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:	**9594**
Team Members:	

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:

Abstract

The college social media app aims to create a platform for seamless communication, collaboration, and networking among university students. It facilitates user registration, login, a home feed, chat with friends, profile management, and a marketplace for buying and selling items within the college community.

Introduction

Purpose

The purpose of this project is to develop a college-specific social media app that enables students to connect, share resources, engage in discussions, and build a sense of community within the college.

Scope

The college social media app provides a user-friendly experience for student registration and communication within the college community, prioritizing data security and privacy.

References

[1] Smith, J., Johnson, A., "Social Media Usage among College Students," Journal of Higher Education Studies (2019).

[2] College Campus Events, Official Website.

[3] Firebase Documentation, Google.

General Description

Product Functions Overview

The college social media app offers the following functions:

- User registration and authentication
- Home feed displaying posts from friends and college events
- Chat feature for one-on-one messaging with friends
- Profile management for updating personal information and profile picture
- Marketplace for buying and selling items within the college community

User Characteristics

The main users of the app are college students with diverse interests and backgrounds, making it versatile and adaptable.

General Constraints

The app is compatible with most Android devices capable of internet access and browser support.

Specific Requirements

Inputs and Outputs

Input: User Registration

- Full name

- Email address

- College ID or username

- Password

Input: Chat Messages

- Recipient's username or college ID

- Message content

Input: Marketplace Listings

- Item name

- Item description

- Price

- Photo of the item (optional)

Output: Home Feed Content

The home feed displays posts from friends, college events, and relevant announcements.

Output: Chat Messages

The app displays received messages in the chat interface.

Output: Marketplace Items

The marketplace displays listings of items for sale within the college community.

Functional Requirements

1. User Registration and Login:

   - Users can register using their college email or social media accounts.

   - Existing users can log in with their credentials.

2. Home Feed:

   - The home feed shows posts from friends and college events in real-time.

   - Users can like, comment on, and share posts.

   - The app provides notifications for new posts and events.

3. Chat with Friends:

   - Users can search for friends by username or college ID.

   - The chat feature enables one-on-one messaging between users.

   - Users can send text messages and media files.

4. Profile Management:

   - Users can update their profile information, including profile picture and bio.

   - Privacy settings allow users to control the visibility of their profile to others.

5. Marketplace:

   - Users can create listings for items they want to sell within the college community.

   - The marketplace displays listings with item details and contact information.

   - Users can message sellers for inquiries and negotiations.

External Interface Requirements

User Interface: The app will have an intuitive and user-friendly interface, with sections for the home feed, chat, profile, and marketplace.

Performance Constraints

The app should provide real-time data retrieval and messaging for a seamless user experience, maintaining responsiveness during peak usage.

Design Constraints

Software Constraints: The app will be developed for Android devices using Android Studio and Java programming language.

Hardware Constraints: The app should run on Android devices with internet connectivity.

Acceptance Criteria

Before acceptance, the app will undergo rigorous testing to ensure smooth functionality and bug-free operation. User feedback will be taken into account for improvements, ensuring a flawless user experience.

Postlab Questions:

a) Evaluate the importance of a well defined Software Requirement Specification (SRS) in the software development lifecycle and its impact on project success.

Some key reasons why a well-defined SRS is important and how it impacts project success are:
1. Clear and Shared Understanding: The SRS document outlines the project's objectives, features, functionalities, and constraints in a structured manner. It ensures that all stakeholders have a clear and shared understanding of what needs to be built, which helps avoid misunderstandings and discrepancies throughout the development process.

2. Scope Management: A well-defined SRS helps in defining the project's scope accurately. It outlines the in-scope and out-of-scope functionalities, which assists in preventing scope creep (uncontrolled expansion of project scope) and helps manage changes efficiently.

3. Requirement Validation: The SRS document allows stakeholders to review and validate the requirements early in the project's lifecycle. This validation process helps identify potential issues and ambiguities, reducing the risk of costly changes or rework later on.

4. Basis for Development: Developers rely on the SRS as a reference to design, implement, and test the software. A well-documented SRS provides developers with the necessary details, reducing the chances of misinterpretation and ensuring that the product aligns with the client's expectations.

5. Project Planning and Estimation: The SRS serves as the basis for project planning and estimation. It helps project managers determine the required resources, timeline, and budget for successful project execution.

6. Risk Mitigation: By identifying and documenting requirements clearly, the SRS helps in risk assessment and management.

7. Client Satisfaction: When the SRS accurately captures the client's needs and expectations, it enhances the likelihood of delivering a product that meets or exceeds those requirements. This, in turn, leads to higher client satisfaction and better chances of future business opportunities.

8. Traceability and Accountability: A well-structured SRS allows for easy traceability of requirements throughout the development process. This traceability aids in maintaining accountability, as each requirement can be tracked from conception to implementation.

9. Reduced Development Time and Cost: With a clear SRS in place, development teams can work more efficiently and avoid unnecessary rework or iterations, resulting in reduced development time and cost.

10. Legal and Contractual Compliance: In projects with formal contracts, the SRS serves as a legal document that defines the scope of work and ensures compliance with contractual obligations.

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

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 aspects that 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 post-conditions.
   - 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.

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 of stakeholder 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 of stakeholders to collect their opinions, preferences, and requirements.

Strengths:

- Efficient for gathering data from a large number of stakeholders 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 of system 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.