NFC card reading User Accounts

3.4 Data Storage

Description	Sonic must be able to store a historical log of where a user has scanned their NFC card. This is required for contact tracing purposes.
Criticality	Critical as Sonic needs somewhere to store all the information it gathers.
Technical Issues	• Privacy: Sonic must be able to store all of this information in a secure and privacy oriented way as to not expose user information.
Dependencies	n/a

3.5 Check Close Contacts

Description	When a user presses the button in the app saying that they have Covid, there must be a check to see what other users have been in the same location as the infected individual.
Criticality	Highly critical as this is required for adequate contact tracing.
Technical Issues	• Reliability: This process must be reliable and have good logic to determine which users would be a potential close contact based on location and duration of contact for example.
Dependencies	User Accounts Data Storage

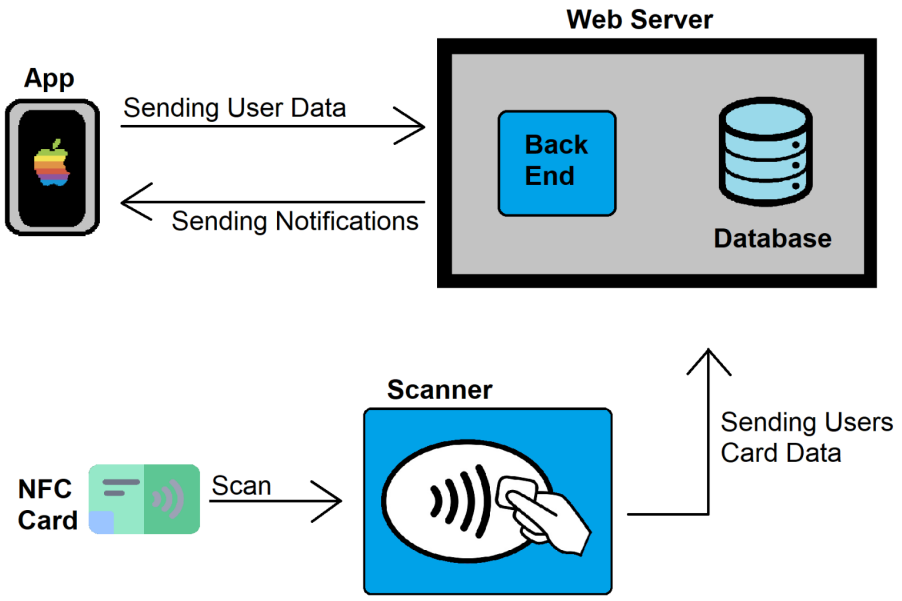
3.6 Send Close Contact Notifications

Description	Users must be sent a notification in-app if they are considered by Sonic to be a potential close contact.
Criticality	Highly critical as users must be notified when they are considered a potential close contact.
Technical Issues	• Notifications: Push notifications must be implemented into the app, Firebase Cloud Messaging is a potential solution for this but has not been fully looked into as of yet.
Dependencies	User Accounts Check Close Contacts

3.7 Show Map of Visited Locations

Description	Sonic will have the ability to show a user a map of their own previously visited locations.
Criticality	Not critical, as this is simply just a nice feature to have to improve the user experience of using Sonic.
Technical Issues	• App Complexity: This adds complexity to the mobile app which must be planned and developed in a way that results in a feature we are satisfied with.
Dependencies	User Accounts Data Storage

