



Daffodil
International
University

SRS REPORT

Submitted on: Sept. 21, 2023

Submitted By:

Abu Hasnat Tamim | 221-15-5458

CSE236 | SOFTWARE PROJECT - 2

Submitted to: Mr. Golam Rabbany

Dept. of CSE, Daffodil International University.

Dept. of CSE | 61_U

Lyte

“Lyte for All”

Table of Contents:

1. Introduction:	2
1.1: Purpose of the project:	3
1.2: Intended Audience:	3
1.3: Project Scope:	3
2. Overview:	4
2.1: Features:	4
2.2: Platform:	4
3. Interface Requirements:	5
3.1: User Interface:	5
3.2: Hardware Interface:	5
3.3: Software Interface:	5
4. Non-functional attributes:	6
4.1: Performance:	6
4.2: Security:	6
4.3: Reliability:	6
4.4: Usability:	6
4.5: Compatibility:	6
4.6: Maintenance & Support:	6
4.7: Backup & Recovery:	6
5. Diagrams:	7
5.1: UML Diagram:	7
5.2: Use-Case Diagram:	8
6. Conclusion:	9

1. Introduction:

1.1: Purpose of the project: There are a lot of times in the life of a student/learner when they might need notes, information, tips, etc. about certain topics and subjects. But, to find actually helpful resources is easier said than done. That's why we feel the need for a hub to easily share, find, and manage notes on topics of the user's choice. Our solution to this problem is 'Lyte', a platform where students and academics can create, share, find, and manage notes, lectures, study materials, etc. on different topics and subjects.

1.2: Intended Audience: Our platform would be accessible to people of all ages and identities, and would be beneficial to any learner on any educational level.

1.3: Project Scope: We aim to achieve the following goals with this project:

- User Profiling
- Notes Creation & Management
- Connectivity & Social Features
- Search & Discovery
- User Collaboration
- Features to incentivize user engagement
- Security & Privacy
- Mobile Accessibility
- User Support & Moderation
- Maintenance & Scalability

2. Overview:

2.1: Features: Users of 'Lyte' will be provided with the following features:

- Create, edit, share, and manage notes
- Content searching and AI-driven recommendation
- Engagement features (Rating, Comment, and Reshare notes) • Direct message and private note-sharing
- Badges, awards, and, leaderboards.
- Contests and challenges
- Customizable user-profiles
- Privacy & security options
- Reporting & Moderation
- Monetization from subscriptions and ads

- Accessibility features for people with disabilities and/or different languages
- Account verification and authentication
- Data export and Backup
- User support & feedback

2.2: Platform: 'Lyte' will be accessible through our website and Android/IOS apps.

3. Interface Requirements:

3.1: User Interface:

- Front-end software
- Back-end software

3.2: Hardware Interface:

- Server Infrastructure
- Storage System
- Load Balancing
- Security Hardware

3.3: Software Interface:

- Operating Systems
- Browsers
- Database management systems
- Messaging services
- Security Software

4. Non-functional attributes:

4.1: Performance:

- **Response Time:** Ensure that the platform responds to user actions within a reasonable timeframe.

- **Scalability:** The platform should be able to handle increased user activity and data growth without significant performance degradation.
- **Load Testing:** Perform load testing to determine the maximum concurrent user capacity and optimize system resources accordingly.

4.2: Security:

- **Data Encryption:** Encrypt sensitive data both in transit and at rest to protect user privacy.
- **Authentication and Authorization:** Ensure secure user authentication and role-based access control to prevent unauthorized access.
- **Data Backup:** Regularly back up user data and implement data recovery procedures in case of data loss or corruption.

4.3: Reliability:

- **Error Handling:** Implement robust error handling and reporting mechanisms to gracefully handle unexpected errors without service disruption.
- **System Redundancy:** Use redundant servers, databases, and components to minimize single points of failure.

4.4: Usability:

- **User Experience (UX):** Create an intuitive, user-friendly interface to maximize user satisfaction.
- **Accessibility:** Ensure the platform is accessible to users with disabilities by complying with accessibility standards (e.g., WCAG).

4.5: Compatibility:

- **Cross-Browser Compatibility:** Ensure the platform works consistently across a range of web browsers and mobile devices.

