Department of Computer Engineering

Academic Term: First Term 2023-24

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	**1**
Title:	**Software Requirement Specification**
Date of Performance:	**27/07/2023**
Roll No:	**9604**
Team Members:	**9540,9561**

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Content Quality (03)	03(All used)	02 (Partial)	01(rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

DigiLocker+ Software Requirement Specification Document

1 Abstract

DigiLocker+ is a mobile application designed to make it easier for people to store and access all the information and documents they need while driving. With DigiLocker+, users can keep their driving license, vehicle registration certificate, insurance papers, pollution certificates, and other important documents in one place, right on their smartphones.

The app is designed to be easy to use, with a simple and intuitive interface that makes it easy to find and access the documents users need. Users can quickly search for a specific document, view all their documents at once, or organize them into folders for easy access.

2 Introduction

2.1 Purpose

Digilocker+ provides users with a hassle-free way to store and organize their documents such as passports, driver's licenses, and insurance papers, among others. With just a few clicks, users can upload their documents to the app's cloud storage system, ensuring that they are always available and protected from loss, theft, or damage. Additionally, Digilocker+ allows users to store all relevant information about their cars, including registration, insurance, and service history, making it easy to keep track of their vehicles' details.

2.2 Scope

- Increasing Digitization
- Paperless Initiative
- Cross Platform Integration
- Enhanced Security
- Mobile First Approach

2.3 Definitions, Acronyms, Abbreviations

Not applicable.

2.4 References

Not applicable.

2.5 Developers Responsibilities

The developer is responsible for (a) developing the system(b)keeping the app updated.

3 General Description

3.1 Product Functions Overview

Digilocker+ is a unique app that uses Flutter and Firebase technologies to provide users with a highly secure and customizable platform for storing and managing their personal information. Flutter is an open-source mobile application development framework that allows for the development of highly responsive and visually appealing mobile applications. Firebase, on the other hand, is a mobile and web application development platform that provides a range of services, including cloud storage, authentication, and real-time database functionality.

The combination of these technologies in the development of Digilocker+ provides users with a highly efficient and secure platform for managing their personal information. For example, the app's cloud storage functionality ensures that users' data is protected from loss or damage, while its real-time database functionality enables users to access their information quickly and easily. Additionally, the app's integration with other services, such as Google Drive and Dropbox, provides users with even greater flexibility and customization.

3.2 User Characteristics

The main users of this app will be the masses, as this app is an app that is beneficial to every citizen as long as they have a mobile phone.

3.3 General Constraints

The system should run Android 4.0 or above.

3.5 General Assumptions and Dependencies

Not applicable.

4 Specific Requirements

4.1 Inputs and Outputs

The system has .pdf/.jpg/.docx inputs as files when uploading documents.
The system displays all documents of the user stored on the cloud in the dashboard.

4.2 Functional Requirements

To make sure the app is functional, there is a requirement for certain modules in the

development process.

1. Authentication Module: This sub-module provides secure authentication mechanisms for users to sign up and log in to the app. The authentication module leverages Firebase's authentication services, including email and password, phone number, Google, and Facebook authentication.
2. Document Storage Module: This sub-module allows users to upload, store and manage their essential documents securely. The document storage module leverages Firebase's cloud storage to store the documents, which are encrypted to ensure their security. Users can upload documents such as driving licenses, identity cards, and insurance documents.
3. Car Information Management Module: This sub-module enables users to store and manage their car information securely. Users can upload and store their car registration, insurance, and other relevant documents. They can also keep track of important dates such as MOT and tax expiry dates. The car information management module leverages Firebase's real-time database to store and retrieve car information quickly and easily.
4. User Interface Module: This sub-module is responsible for designing the app's user interface to be intuitive and easy to use. The module features a clean, modern design that enables users to navigate and access its features quickly. The user interface module includes screens for document storage, car information management, and user settings.

4.3 External Interface Requirements

User Interface: The user can interact with the various parts of the app via buttons and tabs that are built into the UI via touchscreen functionality.

4.4 Performance Constraints

Loading time of documents should not exceed more than 1 minute.

4.5 Design Constraints

Software Constraints

The system is to run under the Android Operating system.

Hardware Constraints

The system will run on any mobile phone as long as it is capable of supporting Android 4.0 and above.

Acceptance Criteria

Before accepting the system, the developer must demonstrate that the system works on the Different types of identity proof including aadhar card, pan card, and driver's licenses. The developer will have to show through test cases that all conditions are satisfied.

Postlab

(a)Evaluate the importance of a well- defined Software Requirement Specification (SRS) in the software development lifecycle and its impact on project success.
Ans:
A well-defined Software Requirement Specification (SRS) is a crucial document in the software development lifecycle, as it plays a pivotal role in ensuring project success. Here are the key reasons why a well-defined SRS is important and its impact on project success:

1. Clear Communication: The SRS serves as a bridge between stakeholders, including clients, project managers, developers, and testers. A well-written SRS clearly outlines the project's objectives, functionalities, constraints, and requirements, ensuring everyone involved has a common understanding. This clarity minimizes misunderstandings and misinterpretations that could lead to costly mistakes during development.

2. Scope Definition: The SRS defines the scope of the project, detailing what functionalities will be included and what will not be. This helps manage expectations and prevents scope creep, which is a common cause of project delays and cost overruns.

3. Basis for Estimations: The SRS provides a basis for estimating the project's timeline, budget, and resource requirements. Accurate estimations are crucial for project planning and successful execution.

4. Guidance for Development: A well-defined SRS acts as a roadmap for the development team. It outlines the software's architecture, design, and functional requirements, helping developers understand what needs to be built and ensuring consistency throughout the development process.

5. Verification and Validation: The SRS serves as a reference for testing and quality assurance teams. They can verify that the developed software meets the specified requirements and performs as expected.

6. Risk Management: By having a detailed SRS, potential risks and challenges can be identified early in the project. This allows the team to devise mitigation strategies and contingency plans to address these issues, reducing the likelihood of project failure.

7. Customer Satisfaction: When the software aligns with the client's expectations, it leads to higher customer satisfaction. A well-defined SRS ensures that the client's requirements are understood and implemented correctly, increasing the chances of delivering a successful product.

8. Reduced Development Costs: Well-defined requirements reduce the likelihood of rework and changes during development, saving time and resources. Changes made later in the development process can be costly and may even disrupt the project schedule.

9. Maintainability and Scalability: A clear SRS facilitates future maintenance and updates to the software. When requirements are well-documented, it becomes easier to identify areas that need modification or expansion, ensuring the software remains relevant and adaptable to changing needs.

Overall, a well-defined Software Requirement Specification lays the foundation for a successful software development project. It ensures that everyone involved is on the same page, reduces misunderstandings and risks, and leads to a higher likelihood of meeting the client's expectations. Failing to establish a clear and comprehensive SRS can result in misalignment, increased project complexity, and ultimately project failure. Therefore, investing time and effort in creating a solid SRS is essential for project success.

(b)Analyse a given SRS document to identify any ambiguities or inconsistencies and propose improvements to enhance its clarity and completeness.
Ans:

1. Ambiguous Language: - Look for vague or unclear statements that could lead to different interpretations. - Identify terms or phrases with multiple meanings or lack specific details. Improvement: - Replace ambiguous terms with specific and well-defined vocabulary. - Provide clear and concise descriptions of requirements.

2. Inconsistent Information: - Check for conflicting or contradictory requirements within the document. - Look for discrepancies in terminology, measurements, or formatting. Improvement: - Cross-reference related sections or requirements to ensure consistency. - Standardize terminology and units of measurement throughout the document.

3. Missing Information: - Identify any gaps or incomplete requirements that lack necessary details. - Look for omitted sections or aspectsthat should be addressed. Improvement: - Fill in missing information to provide a comprehensive view of the project. - Include relevant context, assumptions, and dependencies to avoid ambiguity.

4. Ambiguous Use Cases or Scenarios: - Review use cases or scenarios for unclear steps or undefined inputs/outputs. - Check for inconsistent use case representations or missing alternative flows. Improvement: - Ensure each use case is well-defined with clear steps, preconditions, and postconditions. - Add alternative flows and exceptions to cover various scenarios comprehensively.

5. Unverifiable or Unrealistic Requirements: - Identify requirements that cannot be objectively measured or validated. - Look for requirements that may be impractical or beyond the project scope. Improvement: - Make sure all requirements are verifiable and measurable. - Remove or revise requirements that are unrealistic or unattainable.

(c) Compare and contrast different techniques for requirement elicitation, such as interviews, surveys, and use case modelling, and determine their effectiveness in gathering user needs.
Ans:

1. Interviews:
Description: Interviews involve one-on-one or small group interactions between the requirement analyst and stakeholders. It allows for direct communication and discussion of specific topics.
Strengths:
- Real-time communication enables in-depth exploration ofstakeholder needs.
- Analysts can ask follow-up questions to clarify ambiguities or delve into details.
- Personal interactions build trust and rapport with stakeholders, leading to more honest and open responses.
Limitations:
- Time-consuming, especially when dealing with multiple stakeholders.
- Responses may be biased due to the presence of the interviewer.
- Stakeholders may not always be available for interviews, leading to scheduling Challenges.

2. Surveys:
Description: Surveys involve distributing questionnaires or forms to a large number ofstakeholders to collect their opinions, preferences, and requirements.
Strengths:
- Efficient for gathering data from a large number ofstakeholders simultaneously.
- Responses can be collected anonymously, encouraging honest feedback.
- Cost-effective, especially when dealing with geographically dispersed stakeholders.
Limitations:
- Limited scope for follow-up questions, which may result in less detailed responses.
- Stakeholders may not respond to the survey, leading to potential non-response bias.
- It might be challenging to capture complex or nuanced requirements through fixed-choice questions.

3. Use Case Modeling:
Description: Use case modeling is a technique used to capture functional requirements of the system by representing interactions between users and the system through scenarios.
Strengths:
- Provides a visual representation of how users interact with the system, making it easier to understand requirements.
- Helps in identifying system functionalities and boundary conditions.
- Encourages stakeholders to think in terms of user interactions and system responses.
Limitations:
- May not fully capture non-functional requirements or system constraints.
- Requires a good understanding ofsystem behavior and user interactions for effective modeling.

- Focuses on what the system should do, but not necessarily on how it should be implemented.

Effectiveness in Gathering User Needs:

- Interviews are highly effective in gathering user needs, especially when in-depth understanding and clarification are required. They foster rich communication and allow for a deeper exploration of requirements.
- Surveys are efficient for gathering a wide range of opinions from a large number of stakeholders. However, they may not capture the same level of detail as interviews or use case modeling.
- Use case modeling is effective in capturing functional requirements and illustrating system-user interactions. It is particularly useful for understanding the system's behavior from a user's perspective.