Software Requirements Specification

for

SharEat

Version 1.0 approved

Prepared by Group M

Lut University

08.10.2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Group M	11.10.2021	Initial draft	1.0
Group M	13.10.2021	Revisions based on development updates	1.1

1. Introduction

1.1 Purpose

The purpose of this document is to describe SharEat application system and its requirements. The document will describe the functional and nonfunctional requirements, features, interfaces, and constraints relevant to the development of the SharEat application.

1.2 Document Conventions

This document uses the following conventions.

Document format/ standards	IEEE
Requirement's prioritization technique	MoSCoW Must (Mandatory), Should (Of high priority), Could (Preferred but not necessary), Would (Can be postponed for future)

1.3 Intended Audience and Reading Suggestions

This document is primarily intended for the SharEat product owner, developers, UI/UX designers, requirement analyst, and testers. Within this document you will find the agreed upon vision for the SharEat product. This document is organized to first define the application at a high-level, by stating its intended purpose, scope, and audiences, and as the reader moves further, they are presented with more detailed information relating to the product such as design ethos, system and technology specifications, interface specifications, and requirements listing.
It is recommended that each relevant party would fully go through the 1. and 2. segments of the document to have an understanding of the general view of the products direction and development. After this, each party should seek to go through the segments that best correlate with their individual workflows.

1.4 Product Scope

Refer vision and scope document [1]

1.5 References

1. Vision and scope document - Link
2. Use case specification - Link
3. Requirements management plan - Link
4. Material design – Link
5. React Native Paper for Material Design - Link

2. Overall Description

2.1 Product Perspective

SharEat is a stand-alone application that provides a platform for buying and selling food at a subsidized price. The application acts as a platform to connect individual customers with both private users as well as commercial entities selling their excess meals, in an effort to reduce food wastage and to provide its meal providers with additional revenue streams.

2.2 Product Functions

Major functions of this application will be:

- Account creation and management
- Adding a new meal listing
- Browsing nearby meal listings
- Ordering a meal
- Rating a meal

2.3 User Classes and Characteristics

User class	Characteristics	Goals	Priority
Restaurant	- Commercial entity - Food provider - Consistent user - Multiple listings - Unlikely to purchase meals	- Avoiding food wastage - Increasing revenue - Environmental goals	Absolutely necessary
Students	- Individual person - Frequent user - Low funds	- Buying subsidized meals. - Not able to cook - Environmental goals	High
General populace	- Occasional user - Low-income households are more likely to be recurring users	- Buying subsidized meals. - Not willing to cook - Environmental goals	Moderate

2.4 Operating Environment

Environment can be split up into two parts: Backend server and mobile applications. The backend server will be deployed to AWS (Amazon Web Services) as EC2 instance. MongoDB database will be hosted in AWS as a service. The mobile application will be built to support old versions of iOS and Android. Versions for previously mentioned platforms can be found in the Table below.

Platform	Supported versions
Android	4.4 - latest
iOS	12 - latest
Node.js	16.x

2.5 Design and Implementation Constraints

1. The minimum system requirements for the app: Space (200 MB) and performance (2 GB RAM)
2. The app shall be available only in three languages: Finnish, Swedish and English
3. The collection and management of user data must adhere to GDPR regulations
4. The design of the application must adhere to both the App Store's and Play store's individual platform policies

2.6 User Documentation

SharEat will not provide a separate manual for its users. Instead, we will provide simple tutorials on the basic functions of the app on our homepage. These videos will be in a narrated short video format, hosted on YouTube and embedded into the homepage. In addition to the videos, SharEat will provide written steps on how to achieve these functions. These tutorials will be referred to, linked to, and/or embedded within the application itself.

2.7 Assumptions and Dependencies

Refer vision and scope document [1]

3. External Interface Requirements

3.1 User Interfaces

- Generic ads for non-premium users.
- Error message display- Based on user registration, logins, payment information, meal listings, meal orderings etc. when these actions do not follow the specified guidelines, a pop-up error message is displayed.
- Map view marked with the meal listings as clickable elements.
- Help button at the bottom of the screen in all windows.
- A cart button at the top of the screen after logging in.
- In case of crashes, bug report features are displayed to the user.

The UI should follow Google's Material Design guidelines and practices. React Native Paper will be used to provide mobile components which follow Material Design guidelines [4].

3.2 Hardware Interfaces

Any smartphone supporting the following features:

- Android (4.4 or higher) or iOS (12 or higher)
- GPS
- Camera (3 MP or higher)
- 200 MB of free space
- 2 GB RAM

3.3 Software Interfaces

Following is the software used for the SharEat application,

Software	Description
Mongo DB	Provides a flexible schema which makes it easy to evolve and store meal details and user ratings.
Google Maps API	Allow to create a custom map, a searchable map, check-in functions, display live data synching with location that we can use to display the meal listings on the map.
React Native and Typescript	To make it easier to deploy to both iOS and Android devices.

3.4 Communications Interfaces

- Back-end server for administration use.
- FTP protocols for administration use electronic forms in the form of customer support feedback.
- Communication between the mobile application and the backend server will be HTTPS using JSON to encode messages.

4. System Requirements

Req ID	Requirement	Description	Date of creation	Source of requirement	Priority	Addl. Remarks	Related system feature
001	User registration	The user must be able to register an account for the service.	09.10.2021	Brainstorming session	M	First name, last name, e-mail, phone number, password. Optional details: banking and operating name	User accounts
002	Log in	The user must be able to input their details and log in	11.10.2021	Brainstorming session	M	E-mail for login	User accounts
003	Location data usage	The user must allow the application to access his/her GPS data	11.10.2021	Brainstorming session	M	OS approval	Food map
004	Google map integration for food map view	The user is shown a google map image of their location	10.10.2021	Brainstorming session	M	Google Maps API	Food map
005	List-a-meal functionality	The user can press a “List a meal” -button in order to begin the meal listing process	11.10.2021	Brainstorming session	M		Food map

006	Meal attributes	The user is able provide descriptive information about the meal	11.10.2021	Survey	M	Mandatory fields: description, quantity, photo, condiments, price, offer expiry time. Pop-up error if all fields are not filled.	Meal listing
007	Camera access	The user must grant access to use their devices camera	11.10.2021	Brainstorming session	M	OS approval	Meal listing
008	Modifying a listed meal	The user (food provider) must be able modify or delist their meal	10.10.2021	Stakeholder interview	M		Meal listing
009	Order confirmation	The user must be notified when an order has been placed on a meal, they have listed	09.10.2021	Brainstorming session	S	Pop-up in top right corner	Buy food
010	Meal list management	The application must automatically remove listings that have been ordered.	11.10.2021	Brainstorming session	M		Foodmap
011	Viewing listed providers	The listed food providers must appear as clickable bubbles.	11.10.2021	Stakeholder interview	M		Foodmap
012	Vendor specific meal lists	Upon choosing the provider, the user would be able to see the listed meals by the provider.	11.10.2021	Stakeholder interview	M		Buy food
013	Choosing the meal specifics	The user could choose the specific meal he/she wants to order and add it to the cart.	11.10.2021	Stakeholder interview	M	Quantity and condiments as choices	Buy food
014	Payment processing	The application must be able to process funds between transactions in a secure manner	11.10.2021	Stakeholder interview	M		Online payment
015	Recipient-side order confirmation	Upon a successful order, the user is notified	11.10.2021	Stakeholder interview	M	Pop-up	Buy food

016	Pick-up confirmation	The user must be able to confirm they have successfully picked up the meal they have paid for	11.10.2021	Stakeholder interview	M	Option to report in case of failed pick-up	Buy food
017	Rating a meal	The user must provide a rating for every meal	11.10.2021	Stakeholder interview	M	Stars (1-5), “How well did the meal match the description?”, “How satisfied were you with the meal?”, additional feedback text-box	Food rating
018	Generic Ads	The application must show Google ads to non-premium members.	11.10.2021	Stakeholder interview	S	Premium service = no ads	Serving ads
019	Premium subscription ads	The application must show ads about premium subscription to non-premium members.	11.10.2021	Stakeholder interview	S	Ad will redirect to premium service registration page	Serving ads
020	Premium service registration	The application must allow a non-premium member to upgrade to premium services	11.10.2021	Brainstorming session	S	One-time payment	User account
021	Removal of expired listings	The system must automatically remove listings that have expired	11.10.2021	Stakeholder interview	M		Meal listing
022	Administrator privileges	Administrators must be able to delete ratings, meal listings and user accounts	11.10.2021	Brainstorming session	M		Admin
023	Default language	The application will pick up the default language from the system OS	11.10.2021	Survey	S	In-cases where the system language is different from Finnish, Swedish, or English, default to English	App wide
024	Dark mode	The application should have support for system supported dark mode functionality	11.10.2021	Survey	C	Uses OS specified setting, no toggle	App wide
025	User database	User database using MongoDB	11.10.2021	Brainstorming session	M		Admin

026	Support for aspect ratios	The application must look consistent and usable across different devices (smartphones, tablets)	11.10.2021	Stakeholder interview	C		App wide
027	Checking for an internet connection	The application must try to confirm access to the internet	11.10.2021	Brainstorming session	S	Error pop-up in case of no connection	App wide
028	Payment process failure notification	The application must inform the user when the given details fail to fulfil a purchase	11.10.2021	Stakeholder interview	S	Pop-up notification, and order process stopped	Online payment
029	Notifying users of deleted meal listings	A pop-up notification will be shown to the user when his meal listing is deleted	11.10.2021	Brainstorming session	S		Admin
030	Prefixed location range	The user is automatically shown a radius within which all listed meals are shown	11.10.2021	Survey	M	~5km radius (subject to change)	Foodmap
031	Collection of meals “saved” data	The system should allow for the collection of statistics on meal transactions	11.10.2021	Brainstorming session	S	For use especially in social media campaigns	User accounts

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The app loads quickly and if it takes longer than usual to load, it provides user with the information they need to see as to why the app is taking time to load. The app should render frames every 16ms to achieve the goal of rendering 60 fps (frames per second). The app is designed in such a way so that in rare cases if the app crashes, it launches a window to report the crash/bug.

5.2 Safety Requirements

Passwords are not displayed in the interface. Only hints are provided for auto filling passwords and other sensitive information such as credit card information, phone numbers etc. Sensitive information is not stored within or outside the application’s data storage. Sensitive information is not stored in backup servers.

5.3 Security Requirements

The app requests permission to access sensitive information such as use of camera, use of GPS for exact location, use of bug reports for making the app safer and better to use. The app complies with the GDPR act. Users can request that their data to be removed.

5.4 Software Quality Attributes

- **Availability**: The software will have 99.9% uptime. SharEat application is available for free in iOS app store and Android Google play.
- **Correctness**: Food listing should be up to date and provide accurate details on the map.
- **Maintainability**: Application feature implementations and bug fixes need to be deployed continuously without downtime.
- **Usability**: Application interface should be easy to learn using the narrated short video and allow users to accomplish their goals without errors.

5.5 Business Rules

- One must have been registered with SharEat with a verified email address
- One needs a SharEat account to login
- One must be allowed location sharing while using the app
- The user (food provider) must be able to list his/ her meals
- The user (food receiver) must be able to view all meal listings in his/ her area
- The food receiver must be able to purchase the meals
- The food provider must be notified about all his/ her meal purchases
- The food provider must be able to delist his/ her meal listings
- The user must rate the food in order to continue using the application

6. Other Requirements

Appendix A: Glossary

UI	User interface
UX	User experience
GDPR	General Data Protection Regulation
AWS	Amazon Web Service
HTTPS	Hypertext Transfer Protocol Secure
GPS	Global Positioning System
EC2	Amazon Elastic Compute Cloud
JSON	JavaScript Object Notation
FTP	File transfer protocol
FPS	Frames Per Second
API	Application programming interface
RAM	Random Access Memory