**1. Introduction**

1. **Purpose**

This document provides a detailed view of the architecture and the user interface design of the CLup system. Building on the RASD document, it gives a more refined technical and functional description of the system, explaining it on a much lower level. To provide the full description of the system, UML diagrams will be used since they are the de facto industry standard. While actual implementation is not part of the document, it outlines the presumed implementation, integration and test plan, to help the software development team in the realization of the application. The purpose of the document is primarily to guide software developers, but it can also provide useful information to end users and investors.

1. **Scope**

CLup is a simple application that helps store managers with handling large crowds inside their store and store customers with planning more efficient and safe grocery shops. The target audience for this application includes every person that shops for groceries in a store, which includes almost all demographics fall into this category. Faced with a worldwide pandemic of the COVID-19 virus countries across the world imposed strict health measures in line with the recommendations of the WHO. To combat the spread of the virus, governments introduced decrees that limited the movement of the population to a certain degree. Only essential movement, such as: going to work, grocery shopping or outdoor exercise, was deemed acceptable. Although successful in the mitigation of the disease, the act put a serious strain on society on many levels. To help reduce the stress and anxiety, many aspects of everyday life involving close contact can be considered and improved upon. This project aims to help with, and resolve the issues surrounding grocery shopping. As we all know, grocery shopping is an essential activity which involves close contact inside the store. Since the COVID-19 virus spreads mainly through airborne particles, this activity plays a key role in its mitigation. To reduce crowding inside the stores, supermarkets need to restrict access to their store and keep the number of people inside below the optimal maximum capacity. The main idea is to enable store customers to enter a queue from home (or wherever they find themselves) through simple interaction with the application. Besides that, the application will give customers the option to "book a visit" to the grocery store. This feature will allow them to view available time slots for their grocery shop, book the most convenient one, and optionally indicate an approximated duration of their visit to further improve the accuracy of the waiting time estimation of the system.

1. **Definitions, Abbreviations, Acronyms**
   1. WHO-World Health Organization
   2. RASD-Requirements Analysis and Specification Document
2. **Revision history**
3. **Reference documents**
4. **Document structure**

**2. ARCHITECTURAL DESIGN**

1. **Overview:** High-level components and their interaction
2. **Component view**
3. **Deployment view**
4. **Runtime view:**You can use sequence diagrams to describe the way components interact to accomplish specific tasks typically related to your use cases
5. **Component interfaces**

The complexity of developing a mobile application encourages the engineers to split the paramount task into smaller subproblems, or components. Each component is then encapsulating a certain part of the applications function, and communicating with other components through interfaces. The use of interfaces enables the developer of a component to be unaware of the concrete implementation of other components, but just know the syntax of their method calls.

This section gives a thorough overview of the systems interfaces, divided by components they belong to. It is important to note that the interfaces and their methods proposed in this section, do not necessarily represent the exact written counterparts in the implementation, but offer a basic guideline of the component communication.

General interfaces:

AndroidAppManager/iPhoneAppManager

-RequestTicket

-EntranceCheck

-GiveTicket

-InformUserToEnter

-CheckTicket

CalculateTime

ManageRequests

StoreSelector

ManageStore

ManageLogin

ManageData

DatabaseManager

GoogleMaps

ManageTickets

ManageBAV

CalculateDistance

ManageSchedule

ManageQueue

To begin with, the application on the users phone communicates with the system through the AndroidAppManager and IPhoneAppManager interfaces, according to the type of the users OS. They enable communication between the ApplicationServer, or more specifically, the Director component, and the smartphone application. Depending on the type of the user, the interface offers all methods for the interaction with the system.

For example, upon the push of a button in the UI, the application invokes the RequestTicket method on the Director to propagate the request. Similar to that, an application of a store manager can invoke the CheckTicket method on the Director, to propagate the spcific request to the ApplicationServer.

The systems main global component, the ApplicationServer, is divided into smaller core components Director, LoginManager, StoreSelectionManager, DBService, GoogleMapsService and RequestManager. The communication between the core components is also done through interfaces.

For the Director to propagate users requests, the RequestManager offers methods contained in the ManageRequests interface. Other than that, the two components also communicate to calculate wait time through the CalculateTime interface.

Furthermore, the Director component communicates with the LoginManager and StoreSelectionManager components. For the LoginManager, it does so through a ManageLogin interface, which enables the propagation of the credential check used when a store manager accesses the application. For the StoreSelectionManager, it uses methods offered by the StoreSelectionManager in the ManageStore interface, so that the store manager can manage information about his store.

To enable users to choose between the stores using the CLup system, the StoreSelectionManager also offers the StoreSelection interface, used by the LoginManager and the RequestManager.

The calculation of the distance between the user and the store is done with the help of Google Maps. To implement that, the GoogleMapsService offers a GoogleMaps interface to the RequestManager. Upon gathering information, RequestManagers subcomponents BookAVisitService and DistanceService perform the distance calculation through the CalculateDistance interface.

To manage requests from different users, the RequestManager branches into more subcomponents, the TicketService and QueueService for the store managers, and the BookAVisitService and ScheduleService for the basic users. For communication, the former uses the ManageTickets and ManageQueue interfaces, whereas the latter uses the ManageBAV and ManageSchedule interfaces.

Lastly, in order to persist the applications data, the DatabaseManager interface is used. The interface enables the Application server, or more precisely, the DBService component, to invoke methods to write data to, or read data from, the database. The DBService itself, offers the ManageData interface to all other subcomponents of the ApplicationServer to connect them to the database.

1. **Selected architectural styles and patterns:** Please explain which styles/patterns you used, why, and how
2. **Other design decisions**