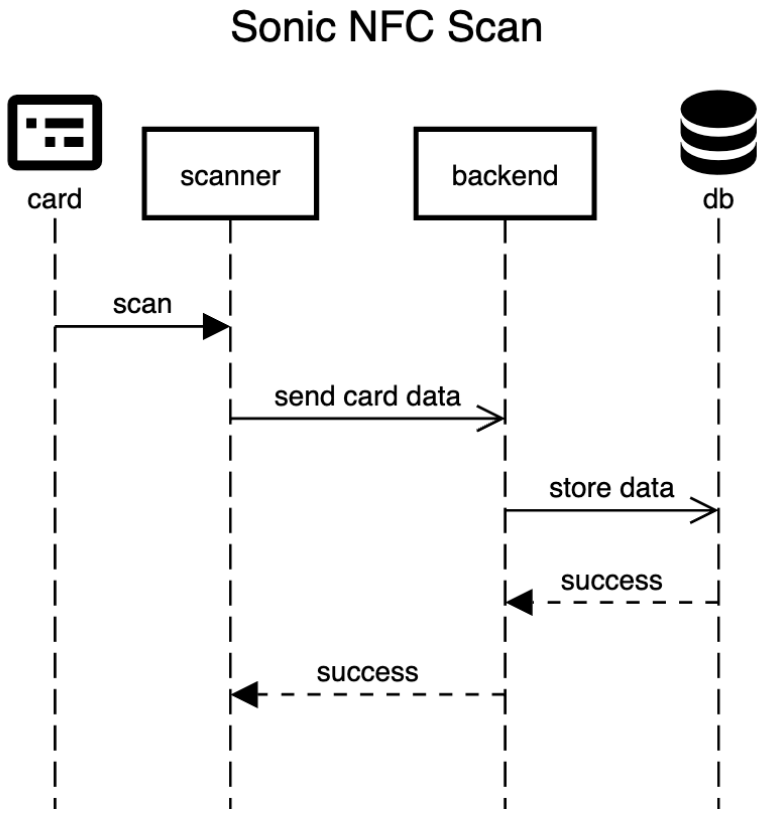
4. System Architecture



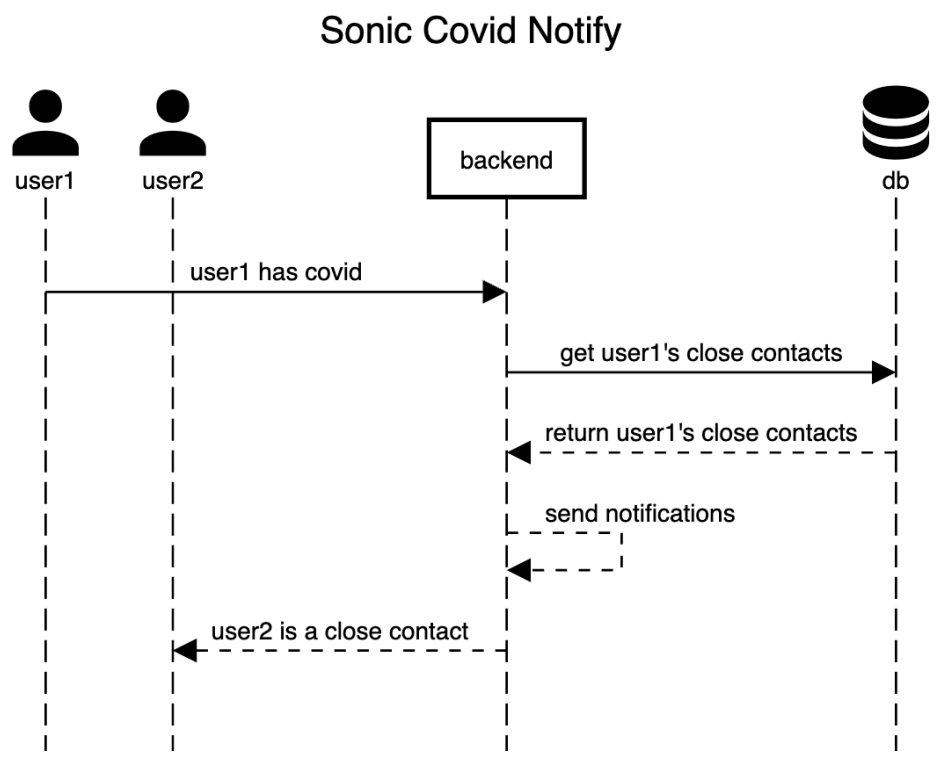
The system architecture diagram above represents how we want Sonic to work. The system is run by our web server which consists of a RESTful API backend made in Go, and a database that stores user information. A user can simply scan their NFC card on the scanner which will then send the user data from the card to the server using the scanner. The users can access their own personal information on our app on their phone and while using the app our backend receives the user data while sending the user notifications.