- **Mobile Responsiveness:** Optimize the user experience for mobile users, including native mobile apps for iOS and Android.

4.6: Maintenance & Support:

- **Regular Updates:** Commit to regular software updates to address bugs, security vulnerabilities, and feature enhancements.
- **Customer Support:** Offer responsive customer support channels (e.g., email, chat) to assist users with issues and inquiries.

4.7: Backup & Recovery:

- **Data Backup:** Implement automated data backup procedures and maintain backup copies in secure locations.
- **Recovery Testing:** Regularly test data recovery procedures to ensure data integrity.

5. Diagrams:

5.1: UML Diagram:

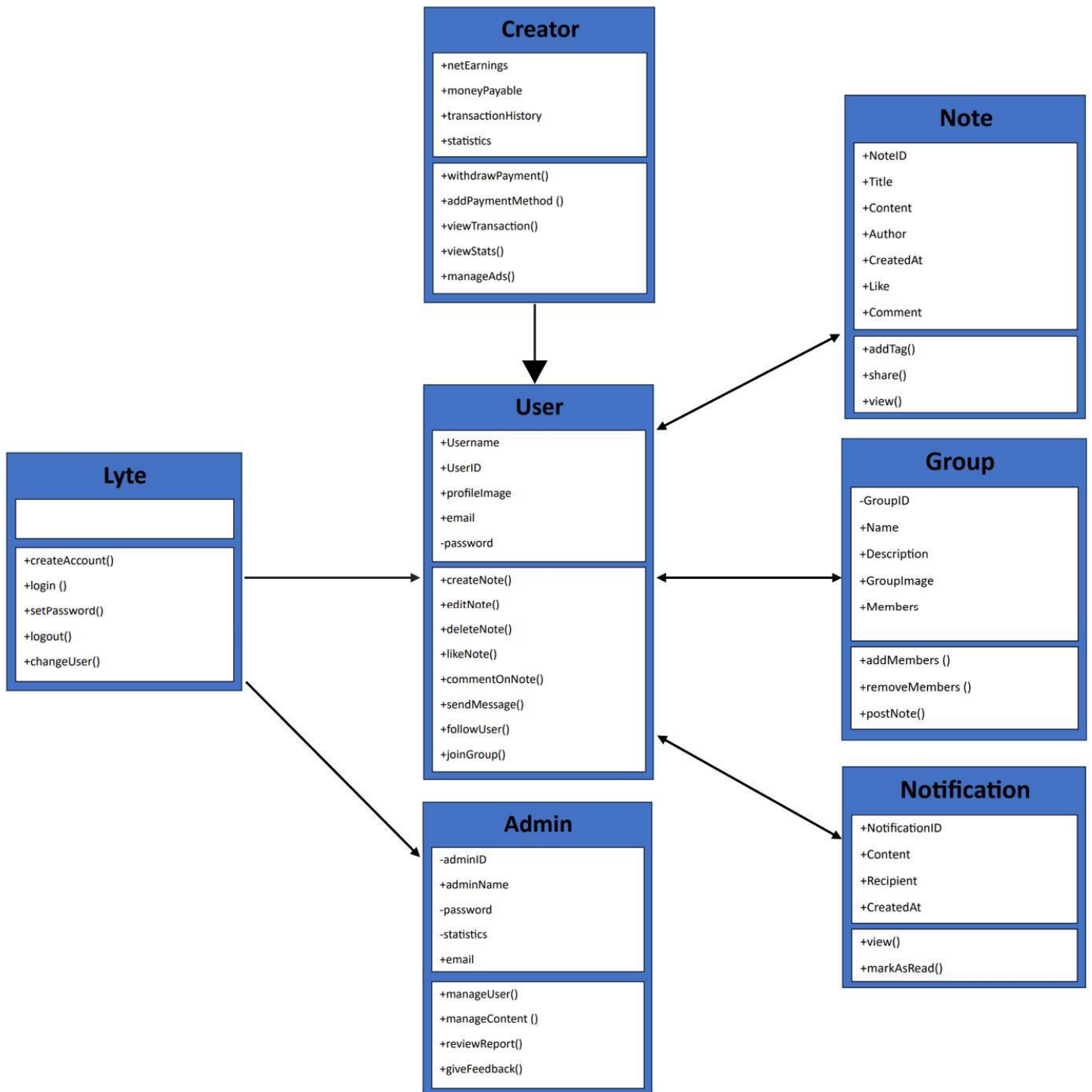


Fig: UML Diagram for our projected platform

5.2: Use-Case Diagram:

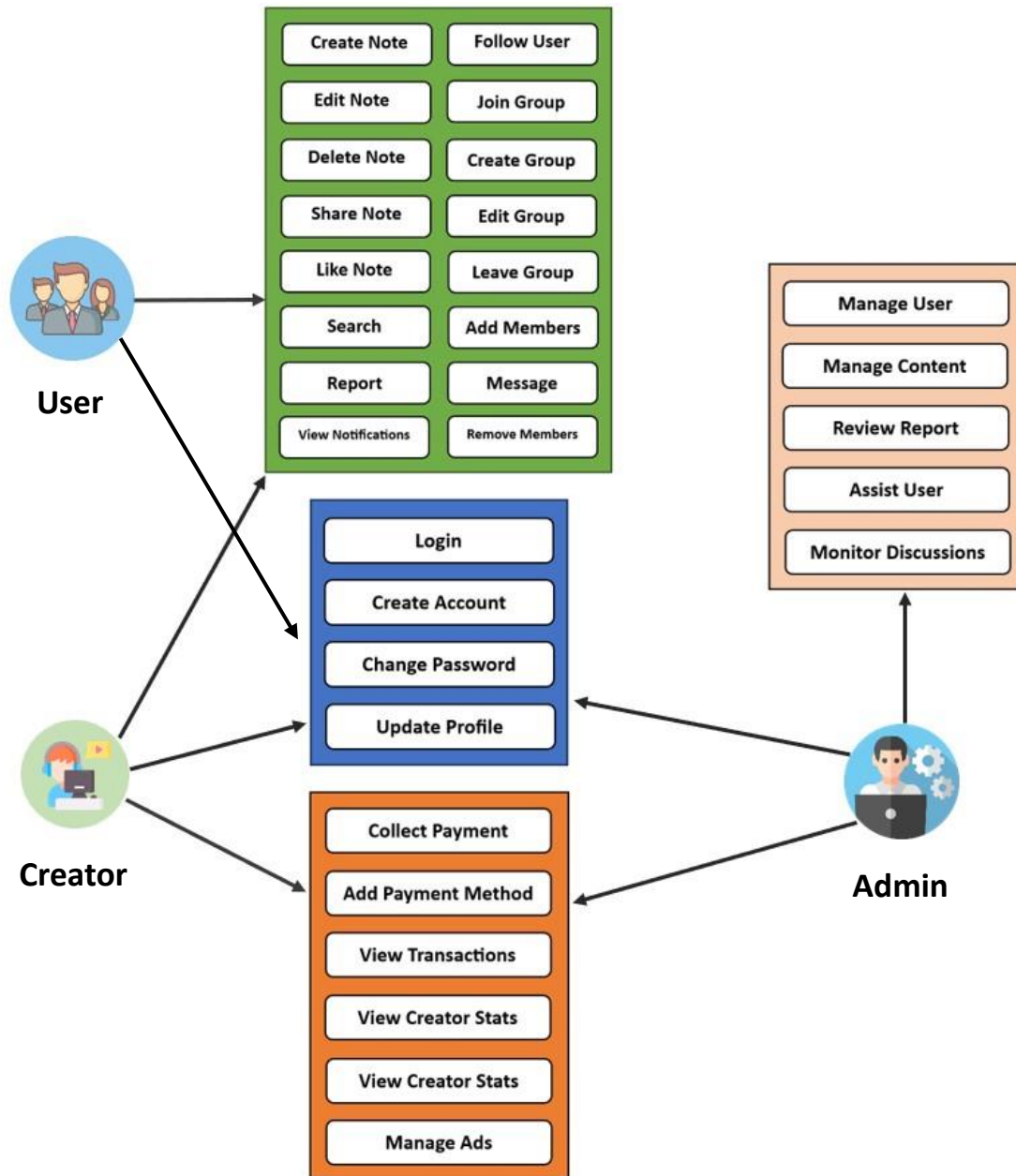


Fig: Use-Case Diagram for our projected platform

6. Conclusion:

Work on the project can be started as soon as all the necessary resources are acquired, and the entire team has been assembled and briefed.

Methodologies:

The Software Development Life Cycle (SDLC) is a systematic process for planning, creating, testing, deploying, and maintaining information systems. There are various models within the SDLC framework, each with its own set of methodologies, phases, and practices. Here are definitions for eight common SDLC models:

Waterfall Model:

Definition: The Waterfall model is a linear and sequential approach to software development. It progresses through defined phases such as requirements, design, implementation, testing, deployment, and maintenance. Each phase must be completed before the next one begins.

Iterative Model:

Definition: The Iterative model involves repeating cycles of the development process, with each iteration building upon the previous one. It allows for feedback and adjustments throughout the development lifecycle, leading to incremental improvements.

Incremental Model:

Definition: The Incremental model divides the software development process into smaller, manageable parts called increments. Each increment represents a portion of the system's functionality and is developed and delivered independently. New increments are added until the system is complete.

Spiral Model:

Definition: The Spiral model combines elements of the Waterfall model and iterative development. It emphasizes risk assessment and management throughout the project. The development process progresses through a series of spirals, with each loop representing a phase of the project.

V-Model (Verification and Validation Model):

Definition: The V-Model is an extension of the Waterfall model that incorporates testing at each stage of development. It emphasizes the relationship between each development phase and its associated testing phase, forming a V-shaped structure.

Agile Model:

Definition: The Agile model is an iterative and flexible approach to software development that prioritizes collaboration, customer feedback, and the ability to respond to changes quickly. It involves short development cycles known as iterations or sprints.

Scrum Model:

Definition: Scrum is an Agile framework that organizes development work into fixed-length iterations called sprints. It emphasizes collaboration, adaptability, and regular review meetings (sprint reviews and retrospectives) to improve the development process continuously.

RAD Model (Rapid Application Development):

Definition: The RAD model is a type of incremental model that focuses on rapid prototyping and quick feedback. It involves user feedback and iteration to refine the software quickly. RAD is particularly suitable for projects with high user involvement and changing requirements.

These models provide different approaches to software development, and the choice of a particular model depends on factors such as project requirements, timeline, budget, and the level of flexibility needed. Based on the provided Software Requirements Specification (SRS), an Agile development model would be suitable for the "Lyte" note-sharing platform. Here's why:

Iterative Development:

The SRS outlines several features and goals, such as user profiling, note creation and management, connectivity and social features, and more. These can be developed incrementally and iteratively, allowing for continuous improvement based on user feedback.

Flexibility for Changes:

Agile is well-known for its flexibility in accommodating changes during the development process. Given the dynamic nature of user needs and the evolving nature of technology, Agile allows for adjustments to be made easily in response to feedback and changing requirements.

User-Centric Approach:

The Agile model prioritizes user involvement and feedback at various stages. Since "Lyte" is a platform focused on user-generated content and engagement, an iterative approach that involves users in the development process aligns well with the goals outlined in the SRS.

Regular Releases:

Agile involves delivering a minimum viable product (MVP) in short iterations or sprints. This aligns with the goal of providing regular updates and features to the users of "Lyte." It allows for a faster time-to-market for essential functionalities.

Continuous Improvement:

The SRS mentions features like badges, awards, leaderboards, contests, and challenges. An Agile approach allows for the continuous improvement of these gamification elements based on user engagement and feedback.

Collaboration and Communication:

Agile methodologies, particularly Scrum, emphasize collaboration and communication among team members. Given the distributed nature of responsibilities (as outlined in the SRS), effective communication and collaboration are crucial.

Adaptation to Technology Changes:

The technology landscape is subject to rapid changes. Agile's iterative and adaptive nature allows the development team to incorporate emerging technologies or industry best practices during the development process.

Visibility and Transparency:

Agile provides a transparent development process with regular sprint reviews, allowing stakeholders to have visibility into the progress of the project. This transparency is beneficial for tracking progress against the outlined goals in the SRS.

Considering these factors, an Agile model, such as Scrum, would be a suitable choice for developing the "Lyte" note-sharing platform, ensuring responsiveness to user needs, efficient collaboration among team members, and a user-centric development process.