SNAPSTER$_{TM}$

CSCE5430– Software Engineering Project

Fall 2024 – Section 409 - University of North Texas

Meghana Choudary Nukavarapu - 11702507
Rakesh Aviraboina - 11656475
Sahithi Tallapaneni - 11759900
Venkata Ravi Tarun Elisetty - 11591910

1 Project Background

Today's world of technology is expanding at a high speed due to which there is an increasing amount of user interaction and content creation. Snapster can become a more popular and effective alternative to traditional centralized data storage for a range of applications. Snapster allows users to share audio and videos from their devices by using decentralized data sharing methods. It is made up of individual user accounts that we may utilize to increase our friend list. Another option is to share tales in public mode, which is accessible to everyone.

Ordinary folks may also use this software to communicate with friends and widen their social circle. The developers of the software, managers, marketers, and designers of the “user interface” will be working on this application. Depending on the chosen team structure, the entire development team may work remotely or in an office. We have our application accessible for Android and iOS smartphones. Examining the application's technological details enables us to have peer-to-peer networking for file sharing with other users. We can design an intuitive “user interface” that makes it easy to switch between the application's many parts.

1.1 Software Overview

The management of all user accounts and user-generated content is done via a backend framework. Numerous more social media marketing tools may be used to promote this program. We may publicize the application by putting up adverts. Moving on to the application's financial considerations, we must pay the developers, managers, and designers. In addition, the license for this program and several other automated software products need to be bought. The expense of promoting this application is also included. The user's need for requirements will have an influence on the application's demand in the market. If our product is unable to differentiate itself from the competition, it may lose market share to TikTok, Facebook, and Youtube. Nevertheless, our solution stands apart a little bit because of the decision to utilize decentralized technology. Additionally, the program must be adaptable enough to incrementally incorporate new features in response to user feedback.Private user data, including passwords and usernames, may be entered into the system. User-added metadata is another important source of data for the system. Between tales, there's an opportunity to see sponsored advertising that may be uploaded. The output consists of user profiles with graphic aids that illustrate their personalities. The primary result is the display of material to users based on their search history and interests. The primary output of this program is the shared audio and video files among its users. One significant risk that we believe might arise is that the application may experience some slowdown if the number of users significantly rises. Different file sharing services have a virus risk.

1.2 Organization / Environment Overview

The product manager is ultimately in charge of creating the project's plan and liaising with different teams to guarantee development. The business analyst will do research on the characteristics that users

require and the state of the market. Software engineers, mobile engineers, and backend developers are in charge of creating the application's architecture, putting different sharing protocols into place, and making sure the program is scalable and performs well. The application's visual elements, such as icons, animated stickers, and filters, are the main emphasis of the UX/UI and graphic designers while creating the user interface. The development of test cases and other forms of application testing are the main focuses of the QA engineers and testers. The marketing experts are in charge of organizing campaigns to advertise the application. Data analysts do analysis on the generated data to improve the application's performance. Support engineers and technical personnel concentrate on product maintenance in response to diverse user inquiries. Devops engineers enforce CI/CD to ensure the product is deployed safely. Tools for managing projects such as JIRA, Canva for design, Visual Studio Code for development, React for frameworks, decentralized storage options, NoSQL databases, Selenium for testing, Git for version control, Microsoft Teams for teamwork, AWS for cloud hosting, and so on can be used. The development team is made up of devops engineers, QA engineers, UI/UX designers, backend developers, and mobile developers. We have two options for them to work: either remotely or using a hybrid strategy. Each of these developers has varying levels of experience; junior developers, who work under senior developers, often have one to three years of experience. Some mid-level developers are capable of managing their responsibilities alone. Senior developers are in charge of leading teams, overseeing architecture, and more. They may have more than five years of expertise. The team may work at more than one location, and a typical work week consists of 40 hours. Decentralized storage, user engagement and assistance, flexibility in incorporating new features, and content retention are the main reliability issues that must be taken into account. The program combines decentralized storage, user-involved content sharing, and simple downloading and sharing methods with many features seen in other social networking apps.

1.3 Schedule / Effort Overview

The SNAPSTER project is the progression of various major approbation points particularly the Alpha and Beta and then the final version. In the Integrated Master Schedule (IMS), a preliminary project timeline that covers effective project initiation as well as requirements collection, visual design, software creation, and software testing is provided. Factors such as people for development, design means, testing set-ups and databases have been determined through the Work Breakdown Structure (WBS), and Basis of Estimate (BOE) approaches. Each phase has clear task delineations in order to prevent the project from going off scope as well as being understaffed or poorly equipped. For a more detailed analysis of the activities and their phases, as well as their combinations and resources, please consult the Program Plan Schedules.

Creating a timeline with all the activities that need to be completed will allow us to begin the project's development. We may do market research in the first few months and adhere to our model by being verified on a number of issues, which may take up to three months. Entering the precise development phase is the next step, which may take up to eight months. The following stage, which takes around three months, is to test the product in accordance with consumer feedback as well. You may use the upcoming months to sell the product. The fully created and implemented product has to be periodically

updated and maintained. Content producers and social media influencers will be the main users of this program.

1.4 Management Overview

The managerial and technological approach to the SNAPSTER project will be conceptually designed and systematically implemented. From the other side, an Agile software development model will be employed to manage the project so that there is flexibility and improvements as the work moves from the design to development and testing phases. Such tools as Git, Power BI and JIRA will be used during the process of development and testing to enhance version control, data analysis and task management respectively.

Key project challenges from a technology perspective include preserving media storage in a decentralized manner, ensuring privacy and providing a consistent experience for users across their devices. In order to deal with these risks, testing of the system will be phased and integrate within the management process. In terms of detailed information about the composition of human resources, where staff will work and time management, please check the Program Plan section.

2 Program Plan

The program plan describes the conceptual framework for the design, development, testing and deployment of the decentralized social media application, SNAPSTER TM. This plan presents the cost, schedule and execution of the project. It helps the stockholders as well as the project team to confine themselves in the plan and implement the project.

2.1 Work Breakdown Structure (WBS)

The Work Breakdown Structure (WBS) divides the overall project into smaller, manageable tasks. This is useful to manage and give deliverable outcomes. For SNAPSTER$_{TM}$, the WBS includes tasks from software development to financial management, testing, and project oversight.

1. Requirements Gathering Tasks

In the case of the SNAPSTER TM app, the Requirements Gathering phase will be used to identify users' necessities and collect the most important requirements. Initial interviews with the potential users will be conducted by the UX Research Team that will cost approximately $1,200. Also, surveys are aimed at collecting wider information on trends in users' choices to cost $800. To complement the above plan, the Market Research Team will benchmark the existing social media apps in order to gauge the potential competition, costing $1, 000. Lastly, a Requirements Analyst will consolidate all the functional and non functional physical requirements into a clear document for $1500.

2. Design Tasks

In the Design phase, the goal is to have a nice and productive app. The UI/UX Designers for a good user interface will charge $2000 for the project. Testing sample versions will be built in order to improve usage capabilities, which is estimated at $1,500. Finally, graphic designers will develop logos, icons, and avatars worth $1,200.

3. Software Tasks

Core Feature Development:Company's offerings: This refers to the P2P based file sharing mechanism, whereby the user is able to listen and download audio and video files from other similar devices. Multiple Source Parallel Download: Create settings that help download files at high speeds, through the setting of different sources for files to download from. Search and Categorization Functions: Provide easy to use search capabilities for the content and make it easy for viewers to filter the content based on some categories and subcategories. User Accounts & Authentication: Implement friend system, messaging and notification system.
Additional Feature Development: Avatar Customization and Ranking System: Permission to compile a points based avatar ranking system that would enable the users to make certain points that they earned through sharing content and which they could redeem for avatar upgrades. Video & Camera Features: Design camera with filter and make the filters to be changeable. This task consists of the incorporation of a filter repository that lets users obtain new filters created by the business. Story Mode: Create and initiate the architecture for sharing video/audio on stories, both personal and communal. Make sure that cached playlist status is correctly managed in order to playback a story in an orderly manner.

4. Financial Tasks

Revenue Modeling: Develop the structure for alternative virtual currency, where people can purchase objects and attributes for avatars; include advertising as the second type of income. Budget Allocation: Devise and plan required resources by targeting, development phase, Timings, cost estimates compensation of the development team, tools and other overhead costs.

5.Testing Tasks

Unit Testing: Some of things that you should know include; First check whether there are individual functions (e.g., P2P downloads, file caching) that need to work and must be tested. System Integration Testing: Evaluate the combination of different app components (recording, for example, or decentralized storage). Performance Testing: Use a number of different scripts to put the application through its paces, especially when many people use the application and download content from various sources simultaneously. Security Testing: Make sure that there are no any security concerns particularly in role playing games decentralized file sharing and private story modes.

6. Project Management Tasks

Milestone Tracking: Maintain logs for all features from development phase, through testing to release phase. Stakeholder Communication: Communicate with other stakeholders frequently especially how the project is experiencing some form of a challenge or has encountered a constraint of some sort. Resource Allocation: Distribute work in a way that requires the development teams' specialized skills and time for completion of jobs.

Below is WBS which also suggests the division of tasks and subtasks and requisite planning for visibility during the developmental phase.

Work Breakdown Structure Table

Project Title:	**SNAPSTER TM**	**Prepared by:**	**Meghana Choudary Nukavarapu, Rakesh Aviraboina, Sahithi Tallapaneni, Venkata Ravi Tarun Elisetty**
Project Manager:	**Sahithi Tallapaneni**	**Date / Control Number:**	**9/26/2024**

Element Number	WBS Elements Activity, Task, or Sub-Task Name	Definition of Activity or Task (Description)	Responsible Person or Group	Estimated (E) or Actual (A) Cost (Cross reference to budget)
1	Requirements Gathering Tasks	Activities focused on understanding user needs and documenting requirements for the SNAPSTER TM app.	Project Manager / Requirements Team	-
1.2	User Interviews	Conduct interviews with potential users to understand their needs, preferences, and expectations for the app.	UX Research Team	E: $1,200
1.3	Surveys	Conduct surveys to gather insights on user preferences.	UX Research Team	E: $800
1.4	Competitor Analysis	Analyze features and functionalities of existing social media apps to benchmark against the SNAPSTER TM app.	Market Research Team	E: $1,000
1.5	Documentation of Requirements	Compile and document functional and non-functional requirements into a detailed requirements document.	Requirements Analyst	E: $1,500
2	Design Tasks	Activities related to the design of the SNAPSTER TM app	Design Team	-
2.1	UI/UX Design	Create user interface designs for a seamless user experience.	UI/UX Designers	E: $2,000
2.2	Prototyping and Wireframing	Develop interactive prototypes for user testing useful for feedback collection.	UI/UX Designers	E: $1,500
2.3	Graphic Design	Design logos, icons, avatars, and filters	Graphic Designers	E: $1,200
3	Software Tasks	Development of essential features and functionalities for the SNAPSTER TM app.	Development Team	-
3.1	Core Feature Development	Development of essential features that define the core functionality of the SNAPSTER TM app.	Development Team	-

3.1.1	Decentralized File Sharing	Design and implement the file-sharing system, allowing users to download audio and video files directly from other users' devices.	Development Team	E: $3,000
3.1.2	Multiple Source Parallel Download	Develop algorithms that ensure optimal download speeds by accessing multiple file sources.	Development Team	E: $2,000
3.1.3	Search and Categorization Functions	Implement robust search features that allow users to browse content by categories and subcategories.	Development Team	E: $1,500
3.1.4	User Accounts & Authentication	Build secure user accounts with login and logout capabilities, ensuring user data privacy.	Development Team	E: $2,000
3.1.5	Friend System & Chat	Develop user-to-user communication features, including friending, direct messaging, and notifications.	Development Team	E: $2,500
3.2	Additional Feature Development	Development of additional features that enhance user engagement and interactivity.	Development Team	-
3.2.1	Avatar Customization and Ranking System	Implement a points-based ranking system that allows users to earn points through content sharing, which can be spent on avatar customization.	Development Team	E: $2,000
3.2.2	Video & Camera Features	Build camera functionality with customizable filters, integrating a filter repository for user access.	Development Team	E: $3,000
3.2.3	Story Mode	Design and implement a system where users can upload video/audio to stories, both private and public, ensuring efficient cache management.	Development Team	E: $2,500
3.2.4	Sound	Development of sound features for the application	Sound development team	E: $2000
4	Financial Tasks	Financial planning and budget management tasks for the project.	Financial Team	-
4.1	Revenue Modeling	Create financial models for points-based purchases and integrate advertising as an additional revenue source.	Financial Analyst	E: $1,500
4.2	Budget Allocation	Define and allocate budget for each development phase, including team compensation, tool procurement, and operational expenses.	Financial Team	E: $1,000
5	Testing Tasks	Ensure the application meets quality and performance standards through rigorous testing.	QA Team	-
5.1	Unit Testing	Ensure that individual functions, such as P2P downloads and file caching, work correctly.	QA Engineers	E: $1,200
5.2	System Integration Testing	Test the integration between various app modules, including video recording and decentralized storage.	QA Engineers	E: $1,500
5.3	Performance Testing	Measure the app’s performance under stress, particularly when multiple users download content simultaneously.	QA Engineers	E: $1,500

5.4	Security Testing	Ensure there are no security vulnerabilities, particularly in decentralized file-sharing and private story modes.	Security Specialist	E: $1,000
6	Project Management Tasks	Oversee project execution and ensure timely delivery of features.	Project Manager	-
6.1	Milestone Tracking	Track progress through all phases of the project, from feature development to final release.	Project Manager	E: $800
6.2	Stakeholder Communication	Regularly update stakeholders on progress, addressing any delays or resource constraints.	Project Manager	E: $600
6.3	Resource Allocation	Assign tasks to development teams based on their expertise and availability.	Project Manager	E: $500

2.2 Organizational Breakdown Structure (OBS)

The Organizational Breakdown Structure (OBS) acts as a chart of the project team and as such it shows which engineering groups and teams are responsible for performing specific works outlined in the WBS of the SNAPSTER TM app. Such a structure also makes it possible for all tasks to be articulated to avoid confusion when implementing the project.

The Frontend Development Team focuses on interface designs, user control, friends' system, avatars editing, and some additional applications in the form of a chat. To ensure customers get the best experience after interaction with the created applications, this team adopts technologies such as React Native for the development of mobile applications that are compatible with both operating systems, and Sketch for UI/UX design.

Rather, the Backend Development Team works on the technical part of the application and how it will run. Their responsibilities include the design of the decentralized file sharing structure of Tbit and algorithms for parallel download and synchronization, data synchronization between appliances and user content cache. They make efficient use of Node.js, MongoDB as well as for cache management, utilizing Redis.

This is because error-free file sharing and privacy are important for decentralized file sharing and the Quality Assurance (QA) Team is responsible for ensuring that the system is strong and secure. He stated that they benchmark the performance as they seek to know whether it can cope with the growing demand and still retain the high availability. For this purpose, they use other tools such as JUnit and Selenium used for testing.

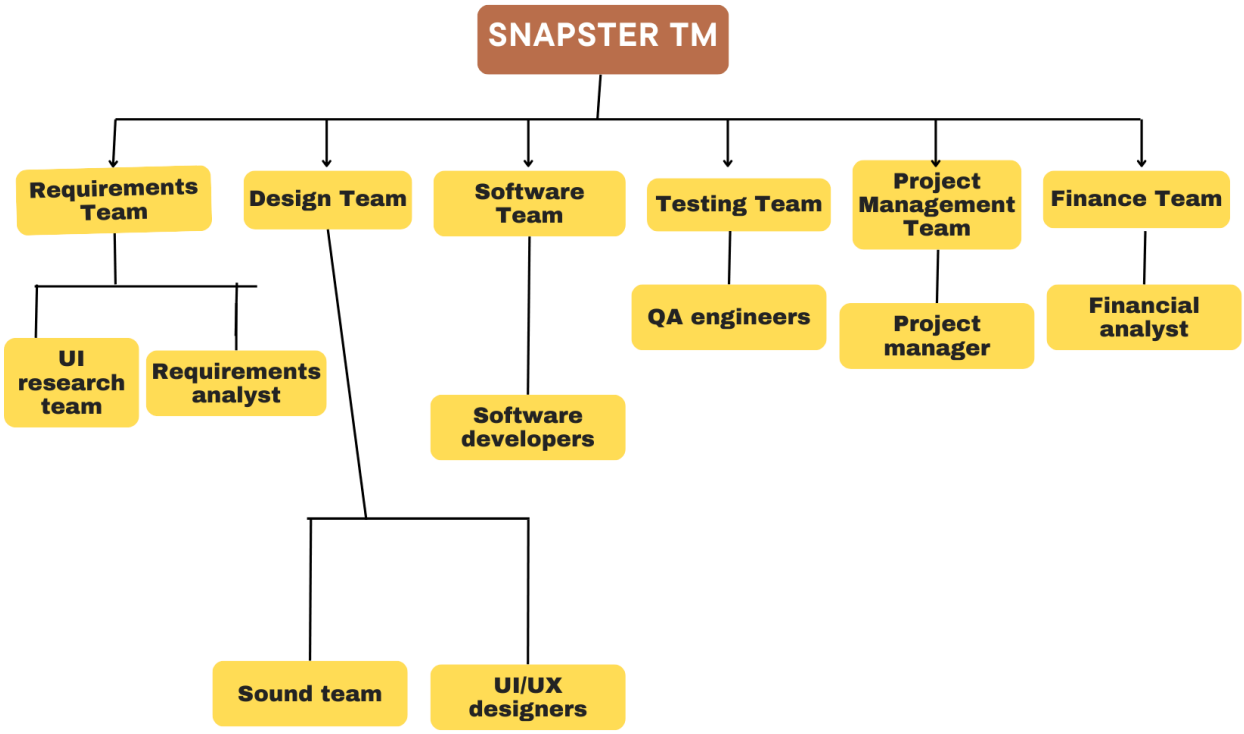
We have an Infrastructure Team that handles the server-side features such as filter libraries, to ensure that the system is capable of multiple update requests. They are also, therefore, held accountable for ensuring data consistency and accessibility throughout content distribution. This team uses AWS for

achieving cloud computing objectives and Docker for achieving containerisation objectives effectively and efficiently.

Finally, the Project Management Team controls the entire duration of the project, costs and availability of resources. Another advantage, they are usually involved in managing the communication between development and QA teams so scheduling and coordinating each module to be delivered on time.

Due to the need to produce a clear-cut and more elaborative view of the organization structure chart the OBS will be developed. This visual tool will improve awareness within the team so that all members will be fully informed of the various expectations in the course of the project implementation.

SNAPSTER TM Organizational Breakdown Structure



2.3 Project Effort Estimation

To estimate the project's effort and cost, the Carnegie Mellon's Constructive COCOMO II model will be used. This model provides a derivative path to quantifying the total amount of manpower as person months that would be expected to be used in software development depending on the size and nature of the project.

COCOMO II (Constructive Cost Model II) is an algorithmic cost estimation model frequently used in software engineering for estimating the effort, cost and duration of an SW development project. It expands the standard COCOMO model and fits it to more recent trends in software development as well as technologies hence it can be useful to a wider range of projects. Here's a breakdown of its key components:

COCOMO II consists of three main components:

- Size Estimation: This includes making a forecast into the proportions of the software project which could be in the form of lines of code (LOC), function points (FP) or any other measure.
- Cost Estimation: The predicted development effort and cost based on specific project attributes, the estimated size is utilized by the model.
- Schedule Estimation: Subsequently, using the estimated efforts, the model generates the project timeline.

Cost Drivers:

COCOMO II consists of a set of cost drivers that influences the amount of effort on a particular project. These include factors related to:

- Product Attributes: Cost issues such as the complexity of the decision, its reliability and the amount of performance required.
- Hardware Attributes: Availability of the computer hardware, the performance as measured by speed and type of the computer hardware, and the limitation of the hardware.
- Personnel Attributes: Comfort with experience and capability of the team, retention of the team.
- Project Attributes: Development environment: Tools used,@Getter application of software

Estimation Equation

The basic effort estimation equation in COCOMO II is expressed as:

$$\textit{Effort (person-months)} = A \times (KDSI)^{B} \times \prod(\textit{Cost Drivers})$$

Where:

- A and B are coefficients that are extracted from data of the prior period.
- KDSI is the activity size measure defined as the estimated size in thousands of the delivered source instructions (or other size metric).
- The cost drivers of the product help in modifying the base effort estimate depending on the characteristics of the project in question.

2.3.1 Project Staffing

The staffing plan categorizes the personnel into three levels of experience:

- Novice: Junior developers to work on UI, its customization for basic aspects of the game, testing, first simple actions, they can contribute, for example, to avatars' settings, or chats' section.
- Journeyman: The mid- level, key developers involved in the project since the decentralized storage for additional downloads.
- Expert: Managerial personnel who have been in charge of designing P2P systems, setting up the network and optimizing the system.

Team Breakdown by Experience Level

Team	Novice	Journeyman	Expert	Total Personnel
Development	2	4	1	7
Testing	1	3	1	5
Quality Assurance	1	2	1	4
DevOps/Deployment	1	2	1	4
Project Management	0	1	1	2
Total Personnel	5	12	5	22

Salary Estimates by Experience Level

Experience Level	Average Annual Salary (USD)
Novice	$70,000
Journeyman	$100,000
Expert	$140,000

Salary Calculation per Team: We multiply the number of personnel in each experience level by their respective salaries to get total salary costs for each team.

Development Team

- Novice: 2 x $70,000 = $140,000
- Journeyman: 4 x $100,000 = $400,000
- Expert: 1 x $140,000 = $140,000

Total Development Salary: $680,000

Testing Team

- Novice: 1 x $70,000 = $70,000
- Journeyman: 3 x $100,000 = $300,000
- Expert: 1 x $140,000 = $140,000

Total Testing Salary: $510,000

Quality Assurance Team

- Novice: 1 x $70,000 = $70,000
- Journeyman: 2 x $100,000 = $200,000
- Expert: 1 x $140,000 = $140,000

Total QA Salary: $410,000

DevOps/Deployment Team

- Novice: 1 x $70,000 = $70,000
- Journeyman: 2 x $100,000 = $200,000
- Expert: 1 x $140,000 = $140,000

Total DevOps Salary: $410,000

Project Management Team

- Journeyman: 1 x $100,000 = $100,000
- Expert: 1 x $140,000 = $140,000

Total PM Salary: $240,000

Total Base Salary Costs

Team	Total Salary (USD)
Development	$680,000
Testing	$510,000
Quality Assurance	$410,000
DevOps/Deployment	$410,000
Project Management	$240,000
Total Salary	$2,250,000

Burdened Labor Rate Calculation: To calculate the burdened labor rate we multiply each salary by 1.5 to account for overhead costs such as electricity, facility space, equipment, and other operational expenses.

Total Burdened Labor Cost = $2,250,000 x 1.5 = $3,375,000

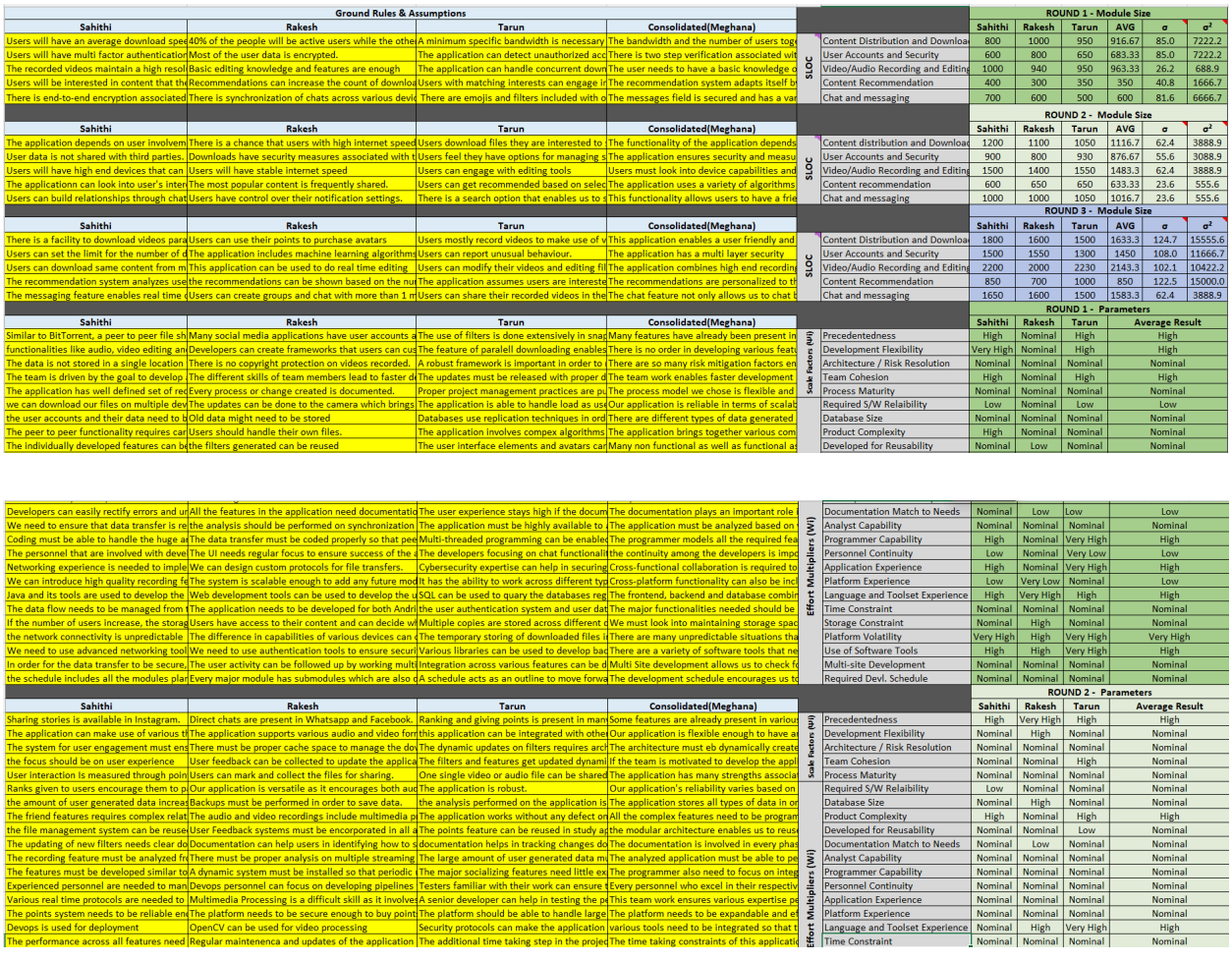
Summary:

- Total Personnel: 22
- Total Base Salary: $2,250,000
- Burdened Labor Cost (with Overheads): $3,375,000

2.3.2 Wideband Delphi Estimates

For SNAPSTER TM, the following estimates will be generated:

- Software Lines of Code (SLOC): Every top-level facility (such as file sharing, a chat system, filters) will have an estimation of SLOC.
- Cost Drivers: A number of issues like the complexity of the utilized platform, experience of personnel, and necessary function capabilities will be addressed.