5. High-Level Design

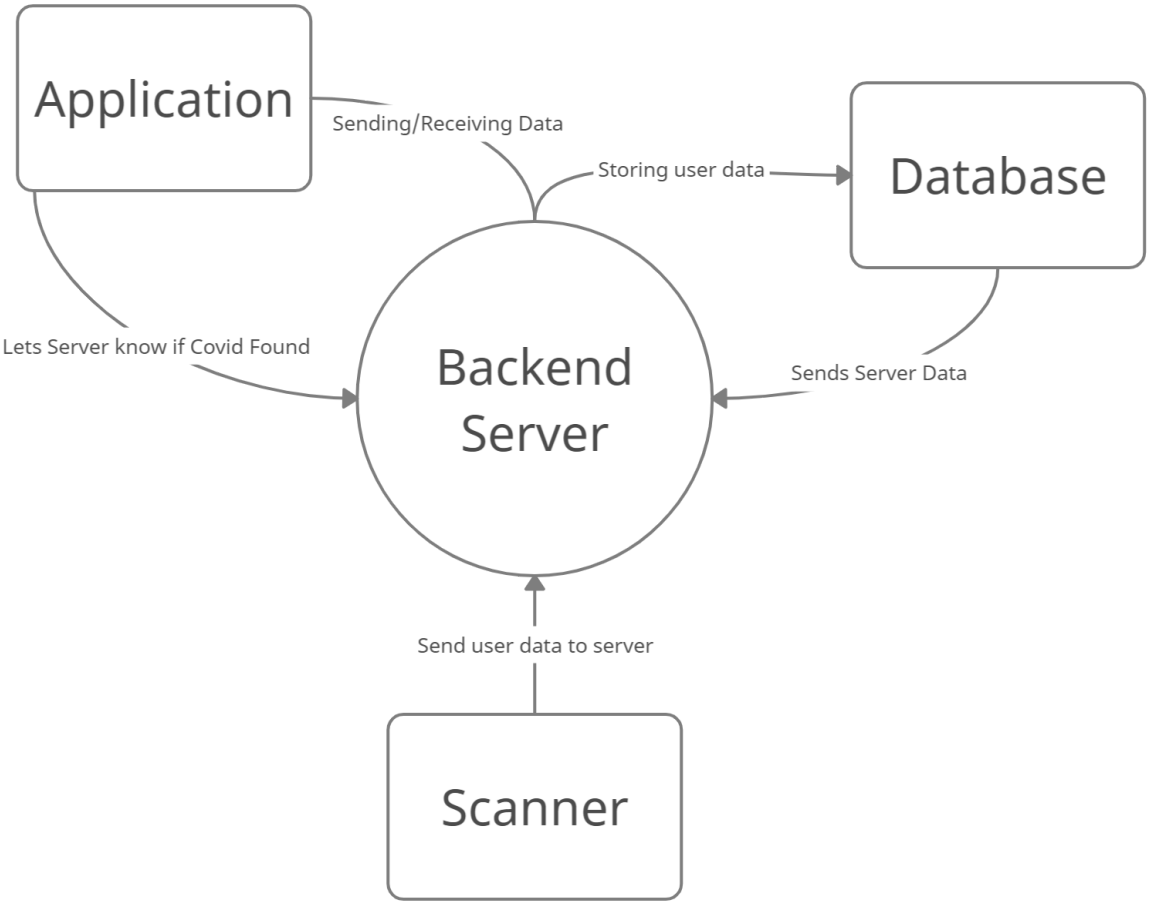
5.1 NFC Scan Sequence Diagram



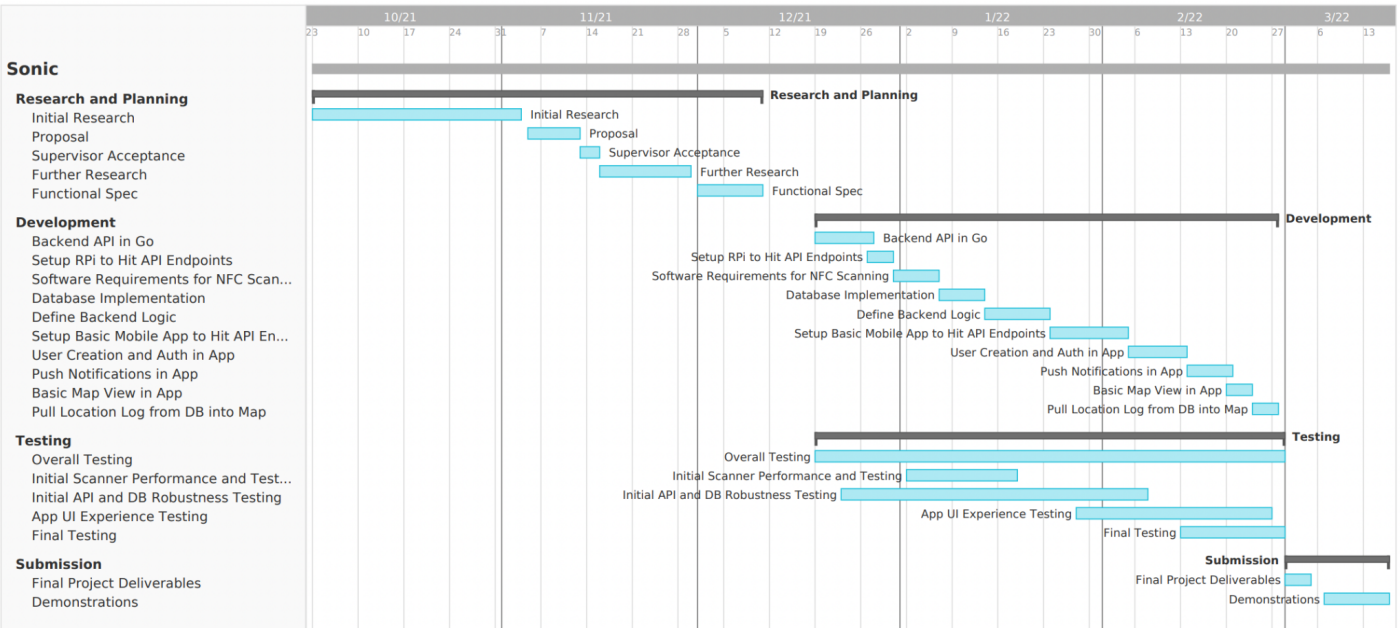
5.2 Covid Notify Sequence Diagram



5.3 Data flow diagram



6. Preliminary Schedule



7. Appendices

7.1 NFC's

https://www.androidauthority.com/what-is-nfc-270730/

7.2 Scanning NFC's

https://www.addictivetips.com/hardware/what-is-nfc-how-it-works-what-are-its-practical-applications/

7. 3 Firebase Cloud Messaging

https://medium.com/nybles/sending-push-notifications-by-using-firebase-cloud-messaging-249aa34f4f4c#:~:text=Firebase%20Cloud%20Messaging%3A%20Firebase%20Cloud,user%20re%2Dengagement%20and%20retention.

7.4 RESTful API in Go

https://tutorialedge.net/golang/creating-restful-api-with-golang/