Software Requirements Specification

for

<Circlify>

Version <1.1>

Prepared by

Group Name: <*place your group name here*>

|  |  |  |
| --- | --- | --- |
| Amartya Mishra | 60004210210 | Amartyamishra2503@gmail.com |
| Gaurang Bhogle | 60004210192 | bgaurangan@gmail.com |
| Darshit Sarda | 60004210208 | darshitsarda10@gmail.com |
| Shubham Mehta | 60004210191 | shubhammehta2003@gmail.com |
|  |  |  |

|  |  |
| --- | --- |
| Instructor: | *Dr. Kiran Bhowmick* |
| Course: | Software Engineering |
| Lab Section: | *Software Engineering Lab* |
| Teaching Assistant: | NA |
| Date: | 07-02-2024 |

Contents

Revisions iii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 1

1.5 Document Conventions 1

1.6 References and Acknowledgments 2

2 Overall Description 3

2.1 Product Perspective 3

2.2 Product Functionality 3

2.3 Users and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3 Specific Requirements 5

3.1 External Interface Requirements 5

3.2 Functional Requirements 6

3.3 Behaviour Requirements 6

4 Other Non-functional Requirements 7

4.1 Performance Requirements 7

4.2 Safety and Security Requirements 7

4.3 Software Quality Attributes 7

5 Other Requirements 8

Appendix A – Data Dictionary 9

Appendix B - Group Log 10

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| Draft Type and Number | Full Name | Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded. | 00/00/00 |

# 

# Introduction

*<TO DO: Please provide a brief introduction to your project and a brief overview of what the reader will find in this section.>*

## Document Purpose

The Software Requirements Specification (SRS) document for Circlify serves as a foundational guide outlining the essential components and expectations for the development of the Circlify software product. It begins with an introduction providing context on Circlify's purpose and scope, followed by a delineation of its boundaries and included features within the scope section. Functional requirements are detailed, specifying the necessary capabilities and user interactions, while non-functional requirements outline quality attributes and constraints.

User interfaces and system architecture are described to illustrate how users will interact with the system and its underlying structure. Data requirements, assumptions, dependencies, and constraints are also documented, providing insights into Circlify's design considerations. Finally, acceptance criteria are defined to ensure stakeholder expectations are met. Overall, the SRS serves as a blueprint for development, facilitating communication between stakeholders and guiding the implementation process to ensure Circlify aligns with user needs and quality standards.

## Product Scope

The Software Requirements Specification (SRS) for Circlify encapsulates the core features and expectations of a social media application designed for both individual and business users. Circlify enables seamless sharing of multimedia content such as audio, video, images, and files, fostering collaboration through posts, stories, and vertical videos. For businesses, Circlify offers a platform to promote ads and other targeted content to engage with their desired audiences effectively. The SRS outlines the application's scope, functionalities, user interfaces, system architecture, data requirements, and acceptance criteria. It serves as a blueprint for development, ensuring Circlify meets user needs and quality standards while facilitating communication between stakeholders and the development team.

## Intended Audience and Document Overview

The Software Requirements Specification (SRS) document for Circlify is primarily directed towards the Circlify engineering and product team. Its secondary purpose is to align leadership and investors with the project vision and requirements. The detailed information provided within the SRS aims to provide the development team with a clear understanding of the expected functionality and characteristics of Circlify. This document serves as a comprehensive reference point throughout the development process, ensuring that all stakeholders are aligned and that the final product meets the defined requirements and expectations.

Following this introductory overview, Section 2 provides a high-level description of Circlify, its context, and primary components. Section 3 delves into the specific functional and behavioral requirements. Section 4 covers non-functional requirements around performance, security, and quality attributes. Finally, supporting appendices provide a data dictionary and project logs.

## Definitions, Acronyms and Abbreviations

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.

TO DO: Please provide a list of all abbreviations and acronyms used in this document sorted in alphabetical order.>

## Document Conventions

In general this document follows the IEEE formatting requirements. Use Arial font size 11, or 12 throughout the document for text. Use italics for comments. Document text should be single spaced and maintain the 1” margins found in this template. For Section and Subsection titles please follow the template.

## References and Acknowledgments

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document.

TO DO: Use the standard IEEE citation guide for this section. An example citation guide is posted for you on the website.>

# Overall Description

## Product Perspective

Circlify will be designed as a standalone social application, not requiring integration with existing platforms. Users will access the platform via a web-based client and native mobile apps on iOS and Android. The backend will consist of cloud-hosted servers providing API-based access to a relational database. Figure 1 illustrates a high-level architecture diagram:

DIAGRAM HERE

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. In this part, make sure to include a simple diagram that shows the major components of the overall system, subsystem interconnections, and external interface. In this section it is crucial that you will be creative and provide as much information as possible.

TO DO: Provide at least one paragraph describing product perspective. Provide a general diagram that will illustrate how your product interacts with the environment and in what context it is being used.>

## Product Functionality

Key functionality:

* User profiles and connectivity
* Sharing posts with text, images, video, audio
* Stories and vertical video
* File and media sharing
* Business features like ads and promotions
* Customizable privacy controls

Data flow diagram here

## Users and Characteristics

* Individual users - Share personal experiences and connect with friends
* Business users - Share promotions, ads, reach target audiences; carry out business research on set target audiences.
* Age groups 13 and above – comprising of teenage and young adults majoritarily.
* Technically apt and novice users

Primary users are individuals looking to share personal content and experiences.

## Operating Environment

The Circlify application is being developed as a cross-platform mobile app for Android and iOS devices. It will target a minimum SDK version of 21 for Android and iOS 13 for Apple devices to maximize accessibility across a range of hardware. Considering the goal of broad user adoption, the app is designed to function smoothly on devices with as low as 2GB of RAM. Along with native mobile apps, users can also access Circlify via any modern web browser for platform flexibility.

On the backend, Circlify will employ a hybrid data persistence architecture. A relational SQL database will efficiently handle structured user profile information and metadata. A NoSQL document store will provide the flexibility to capture dynamic social content like posts and media. Security is paramount for a social platform, so rigorous policies will be enforced for access control, data encryption, and user authentication using industry best practices like OAuth and multi-factor mechanisms. The backend will provide a robust API layer to securely expose functionality to clients. The overall technical approach aims to deliver an accessible, secure, and reliable experience across use cases.

## Design and Implementation Constraints

Given the time and resource constraints of the initial development timeline, the product launch will strategically focus on implementing the core functionality and most critical user needs first. The engineering team will need to make careful design trade-offs and prioritize the MVP feature set guided by technical limitations. For example, while an advanced AI recommendation system is desired, it may be deprioritized for the first release in favor of essential capabilities like content sharing and user profiles. The cloud infrastructure will be scaled to efficiently support the target initial user base. Advanced monitoring and auto-scaling capabilities can be added later to manage growth. Where possible, the team will leverage existing open source libraries and reusable components to accelerate development. With agile methodology, the launch feature set will provide the essential user value while allowing for continuous enhancement and expansion in subsequent releases as future scope.

The other primary limitations are related to security of users and business personals . There will be a need of developing an adaptable security system that can adopt to age restricted content being circulated via the platform. There will bee need to verify the businesses that can actually promote and carry out business on the platform.

## User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.

TO DO: You will not actually develop any user-manuals, but you need to describe what kind of manuals and what kind of help is needed for the software you will be developing. One paragraph should be sufficient for this section.>

## Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.

TO DO: Provide a short list of some major assumptions that might significantly affect your design. For example, you can assume that your client will have 1, 2 or at most 50 Automated Banking Machines. Every number has a significant effect on the design of your system. >

# Specific Requirements

## External Interface Requirements

### User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., Cancel) that will appear on every screen, error message display standards, and so on. Define the software components for which a user interface is needed.

TO DO: The least you can do for this section is to describe in words the different User Interfaces and the different screens that will be available to the user. Those who will be able to provide optional Graphical User Interface screenshots, will be rewarded by extra marks.>

### Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware. You are not required to specify what protocols you will be using to communicate with the hardware, but it will be usually included in this part as well.

TO DO: Please provide a short description of the different hardware interfaces. If you will be using some special libraries to communicate with your software mention them here. In case you have more than one hardware interface divide this section into subsections.>

### Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems (Windows? Linux? Etc…), tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

TO DO: The previous part illustrates some of the information you would usually include in this part of the SRS document. To make things simpler, you are only required to describe the specific interface with the operating system.>

### Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

TO DO: Do not go into too much detail, but provide 1-2 paragraphs were you will outline the major communication standards. For example, if you decide to use encryption there is no need to specify the exact encryption standards, but rather, specify the fact that the data will be encrypted and name what standards you consider using. >

## Functional Requirements

*< Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform. This section is the direct continuation of section 2.2 where you have specified the general functional requirements. Here, you should list in detail the different product functions with specific explanations regarding every function.*

*TO DO: Brake the functional requirements to several functional areas and divide this section into subsections accordingly. Provide a detailed list of all product operations related to these functional areas.*

## Behaviour Requirements

### Use Case View

<A use case defines a goal-oriented set of interactions between external actors and the system under consideration. Since sometimes we will not be able to specify completely the behaviour of the system by just State Diagrams, we use use-cases to complete what we have already started in section 3.3.1.

TO DO: Provide a use case diagram which will encapsulate the entire system and all possible actors. Do not include detailed use case descriptions (these will be needed when you will be working on the Test Plan), but make sure to include a short description of what every use-case is, who are the actors in your diagram. For more information please refer to your UML guide and the MiniThermostat SRS example file.>

# Other Non-functional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

TODO: Provide at least 5 different performance requirements based on the information you collected from the client. For example you can say “1. Any transaction will not take more than 10 seconds, etc…>

## Safety and Security Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied. Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements.

TODO:

* Provide at least 3 different safety requirements based on your interview with the client or, on your ABM related research, and again you need to be creative here.
* Describe briefly what level of security is expected from this product by your client and provide a bulleted (or numbered) list of the major security requirements.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

TODO: Use subsections (e.g., 4.3.1 Reliability, 4.3.2 Portability, etc…) provide requirements related to the different software quality attributes. Base the information you include in these subsections on the material you have learned in the class. Make sure, that you do not just write “This software shall be maintainable…” Indicate how you plan to achieve it, & etc…Do not forget to include such attributes as the design for change. Please note that you need to include at least 2 quality attributes, but it is the mere minimum and it will not receive the full marks.>

# Other Requirements

<This section is **Optional.** Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A – Data Dictionary

*<Data dictionary is used to track all the different variables, states and functional requirements that you described in your document. Make sure to include the complete list of all constants, state variables (and their possible states), inputs and outputs in a table. In the table, include the description of these items as well as all related operations and requirements.>*

Appendix B - Group Log

<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist the Teaching Assistant to determine the effort put forth to produce this document>