Ground Rules & Assumptions

Sahithi	Rakesh	Tarun	Consolidated(Meghana)
Users will have an average download spee	40% of the people will be active users while the othe	A minimum specific bandwidth is necessary	The bandwidth and the number of users tog
Users will have multi factor authentication	Most of the user data is encrypted.	The application can detect unauthorized acc	There is two step verification associated wit
The recorded videos maintain a high resol	Basic editing knowledge and features are enough	The application can handle concurrent down	The user needs to have a basic knowledge o
Users will be interested in content that th	Recommendations can increase the count of downloa	Users with matching interests can engage in	The recommendation system adapts itself b
There is end-to-end encryption associated	There is synchronization of chats across various devi	There are emojis and filters included with o	The messages field is secured and has a va

Sahithi	Rakesh	Tarun	Consolidated(Meghana)
The application depends on user involvem	There is a chance that users with high internet speed	Users download files they are interested to	The functionality of the application depends
User data is not shared with third parties.	Downloads have security measures associated with t	Users feel they have options for managing s	The application ensures security and measu
Users will have high end devices that can	Users will have stable internet speed	Users can engage with editing tools	Users must look into device capabilities and
The applicationn can look into user's inter	The most popular content is frequently shared.	Users can get recommended based on selec	The application uses a variety of algorithms
Users can build relationships through chat	Users have control over their notification settings.	There is a search option that enables us to s	This functionality allows users to have a frie

Sahithi	Rakesh	Tarun	Consolidated(Meghana)
There is a facility to download videos para	Users can use their points to purchase avatars	Users mostly record videos to make use of v	This application enables a user friendly and
Users can set the limit for the number of d	The application includes machine learning algorithms	Users can report unusual behaviour.	The application has a multi layer security
Users can download same content from m	This application can be used to do real time editing	Users can modify their videos and editing fil	The application combines high end recordin
The recommendation system analyzes use	the recommendations can be shown based on the nu	The application assumes users are intereste	The recommendations are personalized to t
The messaging feature enables real time c	Users can create groups and chat with more than 1 m	Users can share their recorded videos in the	The chat feature not only allows us to chat

Sahithi	Rakesh	Tarun	Consolidated(Meghana)
Similar to BitTorrent, a peer to peer file sh	Many social media applications have user accounts a	The use of filters is done extensively in snag	Many features have already been present in
functionalities like audio, video editing an	Developers can create frameworks that users can cus	The feature of paralell downloading enables	There is no order in developing various feat
The data is not stored in a single location	There is no copyright protection on videos recorded.	A robust framework is important in order to	There are so many risk mitigation factors en
The team is driven by the goal to develop	The different skills of team members lead to faster d	The updates must be released with proper d	The team work enables faster development
The application has well defined set of rec	Every process or change created is documented.	Proper project management practices are pu	The process model we chose is flexible and
we can download our files on multiple dev	The updates can be done to the camera which brings	The application is able to handle load as use	Our application is reliable in terms of scalat
the user accounts and their data need to b	Old data might need to be stored	Databases use replication techniques in ord	There are different types of data generated
The peer to peer functionality requires car	Users should handle their own files.	The application involves compex algorithms	The application brings together various com
The individually developed features can b	the filters generated can be reused	The user interface elements and avatars car	Many non functional as well as functional a

ROUND 1 - Module Size

		Sahithi	Rakesh	Tarun	AVG	σ	σ²
SLOC	Content Distribution and Downloa	800	1000	950	916.67	85.0	7222.2
	User Accounts and Security	600	800	650	683.33	85.0	7222.2
	Video/Audio Recording and Editing	1000	940	950	963.33	26.2	688.9
	Content Recommendation	400	300	350	350	40.8	1666.7
	Chat and messaging	700	600	500	600	81.6	6666.7

ROUND 2 - Module Size

		Sahithi	Rakesh	Tarun	AVG	σ	σ²
SLOC	Content distribution and Downloac	1200	1100	1050	1116.7	62.4	3888.9
	User Accounts and Security	900	800	930	876.67	55.6	3088.9
	Video/Audio Recording and Editing	1500	1400	1550	1483.3	62.4	3888.9
	Content recommendation	600	650	650	633.33	23.6	555.6
	Chat and messaging	1000	1000	1050	1016.7	23.6	555.6

ROUND 3 - Module Size

		Sahithi	Rakesh	Tarun	AVG	σ	σ²
SLOC	Content Distribution and Downloa	1800	1600	1500	1633.3	124.7	15555.6
	User Accounts and Security	1500	1550	1300	1450	108.0	11666.7
	Video/Audio Recording and Editing	2200	2000	2230	2143.3	102.1	10422.2
	Content Recommendation	850	700	1000	850	122.5	15000.0
	Chat and messaging	1650	1600	1500	1583.3	62.4	3888.9

ROUND 1 - Parameters

		Sahithi	Rakesh	Tarun	Average Result
Scale Factors (Wi)	Precedentedness	High	Nominal	High	High
	Development Flexibility	Very High	Nominal	High	High
	Architecture / Risk Resolution	Nominal	Nominal	Nominal	Nominal
	Team Cohesion	High	Nominal	High	High
	Process Maturity	Nominal	Nominal	Nominal	Nominal
Effort Multipliers (Wi)	Required S/W Relaibility	Low	Nominal	Low	Low
	Database Size	Nominal	Nominal	Nominal	Nominal
	Product Complexity	High	Nominal	Nominal	Nominal
	Developed for Reusability	Nominal	Low	Nominal	Nominal

Sahithi	Rakesh	Tarun	Consolidated(Meghana)
Developers can easily rectify errors and ur	All the features in the application need documentatio	The user experience stays high if the docum	The documentation plays an important role
We need to ensure that data transfer is re	the analysis should be performed on synchronization	The application must be highly available to	The application must be analyzed based on
Coding must be able to handle the huge a	The data transfer must be coded properly so that pee	Multi-threaded programming can be enable	The programmer models all the required fea
The personnel that are involved with deve	The UI needs regular focus to ensure success of the a	The developers focusing on chat functionalit	the continuity among the developers is impo
Networking experience is needed to imple	We can design custom protocols for file transfers.	Cybersecurity expertise can help in securing	Cross-functional collaboration is required to
We can introduce high quality recording fe	The system is scalable enough to add any future mod	It has the ability to work across different typ	Cross-platform functionality can also be incl
Java and its tools are used to develop the	Web development tools can be used to develop the u	SQL can be used to quary the databases reg	The frontend, backend and database combir
The data flow needs to be managed from t	The application needs to be developed for both Andri	the user authentication system and user dat	The major functionalities needed should be
If the number of users increase, the storag	Users have access to their content and can decide wh	Multiple copies are stored across different c	We must look into maintaining storage spac
the network connectivity is unpredictable	The difference in capabilities of various devices can	The temporary storing of downloaded files i	There are many unpredictable situations tha
We need to use advanced networking tool	We need to use authentication tools to ensure securi	Various libraries can be used to develop bac	There are a variety of software tools that ne
In order for the data transfer to be secure,	The user activity can be followed up by working multi	Integration across various features can be d	Multi Site development allows us to check fo
the schedule includes all the modules plan	Every major module has submodules which are also d	A schedule acts as an outline to move forwa	The development schedule encourages us to

		Documentation Match to Needs	Nominal	Low	Low	Low
Scale Factors (Wi)	Analyst Capability	Nominal	Nominal	Nominal	Nominal	
	Programmer Capability	High	Nominal	Very High	High	
	Personnel Continuity	Low	Nominal	Very Low	Low	
	Application Experience	High	Nominal	Very High	High	
	Platform Experience	Low	Very Low	Nominal	Low	
Effort Multipliers (Wi)	Language and Toolset Experience	High	Very High	High	High	
	Time Constraint	Nominal	Nominal	Nominal	Nominal	
	Storage Constraint	Nominal	High	Nominal	Nominal	
	Platform Volatility	Very High	High	Very High	Very High	
	Use of Software Tools	High	High	Very High	High	
	Multi-site Development	Nominal	Nominal	Nominal	Nominal	
	Required Devl. Schedule	Nominal	Nominal	Nominal	Nominal	

ROUND 2 - Parameters

Sahithi	Rakesh	Tarun	Consolidated(Meghana)			Sahithi	Rakesh	Tarun	Average Result
Sharing stories is available in Instagram.	Direct chats are present in Whatsapp and Facebook.	Ranking and giving points is present in many	Some features are already present in variou	Scale Factors (Wi)	Precedentedness	High	Very High	High	High
The application can make use of various th	The application supports various audio and video for	this application can be integrated with othe	Our application is flexible enough to have a		Development Flexibility	Nominal	High	Nominal	Nominal
The system for user engagement must ens	There must be proper cache space to manage the do	The dynamic updates on filters requires arch	The architecture must eb dynamically create		Architecture / Risk Resolution	Nominal	Nominal	Nominal	Nominal
the focus should be on user experience	User feedback can be collected to update the applica	The filters and features get updated dynami	If the team is motivated to develop the appl		Team Cohesion	Nominal	Nominal	High	Nominal
User interaction is measured through poin	Users can mark and collect the files for sharing.	One single video or audio file can be shared	The application has many strengths associat		Process Maturity	Nominal	Nominal	Nominal	Nominal
Ranks given to users encourage them to p	Our application is versatile as it encourages both aud	The application is robust.	Our application's reliability varies based on	Effort Multipliers (Wi)	Required S/W Relaibility	Low	Nominal	Nominal	Nominal
the amount of user generated data increas	Backups must be performed in order to save data.	the analysis performed on the application is	The application stores all types of data in or		Database Size	Nominal	High	Nominal	Nominal
The friend features requires complex relat	The audio and video recordings include multimedia p	The application works without any defect or	All the complex features need to be program		Product Complexity	High	Nominal	High	High
the file management system can be reuse	User Feedback systems must be encorporated in all a	The points feature can be reused in study ap	the modular architecture enables us to reus		Developed for Reusability	Nominal	Nominal	Low	Nominal
The updating of new filters needs clear do	Documentation can help users in identifying how to s	documentation helps in tracking changes do	The documentation is involved in every phas		Documentation Match to Needs	Nominal	Low	Nominal	Nominal
The recording feature must be analyzed fr	There must be proper analysis on multiple streaming	The large amount of user generated data m	The analyzed application must be able to pe		Analyst Capability	Nominal	Nominal	Nominal	Nominal
The features must be developed similar to	A dynamic system must be installed so that periodic	The major socializing features need little ex	The programmer also need to focus on integ		Programmer Capability	Nominal	Nominal	Nominal	Nominal
Experienced personnel are needed to man	Devops personnel can focus on developing pipelines	Testers familiar with their work can ensure t	Every personnel who excel in their respectiv		Personnel Continuity	Nominal	Nominal	Nominal	Nominal
Various real time protocols are needed to	Multimedia Processing is a difficult skill as it involves	A senior developer can help in testing the pa	This team work ensures various expertise pe		Application Experience	Nominal	Nominal	Nominal	Nominal
The points system needs to be reliable en	The platform needs to be secure enough to buy point	The platform should be able to handle large	The platform needs to be expandable and ef		Platform Experience	Nominal	Nominal	Nominal	Nominal
Devops is used for deployment	OpenCV can be used for video processing	Security protocols can make the application	various tools need to be integrated so that t		Language and Toolset Experience	Nominal	High	Very High	High
The performance across all features need	Regular maintenenca and updates of the application	The additional time taking step in the projec	The time taking constraints of this applicatio		Time Constraint	Nominal	Nominal	Nominal	Nominal

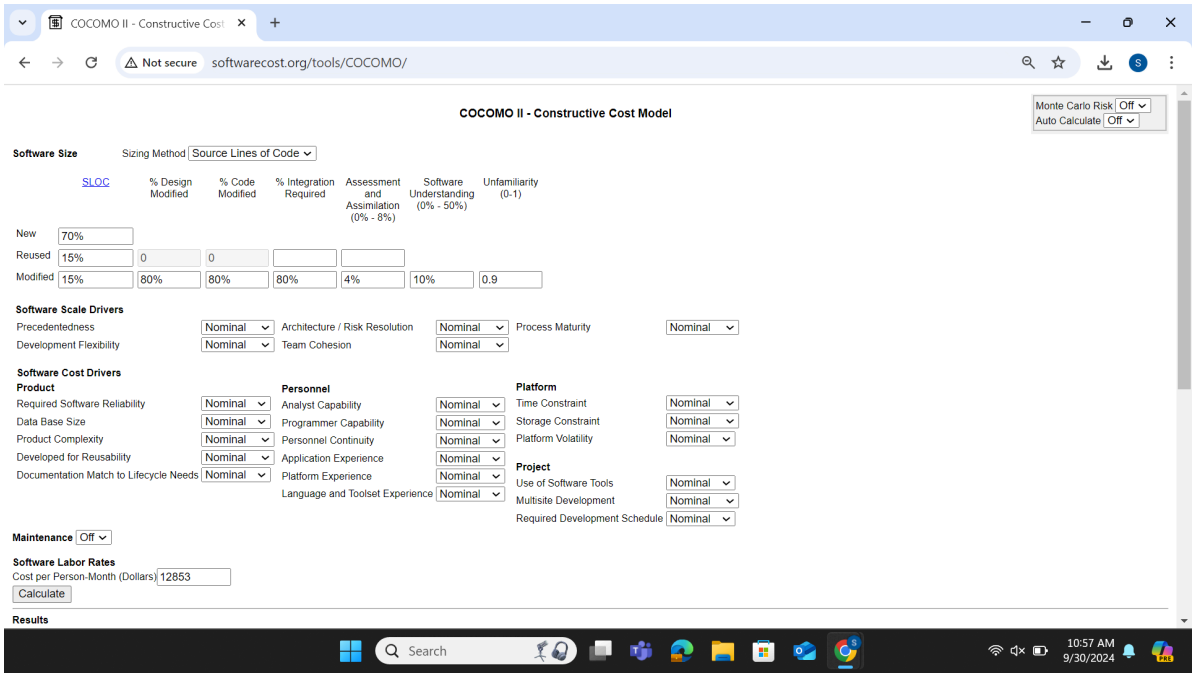
Sahithi	Rakesh	Tarun	Consolidated(Meghana)			ROUND 2 - Parameters			
						Sahithi	Rakesh	Tarun	Average Result
Sharing stories is available in Instagram.	Direct chats are present in Whatsapp and Facebook.	Ranking and giving points is present in many	Some features are already present in various	Scale Factors (Wi)	Precedentedness	High	Very High	High	High
The application can make use of various th	The application supports various audio and video for	this application can be integrated with other	Our application is flexible enough to have ac		Development Flexibility	Nominal	High	Nominal	Nominal
The system for user engagement must ens	There must be proper cache space to manage the do	The dynamic updates on filters requires arch	The architecture must eb dynamically create		Architecture / Risk Resolution	Nominal	Nominal	Nominal	Nominal
the focus should be on user experience	User feedback can be collected to update the applica	The filters and features get updated dynami	If the team is motivated to develop the appli		Team Cohesion	Nominal	Nominal	High	Nominal
User interaction is measured through poin	Users can mark and collect the files for sharing.	One single video or audio file can be shared	The application has many strengths associa		Process Maturity	Nominal	Nominal	Nominal	Nominal
Ranks given to users encourage them to p	Our application is versatile as it encourages both aud	The application is robust.	Our application's reliablilty varies based on		Required S/W Relaibility	Low	Nominal	Nominal	Nominal
the amount of user generated data increas	Backups must be performed in order to save data.	the analysis performed on the application is	The application stores all types of data in or		Database Size	Nominal	High	Nominal	Nominal
The friend features requires complex relat	The audio and video recordings include multimedia p	The application works without any defect or	All the complex features need to be progran		Product Complexity	High	Nominal	High	High
the file management system can be reuse	User Feedback systems must be encorporated in all a	The points feature can be reused in study ag	the modular architecture enables us to reus		Developed for Reusability	Nominal	Nominal	Low	Nominal
The updating of new filters needs clear do	Documentation can help users in identifying how to s	documentation helps in tracking changes do	The documentation is involved in every phas		Documentation Match to Needs	Nominal	Low	Nominal	Nominal
The recording feature must be analyzed fr	There must be proper analysis on multiple streaming	The large amount of user generated data mi	The analyzed application must be able to pe		Analyst Capability	Nominal	Nominal	Nominal	Nominal
The features must be developed similar to	A dynamic system must be installed so that periodic	The major socializing features need little ex	The programmer also need to focus on integ		Programmer Capability	Nominal	Nominal	Nominal	Nominal
Experienced personnel are needed to man	Devops personnel can focus on developing pipelines	Testers familiar with their work can ensure t	Every personnel who excel in their respectiv		Personnel Continuity	Nominal	Nominal	Nominal	Nominal
Various real time protocols are needed to	Multimedia Processing is a difficult skill as it involves	A senior developer can help in testing the pe	This team work ensures various expertise pe		Application Experience	Nominal	Nominal	Nominal	Nominal
The points system needs to be reliable en	The platform needs to be secure enough to buy point	The platform should be able to handle large	The platform needs to be expandable and et		Platform Experience	Nominal	Nominal	Nominal	Nominal
Devops is used for deployment	OpenCV can be used for video processing	Security protocols can make the application	various tools need to be integrated so that t		Language and Toolset Experience	Nominal	High	Very High	High
The performance across all features need	Regular maintenenca and updates of the application	The additional time taking step in the projec	The time taking constraints of this applicati	Effort Multipliers (Wi)	Time Constraint	Nominal	Nominal	Nominal	Nominal
The stored data is distributed through whi	Some specific data within the storage can be retaine	There is no centralized storage so there is a	chance of some bottlenecks slowing down th		Storage Constraint	Nominal	Nominal	Nominal	Nominal
There is a chance of getting synchronizati	Differences in user engagement can result in reducti	There is a chace that high data traffic can ca	One feature is dependent on another which		Platform Volatility	Nominal	Nominal	High	Nominal
Various camera and recording tools need t	Databases need to be used to save the data that nee	Secure encryption protocols are put into plac	The high end software tools can be used if t		Use of Software Tools	High	Nominal	Nominal	Nominal
Real time messages ensure high availabili	Data across different regions can be handled	Multi site development enables fault toleran	The advantages of multi site processing is u		Multi-site Development	Nominal	Nominal	Nominal	Nominal
The individual smaller tasks which can be	The schedule can be a little flexible but not too much	The schedule also includes the deployment (	The devl. Schedule is a way to understand th		Required Devl. Schedule	Nominal	Nominal	Nominal	Nominal

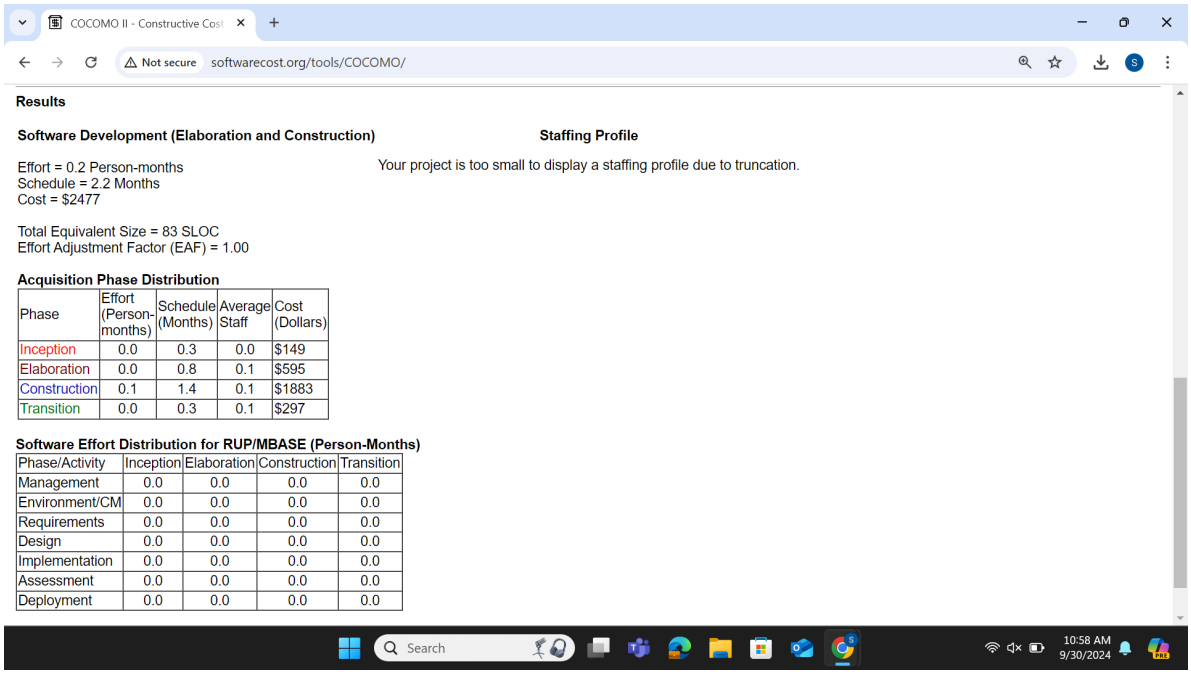
2.3.3 Estimation Modeling

The project estimation modeling process will contribute to undertaking appropriate planning for the SNAPSTER TM project through appropriate time, cost, and resource projections. We have adopted COCOMO II, that estimate software development effort and cost by considering project size, complexity, and team capabilities.

Key assumptions made are as follows:

- SLOC: new code 70%, modified 15% and reused 15%.
- 80% of the design would be modified,
- 10% for software understanding and
- 0.9 for unfamiliarity.
- Application of burdened salary rate to estimate the total labor cost.





2.3.4 Basis of Estimate (BOE)

The BOE for the SNAPSTER TM project involves all the aspects of software development engineering effort such as development, testing, quality assurance, deployment, and project management. This BOE reflects the in-depth understanding of the resources, costs, and efforts required for the project as described in the earlier sections, particularly section 2.3.2.

Labor Estimates: The estimated costs of labor are based on the staffing plan, with three levels of experience for personnel: Novice, Journeyman, and Expert. The salary cost estimates are shown below:

Development Team: $680,000
Testing Team: $510,000
Quality Assurance Team: $410,000
DevOps/Deployment Team: $410,000
Project Management Team: $240,000
These labor costs will further be spread out across various project phases, namely Inception, Elaboration, Construction, and Transition, as elaborated in the Work Breakdown Structure. The appropriate allocation of personnel hours and associated costs for each phase will reflect to ensure that estimates of every aspect of the software development lifecycle have been taken into account in the overall estimation.

Other Costs: Besides labor, the BOE should account for other vital costs beyond the scope of the labor projections. The costs with regard to tools, travels, and other additional resources, put together in

consonance with the project, are all included. For instance, software licenses, testing tools, and any other training that might be needed by team members all add up to the total budget. Contingency funds should be set aside to deal with unforeseen expenses during the project lifecycle.

- Travel Costs: $5,000
- Software Licenses/Tools: $10,000
- Testing Tools/Infrastructure: $8,000
- Contingency Reserve (10% of total labor cost):

10% of 2,250,000 = $225,000

The BOE will summarize the total estimated effort and cost, including:

- **Software Development Effort**: Estimated in person-months, broken down into phases such as Inception, Elaboration, Construction, and Transition.
- **Cost Justification**: Every task, from module implementation to testing, will be costed using the effort estimates derived from COCOMO II.

BOE ID: 456	**WBS: ALL**		
Author Name: Meghana Choudary Nukavarapu, Rakesh Aviraboina, Sahithi Tallapaneni, Venkata Ravi Tarun Elisetty	**Level** **N/A**	**Period of Performance: 12 months**	
		Total Hours	**1630**

Task Descriptions:

1.1 Requirements Gathering

2.1 Design Tasks

3.1 Software Development Tasks

4.1 Financial tasks

5.1 Testing Tasks

6.1 Management Tasks

Basis of Estimate:

The requirements gathering phase can take approximately 60 hours.

The design phase can take approximately 150 hours.

The development phase can take approximately 500 hours. The financial planning phase can take approximately 70 hours. The testing phase can take approximately 150 hours.

The management phase can take approximately 150 hours.

Total Hours for Labor estimate = 1630

Total material cost = $8600

Total Estimate for Trips = $24940

Labor Estimate:

WBS	Task Description	Calculation	Period	Hours
1.1	Stakeholder Interviews	3 members *20 hrs	Inception	60
1.1	Document Requirements	2 members * 40 hrs	Elaboration	80
2.1	UI design	2 designers * 60 hrs	Elaboration	120
2.1	Architectural design	1 architect * 80 hrs	Construction	80
3.1	Frontend Development	2 developers * 160 hrs	Construction	320

3.1	Backend Development	2 developers * 160 hrs	Construction	320
3.1	Integration	2 developers * 80 hrs	Transition	160
4.1	Budget Planning	1 analyst * 40 hrs	Inception	40
4.1	Financial Reporting	1 analyst * 30 hrs	Elaboration	30
5.1	Unit Testing	2 testers * 60 hrs	Transition	120
5.1	Integration Testing	2 testers * 40 hrs	Transition	80
5.1	User Acceptance Testing	2 testers * 40 hrs	Transition	80
6.1	Project Management	1 manager * 80 hrs	Transition	80
6.1	Team Meetings	3 members * 80 hrs	All phases	60

Total Hours for Requirements Gathering = 140
Total Hours for Design = 200
Total Hours for software development = 800

Total Hours for financial tasks = 70

Total Hours for testing tasks = 280

Total Hours for management tasks = 140

Total Hours for Labor estimate = 1630

Material Estimate:

WBS	Purchase Description	Calculation	Cost
1.1	Survey Software for conducting user surveys(Typeform)	1 license * 500/year	500
1.1	Documentation Tools for noting requirements(Notion)	1 license * 600/year	600
2.1	UI/UX Design software for creating prototypes(Figma)	2 licenses *1200/year	2400
2.1	Prototyping tool for interactive prototypes (InVision)	1 license * 600/year	600
3.1	Integrated Development Environment(Visual Studio)	5 licenses * 500/year	2500
3.1	Version Control system(Git)	1 group license * 600/year	600
3.1	Backend Development Frameworks(Django)	Open source	0
3.1	Decentralized storage	Open Source	0
4.1	Financial Planning Software(QuickBooks)	1 license * 500/year	500
5.1	Testing frameworks(Junit)	Open source	0

5.1	Testing Platform(UserTesting)	1 license*1000/year	1000
6.1	Project Management Software(Trello)	1 group license * 500/year	500
6.1	Communication Tools(Teams)	1 group license * 600/year	600

Total material cost for Requirements Gathering = $1100
Total material cost for Design = $1800
Total material cost for software development = $3100
Total material cost for financial tasks = $500
Total material cost for testing tasks = $1000
Total material cost for management tasks = $1100
Total material cost = $8600

Trips Estimate:

WBS	Trip Description	Calculation	Cost
1.1	Meeting with stakeholders in different user locations as a team of 5 members	**Airfare:** $300/person *5 = $1,500 **Accommodation:** $150/night x 5 nights x 3 rooms = $2,250 **Meals**: $50/day x 5 days x 5 people = $1,250 **Local Transport:** $100/day x 5 days = $500	5500

2.1	Collaborative workshops at different design studios as a team of 3 members	**Airfare**: $200/person x 3 = $600 **Accommodation**: $120/night x 4 nights x 3 rooms = $1,440 **Meals**: $50/day x 4 days x 3 people = $600 **Local Transport**: $80/day x 4 days = $320	2960
3.1	Pair Programming sessions at development hubs as a team of 6 developers	**Airfare**: $250/person x 6 = $1500 **Accommodation**: $140/night x 5 nights x 4 rooms = $2,800 **Meals**: $60/day x 5 days x 6 people = $1800**Local Transport**: $100/day x 5 days = $500	6600
4.1	Meeting with financial advisors at corporate headquarters as a team of 2 financial analysts	**Airfare**: $300/person x 2 = $600 **Accommodation**: $130/night x 3 nights x 2 rooms = $780 **Meals**: $50/day x 3 days x 2 people = $300 **Local Transport**: $80/day x 3 days = $240	1920

5.1	Quality Assurance Testings at testing facilities or user locations as a team of 6 testers	**Airfare:** $250/person x 6 = $1,500 **Accommodation:** $120/night x 4 nights x 4 rooms = $1,920 **Meals**: $50/day x 4 days x 6 people = $1,200 **Local Transport:** $100/day x 4 days = $400	5020
6.1	Team meetings with all the teams working on a single project as a group of 4 managers at remote locations or headquarters	**Airfare**: $300/person x 4 = $1,200 **Accommodation:** $150/night x 3 nights x 2 rooms = $900 **Meals:** $50/day x 3 days x 4 people = $600 **Local Transport**: $80/day x 3 days = $240	2940

Total Estimate for Trips = $24940

Rationale/Estimating Methodology:

To estimate the different project expenses, we employed a top-down methodology. This implies that we split down the project into smaller components after starting with a high-level perspective of it. For instance, we begin with the module gathering requirements, divide it into smaller tasks, allocate personnel, and calculate the time and cost of each task separately. The total cost of that portion is then determined by adding together all of the subparts.

The various complexity factors include:

1. Decentralized architecture: Peer-to-peer file sharing and numerous downloads on a single device need advanced networking knowledge, which raises the cost and duration of development.

2. User Features: Estimating becomes more difficult when user features like adding friends, stories, and discussions need extensive backend development in addition to user interface design.

3. User Content: Because of the large volume of user-generated data, it might be difficult to preserve file integrity.

2.3.5 Extra Credit

2.3.2 Wideband Delphi second estimate

Ground Rules & Assumptions						ROUND 1 - Module Size estimates					
Sahithi	Tarun	Rakesh	Consolidated(Meghana)			Sahithi	Tarun	Rakesh	AVG	σ	σ²
This module will require 1500 SLOC and must allow users to create and manage their profiles, including setting up usernames and profile pictures.	This module will require 5000 SLOC and must support user authentication and authorization to ensure secure access to accounts.	This module will require 4500 SLOC and must enable users to update their profile information and view other users' profiles.	This module will require 3666.67 SLOC and must facilitate user notifications for profile changes and friend requests.	SLOC	Module 1 (User Account & Profile)	1500	5000	4500	3666.67	1827.6	0.5
his module will require 3500 SLOC and must allow users to upload media files and share them with friends.	This module will require 2500 SLOC and must support different media formats and handle file storage efficiently.	This module will require 2000 SLOC and must enable users to comment on and like shared media.	This module will require 2666.67 SLOC and must provide a feed of uploaded media for users to browse.		Module 2 (Media Upload & Sharing)	3500	2500	2000	2666.67	763.8	0.3
This module will require 6000 SLOC and must provide a chat interface for direct messaging between users.	This module will require 2800 SLOC and must support multimedia messages (images, videos) in chats.	This module will require 3400 SLOC and must allow users to create group chats and share media within them.	This module will require 4066.67 SLOC and must enable notifications for new messages and mentions.		Module 3 (Chat/DM Functionality)	6000	2800	3400	4066.67	1726.5	0.4
This module will require 7000 SLOC and must enable users to search for and browse content based on categories.	This module will require 4000 SLOC and must allow filtering of search results to find specific media types.	This module will require 1500 SLOC and must provide recommendations based on user interests and activity.	This module will require 4166.67 SLOC and must support trending content displays.		Module 4 (Search & Browse)	7000	4000	1500	4166.67	2821.3	0.7
This module will require 9500 SLOC	This module will require 3200 SLOC and must	This module will require 6000 SLOC and	This module will require 6233.33 SLOC		Module 5 (P2P Download & Storage)	9500	3200	6000	6233.33	3195.6	0.5
						ROUND 2 - Adjustments					
Sahithi	Tarun	Rakesh	Consolidated(Meghana)			Sahithi	Tarun	Rakesh	AVG	σ	σ²
This module will require 1500 SLOC and must allow users to create and edit their profiles with customizable settings.	This module will require 4500 SLOC and must support secure login and account recovery features.	This module will require 4500 SLOC and must enable users to manage privacy settings for their profiles.	This module will require 3500 SLOC and must facilitate user engagement through friend requests and follow options.		Module 1 (User Account & Profile)	1500	4500	4500	3500	1732.1	0.5
This module will require 3500 SLOC and must allow users to upload, edit, and share photos and videos.	This module will require 2300 SLOC and must support simultaneous uploads and batch processing of media files.	This module will require 2000 SLOC and must enable users to tag friends in shared media.	This module will require 2600 SLOC and must provide real-time updates on media interactions.		Module 2 (Media Upload & Sharing)	3500	2300	2000	2600	707.1	0.3
This module will require 5500 SLOC and must enable one-on-one and group messaging capabilities for users.	This module will require 2900 SLOC and must support emoji, stickers, and GIFs in chats.	This module will require 3400 SLOC and must allow users to search chat history for messages.	This module will require 3933.33 SLOC and must provide message encryption for security.	SLOC	Module 3 (Chat/DM Functionality)	5500	2900	3400	3933.33	1292.5	0.3
This module will require 6800 SLOC and must enable users to conduct searches for content across different categories and tags.	This module will require 3600 SLOC and must support sorting and filtering options for search results.	This module will require 1500 SLOC and must allow users to save and bookmark favorite searches.	This module will require 3966.67 SLOC and must display popular categories based on user trends.		Module 4 (Search & Browse)	6800	3600	1500	3966.67	2612.9	0.6
This module will require 9200 SLOC and must facilitate secure file sharing and downloading among users.	This module will require 3500 SLOC and must provide options for downloading multiple files simultaneously.	This module will require 6000 SLOC and must enable users to track their downloaded content and remaining storage.	This module will require 6233.33 SLOC and must ensure data integrity and security during file transfers.		Module 5 (P2P Download & Storage)	9200	3500	6000	6233.33	2829.1	0.5
						ROUND 3 - Final consesus					
Sahithi	Tarun	Rakesh	Consolidated(Meghana)			Sahithi	Tarun	Rakesh	AVG	σ	σ²
This module will require 1400 SLOC and must enable users to quickly access their account settings.	This module will require 4200 SLOC and must support integration with third-party authentication services.	This module will require 4000 SLOC and must allow users to verify their identity via email or phone.	This module will require 3200 SLOC and must provide easy access to help and support resources.		Module 1 (User Account & Profile)	1400	4200	4000	3200	1505.5	0.5
This module will require 3300 SLOC and must allow users to easily upload media from various devices.	This module will require 2500 SLOC and must provide automatic compression for faster uploads.	This module will require 2100 SLOC and must enable users to receive notifications on upload status.	This module will require 2633.33 SLOC and must offer sharing options to multiple platforms.		Module 2 (Media Upload & Sharing)	3300	2500	2100	2633.33	620.2	0.2
This module will require 5700 SLOC and must allow users to send direct messages with read receipts.	This module will require 3000 SLOC and must support video calls and voice messages.	This module will require 3400 SLOC and must enable users to see when friends are online.	This module will require 4033.33 SLOC and must allow blocking and reporting of users.	SLOC	Module 3 (Chat/DM Functionality)	5700	3000	3400	4033.33	1252.2	0.3
This module will require 6500 SLOC and must allow users to browse and explore new content based on interests.	This module will require 3800 SLOC and must support dynamic content updates.	This module will require 1600 SLOC and must provide content categorization and tagging.	This module will require 3966.67 SLOC and must enable users to follow specific topics or creators.		Module 4 (Search & Browse)	6500	3800	1600	3966.67	2213.3	0.6
This module will require 9100 SLOC and must facilitate secure file sharing and downloading among users.	This module will require 3600 SLOC and must provide options for downloading multiple files simultaneously.	This module will require 5500 SLOC and must enable users to track their downloaded content and remaining storage.	This module will require 6066.67 SLOC and must ensure data integrity and security during file transfers.		Module 5 (P2P Download & Storage)	9100	3600	5500	6066.67	2667.9	0.4

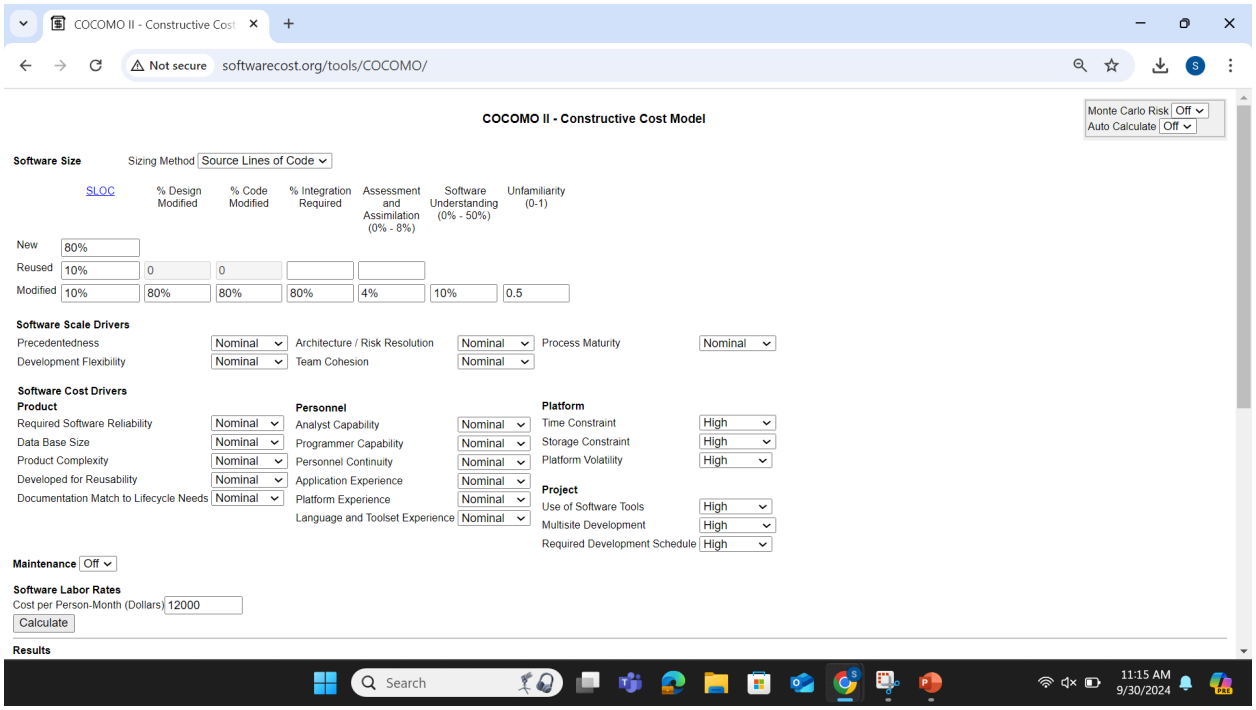
						ROUND 1 - Parameters			
Sahithi	Tarun	Rakesh	Consolidated(Meghana)			Sahithi	Tarun	Rakesh	Average Result
Due to the low precedentedness	Due to the nominal ratings which support	Due to the nominal ratings allowing for	Due to the balanced ratings leading to an	Scale Factors (SU)	Precedentedness	Low	Nominal	Nominal	Nominal
Due to the consistent level of flexibility across all parameters.	Due to the nominal flexibility that allows for some adjustments as required.	Due to the nominal ratings supporting flexibility in development approaches.	Due to an overall consistent approach, enhancing flexibility across the team.		Development Flexibility	Nominal	Nominal	Nominal	Nominal
Due to the balanced risk resolution reflected in architecture ratings.	Due to the nominal architecture ratings providing a basic level of risk management.	Due to low architecture ratings potentially increasing risk factors.	Due to average architecture ratings allowing for moderate risk resolution strategies.		Architecture / Risk Resolution	Nominal	Nominal	Low	Nominal
Due to the varying ratings in team cohesion ensuring collaboration.	Due to team cohesion being rated very high, enhancing collaboration and communication.	Due to the nominal cohesion ratings which provide a basic level of team interaction.	Due to high averages in team cohesion that supports effective collaboration.		Team Cohesion	Nominal	Extra High	Nominal	High
Due to the nominal ratings in process maturity ensuring a steady workflow.	Due to alignment in process maturity promoting efficiency across tasks.	Due to nominal process maturity ratings allowing for consistent workflow.	Due to high averages in process maturity contributing to effective project execution.		Process Maturity	Nominal	Nominal	Nominal	Nominal
Due to varying ratings in required	Due to the nominal ratings ensuring basic	Due to the nominal reliability ratings	Due to average reliability ratings		Required S/W Relaibility	Nominal	Nominal	Nominal	Nominal
Due to the balanced approach to database size, reducing storage concerns.	Due to nominal ratings that minimize issues related to database size.	Due to nominal ratings allowing manageable database sizes for development.	Due to consistent ratings ensuring database size does not hinder progress.		Database Size	Nominal	Nominal	Nominal	Nominal
Due to the very low complexity ratings facilitating a smoother development process.	Due to low complexity ratings that simplify the development process.	Due to the nominal ratings allowing manageable complexity levels.	Due to overall low ratings in complexity supporting easier management of the product.		Product Complexity	Very Low	Nominal	Nominal	Low
Due to nominal ratings indicating some focus on reusability in design.	Due to nominal ratings supporting basic reusability in components.	Due to nominal ratings ensuring some degree of reusability in the development process.	Due to a consistent focus on reusability across all parameters.		Developed for Reusability	Nominal	Nominal	Nominal	Nominal
Due to nominal ratings ensuring alignment with project needs.	Due to high ratings in documentation aligning with development needs for clarity.	Due to high ratings in documentation match enhancing guidance and understanding.	Due to an overall high rating in documentation facilitating clear project understanding.		Documentation Match to Needs	Nominal	Nominal	Very High	High
Due to nominal ratings ensuring basic analyst skills across the team.	Due to nominal ratings reflecting adequate analyst capability.	Due to nominal ratings indicating stable analyst capability across the project.	Due to overall consistent ratings providing sufficient analyst capabilities.		Analyst Capability	Nominal	Nominal	Nominal	Nominal

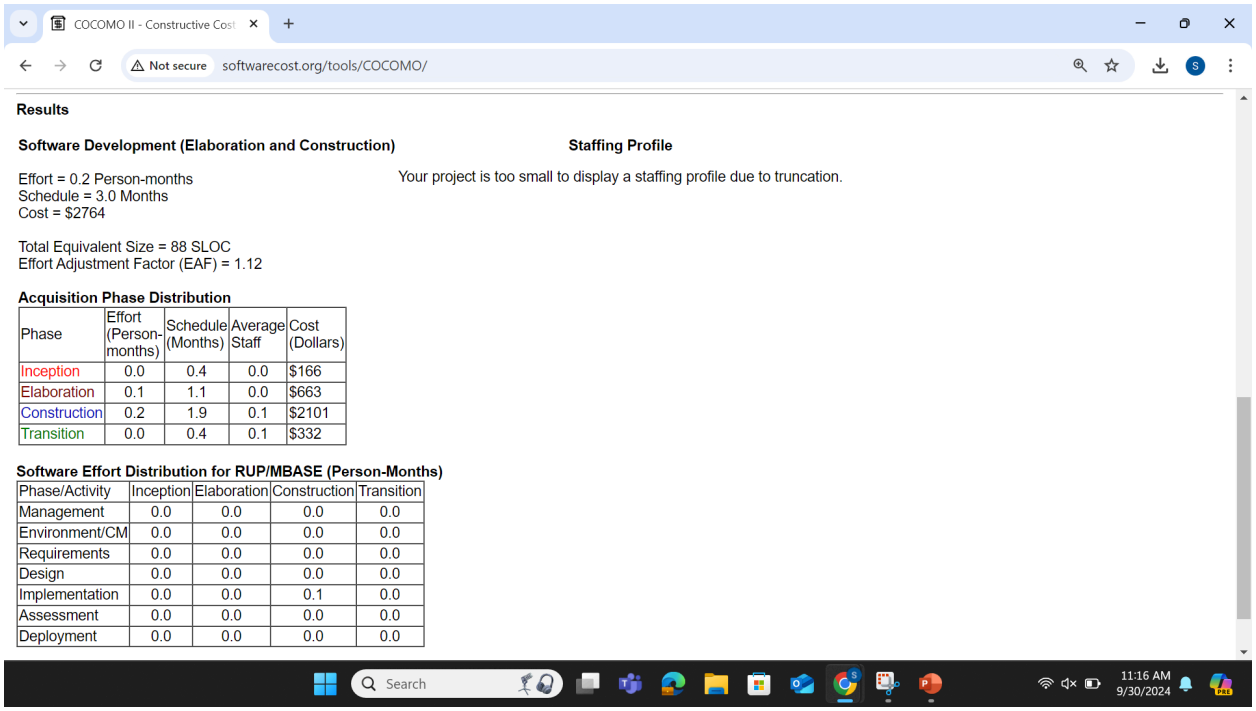
Due to nominal ratings reflecting basic application experience across the team.	Due to high ratings in application experience enhancing problem-solving.	Due to nominal ratings indicating basic application experience among team members.	Due to high averages in application experience supporting effective project handling.	Effort Multipliers (Wi)	Application Experience	Nominal	Nominal	Nominal	Nominal
Due to low ratings in platform experience affecting overall consistency.	Due to nominal ratings ensuring some adaptability in platform use.	Due to nominal ratings providing a stable platform experience.	Due to an average rating reflecting consistent platform usage across the team.		Platform Experience	Low	Nominal	Nominal	Nominal
Due to nominal ratings ensuring familiarity with programming languages and tools.	Due to nominal ratings allowing basic understanding of languages and toolsets.	Due to nominal ratings indicating stable language and toolset experience.	Due to overall consistency in language and toolset experience across team members.		Language and Toolset Experience	Nominal	Nominal	Nominal	Nominal
Due to consistent low ratings in time constraints minimizing stress on timelines.	Due to nominal ratings providing flexibility in managing time constraints.	Due to nominal ratings indicating manageable time constraints for the project.	Due to an overall consistent approach to time management reducing pressure on deadlines.		Time Constraint	Nominal	Nominal	Nominal	Nominal
Due to the balanced approach to storage constraints allowing for adequate data management.	Due to nominal ratings ensuring manageable storage needs.	Due to very high ratings indicating potential challenges with storage constraints.	Due to overall high averages highlighting the need for careful storage management.		Storage Constraint	Nominal	Nominal	Very High	High
Due to nominal ratings indicating stable platform conditions for development.	Due to nominal ratings reflecting consistent platform volatility.	Due to nominal ratings indicating moderate platform stability.	Due to an overall balanced approach to platform volatility enhancing consistency.		Platform Volatility	Nominal	Nominal	Nominal	Nominal
Due to consistent ratings ensuring familiarity with software tools across the team.	Due to nominal ratings indicating stable use of software tools for development.	Due to nominal ratings ensuring basic familiarity with tools among team members.	Due to an overall nominal rating reflecting familiarity with the software tools used.		Use of Software Tools	Nominal	Nominal	Nominal	Nominal
Due to nominal ratings indicating stable interactions across multi-site development.	Due to nominal ratings ensuring basic coordination across multiple sites.	Due to high ratings indicating potential challenges in multi-site coordination.	Due to overall consistent ratings providing a solid foundation for multi-site development efforts.		Multi-site Development	Nominal	Nominal	Nominal	Nominal
Due to consistent ratings in required development schedules that help set realistic timelines.	Due to nominal ratings ensuring manageable development timelines.	Due to nominal ratings indicating basic alignment in development schedules.	Due to an overall balanced evaluation across development schedules enhancing planning capabilities.		Required Devl. Schedule	Nominal	Nominal	Nominal	Nominal

						ROUND 2 - Parameters			
Bob	**Bill**	**Susan**	**Consolidated**			**Sahithi**	**Tarun**	**Rakesh**	**Average Result**
Due to the nominal ratings providing	Due to the low rating affecting overall	Due to nominal ratings allowing for some	Due to the overall nominal approach	Scale Factors (Wi)	Precedentedness	Nomi	Low	Nomi	Nominal
Due to consistent nominal ratings across all team members enhancing development flexibility.	Due to nominal ratings supporting a stable development process.	Due to nominal ratings ensuring consistent flexibility in the development process.	Due to an overall uniform approach facilitating balanced flexibility in development.		Development Flexibility	Nominal	Nominal	Nominal	Nominal
Due to nominal ratings indicating a basic level of risk management across the team.	Due to nominal ratings providing a consistent approach to architecture and risk resolution.	Due to nominal ratings allowing for manageable risk assessment.	Due to the average ratings indicating a moderate capacity for risk resolution.		Architecture / Risk Resolution	Nominal	Nominal	Nominal	Nominal
Due to nominal ratings reflecting a steady level of teamwork and collaboration.	Due to nominal ratings ensuring basic team cohesion among members.	Due to nominal ratings indicating stable interactions within the team.	Due to overall nominal ratings suggesting average cohesion among team members.		Team Cohesion	Nominal	Nominal	Nominal	Nominal
Due to nominal ratings indicating a stable level of process maturity across the team.	Due to nominal ratings providing consistency in processes.	Due to nominal ratings allowing for manageable workflow and process maturity.	Due to overall consistency in process maturity promoting an effective project flow.		Process Maturity	Nominal	Nominal	Nominal	Nominal
Due to nominal ratings reflecting a	Due to nominal ratings ensuring adequate	Due to nominal ratings providing stable	Due to the overall average reflecting	Effort Multipliers (Wi)	Required S/W Relaibility	Nomi	Nomi	Nomi	Nominal
Due to nominal ratings indicating manageable database sizes that do not hinder development.	Due to nominal ratings supporting sufficient database management.	Due to nominal ratings allowing for balanced database size considerations.	Due to overall consistent ratings ensuring no significant concerns regarding database size.		Database Size	Nominal	Nominal	Nominal	Nominal
Due to the very high complexity rating potentially complicating development processes.	Due to nominal ratings that support a stable yet simpler approach to complexity management.	Due to nominal ratings reflecting a manageable level of complexity in product development.	Due to an overall high average indicating a need for careful management of product complexity.		Product Complexity	Very High	Nominal	Nominal	High
Due to nominal ratings indicating some focus on reusability in design.	Due to nominal ratings supporting basic reusability of components.	Due to nominal ratings ensuring some degree of reusability in the development process.	Due to a consistent focus on reusability across all team members.		Developed for Reusability	Nominal	Nominal	Nominal	Nominal
Due to nominal ratings ensuring alignment with project documentation needs.	Due to nominal ratings providing a basic level of documentation match.	Due to nominal ratings ensuring adequate documentation aligns with project requirements.	Due to overall consistent ratings indicating basic alignment in documentation with project needs.		Documentation Match to Needs	Nominal	Nominal	Nominal	Nominal

Due to nominal ratings providing a basic level of analytical skills across the team.	Due to very high ratings in analyst capability enhancing project understanding and execution.	Due to nominal ratings indicating stable analyst capability among team members.	Due to an overall high average reflecting strong analytical capabilities in the team.	Effort Multipliers (Wi)	Analyst Capability	Nominal	Very High	Nominal	High
Due to nominal ratings indicating consistent programming skills among team members.	Due to nominal ratings providing basic programming capabilities.	Due to nominal ratings indicating stable programming capabilities among team members.	Due to consistent ratings across the team enhancing overall programming capabilities.		Programmer Capability	Nominal	Nominal	Nominal	Nominal
Due to nominal ratings ensuring basic team stability across personnel changes.	Due to nominal ratings reflecting a consistent level of personnel continuity.	Due to nominal ratings indicating a steady level of personnel continuity.	Due to overall consistent ratings ensuring team stability during transitions.		Personnel Continuity	Nominal	Nominal	Nominal	Nominal
Due to nominal ratings reflecting basic application experience across the team.	Due to very high ratings in application experience enhancing problem-solving abilities.	Due to nominal ratings indicating adequate application experience.	Due to high averages in application experience supporting effective project handling.		Application Experience	Nominal	Very High	Nominal	High
Due to very low ratings affecting overall consistency and adaptability in platform use.	Due to nominal ratings providing a basic level of platform experience.	Due to nominal ratings indicating stable platform experience among team members.	Due to overall low ratings highlighting potential challenges with platform consistency.		Platform Experience	Very Low	Nominal	Nominal	Low
Due to low ratings indicating limited familiarity with certain languages and toolsets.	Due to nominal ratings ensuring some understanding of programming languages and tools.	Due to high ratings indicating strong familiarity with languages and tools.	Due to overall average ratings reflecting varying levels of language and toolset experience.		Language and Toolset Experience	Low	Nominal	High	Nominal
Due to nominal ratings ensuring manageable time constraints for development.	Due to nominal ratings providing flexibility in managing time constraints.	Due to nominal ratings indicating basic alignment in time management.	Due to an overall consistent approach to time management reducing pressure on deadlines.		Time Constraint	Nominal	Nominal	Nominal	Nominal
Due to nominal ratings indicating manageable storage constraints across the project.	Due to nominal ratings supporting adequate storage management strategies.	Due to nominal ratings ensuring consistent handling of storage needs.	Due to overall consistency in storage management reflecting balanced approaches across the team.		Storage Constraint	Nominal	Nominal	Nominal	Nominal
Due to nominal ratings indicating stable platform conditions for development.	Due to nominal ratings reflecting consistent platform stability.	Due to low ratings indicating potential challenges in platform volatility management.	Due to an overall nominal rating indicating an average approach to managing platform volatility.		Platform Volatility	Nominal	Nominal	Low	Nominal
Due to nominal ratings ensuring familiarity with software tools across the team.	Due to nominal ratings indicating stable use of software tools for development.	Due to nominal ratings ensuring basic familiarity with tools among team members.	Due to an overall nominal rating reflecting familiarity with the software tools used.		Use of Software Tools	Nominal	Nominal	Nominal	Nominal
Due to high ratings indicating potential challenges in coordinating multi-site efforts.	Due to nominal ratings ensuring basic coordination across multiple sites.	Due to nominal ratings indicating stable interactions within the team.	Due to overall consistent ratings highlighting a need for improved coordination in multi-site efforts.		Multi-site Development	High	Nominal	Nominal	Nominal
Due to consistent ratings in required development schedules helping set realistic timelines.	Due to nominal ratings providing adequate planning for development timelines.	Due to nominal ratings indicating basic alignment in development schedules.	Due to an overall balanced evaluation across development schedules enhancing planning capabilities.		Required Devl. Schedule	Nominal	Nominal	Nominal	Nominal

2.3.3 COCOMO -II second estimate





2.3.4 Round 2 Basis of Estimate (BOE) for SNAPSTER Project

1. Labor Estimates

Based on team experience and adjusted hourly estimates due to project complexity here are the updated estimates:

- Development Team: $700,000 (increase due to added complexity)
- Testing Team: $520,000 (slight increase to accommodate additional testing phases)
- Quality Assurance Team: $420,000 (adjusted for better QA procedures)
- DevOps/Deployment Team: $430,000 (increase due to additional deployment requirements)
- Project Management Team: $250,000 (increase to manage better)

Total Labor Cost: $2,350,000

2. Other Costs

Adjustments based on project needs and expenses:

- Travel Costs: $6,000 (increased travel for stakeholder engagement)
- Software Tools: $12,000 (additional tools for development)
- Testing Tools: $10,000 (increased infrastructure for comprehensive testing)
- Contingency Reserve (10% of total labor cost):
  10% of $2,350,000 = $235,000

Summary of Total Estimated Effort and Cost

Total Hours

The total hours for each phase have been increased to better accommodate the complexity of tasks:

Phase	Estimated Hours
Requirements Gathering	80
Design	220
Software Development	900
Financial Tasks	80
Testing Tasks	350

Management Tasks	150
Total Labor Hours	**1,780**

Labor Estimate Breakdown

WBS	Task Description	Calculation	Period	Hours
1.1	Stakeholder Interviews	4 members * 20 hrs	Inception	80
1.1	Document Requirements	2 members * 50 hrs	Elaboration	100
2.1	UI Design	2 designers * 80 hrs	Elaboration	160
2.1	Architectural Design	1 architect * 100 hrs	Construction	100
3.1	Frontend Development	2 developers * 220 hrs	Construction	440
3.1	Backend Development	2 developers * 220 hrs	Construction	440
3.1	Integration	2 developers * 100 hrs	Transition	200
4.1	Budget Planning	1 analyst * 50 hrs	Inception	50
4.1	Financial Reporting	1 analyst * 30 hrs	Elaboration	30
5.1	Unit Testing	2 testers * 80 hrs	Transition	160
5.1	Integration Testing	2 testers * 50 hrs	Transition	100
5.1	User Acceptance Testing	2 testers * 50 hrs	Transition	100
6.1	Project Management	1 manager * 100 hrs	Transition	100

6.1	Team Meetings	3 members * 100 hrs	All phases	300
Total Hours				**1,780**

Material Estimate

WBS	Purchase Description	Calculation	Cost
1.1	Survey Software	1 license * $500/year	$500
1.1	Documentation Tools	1 license * $600/year	$600
2.1	UI/UX Design Software	3 licenses * $1200/year	$3,600
2.1	Prototyping Tool	1 license * $600/year	$600
3.1	Integrated Development Environment	6 licenses * $500/year	$3,000
3.1	Version Control System	1 group license * $600/year	$600
3.1	Backend Development Frameworks	Open source	$0
3.1	Decentralized Storage	Open source	$0
4.1	Financial Planning Software	1 license * $500/year	$500
5.1	Testing Frameworks	Open source	$0
5.1	Testing Platform	1 license * $1000/year	$1,000
6.1	Project Management Software	1 group license * $500/year	$500
6.1	Communication Tools	1 group license * $600/year	$600

Total Material Cost			**$9,100**

Trips Estimate

Updated travel costs based on project needs:

WBS	Trip Description	Calculation	Cost
1.1	Meeting with Stakeholders	Total: $6,000	$6,000
2.1	Collaborative Workshops at Different Studios	Total: $3,000	$3,000
3.1	Pair Programming Sessions at Development Hubs	Total: $7,000	$7,000
4.1	Meeting with Financial Advisors	Total: $2,000	$2,000
5.1	Quality Assurance Testing at User Locations	Total: $5,000	$5,000
6.1	Team Meetings with Project Managers	Total: $3,000	$3,000
Total Estimate for Trips			**$26,000**

Total Estimate Summary

Category	Cost
Total Labor Cost	$2,350,000
Total Material Cost	$9,100
Total Trip Estimate	$26,000

Contingency Reserve	$235,000
Total Estimated Cost	**$2,620,100**

Rationale/Estimating Methodology

The adjustments reflect a deeper understanding of the project complexities, Decentralized architecture, user features, and the extensive backend development needed for user-generated content. Assessing each task's cost and hours based on the breakdown of responsibilities and expertise, a bottom up approach is used.

2.4 Software Process Model

The best-fit software process model for the SNAPSTER TM project is the Agile Software Development Model. First, the iterative development aspect of Agile gives room for flexibility on the project. SNAPSTER TM is an application featuring, among other things, complex functionalities in decentralized file-sharing, user accounts, ranking, and customizable avatars. This is where an iterative approach lets us break down the development of such a system into smaller, manageable sprints. Each sprint develops a functional version of the product that gives us flexibility for adapting changing requirements and feedback as the project is in progress. This has been of particular importance to us in this dynamic social media platform where at any moment new user expectations or trends emerge.

This means that Agile is particularly focused on continuous user feedback. Because SNAPSTER TM will use active user interaction via the features of friending, chatting, and sharing contents intensively, user experience is key. This is also supported by the Agile model, which highly calls for collaboration. Considering how complex SNAPSTER TM is and the number of teams, this is a must. In Agile, teams regularly communicate; for example, the frontend communicates with the backend, QA, and infrastructure. Above all, daily stand-ups and sprint reviews ensure that all teams align with, understand each other's updates, and are informed of what's new. This would be the level of collaboration needed, for instance, in integrating complex features related to decentralized file-sharing, which requires close cooperation between frontend and backend teams.

Other advantages of Agile include rapid prototyping and testing. SNAPSTER TM includes innovative features such as story mode, multiple-source parallel downloads, and custom filters-all requiring early prototyping to ensure that the features will work as expected. Agile enables us to create prototypes early in the process and call for feedback so we can refine these features before we fully implement them. Continuous testing gives us stability and the level of performance which was met with during this app's development.

In addition to these pleasing aspects, Agile reduces risks. Since Agile provides working software in frequent cadences, there is no possibility of having big issues at the end of the development cycle. For example, we can make incremental steps to handle issues such as decentralized file sharing or security vulnerabilities. That way, we avoid costly reworks and delays and stay on schedule with minimal risks.

Agile is finally apt for SNAPSTER TM as it puts much emphasis on the continuous integration and delivery that will be needed when frequent updates with new features prove necessary. In this manner, once the new functionality is developed, it can easily be integrated into the app; this, in turn, will assure smooth and regular releases. This minimizes the periods the app will be down while keeping it updated with new features.

This ends with the Agile software process model being appropriate for the SNAPSTER TM project due to the fact that it allows flexibility, collaboration, and rapid iteration in developing a feature-rich, user-focused social media app. We can adopt Agile to ensure SNAPSTER TM remains change-enabling and responsive to user feedback and also assures efficiency in the delivery of high-quality products to market.

2.5 Integrated Master Schedule

These will be developed using a Gantt chart in the IMS to map out key project milestones, including:

- Design Review: Complete design review on architecture for blockchain-based decentralized content sharing.
- Developmental Milestones Core modules include the P2P file-sharing system, a ranking system, and custom filters.
- Performance, Security, Feature Functionality Testing Phase: This would, in turn, require QA cycles that are performance, security, and feature functionality-based.
- Release Phases: Beta release, full public release, feature updates (e.g., extension of filter repository).

Project Initiation

- **Deliverable**: Project Charter
- **Duration**: 1 week
- **Start Date**: Week 1
- **End Date**: Week 2

Requirements Gathering

- **Deliverable**: Requirements Specification Document
- **Duration**: 2 weeks
- **Start Date**: Week 2
- **End Date**: Week 4

Design Phase

- **Deliverable**: System Architecture and Design Documents
- **Duration**: 3 weeks
- **Start Date**: Week 4
- **End Date**: Week 7
- **Milestone**: Design Review at Week 6

Development Phase

- **Deliverable**: Developed Software Components
- **Duration**: 6 weeks
- **Start Date**: Week 7
- **End Date**: Week 13
- **Milestone**: Alpha Release at Week 12

Testing Phase

- **Deliverable**: Test Plans and Results
- **Duration**: 4 weeks
- **Start Date**: Week 13
- **End Date**: Week 17
- **Milestone**: Beta Release at Week 16

User Acceptance Testing (UAT)

- **Deliverable**: UAT Feedback Report
- **Duration**: 2 weeks
- **Start Date**: Week 17
- **End Date**: Week 19

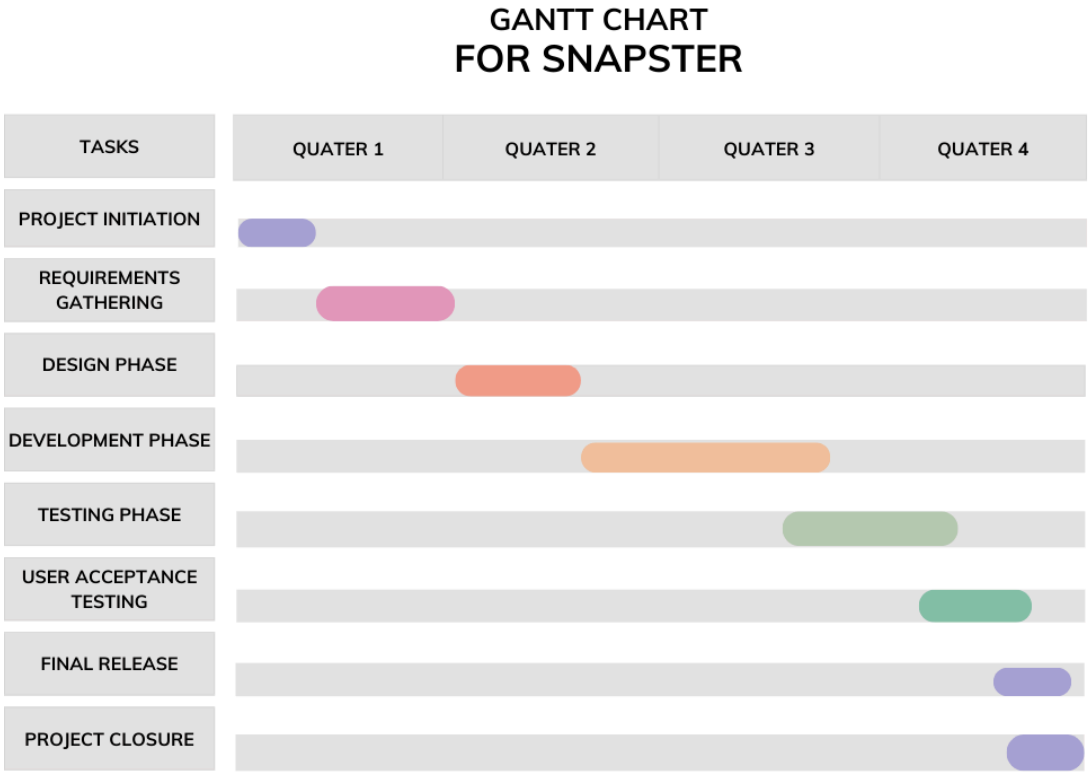
Final Release

- **Deliverable**: Final Product
- **Duration**: 1 week
- **Start Date**: Week 19
- **End Date**: Week 20
- **Milestone**: Final Review at Week 20

Project Closure

- **Deliverable**: Project Closure Document
- **Duration**: 1 week
- **Start Date**: Week 20
- **End Date**: Week 21

The Gantt chart will provide a clear view of the project timeline, major milestones, and deliverables.



3 Problem Definition

This section contains the problem statement and the root causes for the problem using the fishbone diagram. It is then followed by the system boundary diagram, the users and stakeholders of the software and finally ending with constraints of different aspects of the project.

3.1 Problem Statement

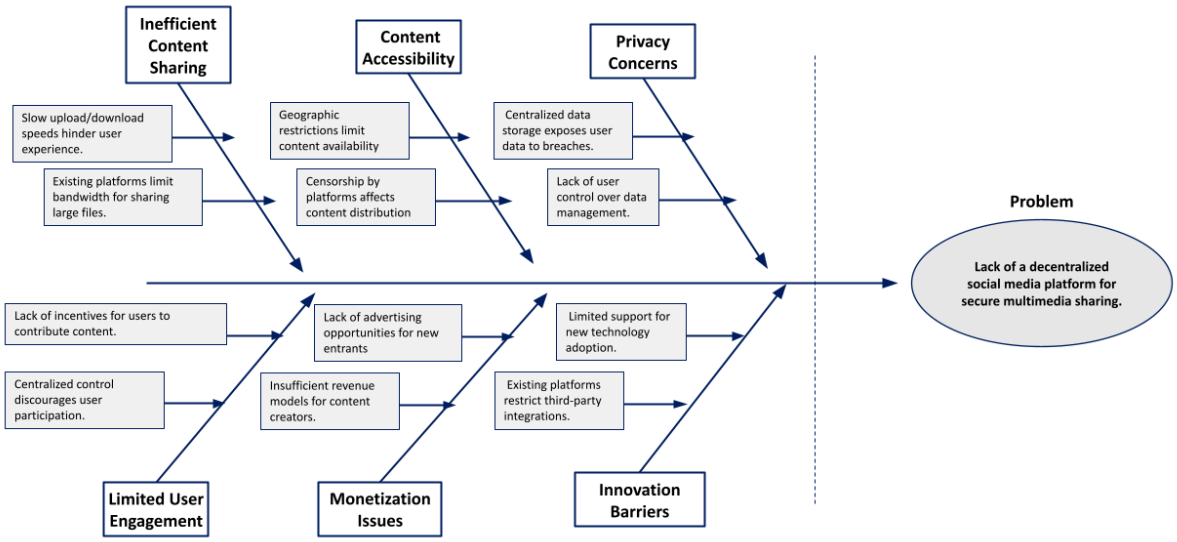
Element	Description
The problem of	absence of a decentralized platform that facilitates the secure exchange or access of multimedia leading to more concern over data exposure than is necessary as well as limited reach for content.
Affects	Users: Data privacy and security are the major issues. Content Creators: Restricting control to them also means, censorship hampers reaching the intended audience.

	Developers: Solutions, which enable users, are hard to come up with achieving such autonomy is rare.
the result of which	Users: Privacy leads to a level of trust that's hard to breach, limited freedom to move about in the platform is quite irritating, and low engagement leads to disinterest in participation. Content Creators: Limited audience reach, leads to poor growth, proscription on sharing leads to self-monitoring and risks loss resources required for future endeavors. Developers: Technical constraints and stifled urge for creativity because of the centralized web environment. Business Activity: Market positioning may be jeopardized by the inability to stand out, ineffective audience engagement hampers revenue that would be derived from advertising and market contains risk of evaluating due to lack of privacy.
benefits of	the social network will improve the privacy level of users, allow more unrestricted access to content and make the distribution process more effective. The main advantages are, more authority of users over their personal information, quicker content distribution and an active user base.

3.2 Root Causes

The root problems identified which explain the SNAPSTER project have a reflection on the challenges of traditional social networking sites issues. The overarching drawback is the presence of privacy issues due to the centralization of data storage, which increases the risk of breaches and diminishes the trust of all users. Moreover, the latitude in terms of content distribution is also dependent on the geographic locations and the platform's will, limiting how far any given creator can go. Combined with non-effective distribution of the content and therefore lack of monetisation opportunities, it creates a case for a new system that is able to stimulate creativity and participation while safeguarding both security and privacy.

Fishbone Diagram



3.3 Stakeholders & Users

The SNAPSTER project stakeholders are made up of different persons and organizations interested in the evolution and success of the platform. Such stakeholders are the final customers of the platform, who interact with it by uploading and consuming multimedia content. Content creators are also important stakeholders since they enhance the richness of the platform by creating content. Beta testers help in testing the platform and give valuable suggestions or comments, which help to correct any drawbacks on the platform before its official launch.

Other important stakeholders are obviously the investors who are the source of funding required to develop and sustain the project. Advertising partners seek to exploit the platform in the marketing of their goods and services which are factors that affect the revenue generation strategies of the platform. Also, thirst parties, such as adjusting authorities, are stakeholders as they have to abide by the laws and policies related to data privacy and other laws protecting users' rights and encouraging content exchange within a safe space.

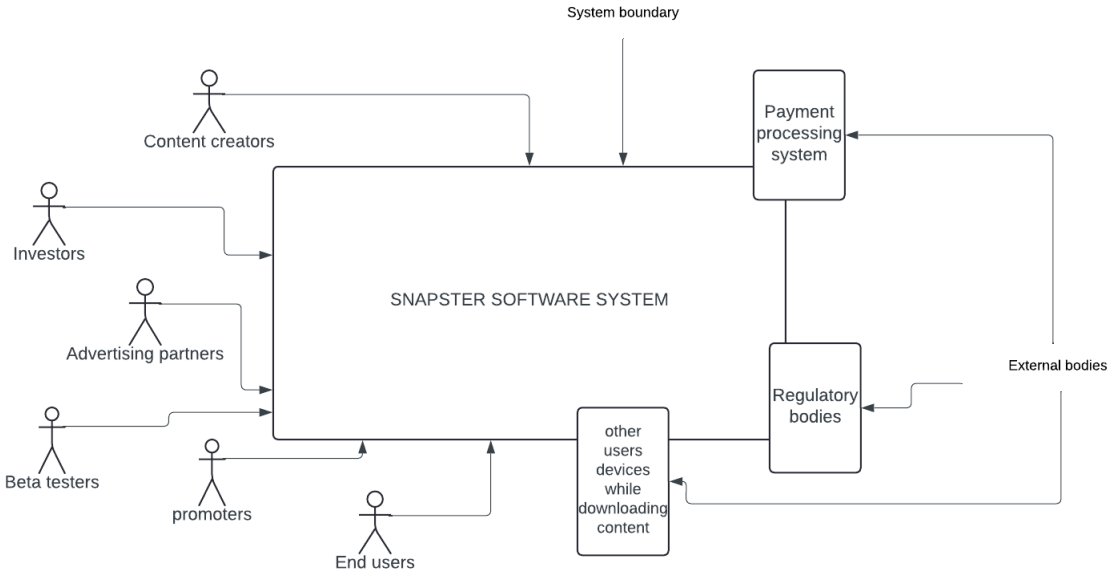
Users can be considered the core part of the SNAPSTER system since they are mainly end users who include themselves in the system and use it to upload, seek and view multimedia content on a daily basis. A better understanding of the target market eases making important strategic decisions such as developing new products, marketing them and even expanding the current covers. This user segment is a core resource within the Snapster platform as they are responsible for creating the content that keeps the users engaged, boosts the platform usage and also aids in community-building on the platform. This group consists of artists, influencers and developers that play a key role in bringing and keeping a wider audience.

Furthermore, beta testers are very important for this phase of development as they help with particular details and issues, which can have a negative impact on user experience. On the other hand, Community advocates marketing SNAPSTER, providing further benefits by integrating users and enforcing their sense of community throughout the platform, which plays a very powerful and vital role in the successes of the platform. Therefore, all of these users have specific purposes in the platform thereby making it easier for them to be able to provide the services that they do in the platform.

Users	Other stakeholders
End users	Investors
Content Creators	Advertising Partners
Beta Testers	Regulatory Authorities
Promoters	Project Management Team
	Technical Support Staff
	Marketing Team
	Feedback Analysts
	Software development team

3.4 System Boundary Diagram

The System Boundary Diagram for the SNAPSTER project illustrates what can be termed the area of the SNAPSTER platform given the interactions with systems and users who are part of the project and other systems which are not included in this project. This diagram is helpful in defining the limits of the project, that is to ask what constitutes the system, and what lies outside these boundaries.



3.5 Constraints

The SNAPSTER project is constrained by several factors that will influence the course of its development and implementation. It should be possible to interface with and integrate into the existing systems and comply with legal and regulatory requirements mainly related to user privacy. Optimistic or uncertain feature scope and pricing strategies will also be undermined by Resource availability policies. Besides that, the project team may encounter organizational barriers to interdepartmental collaboration and technical issues with regard to platform integration, while also having to consider the incremental adoption of new technology whenever possible and appropriate.

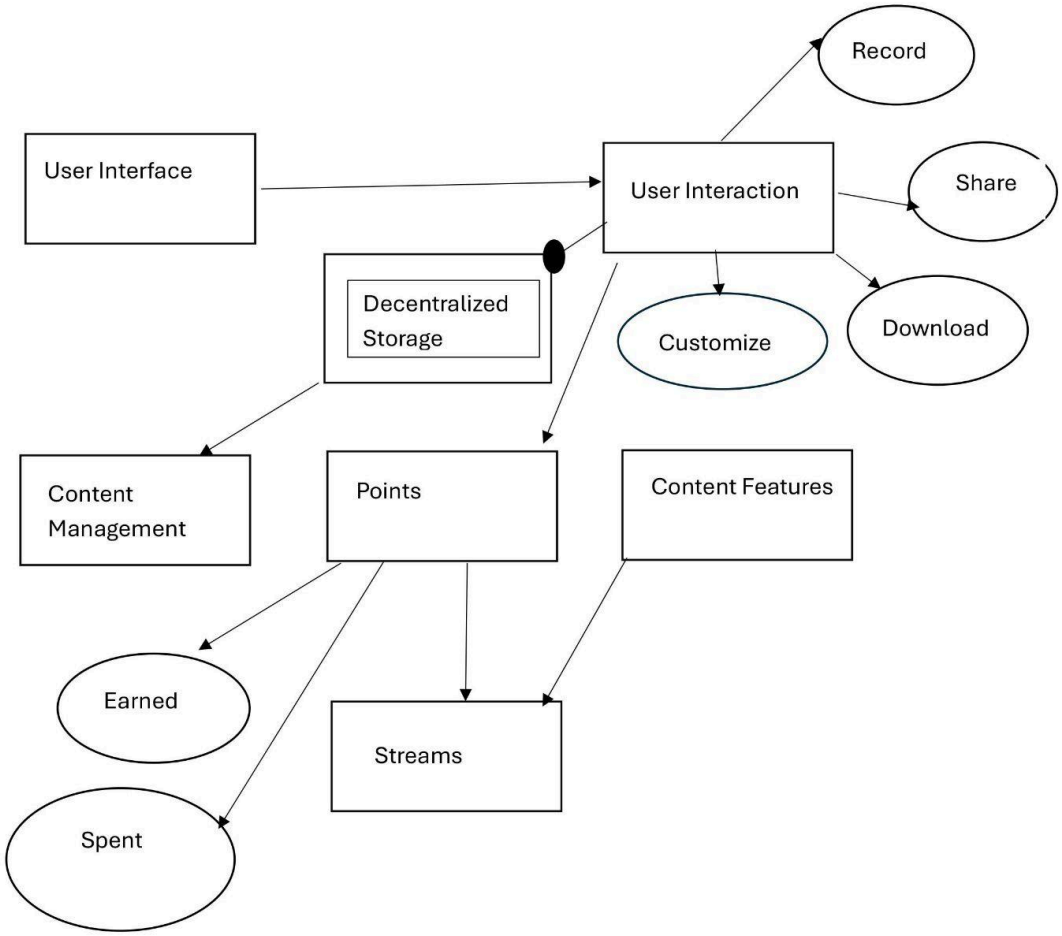
Source	Considerations
System	■ SNAPSTER is to be built on existing systems to take cave on reasonable resources and technology. ■ It is required to provide support for existing systems for better end-user and system interaction and data diversity. ■ The approach applied must also support application development for a number of operating systems including iOS and Android, deployment options both in the cloud and on the premises.
Environmental	■ Data protection laws which are accorded to the users (i.e. European gent General Data Protection Regulation, California Consumer

	Privacy Act, etc) should be considered throughout the development of the project. ■ Certain provisions of copyright regulations and rules concerning content sharing ought to be observed to prevent legal issues. ■ Security specifications imply assuring great levels of security of the application, for example, with the help of encrypting the data and secure authentication of the application user.
Schedule and resources	■ Yes the planning aspect is very well done with strict time bounds and deliverable oriented project timeline. ■ The project is mainly confined within the in-house resources . This may cause some constraints on how developed the project will be, and how fast the development will be done. ■ Inadequate internal labour supply usually allows the use of outside labour, especially for specific tasks or where time is of the essence. ■ Although it is possible to create more resources by either partnering with another company or hiring contractual employees to support the project if necessary.
Economic	■ Currently, the project has a budget that casts certain restrictions upon the aspects that can be enhanced and incorporated into the project without going over budget. ■ Every cost related to goods sold also need to be well controlled, and pricing techniques for subscription and in-app need to be strategically established to make good profits but at the same time fit into the market. ■ The use of third-party tools and libraries may lead to licensing problems as they have their own terms which must be followed and which may involve more cost.
Political	■ Apart from the culture and values that are potentially brought by foreign investors, there is still some weakness about which the stakeholders have different priorities which could affect the decision making and allocation of resources.

	■ Organizational conflicts may arise in the course of strongly related functions being diluted or if there is a lack of synergy between departments like marketing, development and customer support. Communication between departments such as marketing, development, and customer support.
Technical	■ The project is somewhat restricted in the choice of technologies due to the need for compatibility with existing systems. ■ It is constrained to work with existing platforms and technologies to ensure seamless integration and maintain operational continuity. ■ There are no prohibitions against new technologies, but their adoption will depend on resource availability and team familiarity. ■ The project is free to use purchased software packages, provided they align with the budget and meet project requirements.

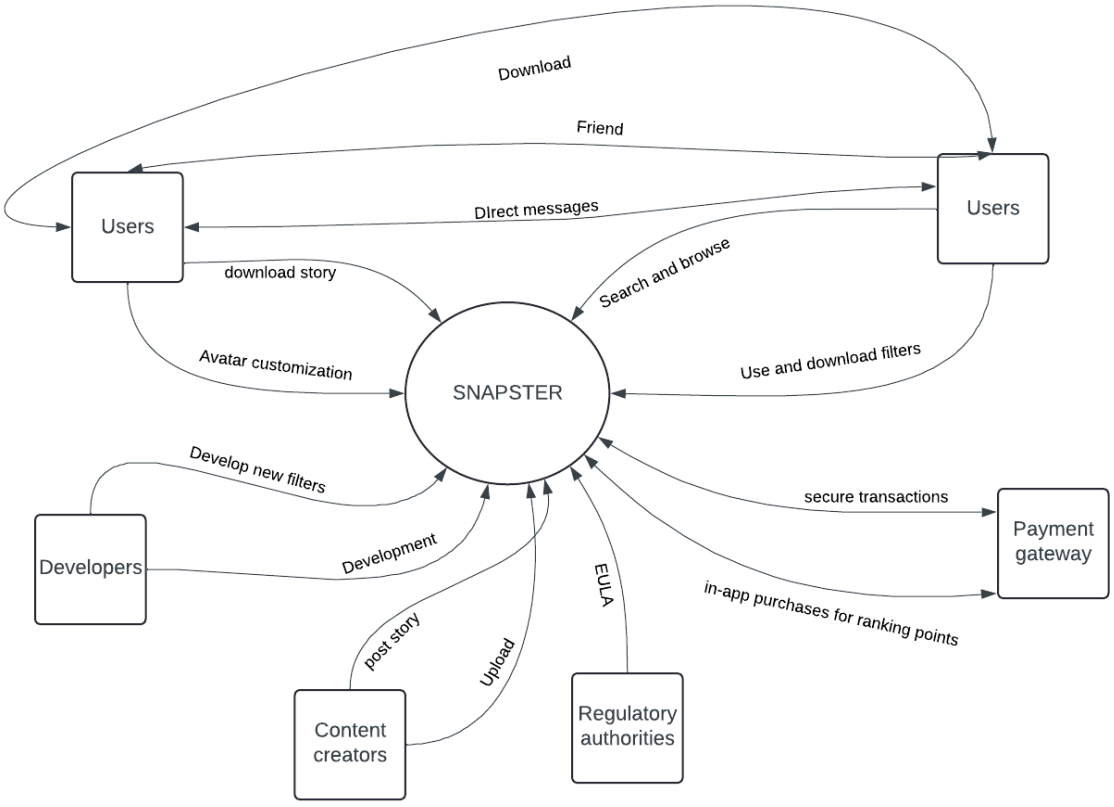
4 Frame Diagram

The frame diagram for the SNAPSTER project provides a visual representation of the system's architecture. The drawing provides a lightweight, pictorial overview of how the major components of the SNAPSTER TM system will interact. It shows the key features: the user interface, shared content, data storage, and externally linked communications. This diagram allows us to better see how different parts are connected with each other, which enables us in determining any potential conflicts or areas which need re-configuration while building our system. It is one of those useful tools to make certain everything fits and works together as it should.



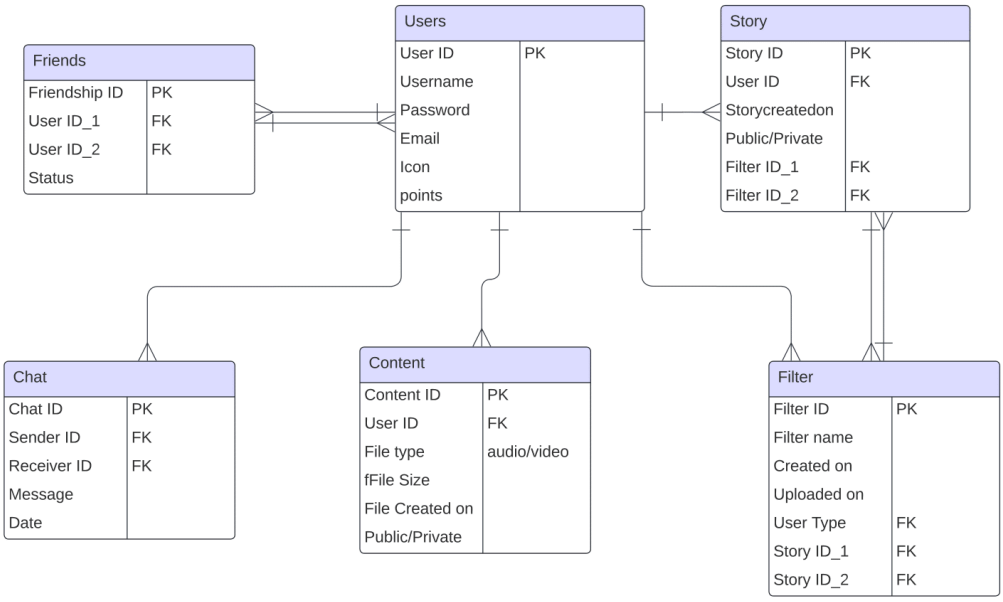
5 Context Diagram

The context diagram for the SNAPSTER distributed multimedia sharing application shows how the system interacts with its environment. The most of them are involved in activities, which can be mentioned as follows: uploading/downloading multimedia and personal images; friending other users and ranking; customizing avatars. Researchers state that content creators upload their contributions while payment systems process secure in-app transactions. Another source of income which comes from ad services is the opportunity to insert advertising into story review mode. Governments stations oversee privacy and content laws while third-party organizations contribute to the growing open source nature of the platform optimizing the experience of the user and adding creative features. This diagram give a broad view of each of the external actors in their relation to SNAPSTER's core system of content sharing, monetization and compliance.



6 Entity Relationship Diagram

The SNAPSTER system ERD shows all the principal entities and the associations between them. It offers an understanding of how information is structured at Collective Coder and how different entities like users, media files and the interactions between these entities (like likes, comments and shares are related). The following is very important as it assists in drawing the structure of the database as well as assisting in the management and retrieval of data. The relationships between entities in the ERD further the both the design and development phases so as to provide clear understanding of the system architecture.



7 Use Cases

Each use case involves a user approach to the specific Snapster's features, it is able to define the purpose of the user and the system. For instance the use cases like "Download Stories," "Earn/Buy Points," "Change Avatar" offer details such as, who, what, before what, what in between and after what and what is expected out of it.

7.1 Use Case Diagrams

Interactions of Primary Users:

User: The user has done the registration in the Snapster application. In addition, people can upload, view, share, and post content from any of their profiles. It shows that they can even earn points, which are earned through dispensing and receiving posts. They could possibly seek out certain rather familiar faces and just become pals virtually. They could perhaps choose an avatar, view adverts and also interact live with other users through the chat and messaging options.

Remote User: They are also users that are comparable to average users and possess similar abilities. In the global view, we ought to observe this remote user from a user's standpoint. The remote user posts stories or uploads and authorizes other users' material. This content can be downloaded by the users through many devices.

Malicious User: A malicious user is a threat actor in the making who aims to capitalize on one or more of the Snapster attributes. They are concentrated on achieving unauthorized entry into personal information and managing download figures. He is abstract as a security threat, a privacy violation, or a data issue.

Sponsor: Advertisers are outside businesses that place a brand or other advertisement within the application. They enable advertisements being used to be aired while in between two different stories. They develop an ad display code, an option that includes a user's interest and his browsing history.

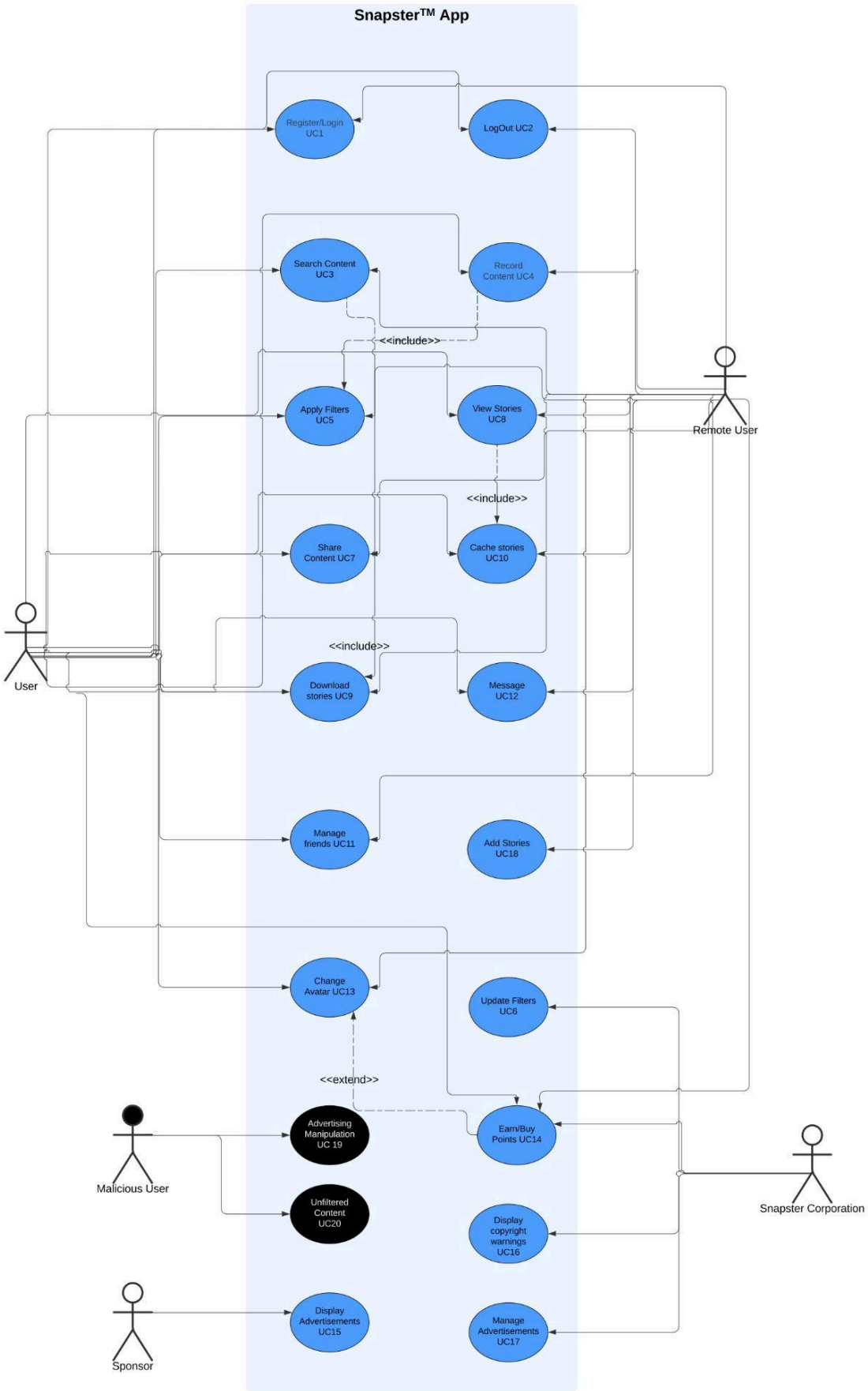
Snapster Corporation: This forms the core of our Snapster app as it will be an important point of contact. It regulates the platform, owns the filter change process, handles user identification and the points system. It also is responsible for content rules, user agreements and partnerships.

Included and Extended Use Cases :

Included: Some use cases, such as applying filters (UC5) and caching stories (UC10), are parts of the more comprehensive actions such as viewing/sharing stories, improving the usability and structures of the content.

Extended: The third and fourth use cases labeled "Advertising Manipulation" and "Unintended Content" show potential for app misuse that come with further security measures imposed on it and monitoring from the side of the app administrators.

The following diagram provides a first glance view of the Snapster™ app emphasizing how different actors interrelate and the normal and misuse case app structure. This also sows the seeds for functional requirements and it outlines the organization of a system with respect to feature development and protection.


Snapster™ App
Register/Login UC1
LogOut UC2
Search Content UC3
Record Content UC4
<<include>>
Apply Filters UC5
View Stories UC8
Remote User
<<include>>
Share Content UC7
Cache stories UC10
User
<<include>>
Download stories UC9
Message UC12
Manage friends UC11
Add Stories UC18
Change Avatar UC13
Update Filters UC6
<<extend>>
Advertising Manipulation UC 19
Earn/Buy Points UC14
Malicious User
Unfiltered Content UC20
Display copyright warnings UC16
Snapster Corporation
Display Advertisements UC15
Manage Advertisements UC17
Sponsor

7. 2 Use Cases

UC ID	UC Name
UC1	Register/Login
UC2	Logout
UC3	Search Content
UC4	Record Content
UC5	Apply Filters
UC6	Update Filters
UC7	Share Content
UC8	View Stories
UC9	Download Stories
UC10	Cache Stories
UC11	Manage Friends
UC12	Message
UC13	Change Avatar
UC14	Earn/Buy Points
UC15	Display Advertisements
UC16	Display Copyright Warnings
UC17	Manage Advertisements
UC18	Add Stories
UC19	Advertising Manipulation
UC20	Unfiltered Content

UC1 Register/Login: In this use case, the two roles defined are: the user and the remote user. The register function is also available for those who do not yet have a Snapster account and helps them to create one. The registration function suggests to the user to create a new username and password and they may enter their email or phone number to restore an account. In case the user decides to join, the will be redirected to the welcome page which shows the user's history of the points and an option of creating the avatar. The security feature of the portal is that the existing users log in their accounts using their user id and password. It can be further improved by configuring multi factor authentication.

UC2 Logout: In this case, the user and the remote user are the two actors in this scenario. To clear the screen, the user types clear and to log out they choose the logout option. From the current session, Snapster safely logs the user off while saving any incomplete work in progress with great ease. After the session ends the user is redirected to the login page. The user account is only vulnerable until the next login session of the user or the next session the user logs in.

UC3 Search Content: The two actors involved in this hectic scenario are the user and the remote user. This is true because the user only scrolls through the program looking for whatever they might feel is interesting to them. The program helps to direct the user to whitelisted information if the user would like to search for specific sort of material for instance music, travel, sports etc. Snapster takes a series of searches and returns them as a list which the user can then choose to view or to send to many others in different locations through a peer-to-peer connection.

UC4 Record Content: The roles in this scenario negate the clients' name but recognizes the user and the remote user as the actors in the system. It lets the user record sound and video through Snapster's built-in camera and microphone options. The captured films looked sharper and less grainy when additional filters that are periodically updated from a central library were applied.Conveying can also be made through direct messages about the content.

UC5 Apply Filters: The role players in this instance are the user, and the remote user. After the user has decided which content to upload as a story the user may now search for filters to add to the story. The filters uploaded enable the user to switch over the audio and the video content material. This filter may be shown in real time. Finally among the functionalities of record content is the use case for apply filter.

UC6 Update Filters: As the actor of this case, it is the Snapster Corporation. In this backend of the program it is possible for the users to get the most recent camera features from the repository. The program then has a pop up message saying that to use the newest camera options, the user has to update the program. Most of these filters can be obtained on the internet and used off-line.

UC7 Share Content: The role players for this scene includes the actor on stage and the remote user. This means that the user selects things to either capture or retrieve. In addition, the user selects where this item should be shared – publicly or with people who are on the friends list only. When the material is posted to the business's Page and then whitelisted users will see the post in their feed. People are rewarded with points and can also spend material to change avatars.

UC8 View Stories: The first target in the scene is the actor and the second target is the remote user. This feature enables the user to view stories uploaded by other users in the public tab. The user can download it using a direct connection of two peers. It is also able to play advertisements in between episodes. Conversely, the downloaded content can be stored and updated based on history at some interval.

UC9 Download Stories: All the available content shared by the friends that are public users is scrutinized by the users when deciding which of the narratives to download. Our program searches for the news sender logged by the user and examines if that sender is on the whitelisted or not. In case the supplied content is appropriate, Snapster will send out a download request. This is because, using Snapster, one can download with the aid of several devices provided that one device has a hither to better download quality than the other. So after reaching some downloaded amount of narrative the program starts playing the narrative back and can be interrupted by ads in the middle.

UC10 Cache stories: The characters in this scene are the actor, and the remote user. This function is to facilitate the Download to temporarily save, after downloading the videos for the purpose of uninterrupted playing. The advantage of using caching is that the user can watch the video even when connection with the peer is lost. To control storage, cache always has the option of cache files being rewritten when playing is done with older content.

UC11 Manage Friends: In this scene, there are two actors, the actor, and the remote user. This tool enables a user to expand his/her friend list through addition or deletion of friends or categorization of friends. The actions that one user can make are to make a request, undo a request, and categorize friends. When connected, the friend list of the user is changed. Users can sort friends into lists which can either be private or friends of friends.

UC12 Message: The role players in this scene include the actor and the remote user. They can engage in real–time discussion through chat of this feature. Some people will be communicating through text messages, voice and even video. A notice is provided to the customer when there is a new message in the system.

UC13 Change Avatar: The participants are the local user and the remote user. The user chooses the change avatar from the other option available on the program. Each of these avatars is offered in a free version and a premium version and customers can cash in the points they have to get the premium version. If the avatar is free, they may advance without spending any points. If it will be in the television program wherein the avatar will be used, and it is one of the premium selections, then the system may check the point level of the user. If the user runs out of points the program will recommend him to buy more points. The avatars can also be edited by a user if he has enough points to make a purchase, the points will be subtracted from the total points of the user.

UC14 Earn/Buy points: The players here are the user, the remote user and the snapster corporation. The program has numerous activities that are presented which could be done in a bid to get the point. The user can select an option and perform the task and mostly it will be completed. The application checks whether or not the action has been completed in question. There are credits awarded to BS account in the parameters of user account balance. When a new point pack is to be bought and a payment is to be made, the user has to make a click on it. Finally, through going further in the program, we get to a payment system to complete the purchase. This – to ensure that our payment has been authorized by

the payment system, is what the application does. Afterward, the snapster then adjusts the new points to the user's balance of the account of the snapster.

UC15 Display Advertisements: The actor here being the sponsor. Snapster insists on the Snapster Corporation to secure the adverts that are stored in their database and provided for by the sponsor. The snapster server then uses the user preference and any other relevant criteria to make selections of adverts to pull. Snapster does these by incorporating them into the graphical User Interface as we will see. The user himself can decide whether he wants to see this advertisement or not and if he wants to interact with them on which topic.

UC16 Display copyright warnings: The actor in this case is the Snapster Corporation. These warnings emerge when a user wants to post an article. The warning informs the user of her/his responsibility to ensure compliance with such copyrights and links the user to the app's EULA. This method not only helps people learn about issues regarding copyright, but it also gathers people who support free or open sharing while following legal rules.

UC17 Manage Advertisements: In this case, the actor is the Snapster Corporation. They pop up when the replay of the story is on. Snapster cooperates with a number of other marketers to make sure that they are in touch with the users' needs. This also tracks the interaction with adverts by users.

UC18 Add stories: In this case, the actor is the remote user. The user can either create or add the tales in real time through filming or using an existing video. After the content is obtained, a number of filters are available to the user. After editing the material, the user can share it with several friends. The story may be added publicly or in private.

UC19 Advertising Manipulation: This is a misuse case. The actor in the context of this threat can be regarded as the malicious user. It frequently achieves specific advertising and realigns them by altering impressions with regard to specific segments. The algorithm at Snapster works with this data which is in its modified form. This results in a wrong placement of advertisements in the consumers' market.

UC20 Unfiltered Content: This is a misuse case. The actor in this case is the malicious user. Insignificant: The malevolent actor tries to release or disseminate such content without pre-filtration. This may include use of rogue language, lewd images or representations or any material likely to offend any person. These people are still able to bypass the detecting mechanism through exchange of new words, use of picture overlay amongst others. The material was successfully released in the application.

7.3 Normal Scenarios

Use Case : Download stories

ID	UC9
Actors	a. User: They would like to view a story. b. Remote User: They look at stories posted and also post their own stories which can be downloaded. c. Snapster corporation: They manage accounts and the content to be downloaded.
Pre-conditions	The user has logged in to their Snapster account. Users may browse the tales of their friends in their accounts. The distant user has completed the content to be shared. The application has an internet connection and can reach a large number of individuals, allowing for ideal download speeds.
Flow of Events	1. The users check through all of the accessible content posted by their friends who are public users before deciding which narrative to download. 2. Our program looks for the user who uploaded the news and determines if it is whitelisted or not. If the content supplied is suitable, Snapster will launch a download request. 3. Snapster allows us to download simultaneously using several devices if one has a faster download speed than the other. 4. After the narrative has been downloaded to a certain amount, the program begins playing it and may include adverts in the midst. 5. The downloaded content might be cached throughout the playing process. This cache may be overwritten when fresh material is downloaded. 6. After watching a narrative, the viewer may be rewarded with points that may be used to customize their avatars.
Post-conditions	The user has successfully downloaded the tales. The downloaded files are stored in the cache until they are refreshed. The user's viewing history is periodically updated in accordance with Snapster's policy.

Use Case : Change Avatar

ID	UC13
Actors	User: They use the earned points in the application to modify their avatar. Remote User: They can have the same functionality as a normal user and can use their points to change the avatar.
Pre-conditions	The user has logged in to their Snapster account. They have access to the alter avatar option and have the necessary points.
Flow of Events	1. The user selects the change avatar option in the program. 2. The program presents a selection of avatars, some free and others premium, which may be redeemed using the points they have. 3. If the avatar is free, they may advance without expending any points. 4. If the avatar is a premium option, the system may verify the user's point balance. 5. If the user doesn't have enough points, the program prompts them to buy more. 6. If the user has enough points to purchase an avatar, they can edit it and the points will be taken from their points total.
Post-conditions	The user will be able to view the freshly changed avatar in their profile. When you change your avatar, the points balance changes.

Use Case : Earn/buy points

ID	UC14
Actors	User: The primary snapster user who would like to buy extra points. Payment System: This is a new external system that manages the transactions made. Snapster Corporation: It manages point balances and processes the earned points after the payment is processed.
Pre-conditions	The user has logged in to their Snapster account. The user’s account must have a good history with no restrictions or bans. The payment system must be perfectly integrated with snapster.
Flow of Events	1. The user navigates to the Snapster app's earn points area. 2. The program presents a variety of tasks that may be accomplished in order to earn points. 3. The user can choose an option and complete the task. 4. The application validates whether or not the action has been performed. 5. Points are credited to the user's account balance. 6. Scenario to buy points

	7. The user clicks on a point pack to purchase it and make a payment. 8. The program takes us to a payment system to finish the transaction. 9. The application validates our payment through the payment system. 10. Following a successful purchase, the snapster updates the user's account balance with the new points. 11. The user receives a message that the updated points have been added to their account.
Post-conditions	The user's point balance is updated depending on the most recent purchase. Users may use their points to access any feature inside the program. The purchase history is stored in the program for future reference.

Use Case : Display advertisements

ID	UC15
Actors	Sponsor: The person who finances to display the advertisements in the application. Snapster Corporation: The advertisements to be displayed are stored in a database.
Pre-conditions	The sponsor has a database full of advertisements ready to be presented on the application. The user's browser history was gathered to enable targeted ad distribution.
Flow of Events	1. The user continues to go to other faces inside the Snapchat account. 2. Snapster requests that the Snapster Corporation obtain the adverts held in its database, which have been supplied by the sponsor. 3. The snapster server retrieves appropriate adverts based on user preferences and other pertinent factors. 4. Snapster shows these advertising by integrating them into the UI. 5. The user may choose whether to see, skip, or interact with the advertisement based on their interests.
Post-conditions	The sponsor can monitor how users interact with adverts. Apart from engaging with or watching adverts, the user can continue to use their program. The comments on the adverts can be provided to the Snapster Corporation, which will transmit it to the advertiser.

7.4 Abnormal Scenarios

Use Case : Earn/Buy Points

ID	UC14
Actors	User: The primary Snapster user who would like to buy extra points. Payment System: This is a new external system that manages the transactions made. Snapster Corporation: It manages point balances and processes the earned points after the payment is processed.
Pre-conditions	The user has logged in to their Snapster account. The user's account must have a good history with no restrictions or bans. The payment system must be perfectly integrated with Snapster.
Flow of Events	1. The user navigates to the Snapster app's earn points area. 2. The program presents a variety of tasks that may be accomplished in order to earn points. 3. The user can choose an option and complete the task. 4. The application validates whether or not the action has been performed. 5. Points are credited to the user's account balance. 6. The user clicks on a point pack to purchase it and make a payment. 7. The program takes us to a payment system to finish the transaction. 8. The application validates our payment through the payment system. 9. Following a successful purchase, the Snapster updates the user's account balance with the new points. 10. The user receives a message that the updated points have been added to their account.

Post-conditions	The user's point balance is updated depending on the most recent purchase. Users may use their points to access any feature inside the program. The purchase history is stored in the program for future reference.
Extensions / Alternate Flows:	
3a: Invalid credit card:	3a1 The machine displays invalid credit card. 3a2 Return to step 2.
Post-conditions	System is ready for credit card processing again.

Use Case: Unable to Download Stories

ID	UC9
Actors	a. User: They would like to view a story. b. Remote User: They look at stories posted and also post their own stories which can be downloaded. c. Snapster corporation: They manage accounts and the content to be downloaded.
Abnormal Flow of Events	1. The user browses and decides to download a story. 2. The Snapster review system looks for reviews on the story. 3. The download has been disabled. 4. This is due to the story being recently banned or blacklisted by Snapster's review system.
Post-conditions	The download will not proceed, and the user will receive a warning stating that this cannot be downloaded. The download history is saved, and the user is prompted to view further stories to download.

Misuse case scenarios(Extra credit):

Use Case : Misuse Scenario for Advertising Manipulation

ID	UC19
Actors	Malicious user who injects malicious advertisements into the normal user content.
Pre-conditions	The malicious individual created a false Snapster account and signed in. This malevolent individual will be able to see the adverts shown.

	Snapster's advertising system is active, displaying adverts depending on user interactions.
Flow of Events	1. The hostile user gains access to and abuses the user's data. 2. It often targets certain advertising and misaligns them by modifying impressions for specific audiences. 3. Snapster's algorithm processes this modified data. 4. This leads to an incorrect distribution of advertisements among the consumers.
Post-conditions	Snapster's advertisement algorithm is warped, causing the system to focus on illicit material. Regular users may get unrelated adverts, which diminishes their interest in the application. The application's security system may exhibit strange behavior.

Use Case : Misuse Scenario for Unfiltered Content

ID	UC20
Actors	Malicious user who posts inappropriate content from their accounts.
Pre-conditions	The malicious individual created a false Snapster account and signed in. The malicious user may upload or distribute content from their accounts. The Snapster's detecting algorithm does not instantly identify the material.
Flow of Events	1. The malicious user attempts to publish or distribute unfiltered material. 2. This might include foul language, unsuitable imagery, or unfiltered content. 3. They can get around the detecting mechanism by employing changed language, graphic overlays, and other techniques. 4. The material was successfully shared in the application.
Post-conditions	Unfiltered content is removed from the platform. The offending user's account may be temporarily suspended or have their user privileges restricted. The users are notified that this content has been removed.

8 Activity Diagrams

The situation to purchase points is depicted in our activity diagram. The user is first presented with a variety of point alternatives via the display points package. The consumer chooses a bundle that appeals to them. The user is prompted by the system to submit their payment information. The user inputs their payment details. A decision node in the system verifies the information entered. We verify the purchase and provide the user with a confirmation message if the information is accurate. We ask the user to reenter the information if any of the details are incorrect. To make sure the process waits for all previous actions (such confirmation or error display) to finish before proceeding, the pathways converge after validation. We leave without completing a transaction if the user decides to quit or cancel at any time.

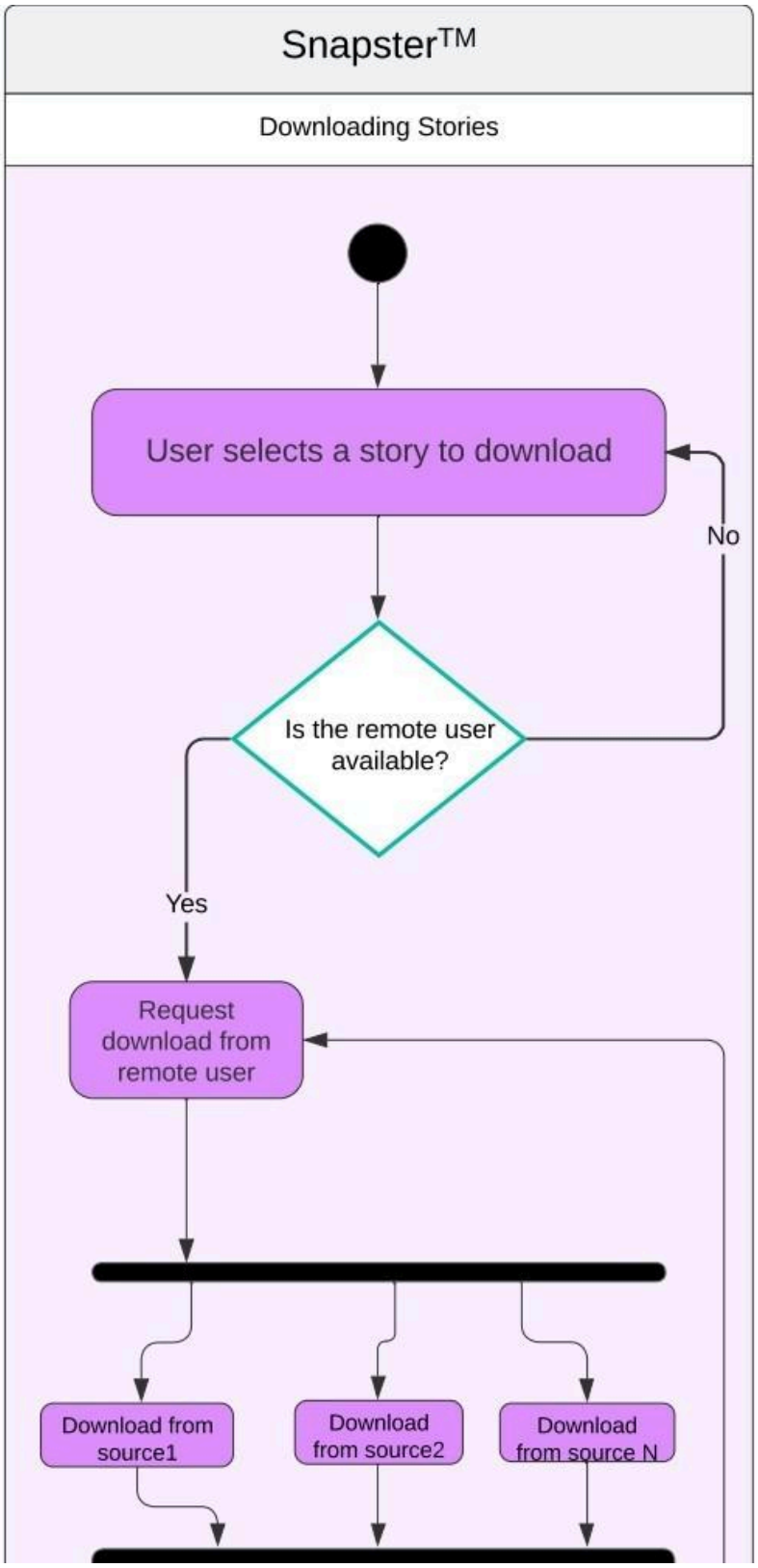
In activity diagrams, the normal scenarios describe a typical state of an affair, in which every action is followed by a proper outcome that is free of complications. For example, in Snapster™, a typical scenario of story download might entail selection of a story, probe for remote user availability, initiating downloads from various sources, caching the content, and playing the downloads with ads. This I ensured would follow a predetermined sequence to ensure first time run through and its successful completion.

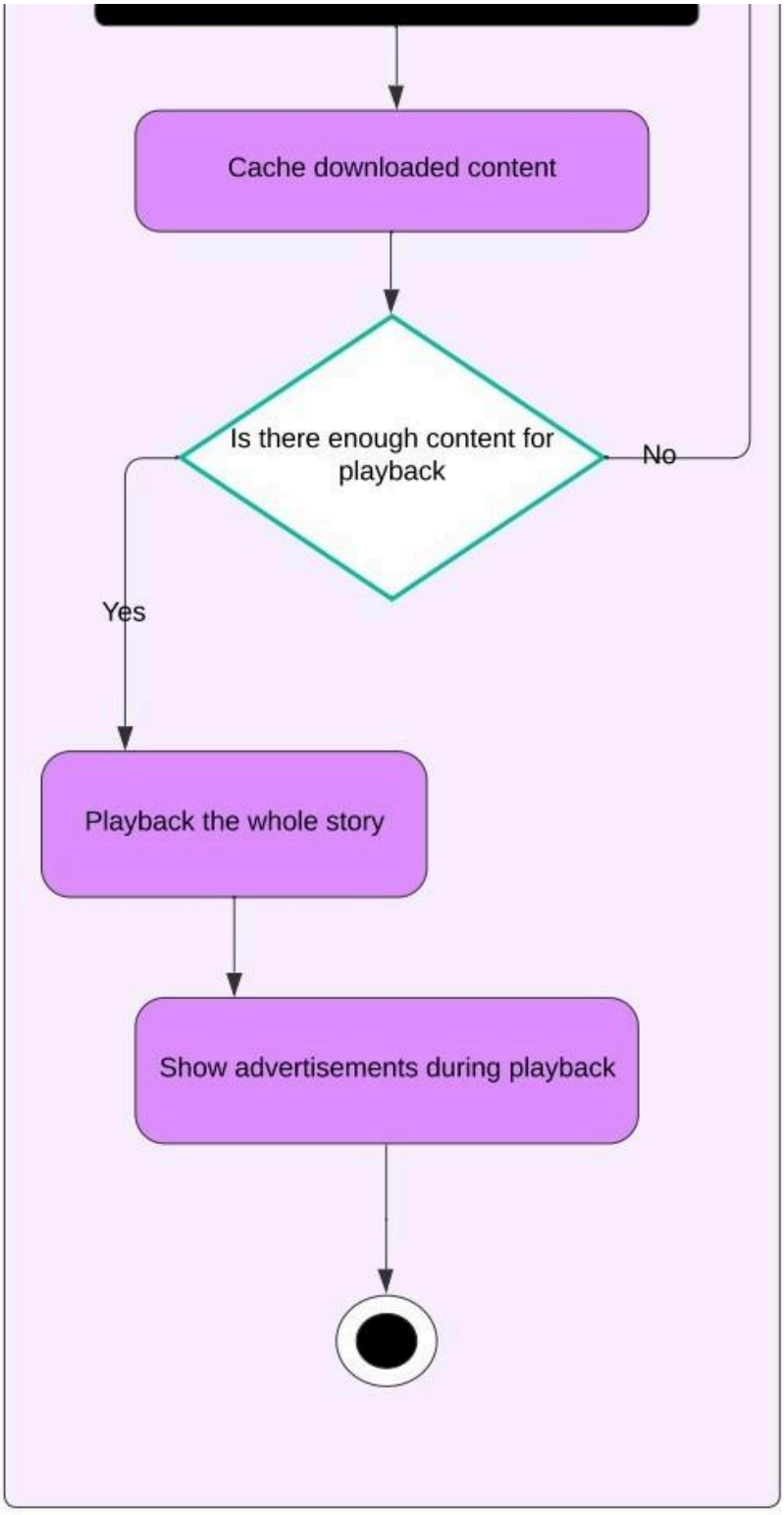
Anomalous sequences, on the other hand, capture other paths, usually initiated by some form of failure or other abnormal events. These scenarios are indeed very essential for the increase of the application design's reliability as they … For instance in Snapster™, if a user chooses a story but the remote user is not available, the diagram would follow an unusual path where the system could show an error message or recommend what other action to take. Likewise during a purchase when payment validation fails the system can display an error message to enter payment details again or end the purchase activity.
Due to the specification of activity diagrams, possible normal and abnormal situations in regard to Snapster™ are described, and thus, reliability is considered, and possible disruptions make the user experience better by handling errors.

8. 1 Normal Scenario

This activity diagram presents the steps of download and viewing stories in the Snapster™ application. It begins with a user choosing a story to download, the system first checks for availability of the remote user, a source. If available, the app sends a request for the story download to multiple sources to guarantee reliability and possibly faster downloading from parallel sources. It should be noted once the download is achieved, the content itself is buffered for straight playback. Before playback is to start, the application checks whether sufficient content has been downloaded to facilitate smooth playback of the video. Playback also incorporates ad insertion, thereby improving advertisement monetization during user engagement. As this design showed that the tutorial could be delivered efficiently, and the ads could be placed that brought revenue and not hinder the user experience.

Normal Scenario activity diagram for use case of downloading stories:




Cache downloaded content
Is there enough content for playback
No
Yes
Playback the whole story
Show advertisements during playback

8.2 Abnormal Scenario

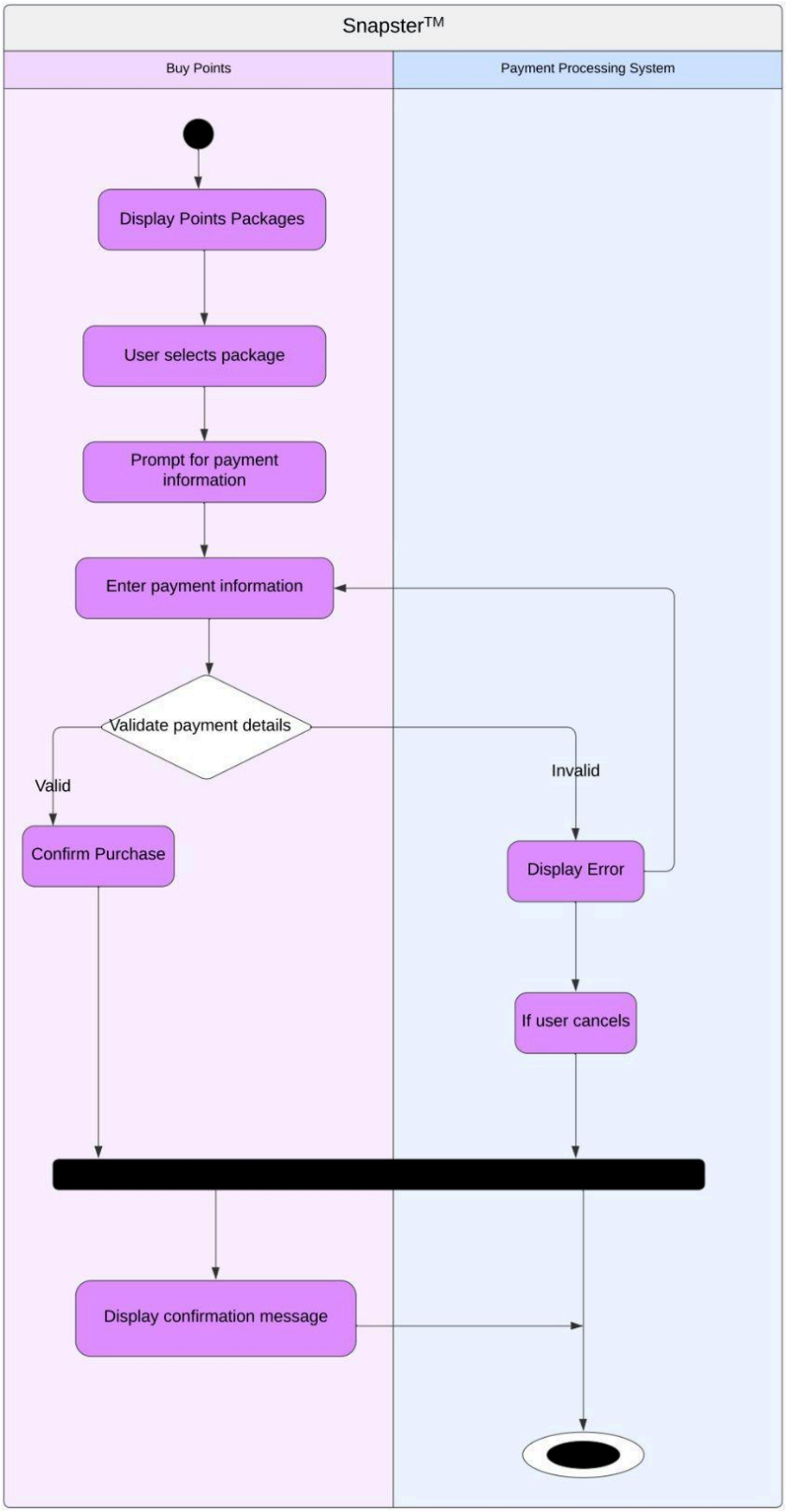
Abnormal Scenario activity diagram for Use Case - Earn/Buy Points : This activity diagram illustrates the process of purchasing points within Snapster™, which are likely used for in-app features or customizations. Users begin by viewing available points packages, selecting one, and then entering their payment information. The system validates the payment details by interfacing with the payment processing system, ensuring the information is correct before confirming the purchase. If valid, the transaction proceeds, and the user receives a confirmation. If not, an error displays, and users can either correct the input or cancel the process. This flow effectively balances user engagement with secure payment handling, prioritizing both ease of use and transaction security.

1. **Invalid Payment Information**: Anytime a user introduces his or her payment details, the system leads to a validation check whereby it goes through a payment processing system. Should the card details (credit card number, credit card expiry date or credit card CVV) filled by the client be incorrect or incomplete they are easily noticed by the system. After an error message is generated, the user is given a message that he or she needs to analyze and input correctly. They can either type in their details once again or, if they have changed their mind they can choose to abort the transaction and go back to the basic menu points.

2. **Payment Authorization Failure:** Sometimes, payment might be declined even when entering correct payment details hence, low balance or an expired status on the card or restrictions from the user's bank.When this happens, the system informs the user that the identified account failed authorization and describes the problem to the user in as much detail as possible minus putting the user at risk. Customers are encouraged to employ another payment type or get in touch with their bank if necessary.

3. **Network or Processing Errors:** Finally during the transaction process, network problems or system failure at the payment processor end can interfere with the purchase. In such cases, the system gets a hint of the interruption and does not allow the transaction to actualize to minimize confusion to the user and an instance of allowing another similar transaction to go through. It shows some error message that indicates to the user an unsavory temporary condition. The system automatically gives the choice of repeating the transaction or otherwise canceling and trying again. This makes users not to be in a 'limbo' as to what may be happening with regards to the transaction.

4.**Session Timeout:** To ensure security, a user may spend a long time making the purchase, and the session may end if it takes, especially for targeted data such as payment information. This is against the condition if the user is logged in and abound the app, it should protect from access by other users. At the same time, the system presents a timeout warning and informs the user that they have to restart purchasing from the first step. The above precaution serves to enhance security of the transactions.


Snapster™
Buy Points
Payment Processing System
Display Points Packages
User selects package
Prompt for payment information
Enter payment information
Validate payment details
Valid
Invalid
Confirm Purchase
Display Error
If user cancels
Display confirmation message

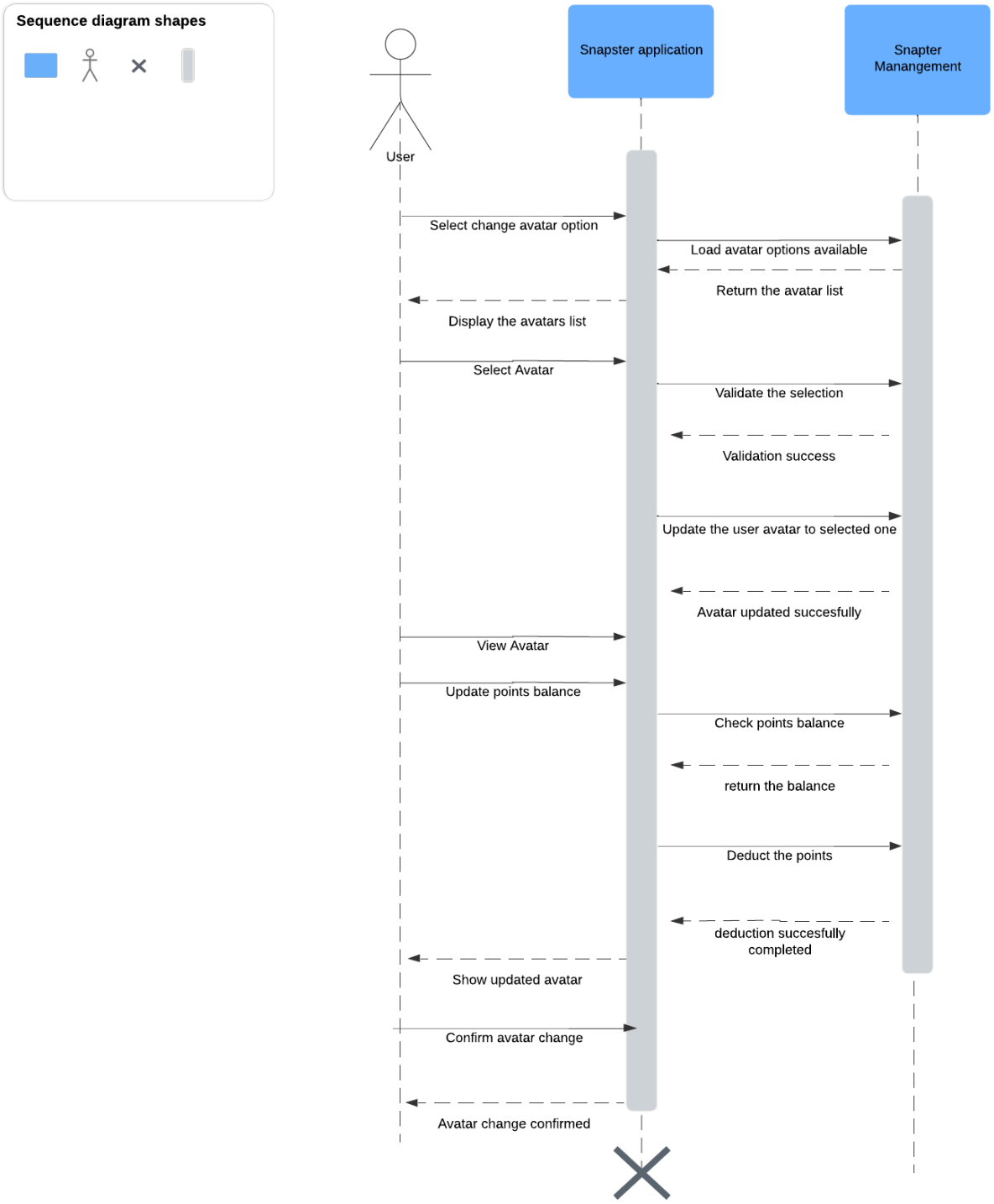
9 Sequence Diagrams

The important constituent of UML is a sequence diagram, which narrates how the various actors and systems interact with each other on account of time. They present a sequence of messages among components that are involved in a specific use case, showing both normal and abnormal situations. This sequence diagram depicts how the user will interact with the Snapster system concerning the use case Change Avatar, updating a new selected avatar, verifying selections and update of database and balance management of the points. The scenario of Unable to Download Stories describes how a set of interactions will take place when one tries to download a story to which access is restricted. It describes how the system is going to process the request by checking reviews and statuses prior to informing that it is unable to download it. These diagrams let functional requirements be illustrated and, in turn, improve system behavior understanding by mapping the relationships and workflows involving users, the system, and whatever external entities exist.

9.1 Normal Scenario

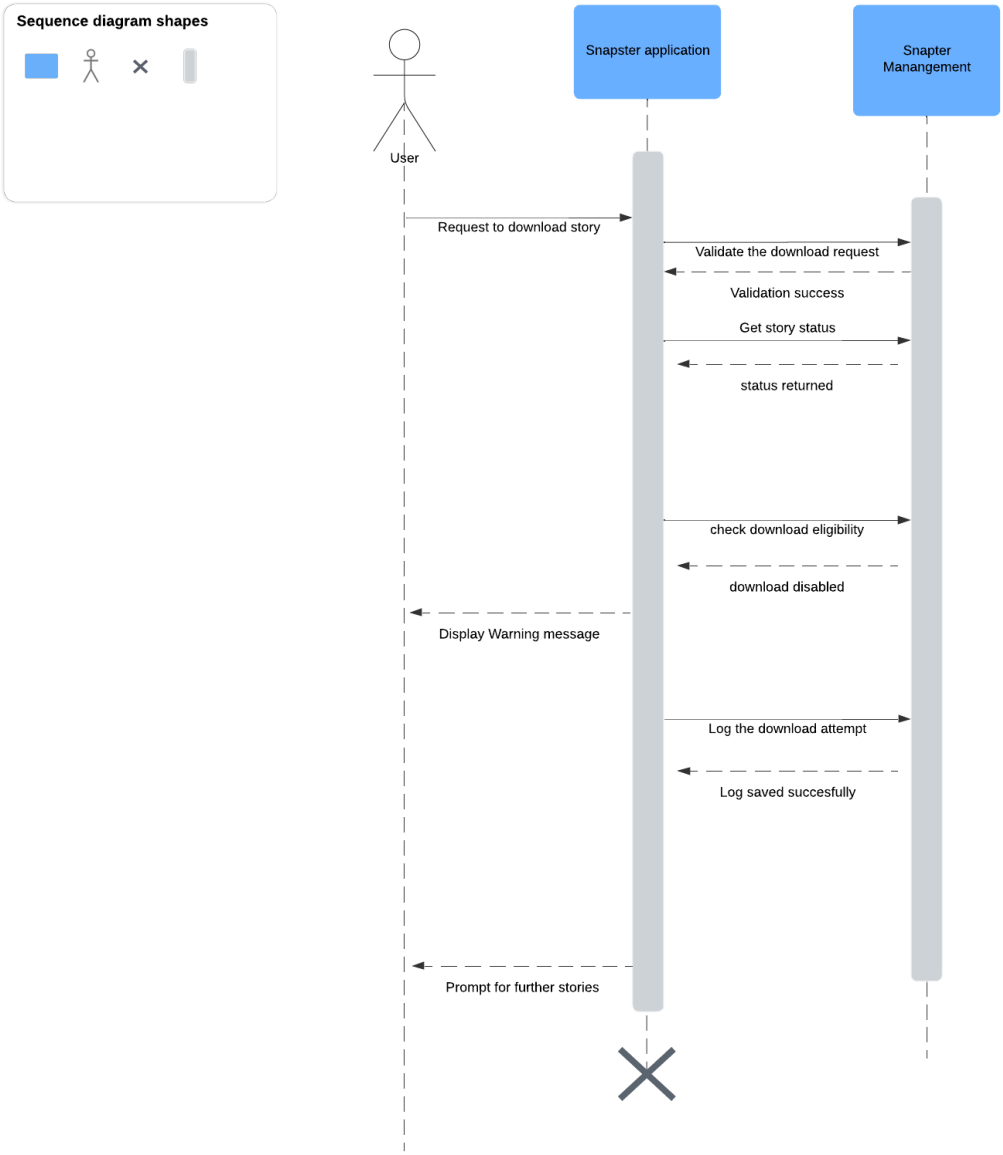
Sequence Diagram for Normal Scenario: Change Avatar

1. User selects "Change Avatar" option and the Snapster System loads list of available avatars
2. System asks Snapster Corporation the list of available avatars
3. Corporation responds the list of available avatars and presents to user
4. User selects such an avatar that is free of cost
5. System checks if the chosen avatar is a valid one.
6. If valid then it changes the user's avatar in the Snapster Corporation's database
7. Corporation responds by sending a message that the avatar has changed successfully
8. The system notifies the user about the change of avatar
9. User requests the system to update the balance of the user points
10. System requests the corporation for the points balance of the user
11. The corporation sends to the system the points' balance
12. System deducts the applicable points for change of avatar
13. Corporation confirms the deduction of points
14. Updated avatar is presented to the user.
15. User confirms change of avatar; system provides confirmation feedback.

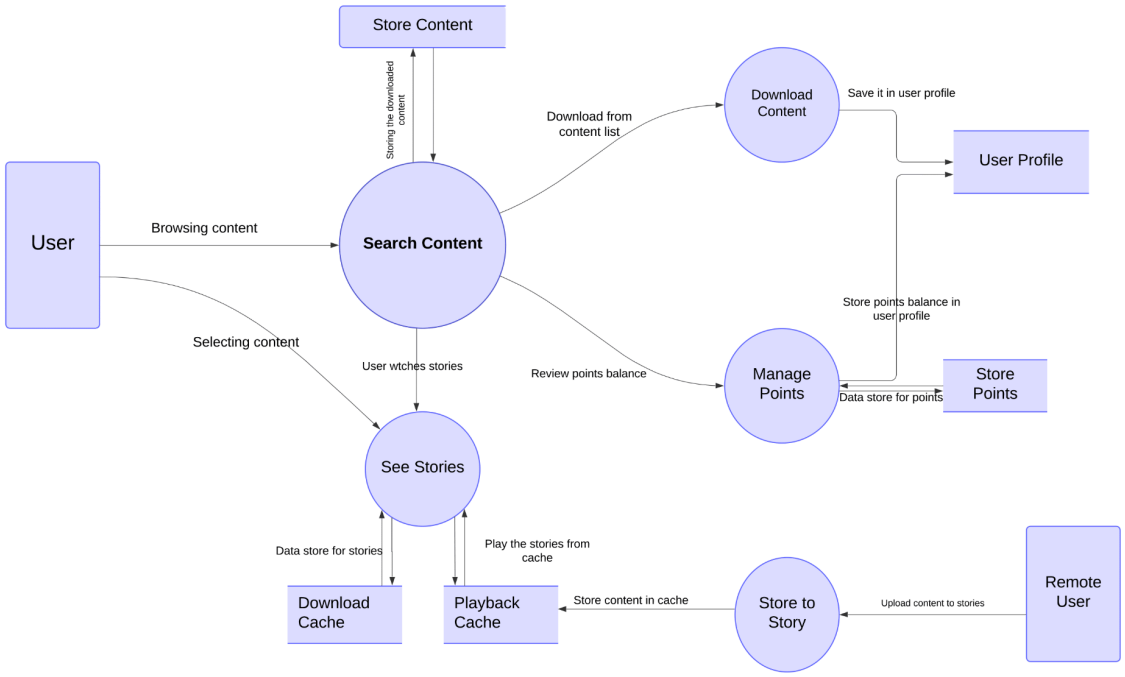

Sequence diagram shapes
Snapster application
Snapter Manangement
User
Select change avatar option
Load avatar options available
Return the avatar list
Display the avatars list
Select Avatar
Validate the selection
Validation success
Update the user avatar to selected one
Avatar updated succesfully
View Avatar
Update points balance
Check points balance
return the balance
Deduct the points
deduction succesfully completed
Show updated avatar
Confirm avatar change
Avatar change confirmed

9.2 Abnormal Scenario

Abnormal Scenario: Failure of Stories Download The user selects to download a story, and Snapster System checks if download request is valid. After the valid download request, the system checks if there are any reviews attached with the particular story for which the download request has been made and sends a request to Snapster Corporation in order to check the status of the story. Based on this request, the corporation sends the review result back to the system where the overall status of the story would be checked. The company confirms that the download is disabled; it might be blacklisted or for any other reason. So the system warns the user regarding failure in downloading. The system keeps a record of the history of the user regarding downloading, logs the attempt made by the user to download, and notifies the corporation regarding the attempt. The corporation confirms the log entry regarding the attempt to download. It will, at last, take the user to check for further stories.



10 Data Flow Diagram



This data flow diagram represents a content-sharing platform where the user will browse, download, and then watch the stories while managing his or her points gained. The interaction of the system revolves around the User, who, in this respect, is to be considered the entity instigating or initiating the interactions in the form of browsing and choosing the content through the process called Search Content. It will view/download the content from the media library, stored in Store Content. It selects the process of User Selection through which process it feeds into a smooth way to browse through the categories of contents and select items to be downloaded, which will be stored in his User Profile. It could make the Download Content process available to the media files for viewing by the user and download. Every time the user downloaded the content selected from the Search Content process, it would store it in his/her User Profile. This is in order to allow every custom profile to maintain a balance of points for the user so that he may track rewards or incentives that may have accrued due to interactions through the platform. This process shall be named Manage Points while the user can view and edit the balance of the point kept in Store Points.

Here, it allows for the process of See Stories, whereby a user is able to view the video stories in snippets. The system uses Download Cache and Playback Cache to first download the content and then be played smoothly for continuous viewing. Such a structure of caching reduces the time a page takes to load and provides seamless access to story content by users. Another great thing which Playback Cache does is allow playing directly from cached data. What that means is, while the users go through the stories many times, playback of the experience would be smooth.

The other core functionality is a process called Store to Story, which can be added by not just the main User but also Remote Users, especially in this form or feature called "stories". That is the gist of both, making the content on it diversified since other users can upload any form of media on the platform and the others viewing it. The uploaded stories cache in the Playback Cache, fetching and playing efficiently in order to make sure that accessing the story feed-for the user-is easy. The DFD here, in general, represents how this whole platform effectively arranges data flow in the interest of increased user experience. Support for fast content access, smooth playing, and efficient points management should be provided by structuring data storage, retrieval, and caching. This DFD enables the User to interact with the system seamlessly and provides powerful functionality on the backside for uploading remote content, thereby delivering a community-driven content-sharing environment.

11 Functional Requirements

These specifications are the functional requirements for Snapster™, that describe functional needs and features the application has to provide in order to offer users a great experience. The following requirements describe how Snapster™ will enable users to capture, post, and navigate video and audio materials in a decentralized social network context. Some of the core operational and social features are registration of accounts, making of friends lists and real-time communication channel and posting of messages. Multimedia functionalities including camera recording, filters and organization and download capabilities will also be provided through the point to point system in the app. Furthermore, there will be a Snapster™ rating procedure: when a user acquires or pays for points, he/she can have access to custom avatars and bonuses within the application. These functional requirements form the basis of creating Snapster™ as a dynamic and responsive use-centric platform that is designed with an emphasis on creative endeavors and connections, security, and safe sharing of media.

FR1 The app shall allow uploading and downloading of content on a decentralized peer-to-peer network to users.

Requirement ID	Description
FR1.1	Uploading: Users shall be able to upload video and audio files directly to the network.
FR1.2	Downloading: Users shall have the ability to download content files from other users on the network.

FR1.3	Parallel downloads: Downloads shall be optimized by allowing parallel downloads from multiple sources.
FR1.4	Privacy: Users shall have control over privacy settings, specifying who can view and download their content.
FR1.5	Notifications : Users shall be notified if their content is downloaded or shared by others.

FR2 The app shall implement a user engagement system with ranking and rewards.

Requirement ID	Description
FR2.1	Points system: Points shall be awarded to users for engaging in activities like sharing, commenting, and liking.
FR2.2	Redeem points : Users shall be able to redeem points for customizing avatars or making in-app purchases.
FR2.3	User ranking: The app shall display user rankings on their profiles, reflecting their engagement level.
FR2.4	Rewards : Rewards shall include unlockable content, avatar upgrades, and exclusive filters.
FR2.5	New ranks: Users shall be notified when they reach a new rank or unlock a reward.

FR3 The app shall support communication features, including messaging and group interactions.

Requirement ID	Description
FR3.1	Directing messaging: Users shall be able to send private messages to friends.
FR3.2	Group chats: Group chats shall allow multiple users to communicate in a single conversation.
FR3.3	Multimedia: Users shall be able to attach media files, such as images or short videos, in chats.
FR3.4	Secure chats : Chats shall be secured with end-to-end encryption for user privacy.
FR3.5	Notifications : Users shall receive notifications for new messages and group mentions.

FR4 The app shall include advertisement features integrated into the user experience.

Requirement ID	Description
FR4.1	Ads: Ads shall appear intermittently within the user's story feed, with a maximum of one ad per three stories.
FR4.2	Skip ads: Users shall be able to skip advertisements after 5 seconds.
FR4.3	Ad interactions : Ad interactions, including clicks and views, shall be recorded for advertising analytics.

FR4.4	Ads relevance: Advertisements shall be targeted based on user engagement data to ensure relevance.

FR5 The app shall allow users to search, browse, and filter content efficiently.

Requirement ID	Description
FR5.1	Search content : Users shall be able to search content by keywords or tags.
FR5.2	Filter search: Filter options shall allow users to sort content by type (e.g., video, audio) and popularity.
FR5.3	Categories search: The app shall allow content browsing by predefined categories.
FR5.4	Creator search: A creator search feature shall allow users to find content by specific creators or usernames.
FR5.5	Recommendations: Content recommendations shall appear based on user viewing and interaction history.

FR6 The app shall provide story and feed customization options for users.

Requirement ID	Description
FR6.1	Filters: Users shall be able to apply filters to their stories, with options like brightness, contrast, and effects.
FR6.2	Order of stories: Users shall be able to customize the order in which stories appear in their feed based on categories.

FR6.3	Save stories: The app shall provide options for users to save stories as drafts before posting.
FR6.4	Delete/edit stories: The app shall allow users to delete or edit a posted story within 24 hours of posting.
FR6.5	Track of stories: Story interactions, including views, likes, and comments, shall be tracked for each post.

12 Nonfunctional Requirements

Performance: For our system to be capable of providing fast transfer rates to several target sites, it has to enable quick, parallel downloads from different source sites. Depending on the quality of user experience which is expected, the time taken in downloading a file should be brought down. Snapster should be able to download from different users at the same time.

Availability: They continue that the availability of the application should be above 99.9% for end users of the site to have constant access to the search, view, and upload functions. Due to the fact that snapster can be used in a decentralized manner it is important that it supports downloading from multiple sources in case one is offline.

Flexibility: New users should be occasioned by no performance problems when using Snapster to create an account and begin using the services identified above. A large volume of content has to be processed by the system in terms of user created contents without impacting search relevancy or download rate.

Security: The user account and any of the user's credentials and/ data ought to be well protected. Consumers' messages and conversations have to be protected. Compliance warnings must be provided by Snapster from time to time in order to warn its users against sharing legal info.

Maintainability: The filter update process of the Snapster Corporation should not cause any inconvenience to the users. There, the system has to be easy to upgrade and introduce new functions without any interruption in the work.

Usability: The user interface must be easy to manage for the master and no technical skills must be required to master a program. It should attract many people and be easy to work with for every individual that will be using it. In a consumer's view, it will not require much effort for them to look for and acquire the material in question. Alabama The strategic search algorithm has to work, while the material can be very structured.

Reliability: The caching method by Snapster is effective. It has to recognise content deletion without distorting the value of playback. Functions such as downloading and displaying the material must have fail-safes incorporated in order to have minimum impact in worst-case scenarios involving problems with servers or the users' device capabilities.

The non-functional requirements are identified below – The data in this application has to meet certain performance analysis, security, usability, scalability and ;maintenance standards. These NFRs are to guarantee the reliability of the app with the specified conditions while also allowing for a positive user experience and future development. That is why, following these specifications, the application is targeted to satisfy the users' needs and requirements as well as to correspond to the common worldwide tendencies and legal demands.

NF1. For the application to be effective, it shall own a crash rate of not more than one in every thousand user sessions.

NF2. File upload, download and in-app messaging are functionalities that will be core to Allcore; they must have an up-time of 99.9%.

NF3. On mobile devices connected through 4G or higher connection speeds, the app shall launch in less than 3 seconds.

NF4. It should take no longer than 2 seconds for content search and browsing results to appear.

NF5. File download speeds shall be at the best network link available, the duration required to download a file of up to 100MB shall be less than 20 seconds.

NF6. The app shall be able to support up to 50, 000 concurrent connections without affecting the app performance.

NF7. There should be flexibility in the creation of the application System architecture to easily provide for growth in that the application should be capable of handling double the traffic it currently experiences without having to shut down.

NF8. It shall be able to accommodate up to 10,000 uploads at a given instance and make 20,000 downloads concurrently.

NF9. Any and all messages conveyed between users, personal and individual chats and file shares included, must be encrypted with end-to-end encryption.

NF10. Users will be expected to turn on MFA for login in addition to following the Usability Principle.

NF11. The different operation paradigms of the application implemented shall include rate limiting that will protect the application or the server against DDoS attacks.

NF12. The security audit should be performed at least twice per year, to cover all or most of the areas that might be at risk or could be compromising the standards set in the company's security policies and procedures.

NF13. The application interface shall be rather uncluttered, it should take no more than three taps to get to any basic functionality of the application like uploading a content or trying to see the feed.

NF14. Specific features within the app shall include: the app shall be accessible to visually impaired individuals with features such as screen reader, and text-to-speech functions.

NF15. The app shall offer multilingual support for at least five languages: ,English, Spanish, French, Deutch and mandarine.

NF16. The app shall operate on and run on Android and IOS platforms and devices up to a version of three years old.

NF17. The web version shall be compatible with the current two versions of Chrome, Firefox, Safari and Edge browsers.

NF18. It is mandatory to ensure that all user's data, including media and profile information is backed up each 24 hours.

NF19. It shall also provide ways of checking replication to ensure that no duplication of files will occur and that files being uploaded are not affected by some virus.

NF20. Any and all user activity shall be logged for six months and then deleted to ensure privacy and to save space.

NF21. The application code base should be modulated so that the different application modules can be updated without necessarily updating the whole application.

NF22. This further implies that all the code as well as the API endpoints shall be facilitated by good documentation.

NF23. The app shall be for swift patch distribution specifically for fixing of high ends in less than 24 hours of identification.

NF24. The app shall respect data protection laws for the European Union market, which is GDPR and for the California market known as CCPA.

NF25. Collecting personal data and information shall be mentioned as policies to which the users shall consent to before creating an account.

NF26. The application shall implement a caching mechanism to minimize the load placed on the server as well as the time taken to respond to most frequently used data.

NF27. Information that may identify the user shall be protected by the use of secure encryption both during transit and while in storage.

NF28. The application shall have automated monitoring, the ability to detect unwanted performance drop off or activities.

NF29. All media files that users upload shall be saved in a scalable cloud storage solution that has redundancy to avoid loss of data.

NF30. This is in light of the fact that the application shall include disaster recovery plans whereby the complete system shall be recovered within 2 hours of occurrence of the disastrous event.

13 Traceability Tables

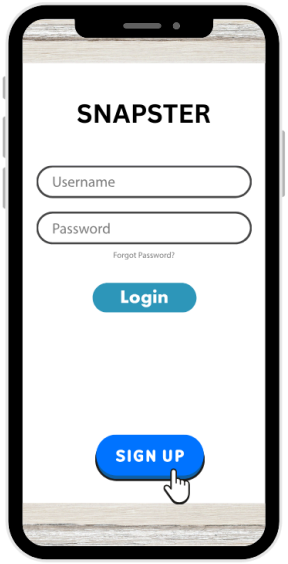
The SNAPSTER project consists of different use cases intended to improve interaction and operations in various ways. UC1: User Registration enables the creation of a new user account as well as verifying the user's e-mail address (Functional Requirement 1.1, Functional Requirement 1.2). After registration, the FT3 is enabled and the users can deliver new content like videos or audios (FT2.5). User Profile Management (UC3) allows access to the user's profile so as to permit him to customize the environment as specified by FR5.1 and FR5.2. To get into the platform, the users conduct User Authentication (UC4) with their identity number (FR1.1). Also, the Search Functionality (UC5), which enables the user to realize the content search by keywords (FR6.1). Content Interaction (UC6) allow users to like and comment on the shared content – Liking and commenting (FR6.2, FR6.3); whereas Friend Request Management (UC.7) allows users to connect with other users – Connection (FR1.4). The platform also enables Content Sharing (UC8) with friends through direct messages (FR5.3, FR6.5) and updates users through Notifications Management (UC9) (FR1.3). Users can customize their particular options Privacy Settings (UC10) which define who can view users' content (FR1.3, FR1.4). Also, they are able to submit objectionable content through feature Content Reporting (UC11) and prevent users through User Block Functionality (UC12) (FR3.1, FR3.2, and FR3.4). Further, the project comprises of Content Categorization (UC13) which conform to the Functional Requirements FR2.2 and FR2.4 and Content Tagging (UC14) which will improve the search aspect as captured in Functional Requirement FR2.1 and FR 2.2. Users get User Engagement Metrics (UC15) where they can monitor the performance of their content (FR4.1, FR4.4) and they get Advertising Integration which is a feature showing them relevant advertisements

alongside their performance statistics (FR4.3, FR 5.5). By reaching these use cases, SNAPSTER is planning to develop a secure and versatile social media that focuses on satisfying user experiences.

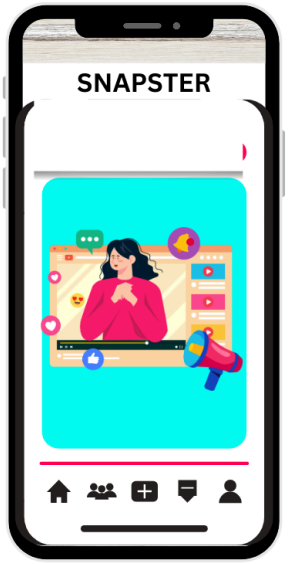
Use Case	Requirements
UC1	FR1.1, FR1.2
UC2	FR2.5
UC3	FR5.1, FR5.2
UC4	FR1.1
UC5	FR6.1
UC6	FR6.2, FR6.3
UC7	FR1.4
UC8	FR5.3, FR6.5
UC9	FR1.3
UC10	FR1.3, FR1.4
UC11	FR3.1, FR3.2
UC12	FR3.1, FR3.4
UC13	FR2.2, FR2.4
UC14	FR2.1, FR2.2
UC15	FR4.1, FR4.4
UC16	FR4.3, FR5.5
UC17	FR4.1, FR4.2
UC18	FR1, FR6
UC19	FR5.1, FR5.2
UC20	FR5.5
UC21	FR2.3, FR2.5
MUC1	FR3.1, FR3.2

14 User Interface Design

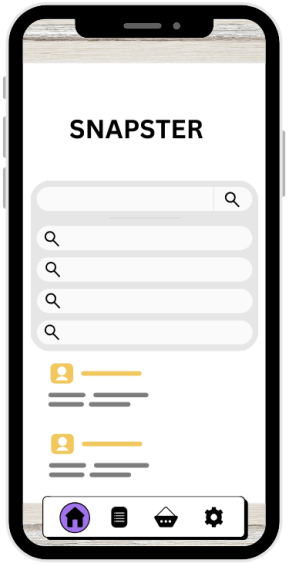
14.1 Storyboards



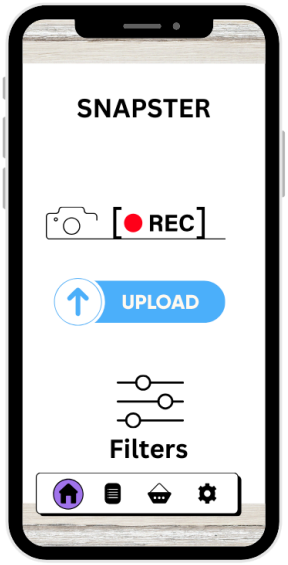
Login/sign up page



Homescreen

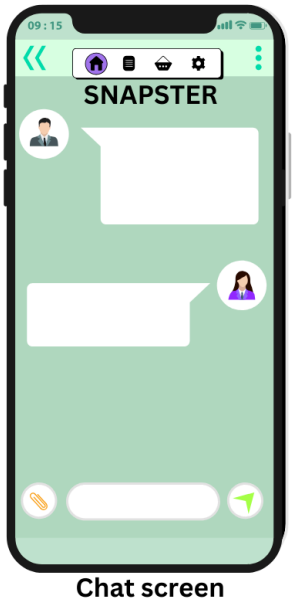


Search screen

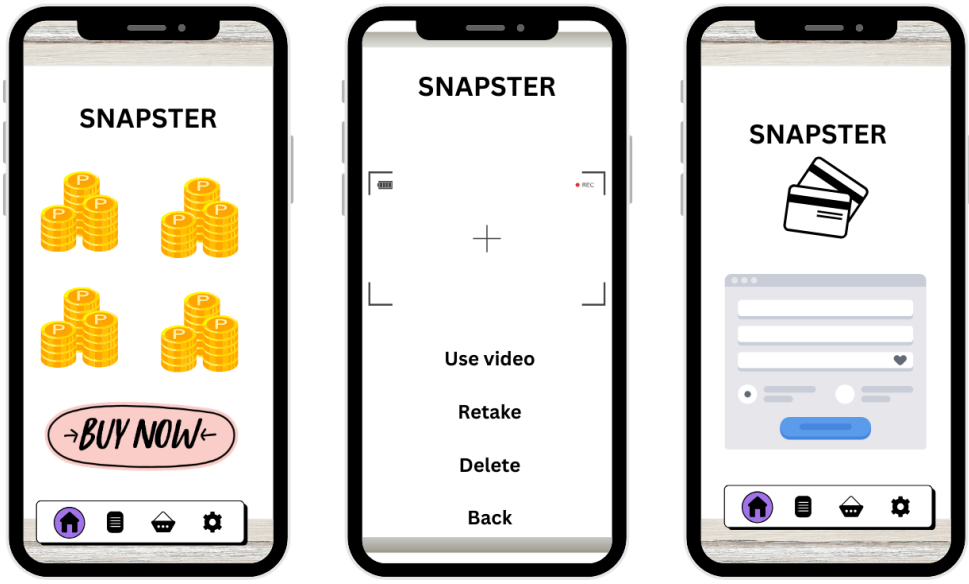


Upload screen

Profile screen



Chat screen



Buy points screen | Video creation page | Billing screen

1.Login/Sign-Up Screen

Components:

- "Email/Username" and "Password" text boxes one above the other with 'Forgot password' link below the password text box.
- Login, join, and password connection buttons.

Purpose: Allows a customer to sign in, or sign up for a new account.

2. Home Screen

Components:

- Video posts: a feed for the content cards of videos and audio posts with related thumbnails, titles, and the user information.
- A bar of rounded rectangular pictures at the bottom of the page with Home, Search, and Upload, Notifications, and Profile.

Purpose: Here users engage with the content-feed or they call other sections of the app.

3. Search Screen

Components:

- A box at the top for entering the whole keyword.
- Addition options such as Content Type Filter, Popular Posts, and Recent Posts.
- Organized results in the form of cards identified based on filter types.

Purpose: The feature enables one to look for content or particular personalities of one's choice.

4.Upload Screen

Components:

- Record and upload existing file buttons.
- The available choices for the inclusion of meta tags including the title, description and the list of tags.
- A panel for custom filters "overlay" to be prescribed on the uploaded videos/audio.

Purpose: Allows a user to generate new content or contribute with existing one.

5. Profile Screen

Components:

- A profile banner that shows the user's avatar image, the username, short bio.
- Points stats, uploads stats, followers stats, and ranking stats.
- An option button for adjusting profile details such as changing an avatar or settings of privacy.

Purpose: People have detailed individual information that is shown and where the user profile can be regulated.

6. Chat/DM Screen

Components:

- A list of active threads with the user names/nicknames and photos in their avatars.
- A messenger for to send messages or file sharing.

Purpose: It promotes one-to-one communication activities such as the transfer of files.

7. Buy Points Screen

Components:

- Input used in determining what points to purchase a particular product (for instance 100 points, 500 points, one thousand points among others).
- Credit card info, use of PayPal.
- Transaction summary X cost X points.
- Confirm purchase button.
- Success/failure notification.

Purpose: Enables the users to purchase points for use in the application acquisitions.

8. Video Creation Screen

Components:

- "Start Recording" button.
- Video preview.
- Pause/Stop button.
- Title, description and tags section.
- Filter/effects options.
- Save/Upload button.

Purpose: It allows the capturing and uploading of videos.

9. Billing Page

Components:

- Account statement (old bill).
- Payment method management.
- Listed below are the governing plan details with regards to the subscription:
- Download invoice button.
- Contact support link.

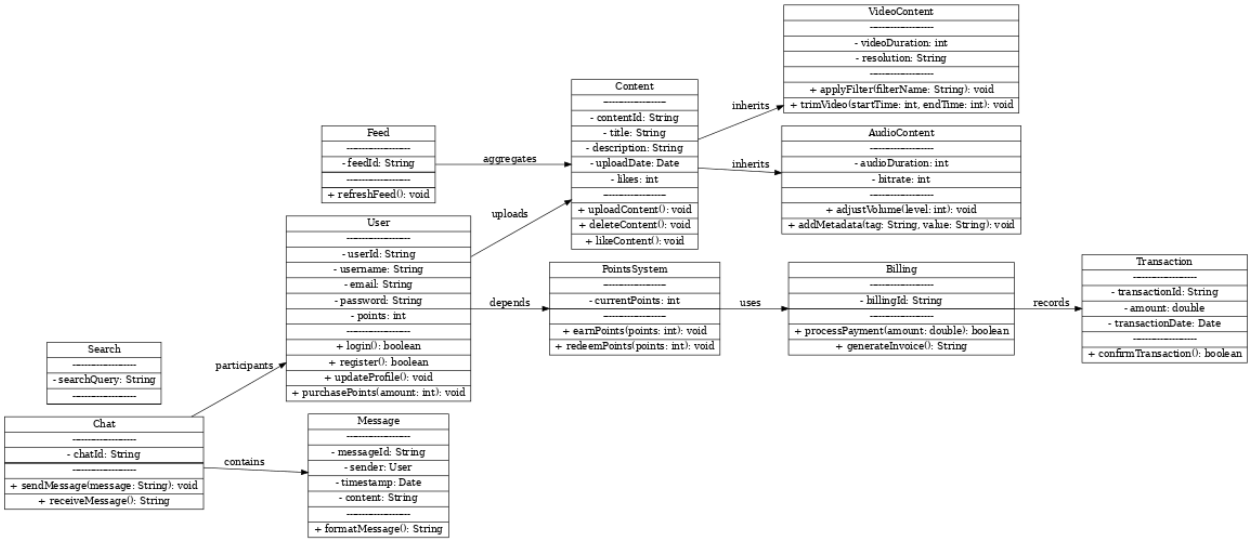
Purpose: Store and Organize all the other activities related to payments, bills, and subscriptions.

14.2 Link Analysis

Source Screen	Action/Trigger	Destination Screen	Description
Login/Sign-Up Screen	User clicks "Log In"	Home Screen	After successful login, users are taken to the home screen to interact with content.
Login/Sign-Up Screen	User clicks "Sign Up"	Sign-Up Screen	If the user chooses to create an account, they are directed to the sign-up page.
Login/Sign-Up Screen	User clicks "Forgot Password"	Forgot Password Screen	If the user needs to reset their password, they are directed to the password recovery page.
Home Screen	User clicks on a content card (video/audio)	Content Detail Screen	Users are taken to a detailed view of the selected content (video/audio post).
Home Screen	User clicks the "Search" icon	Search Screen	Clicking the search icon takes the user to the search page to find content.
Home Screen	User clicks the "Upload" icon	Upload Screen	Users are directed to the upload page to add new content.
Home Screen	User clicks the "Notifications" icon	Notifications Screen	Displays notifications for new activity or content updates.

Home Screen	User clicks the "Profile" icon	Profile Screen	Takes the user to their profile page to view and manage their profile.
Search Screen	User selects a filter option	Search Results Screen	Displays search results filtered by the selected criteria.
Search Screen	User clicks on a content card	Content Detail Screen	Clicking a content card will show the details of the selected content.
Upload Screen	User clicks "Record"	Video/Audio Recording Screen	Takes the user to the screen where they can record a new video or audio post.
Upload Screen	User clicks "Upload Existing File"	File Selection Screen	Opens a file selection screen to choose an existing file to upload.
Upload Screen	User submits metadata (title, description, tags)	Home Screen	After uploading, the user is redirected back to the Home Screen.
Profile Screen	User clicks "Edit Profile"	Profile Edit Screen	Allows the user to customize their profile (e.g., change avatar, bio).
Profile Screen	User clicks "Buy Points"	Buy Points Screen	If the user needs to purchase more points, they are directed to the buy points page.
Profile Screen	User clicks "Billing"	Billing Screen	Redirects to the billing page to view or manage payment information.
Profile Screen	User clicks "Log Out"	Login/Sign-Up Screen	Logging out brings the user back to the login or sign-up page.
Chat/DM Screen	User clicks on a conversation	Chat Window	Clicking a conversation takes the user to the chat window where they can send messages.
Chat/DM Screen	User sends a message or file	Chat/DM Screen	After sending a message/file, the user remains in the chat to continue the conversation.

15 Class Diagram



In the case of the SNAPSTER project the above class diagram describes the fundamental architecture of the application and the interactions between the different parts of the SNAPSTER project. In the middle is the User model that handles the users' authentication and account creation and modification functions in addition to attributes such as 'username' and 'password'. It also manages the acquisition of the points by use of the purchasePoints method.This is in addition to making payment for the in-app points. The Content is an abstract class of media uploads, like a video and audio, and has fields common with other recursive media types like 'title' and 'uploadDate'. Further subclasses, VideoContent and AudioContent, enhance this capability with additional characteristics as video duration and bitrate and methods as filter applying for videos and volume controlling for audio.

The Feed class handles the content presentation by updating and retrieving new content to present to the users, the Search class that implements the queries and filters that allow users to discover the content they need. What it turns out is that in order to interact with users and enable a means of sending and receiving messages, there are specific classes, the Chat and Message classes, that encompass the abilities such as the sender of a message as well as the time stamp from the builder of the object with attributes like 'sender' and 'timestamp' respectively. The PointsSystem class controls the earning and using of points, collaborating with the Billing class, which controls payments and contains records of transactions in the Transaction class.

The current design also reflects basic key object-oriented design patterns. Altogether, these classes and patterns compose a solid, composite structure where each component is effectively segregated and loosely coupled from the other, to serve the fundamental functions of SNAPSTER while remaining highly scalable for subsequent improvements.

Object-Oriented Design Patterns in the SNAPSTER Project

To enhance the architectural integrity and scalability of the SNAPSTER project, two widely used object-oriented design patterns were implemented: those that include the Factory Pattern and the Observer Pattern. Both densities are organizational in maintaining the essential characteristics of the growing and changing social media platform like SNAPSTER which include flexibility, easiness of maintaining and future expandability.

1. Factory Pattern (Content Creation)

The Factory Pattern is used to control the creation of the content types like VideoContent and AudioContent, dynamically. Its fundamental concept can be best summed up as to encapsulate the creation of objects of different types of contents within this pattern so that it becomes easier and more efficient to integrate new types of contents with the system without having to write a great deal of new code from scratch. This is a significant fact for the innovative social media platform like SNAPSTER, its new content formats & types may be invented over the time, hence a maintainable and scalable approach is needed.

In this case there is required a ContentFactory class that accepts requests to create instances of content objects. There are no direct calls to VideoContent or AudioContent creation in the system which instead relies on the ContentFactory class to order the creation of a desired content type based on client input. The ContentFactory class contains a method createContent(type, metadata) where the type parameter defines whether a content object to be created is a video or an audio. This information is used by the factory method to create and return the proper object: a VideoContent or an AudioContent thus allowing the main application to keep no concern to how the objects are created or what class the objects are an instance of.

Therefore, the SNAPSTER system becomes much more flexible by applying the Factory Pattern. All of this means that if a new content type (like images or text-based post) need to be inserted in the future; it can be done easily without brainstorming about the core logic behind the app. Rather, only the factory needs to be extended in order to create the new content type For this, a new object type which respects to the open/closed principle of designing object oriented systems (the system is opened to extension but closure to modification).

2. Observer Pattern (Real Time Update)

Observer Pattern is used in SNAPSTER to allow constant update to be made. This pattern is actually useful if you are in a social media where users want real time updates of the contents posted or the messages and other activities concerning the social media. The Observer Pattern in a way future proofs the design by allowing the Feed and Chat components, which are the subject, to update many users or observers in real time.

In the case of the Feed class, which is responsible for displaying new content to the users, the pattern works as follows: when new content for example a video or an audio post is posted, the Feed class informs all subscribed users of the new content. In Feed class, there's an array of references to Observer (people who want to receive the update). Every time the new content is published to any feed, the notifyObservers() method is invoked and all the feeds of users who subscribed to the content are updated at once. This makes it easy for the users to always know when there is new content posted since the content page updates immediately there is new content posted.

Likewise, in the Chat class, the Observer Pattern is employed so that the users retrieve updates on active conversations in real time. In case a user posts a message, all observers are informed, and they appear in the conversation, receiving information that a new message has been posted. This implementation makes the communication sound, or else puts it in harmony with whoever is communicating in the chat which is essential in a chat system.

The Observer Pattern further helps by offering the user instant feedback and make sure the user is always able to see the latest in on interactions; be it updated content or messages in one's chats. This is essential for the success of a facebook application because it constantly provides the users with the vivo feedback of the running activities.

Why These Patterns?

The Factory Pattern was adopted as part of the SNAPSTER project, primarily to fulfill some of the flexibility, scalability, and response needs required in a social media application also, the Observer Pattern was selected to augment this flexibility in the form of a notification mechanism.

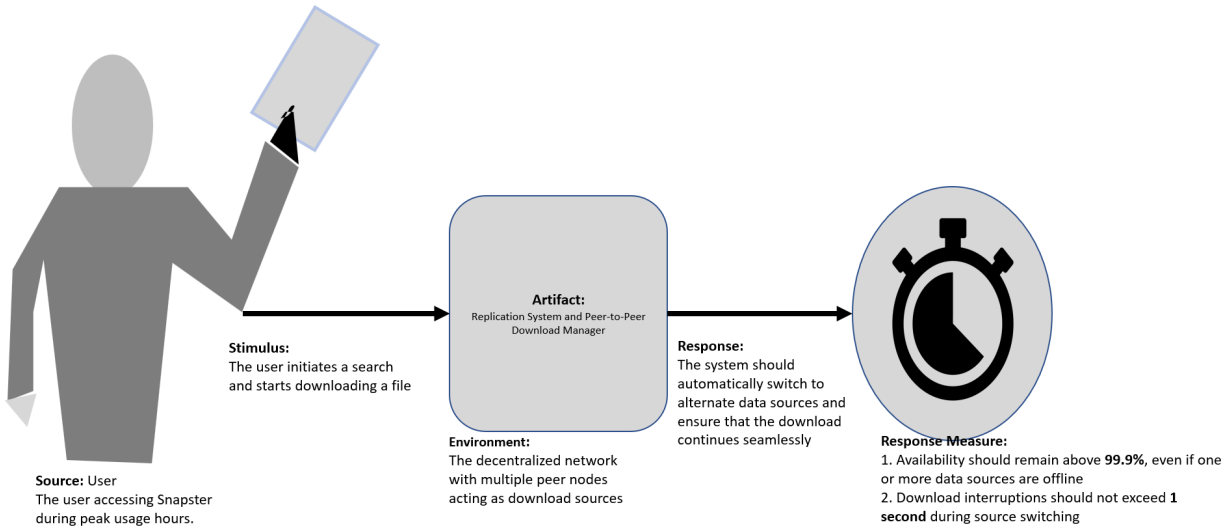
Factory Pattern: Generally, this pattern faces the execution and management of content types in a highly flexible and easily scalable fashion. Their separation allows cementation of new types of content without affecting existing logic, as the content creation logic is separated. When the platform is expanded and additional content types are added into a stream, it is easy to bring the changes into the factory without disturbing the other components.

Observer Pattern: Social sites require real-time updates to provide the experience that users are looking for, thus, the Observer Pattern gives the system an opportunity to communicate asynchronously. Both in case of a new piece of content in the timeline and in case of a new message in a conversation, the user gets a notification which improves his overall experience.

Combined, these design patterns allow adaptive development of the SNAPSTER platform to be maintainable, extensible, and prepared for future expansions; providing a stable foundation to support the growth of SNAPSTER.

16 Quality Attribute Scenarios / Tactics

16.1 Availability Scenario



Implementation of Tactics for Availability in Snapster

As earlier discussed, to maintain more than 99.9% availability, and ensure that a user can smoothly download files from its network, Snapster's decentralized system calls for mechanisms that would guarantee minimal disruption of service. These objectives are to be achieved by using two basic strategies namely Node Health Monitoring with Failover and Data Replication with Redundancy. These tactics will be employed considering Snapster as a decentralized system so that highest availability can be provided even due to failed nodes, network problems, or loaded conditions.

1. Node Health Check Management and Sensor Failure Management

Overview and Purpose:

Node Health Monitoring and Failover makes Snapster capable of saying that they will be able to handle node failures and make sure that download is not interrupted. The decentralized network is made up of multiple peer nodes and one of them can be a source of download. To ensure availability, there is a need to periodically check nodes for faults, and ensure that the numerous requests Downloaded by the java application are redirected to working nodes.

Detailed Implementation:

Health Check Service: Currently the Health Check Service that monitors the status of each peer node running in Snapster will be active all the time. Such information may be obtained through generating traffic to the connected nodes and periodically checking on their response time to our request, the bandwidth capacity and the quality of the updated data. Latency and packet loss will be measured and therefore compared with specific standard levels.

Heartbeat Mechanism: An important feature of the monitoring system is called Heartbeat Mechanism. Albeit, every peer node will broadcast constant signals known as 'heartbeat' to the central monitoring part. If there are unprocessed or late heartbeats for a certain period (2-3 seconds), the node will be specified as "unavailable".

Dynamic Failover System: If a failure is sensed, then the failover mechanism will be automatically initiated by the system. Continuing with the download, the current connection shall be shifted to another with other suitable nodes in accordance to parameters such as latency or throughput. As is the case with the previous process, this is designed to take one second in order not to inconvenience the user, downloading large files.

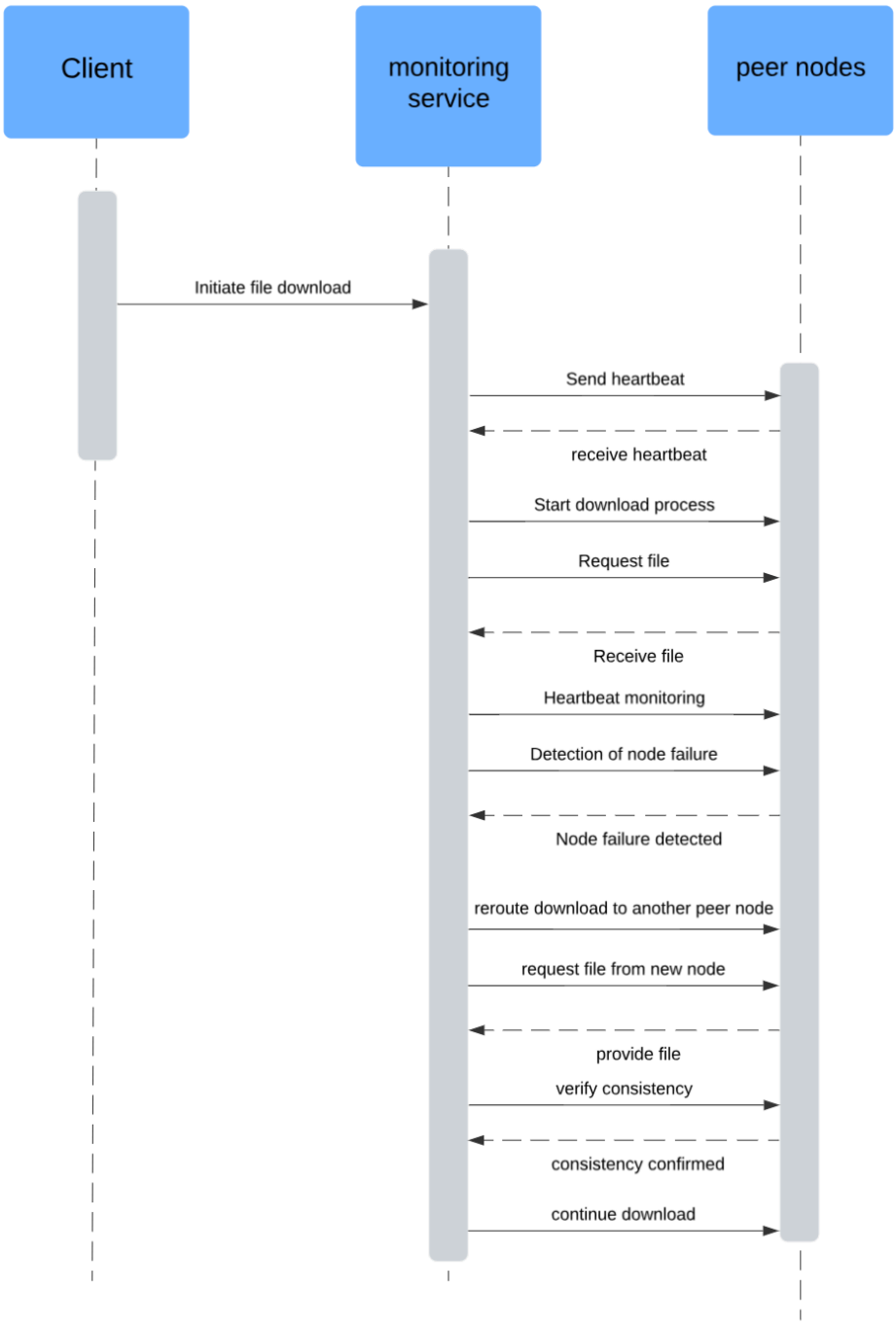
Consistency Verification: To avoid having replicate or corrupted data during the switch, Snapster will employ a consistency verification method. In the failover system, the data segment will be checked and the new node will confirm the segment it has downloaded before the failure and start from the correct point of the data segment to avoid overlapping segments or missing segments.

Example Scenario:

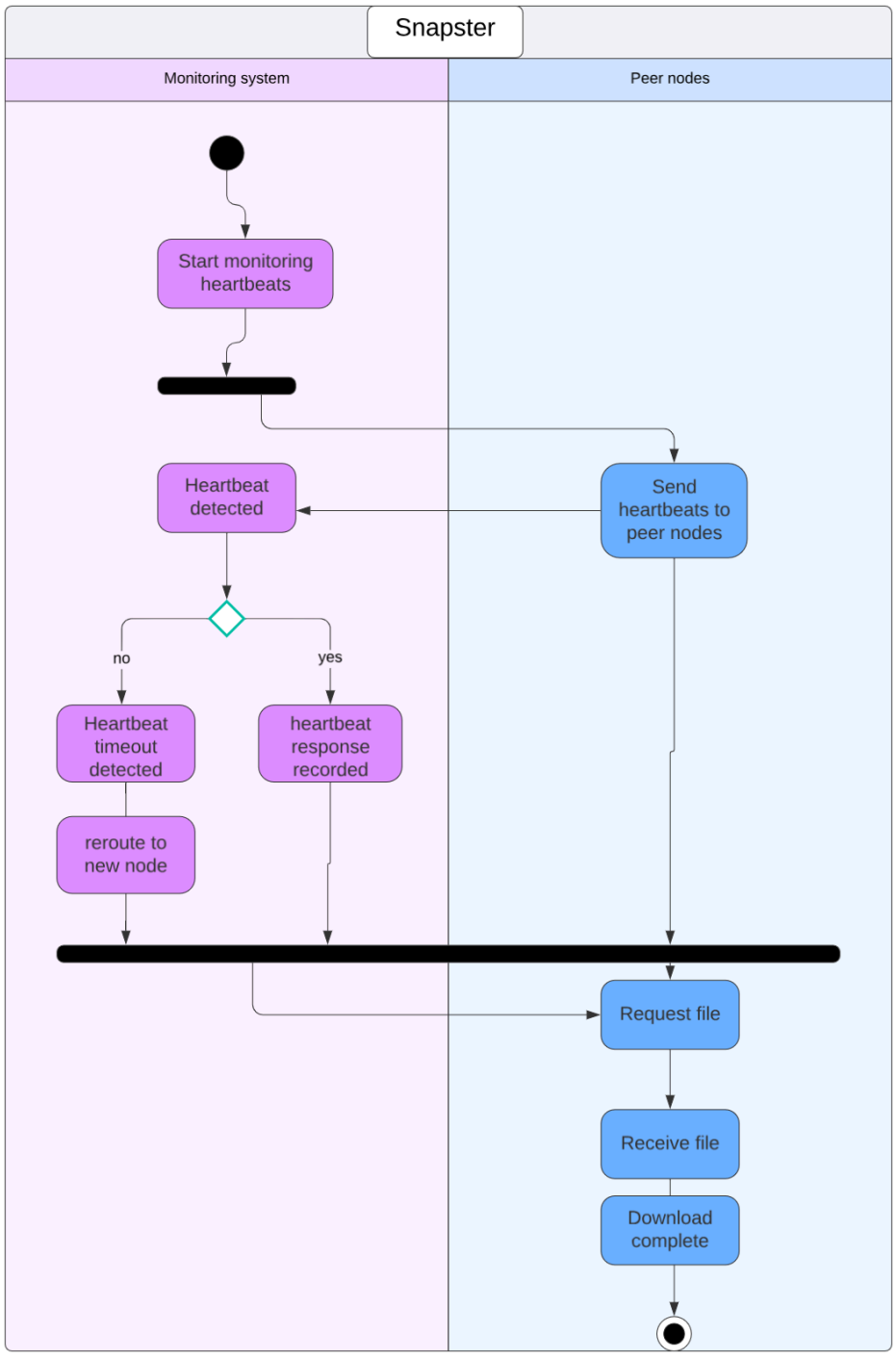
A large video file being downloaded by a user may find that half way through the download the particular node is switched off. In the event, Snapster's Health Check Service identifies the problem by missing heartbeat and quickly the Dynamic Failover System switches the downloading to the nearby node within one second. This completes the download without much interruption.

Artifacts Implementation:

1. **Sequence Diagram:** The sequence diagram will show how the interactions between the client, monitoring service and the peer nodes take place with regard to the heartbeat check, node failure check and the download path.



2. **Activity Diagram:** Here, the activity diagram will reflect on more explicitly provided incremental steps in health monitoring, failure detection process, failover commencement, and confirmation of consistency processes.


Snapster
Monitoring system
Peer nodes
Start monitoring heartbeats
Heartbeat detected
Send heartbeats to peer nodes
no
yes
Heartbeat timeout detected
heartbeat response recorded
reroute to new node
Request file
Receive file
Download complete

2. Data Replication and Redundancy Strategy

Overview and Purpose:

Data Replication and Redundancy are crucial for the reliable availability of content consumed from one user and may be stored on multiple nodes. With files duplicated across multiple nodes, Snapster can ensure that the content being accessed does not stop. This tact ensures that in the event of a node failure there is a backup node with a copy of the file reducing down time.

Detailed Implementation:

Replication Policy and Factor: Snapster will follow a replication scheme in which every file will be allocated to at least five different peer nodes. These nodes will be chosen randomly so that the nodes in each network are distributed physically, and thereby provide backup for each other in the event that a geographical network is inhibited from operating by conditions such as lack of power or mechanical breakdown.

Automatic Replication Management: The system will also use an Automatic Replication Manager to oversee the addition and removal of replicate copies. Whenever any node with replica copy becomes offline, the Replication Manager will ensure that the redundancy level requirement is met by replicating the data to another healthy node. It also provides a way to have continuous data availability throughout the process.

Load Balancing and Optimized Reads: In the case where users start downloading, load balancing will be used at Snapster to spread out download requests among the nodes. The system will also be able to choose the node that best reaches the user or the node with less downloads at any given time.

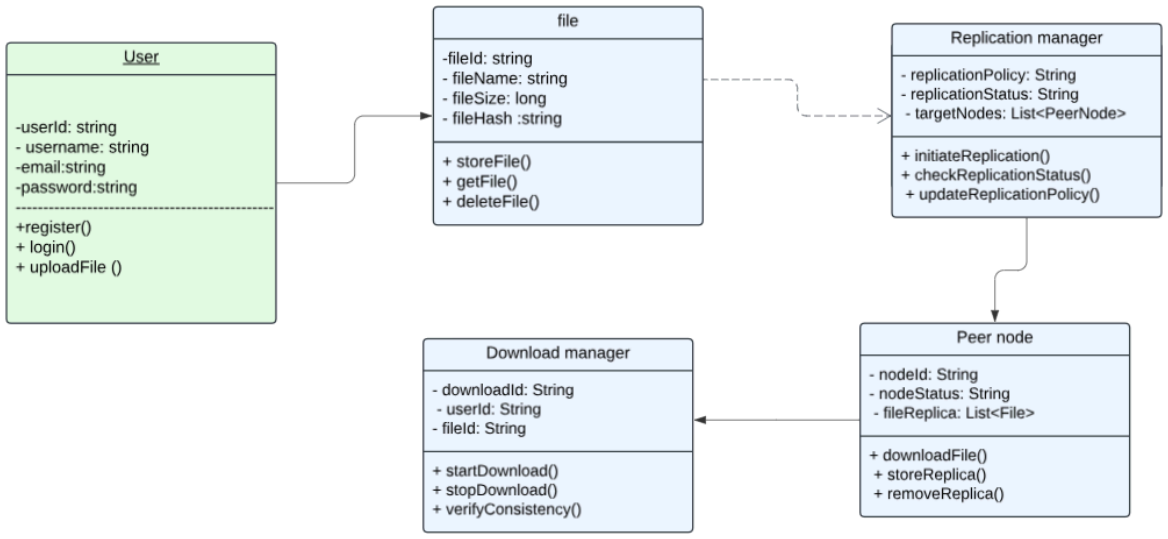
Data Integrity and Version Control: To maintain the quality of the replicated data, the use of checksum validation will be implemented in Snapster. The regular check is also laid down for each replicated copy to ensure that the data is not corrupted. Besides, version control will also be applied in the system, so that each time connected user will get the updated version of the required file, if there are several versions at different nodes.

Example Scenario:

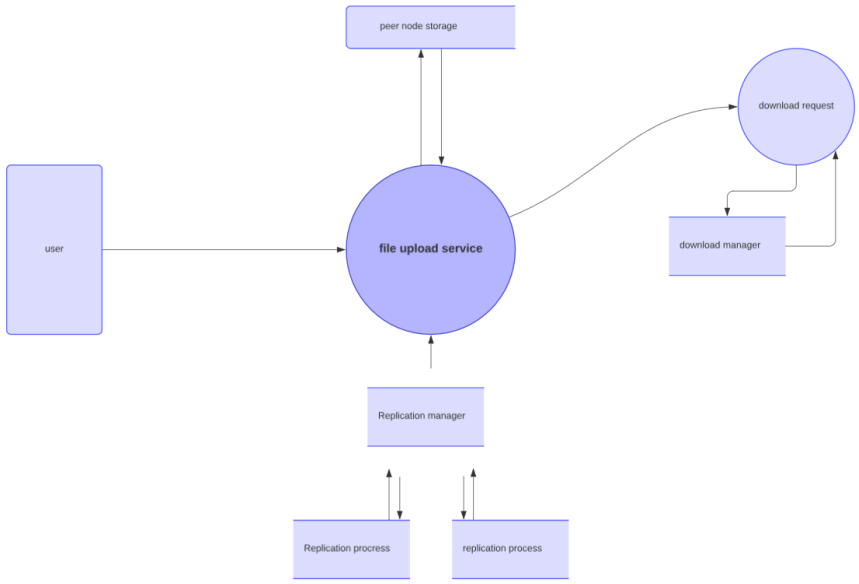
In Snapster there is the option that when a user uploads a video to be stored in Snapster the video file will be copied five times on different nodes in different geographic locations. In the event that one of these nodes becomes unavailable because of a network problem, Replication Manager promptly creates another copy for the user’s file in the other node to be downloaded without interruption.

Artifacts Implementation:

1. **Class Diagram:** The class diagram will have a ReplicationManager class, responsible for maintaining replication policies and ensuring data integrity.



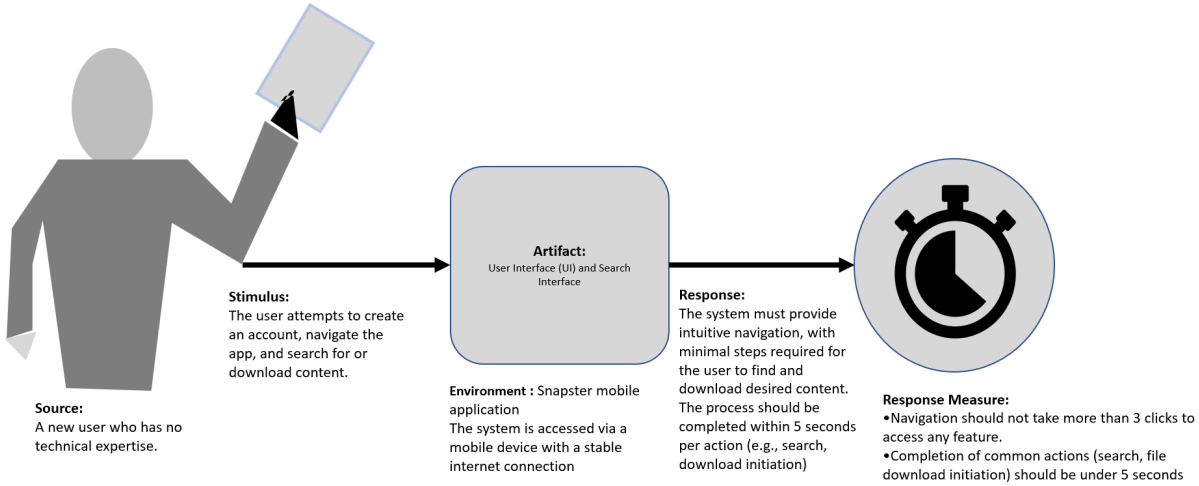
2. **Data Flow Diagram:** Data flow diagram will illustrate the flow of data from the user upload to replication across multiple nodes and how data retrieval is managed during a download request.



Conclusion:

By practicing Node Health Monitoring with Failover and Data Replication with Redundancy, Snapster is confident to maintain an availability rate of over 99.9%. Such strategies allow the system to adaptively redirect downloads in the case of node failures and have multiple copies of users' materials on nodes to provide uninterrupted work. This boosts the user experience as files can rapidly be accessed instantly for use despite being dense, during high traffic or during network irregularities.

16.2 Usability Scenario



Implementation of Tactics for Usability in Snapster

For Snapster to attain a high usability of this system, the activities, such as file search and download should have a smooth and natural interface, as perceived by users. It is the key aspect to sustain the website user engagement and satisfaction. Two strategies here will be used: Responsive UI Design and Progressive Feedback Mechanism. These tactics will meet the identified user need concerning the real-time servicing, rapid operating, and feedback of the action like search and download.

1. It is also important in creating a Responsive User Interface (UI) Design.

Overview and Purpose:

The responsive UI design also helps in making sure that Snapster fits well in all the available devices, screen size as well as the user's touch on any of the devices. This tact is very crucial in the user decentralized environment whereby users can press the application from different devices such as mobile phones, tablets and detached computers all with different network connectivity strength.

Detailed Implementation:

Adaptive Layout Design: The UI will be done using principles of RWD includes the use of fluid grid, flexible images, and media queries. The layout will respond to the screen and adapt and fit it so that users can view and or interact with it on any device. This shall help to avoid such problems like having two similar buttons or images that are not well placed due to failure to recognize their scale.

Dynamic Content Loading: With Snapster, section contents such as search result list, recommended files, and peer presence will be loaded asynchronously. Since only the selected parts of the site are updated, its loading will not require the time of several minutes as it is the case with common practices and the user will be able to enjoy the real-time update of the selected sections. This will be accomplished through the use of asynchronous JavaScript and XML AJAX and the WebSockets for dynamic updates that do not require a full page reload.

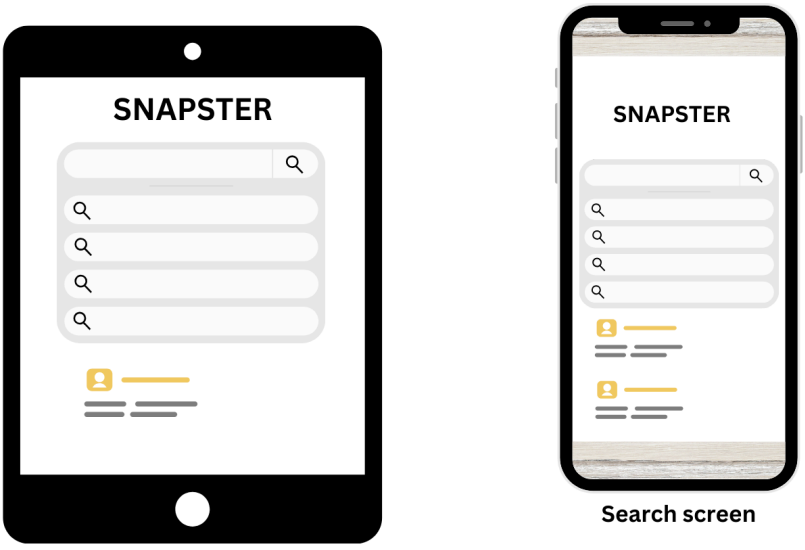
Optimized UI for Network Conditions: The second feature is adaptable UI, which will inform Snapster of the user's network situation and adjust the interface. For low bandwidth users, the application will have low quality of graphical and animated images and only fetch important content that can easily be loaded with less demand for network resources.

Cross-Platform Compatibility: But the problem is that the system will be made compatible for use across a number of operating systems and browsers and this testing will be done rigorously to ensure uniform look and feel. This is across portable devices such as iOS and Android, and fixed environments such as Windows, MAC and Linux.

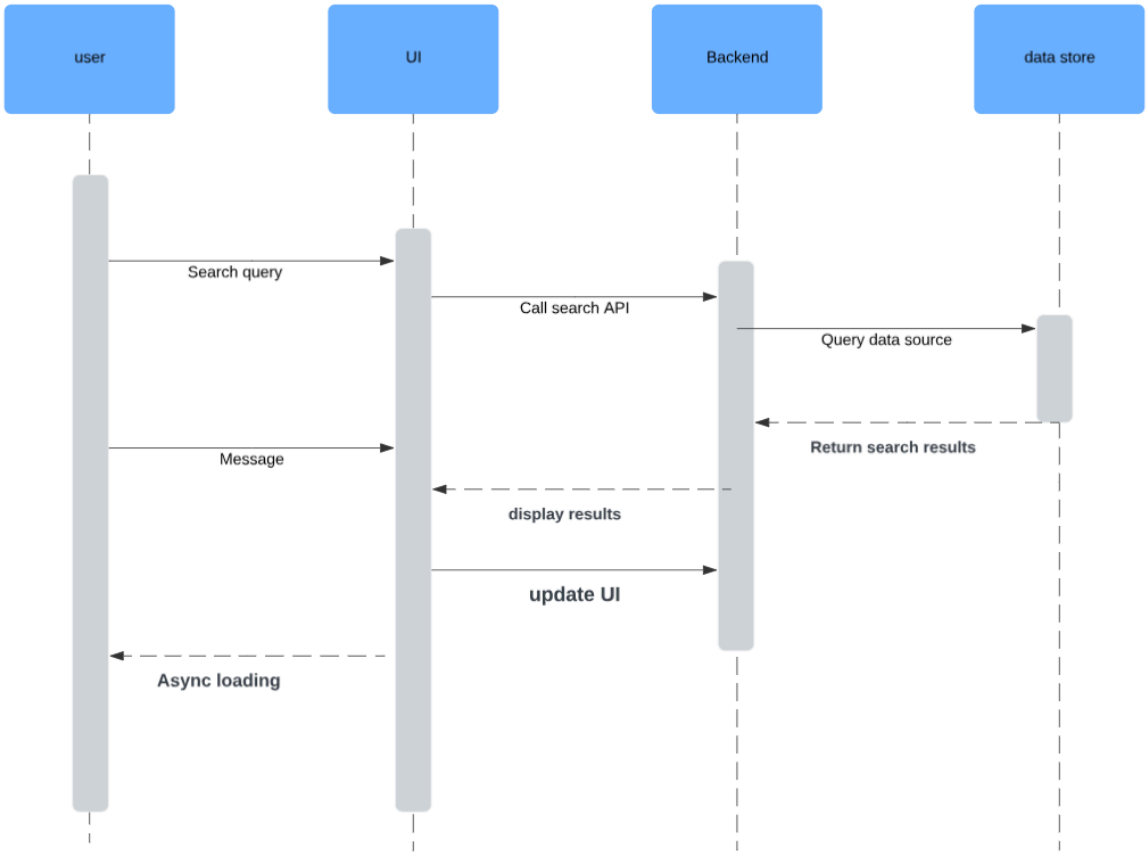
Example Scenario: Specifically, a user installing Snapster on a smart phone with a slow network connection starts a search for video files. The layout feature is adaptive, hence enables the user interface to fit the small screen appropriately while the dynamic content loading mechanism enables the search results to be delivered in chunks without requiring the page to be refreshed. The system minimizes such functions of graphical interface, which do not affect the function's utilization and thereby improves real-time result rates for the users.

Artifacts Implementation:

1. **User Interface Storyboard:** This will show how changes are made to the UI upon transition from one size to another as well as how the content changes depending on the available network bandwidth.



2. **Sequence Diagram:** The particular attention will be paid to dynamic content loading and asynchronous updates during the search operation, so the sequence diagram will illustrate the client and UI components interacting with backend services.



2. Progressive Feedback Mechanism

Overview and Purpose:

A Progressive Feedback Mechanism gives users long-term feedback about their actions in real-time. It also increases usability by letting users know the status of the tasks performed on the program like search and download of files and making connections to other peers. Clear feedback makes the users feel that they are in charge and that makes them less frustrated especially when performing extended or complicated processes.

Detailed Implementation:

Loading Indicators and Progress Bars: Some aspects like paying attention to are – Snapster loading search icon, and Loading icons -> searching… or Downloading… bar with icon. These indicators will provide a real-time progress bar in form of the percentage of the search or download, the estimated time remaining for the search or download, and the current rate of the download.

Status Messages and Notifications :The system will offer contextual status messages as like 'Searching for peers…' or "Changing the source of download..". These messages will be displayed bright for users to notice them when they are informing them of ongoing processes. Notifications will appear in form of pop-ups that will notify users when downloads are done, paused or interrupted.

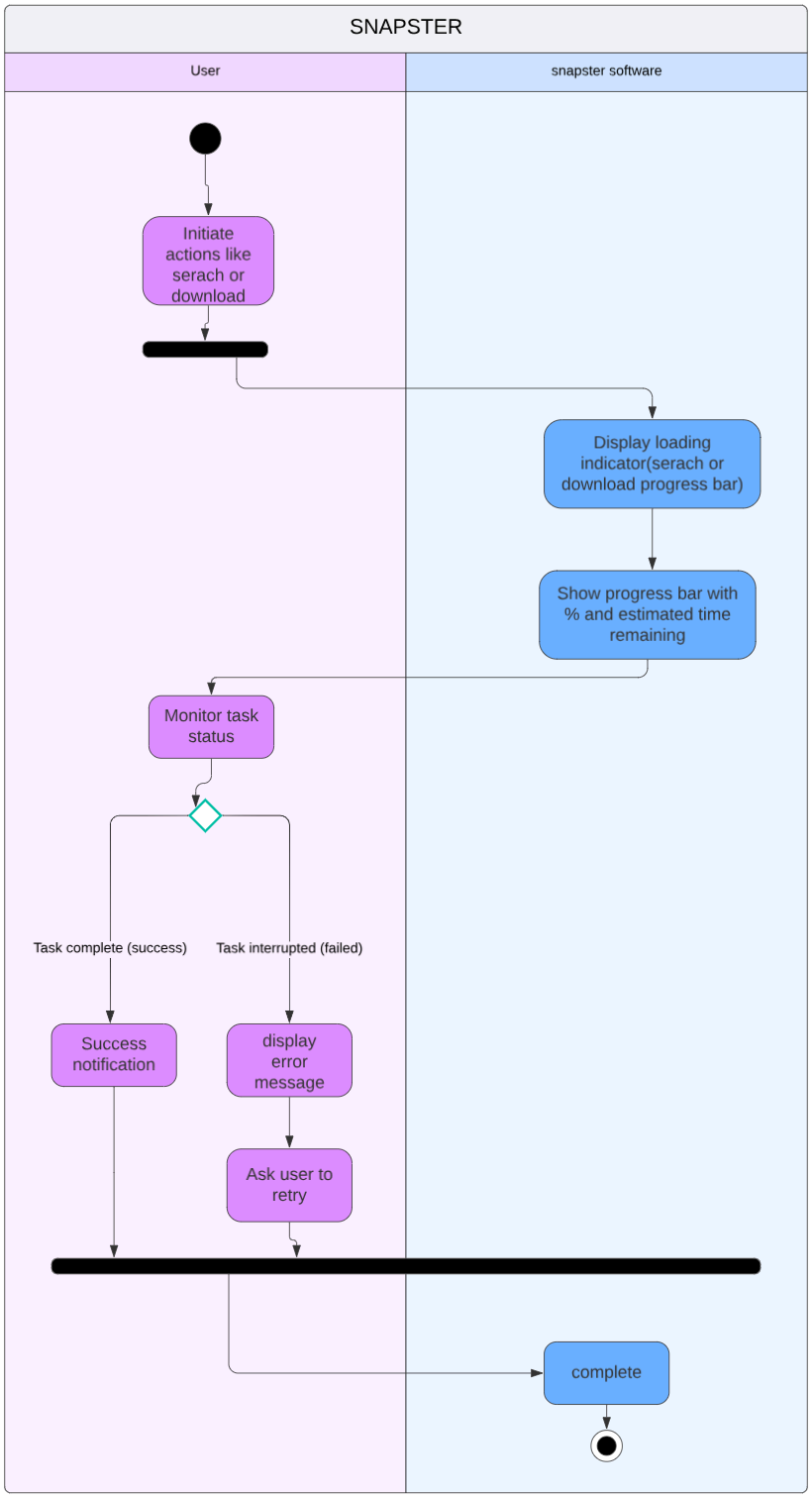
Interactive Error Handling: During the download failure or connection breakdown, Snapster will display sensible and definite error messages. Apopia, users will be prompted to retry, to download from another source, or to contact help desk for support. This means that the users will not be left in the dark trying to figure out what next when mistakes occur.

User Customization Options: App users will be able to control the kind of feedback they wish to receive including disabling specific notifications or whether they want detailed or brief progress alerts. This makes it possible for the users to augment the feedback depending on the extent of their needs and desires.

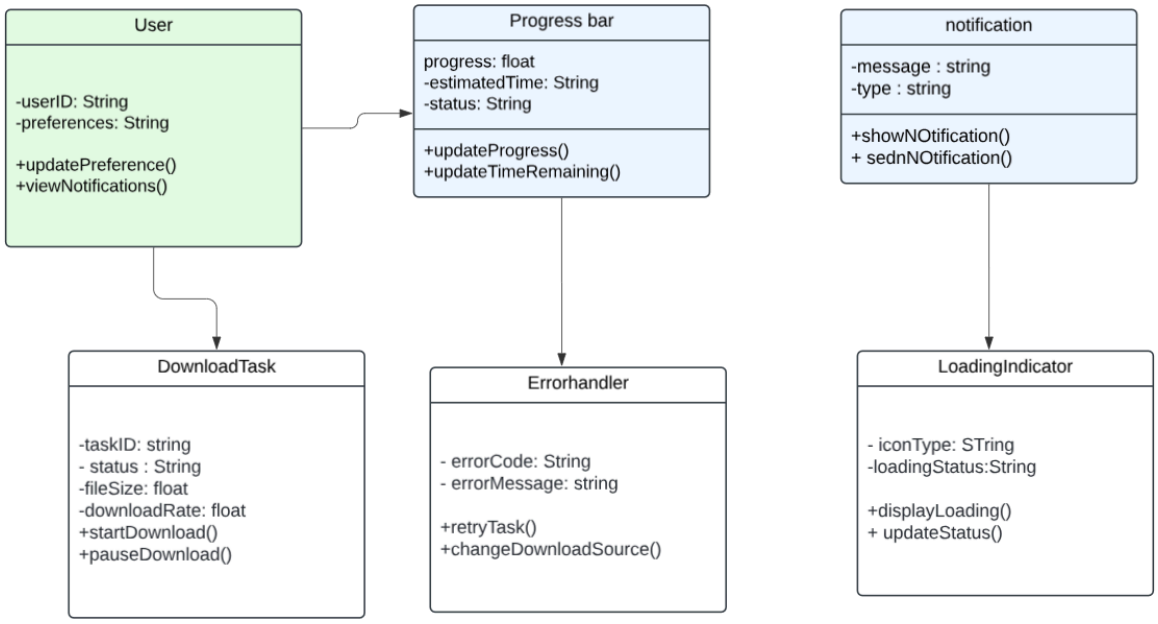
Example scenario: A user begins to download a large file of audio type. Snapster shows the progress bar with the view of 45% of the file downloaded and the estimated time left 2 mins. Half way through the download one of the peer nodes shuts down. Above the progress bar there is a status message "Switching download source…", the bar stops and then continues immediately. In one second, downloads resume right away, and a notification pops up to let the users know the switch happened; this also increases customer trust in the system.

Artifacts Implementation:

1. **Activity Diagram:** Activity diagram will contain further specification of the feedback loop: Loading activity indicators; updates on the status of the search and download; and error messages.


SNAPSTER
User
snapster software
Initiate actions like serach or download
Display loading indicator(serach or download progress bar)
Show progress bar with % and estimated time remaining
Monitor task status
Task complete (success)
Task interrupted (failed)
Success notification
display error message
Ask user to retry
complete

2. **Class diagram:**



Conclusion:

This interaction improvement is achieved with the incorporation of Snapshot's Responsive UI Design and the Progressive Feedback Mechanism in the usability of the Snapster software. These tactics allow users to find out the location of files to download and make appropriate feedback, as well as to have a smooth interaction with the program through various devices and various networking environments. Both tap into and help create a usability system that makes users happy and willing to continue using the given system.

17 Config Management Overview & QA Metrics

The team can monitor, manage, and control changes to software artifacts while preserving consistency, traceability, and reproducibility thanks to Software Configuration Management (SCM), which makes sure that every part of the software development lifecycle is handled methodically.

17.1 Configuration Identification

Configuration identification entails identifying and classifying all project components, objects, and aspects that are under configuration control.

- Configuration Items(CIs): CIs are project components that are handled using SCM to assure version control, consistency, and traceability.
    A. Source Code: It is the major application code and also includes various supporting scripts.
    B. Documentation: This includes the requirements documentation, design documentation, user manuals and technical documentation.
    C. Test Artifacts: These involve the test cases, scripts, test data and the results of both manual and automated testing.
    D. Executable and Deployable artifacts: These are the compiled binary codes and configuration files for deployments.
    E. Development and Build Tools: This includes the configurations for the Integrated Development Environments and Continuous Integration/ Continuous Deployment pipelines.
    F. Third-Party Dependencies: These include the libraries, frameworks and APIs used. These typically involve the dependency manifesting files as well.
    G. Task Artifacts: These are issue trackers that store backlog items, problem reports, and feature descriptions.
- Versioning Identification Strategy: To make software artifacts traceable and reproducible, versioning entails giving distinct IDs to various configuration item states. Clarity and uniformity are guaranteed by a well stated approach. Semantic versioning (major.minor.patch) and personalized versioning systems adapted to particular project requirements are often employed techniques.
    a. Semantic Versioning(Preferred)
        - Major: Increased if modifications are made that are incompatible with the past.
        - Minor: Backward-compatible new features are incremented.
        - Patch: Backward-compatible bug fixes are incremented.

        Ex: v2.3.1(Major Version2, Minor version3, patch1)

    b. Branching and Tags in Version Control Systems
        - Main Branch: Includes code that is reliable and suitable for production. Ex: main or master
        - Development Branch: Tracks current work before merging it into the main branch. Ex: develop
        - Feature branches: Isolated development of certain traits. Ex: feature/add-auth
        - Release Branches: Creates a release-ready stable build. Ex: release/2.3.0
        - Hotfix Branches: Identifies and addresses major production concerns. Ex: hotfix/2.3.1
        - Tags: Immutable markings are applied to commits. Ex: v2.3.1

c. File Naming Conventions: Add version markers to file names. Ex: user_guide_v1.2.0.pdf
d. Version Control System Metadata: To monitor individual changes, use commit hashes or the version control system's unique IDs.

17.2 Configuration Control

Configuration control is the methodical process of controlling modifications to software assets, such as test cases, documentation, source code, and related configurations. The major aspects include:

1. Change Request Management: It is the formal process for suggesting, evaluating, and approving modifications.
2. Versioning and Archiving: It includes utilizing version control systems such as Perforce, SVN, or Git to guarantee that each change can be tracked down.
3. Impact Analysis: It is the assessment of a change's possible impacts before to implementation.
4. Audits and Reviews: It performs routine inspections to confirm adherence to project baselines and requirements.

   Baseline Change Board workflow(BCB): Proposed modifications to existing baselines are overseen by the Baseline Change Board, which makes sure the right parties examine and approve the changes. The states include:

   a. Proposed: The change, its justification, and the components that will be impacted are all included in a change request (CR).An initial priority and severity are assigned to the task.
   b. Under Review: Stakeholders assess the CR's commercial, operational, and technological ramifications.This might entail the submitter making further changes or clarifications.
   c. Approved: The modification is approved for use. Baselines and documentation have been updated appropriately.
   d. Rejected: Technical impossibility, aim misalignment, or inadequate rationale are the reasons for declining the CR.
   e. Implemented: The modification is implemented and checked against the acceptance criteria.

The **key stakeholders and governance** for this includes:

A. Project Manager: They guarantee adherence to the project's goals.
B. Configuration Manager: They oversee the application of baseline updates and authorized modifications.
C. Technical Leads: They perform technical evaluations and determine viability.
D. Quality Assurance: They confirm adherence to the criteria.
E. End Users: They contribute to user-focused modifications.

Anomaly Resolution Board Workflow(ARB): Defects or problems reported throughout the software lifecycle are handled by the Anomaly Resolution Board. It guarantees the effective and efficient resolution of abnormalities. The workflow includes:

a. Reported: Enough information is recorded about an anomaly, such as the surroundings, observed behavior, and how to replicate it.
b. Triaged: Priority and severity are determined by how they affect users, functioning, or scheduling. Anomalies are divided into groups.
c. Assigned: The task is assigned to a team or developer.
d. In Progress: The anomaly is either being looked into or fixed.
e. Resolved: A solution is put into practice, examined, and authorized.
f. Closed: After being confirmed by QA or stakeholders, the anomaly is declared as complete.

Severity and Priority

Severity: It displays the anomaly's technical impact. The subcases are:

a. Critical: This causes a major failure or system crash.
b. High: This has a significant functional impact without workarounds.
c. Medium: It needs workarounds or has minimal functionality effects.
d. Low: This means cosmetic problems or minor flaws.

Priority: It shows how urgent a resolution is. The stages are:

- Immediate: This must be decided in order to proceed with development.
- High: This has to be fixed before the following milestone.
- Normal: The resolution can be planned for next sprints.
- Low: This can be resolved as soon as resources permit.

The **key stakeholders and governance** involved here is:

a. Anomaly Manager: They oversee the process of resolving anomalies and make sure they are dealt with quickly.
b. Development Team: The team looks into and fixes the abnormalities that have been assigned.
c. Quality Assurance: They resolve anomalies and validate fixes.
d. Product Owner: They offer suggestions on importance and severity from a business standpoint.
e. Support Team: They help to record and replicate oddities.

17.3 Configuration Status Accounting

The process of documenting and reporting the present state of software configuration items (SCIs) and the modifications related to them is known as configuration status accounting, or CSA. It ensures traceability and adherence to project requirements by offering an auditable history of modifications. The major action it takes is to keep a thorough log of the system's setup at all times, assist with the effect-analysis of suggested modifications, participate in evaluations and audits.

CSA Strategy

1. Artifact Identification: To guarantee traceability, each software artifact—such as code files, documentation, and test cases—is given a unique identification. Artifact Name, Version Number, Author/Contributor, Associated Change Request, or Defect ID are some of the identifiers.
2. Version Control: A version-controlled repository, such as Git or SVN, contains all configuration elements. Authors, timestamps, version numbers, and modifications are all recorded in the repository.
3. Status Tracking: Every configuration item's status is monitored by a centralized system. Typical states consist of:
   Draft: First draft, not yet approved.
   Reviewing: Awaiting permission.
   Approved: Baselined and reviewed.
   Outdated: Taken over by more recent iterations.
4. Reporting and Communication: Reports are produced on a regular basis to give updates on the baseline as of right now and any modifications since the prior update, the status of pending change requests, configuration items have known problems.

Key Attributes/Artifacts

1. Change logs: These are thorough documentation of modifications that include what was altered, why, by whom, and when.
2. Baselines: Fixed setups used as benchmarks for additional deployment or development.
3. Change requests (CRs): These are official records of changes that are sought, together with an effect analysis and approval.
4. Configuration Items (CIs): Software artifacts (documents, executables, and source code) that are tracked.
5. Version histories: These are lists of all a CI's prior iterations, together with any variations between them.

Sample CSA log

Date	Artifact	Version	Author	Change Description	CR/defect ID	Status	Approver
11-10-2024	main.py	v1.1	N.Megs	Modified for reusability	CR-1001	Approved	T.Sahithi
11-11-2024	user.docx	v2.0	J.Hope	Updated to add a new feature	CR-1002	Approved	T.Smith
11-12-2024	config.yaml	v1.0	Z.Bill	Configuration Setup	N/A	Baseline	T.Smith

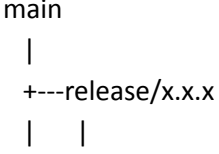
17.4 Version Control

By allowing teams to handle code changes in a methodical manner, version control guarantees that developers may work on features concurrently, monitor changes, and keep the core codebase stable. The principal goals are to keep an accurate record of all modifications (commits, merges, tags), facilitate effective teamwork among members, establish a process for settling disputes and undoing modifications, and make sure that bug fixes and releases are traceable and reproducible.

Git will be used for this project because of its powerful features:

Distributed Model: Facilitates offline cooperation and work.
Branching: Supports several workflows (such as Gitflow and trunk-based development) and is lightweight.
Integration: Compatible with CI/CD, issue tracking, and hosting solutions such as Bitbucket, GitLab, and GitHub.
Community Support: extensively used and well-documented.

Branching Structure



```
|    +---- develop
|            |
|            +---- feature/story-mode
|            +---- feature/avatar-customization
|
+--- hotfix/urgent-fix
```

Repository Structure

```
root/
|----app/
|    |----filters/              # Custom filter features
|    |----camera/               # Camera recording functionality
|    |----chat/                 # Direct messaging/chat system
|    |----avatars/              # Avatar customization
|    |----social/               # Friend management and ranking system
|    |----downloader/           # Decentralized file-sharing logic
|----services/
|    |----points/               # Point system
|    |----ads/                  # Advertisement delivery
|    |----eula/                 # Legal compliance modules
|    |----ranking/              # User ranking algorithm
|----tests/                     # Unit, integration, and performance tests
|----scripts/                   # Automation and deployment scripts
|----docs/                      # Documentation and EULA details
|----config/                    # Configuration files for the app
|----assets/                    # Static Assets
```

17.5 Configuration Auditing

Configuration auditing ensures that all deliverables are in line with the project's configuration management strategy and that SNAPSTER's software and procedures follow established standards. This procedure confirms that all project artifacts are present and consistent across versions, that allowed modifications are applied successfully, and that no unauthorized changes are included.

The checklist for configuration auditing can include:

1. Configuration Identification: Check to make sure that every item—code modules, papers, filters, etc.—is properly named and versioned. Make that the version history of the core modules, filters, and content is preserved.
2. Baseline verification: Verify that all baselines, such as release versions, have been authorized and recorded. Verify that there are no alterations outside of the baseline.

3. Change Management Validation: Verify that every modification has a corresponding change request or issue ticket. Make sure that any modifications to filters, ad modules, or caching systems are approved and recorded.
4. Dependency Tracking: Make sure all dependencies are well described, including filter updates and decentralized download methods. Verify compatibility between legacy and upgraded components.
5. Build Validation: Verify that builds include the authorized versions of settings, libraries, and source files.
6. Testing Documentation: Verify that the integration, performance, and functional tests were successfully completed following significant upgrades.
7. Artifact Review: Verify that all necessary artifacts, such as EULAs, filter update logs, and user manuals are current.
8. Compliance Check: Verify that ads and legal disclaimers (such as the EULA for copyright) adhere to legal requirements.

Frequency of configuration audits: To guarantee the system's integrity, SNAPSTER configuration audits will take place on a regular basis and during significant events:

1. Regular Audit Schedule
   Audits every month: A regular audit to guarantee ongoing adherence to change management and configuration protocols.
2. Event-Triggered Audits
   a. Significant Software Updates: while introducing new features (such updated avatars or narrative filters) following major modifications to the decentralized download system.
   b. Hotfix or Patch Deployments: favor fast updates, particularly those that include speed or security enhancements.
   c. Fresh Marketing Initiatives: to guarantee that sponsor-provided material is properly integrated without interfering with user experience.
   d. Launch of New Features or Filters: Check that new features and filters are implemented from the corporate repository in accordance with the guidelines.
   e. Updates on Law or Compliance: Make sure that the system as a whole reflects any changes made to EULAs or compliance warnings.
   f. Final Milestone Inspections: prior to the end of any stage of development.

17.6 Key Software Metrics

Process Metric: Issue Resolution Time

It calculates the typical amount of time needed to fix issues that have been reported, such as bugs, crashes, or feature requests.

Frequency: Weekly

Rationale: Assists in monitoring the development team's effectiveness and guarantees that serious problems don't impair the user experience or application performance.

Threshold
Typical: ≤ 5 days.
Caution: 6–10 days.
Critical: >10 days
Response: Start a root cause investigation to find bottlenecks if the threshold is more than ten days. Give high-priority topics more resources, and let stakeholders know about any modified timetables.

Project Metric: Active User Engagement
It monitors the quantity of distinct active users who engage with the application on a weekly and monthly basis (e.g., story views, downloads, uploads, and logins).
Frequency: Monthly and weekly
Rationale: This has a direct impact on advertising income and in-app purchases by indicating the app's popularity, user retention, and general health of the user base.
Threshold
Normal: > 80% of monthly active users are retained each week.
Warning: 60–79% weekly retention
Critical: <60%
Response: Use behavioral analytics or user surveys to examine the causes of disengagement below 60%. To increase user participation, add gamification components or improve features like customized avatars.

Product Metric: Parallel Download Efficiency
It tracks the average speed and success rate of parallel downloads (i.e., the proportion of downloads that finish within the anticipated speed and time limit).
Frequency: Daily, with weekly summaries
Rationale: It provides seamless content distribution, particularly considering SNAPSTER's decentralized file sharing methodology. App dependability and user happiness are directly impacted by this statistic.
threshold
Normal: > 75% of anticipated bandwidth with an average speed success rate of 90%.
Caution: success rate of 80–89% or speeds over 50% but less than 75%.
Critical: speeds < 50% or success rate < 80%.
Response: Run peer-to-peer connection protocol diagnostics below 80%. