<Project Name>

Non-Functional Requirements Specification

Usage note: There is procedural guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute allowing you to toggle whether it is visible or hidden in this template. Use the Word menu Tools🡪Options🡪View🡪Hidden Text checkbox to toggle this setting. A similar option exists for printing Tools🡪Options🡪Print.

# Introduction

# System-Wide Functional Requirements

[Statement of system-wide functional requirements, not expressed as use cases. Examples include auditing, authentication, printing, reporting.]

# System Qualities

## Usability

* the app will avoid cluttering; minimalist design so that the screen is not distracting for users with ADHD and is also easy to navigate.
* design pattern will be consistent to increase ease of use and ease of learning.
* provide verbal and/or text feedback (as per user preference) when user interacts with the app; example, *list successfully created* or *Congratulations! Task Completed*.
* include multilanguage for (localised support) support such as user manual, notifications, and instructions.

## Reliability

* detect errors such as user inputs, display error message and give options to user to re-enter or quit instead of crashing.
* test thoroughly to identify and fix bugs.
* Monitor app performance over time using analytics tools and error tracking software.
* App can work with/without internet.

## Performance

* provide scaling to accommodate increase in number of app users.
* database queries can be optimised. Such as not using SELECT\* but only columns needed.
* response time of <= 1 second is envisioned.
* Using Android Profiler tool to measure response time.
* Use Android App Startup which is a library that helps optimise the startup.
* Avoiding memory leaks to optimise shutdows; use onDestroy() method to releases resources.
* the throughput will be identified by collecting perfromance data and analysing perfromance metrics. The code will be also reviewed.

## Supportability

* Documentation
  + how to install the app; user manual in multi-language.
  + explanation about the features included.
  + System to be designed with modularity in mind for easier bug fixing.
  + Use of version control to see how the app evolves over time.
  + Specify the size of the app for user installation.
* App designed to function with and without internet connection.
  + - Clear message to be displayed when offline so that users can know certain features/functions will not be accessible.
    - allows users living in area with poor network connection to use the core features.
* Adaptability: Allow flexible configuration for greater adaptability such as
  + - user changing font size, reminder tone, and frequency of reminders.
    - users can change from auditory notofications to visual notifications.
    - using analytics to identify area where the app can be improved, such as features that are causing dissatisfaction to users.
      * Flurry analytics can be used to collect and store locally on device and once internet connection is available can send it to server such as Firebase analytics.
* Localisation
* use a localisation framework (Internationalisation framework).
* apply appropriate graphics and icons. Example thumbs up gesture is acceptable in Australia but not so in clutures like Middle East.

# System Interfaces

[Interface Requirements are part of the + in the FURPS+ classification of supporting requirements. Define the interfaces that must be supported by the application. It should contain adequate specificity, protocols, ports and logical addresses, and so forth, so that the software can be developed and verified against the interface requirements.]

## User Interfaces

[Describe the user interfaces that are to be implemented by the software. The intention of this section is to state requirements relating to the interface. Interface design may overlap the requirements gathering process.]

### Look & Feel

[Provide a description of the spirit of the interface. Your client may have given you particular demands such as style, colors to be used, and degree of interaction and so on. This section captures the requirements for the interface rather than the design for the interface.]

### Layout and Navigation Requirements

[Capture requirements on major screen areas and how they should be grouped together.]

### Consistency

[Consistency in the user interface enables users to predict what will happen. This section states requirements on the use of mechanisms to be employed in the user interface. This applies both within the system and with other systems and can be applied at different levels: navigation controls, screen areas sizes and shapes, placements for entering / presenting data, terminology.]

### User Personalization & Customization Requirements

[Requirements on content that should automatically displayed to users or available based on user attributes. Sometimes users allowed to customize the content displayed or to personalize displayed content.]

## Interfaces to External Systems or Devices

[Are there any external systems with which this system must interface? Are there any constraints on the nature of the interface between this system and any external system, such as the format of data passed between these systems, and any particular protocol used? Consider both provided and required interfaces.]

### Software Interfaces

[This section describes software interfaces to other components of the software system. These may be purchased components, components reused from another application or components being developed for subsystems outside of the scope of this SRS, but with which this software application must interact.]

### Hardware Interfaces

[This section defines any hardware interfaces that are to be supported by the software, including logical structure, physical addresses, expected behavior, and so on.]

### Communications Interfaces

[Describe any communications interfaces to other systems or devices such as local area networks, remote serial devices, and so on.]

# Business Rules

[Business rules are statements that define or constrain some aspect of the business. Business rules are often represented as production rules when they are meant to be directly executed by an IT System: a production rule is an independent statement of programming logic that specifies the execution of one or more actions in the case that its conditions are satisfied. Production Rules define the operation semantic for the system in a technologic independent way. They constrain the behavior expressed in system use cases.

Organize this document on rule classes, a high level grouping of candidate or actual rules about one **business concept** with a specific kind of **logic processing**, example: Driver Risk Assessment Rules or Customer Validation Rules.]

## <Rule class name>

### <Rule name and ID>

[The description defines the rule. It can be made in natural language typically following a decision table or a pattern like: if [condition-list] then [action-list], example:

If there are at least 3 items of the same type in the customer shopping cart and each item’s value is greater than $30 then give to the customer a voucher whose value is 10% of the cheapest item.]

# System Constraints

* Design Constraints
  + The app will be developed for Android operating system.
  + the cell phone is expected to have a CPU [speed](https://insidetechworld.com/phones/top-android-phones-with-fastest-processors-in-2021/) of 1.8 GHz to 3 GHz or higher.
  + Consume no more than \_\_\_ MB of flash memory and no more than \_\_\_ MB of system memory.
* Implemantation requirements
  + Android studio IDE will be used with Kotlin as the programming language.
  + A distributed version control system will be used to allows simultaneous real-time collaboration and keep track of the project’s progress.
  + The documentations once completed and approved in the team meeting will be uploaded to the remote git repository.
  + Group members will create individual local git branches and push commits to the remote branches to avoid breaking the main branch once coding begins.
  + Google Maps SDK, a third-party component that can provide location-based reminders and notifications.
* Google Maps SDK: An ADHD app could use the Google Maps SDK to provide location-based reminders and notifications. For example, the app could notify the user when they are approaching a specific location where they need to perform a task.
* Interface requirements
  + Home screen must be easy to navigate and provide to features such as a task list, timer, or a journal via a button.
  + App screen to follow phones orientation: landscape or portrait.
  + Setting: allows users to customise such as font size, and notofication setting.
* Size and weight of the physical hardware to be identical to current Android smart phones in the market.

# System Compliance

## Licensing Requirements

[Define any licensing enforcement requirements or other usage restriction requirements that are to be exhibited by the software.]

## Legal, Copyright, and Other Notices

[This section describes any necessary legal disclaimers, warranties, copyright notices, patent notice, wordmark, trademark, or logo compliance issues for the software.]

## Applicable Standards

[This section describes by reference any applicable standards and the specific sections of any such standards that apply to the system being described. For example, this could include legal, quality and regulatory standards, industry standards for usability, interoperability, internationalization, operating system compliance, and so forth.]

# System Documentation

[Describes the requirements, for on-line user documentation, help systems, help about notices, and so on. Set expectations for the documentation and to identify who will be responsible for creating it.]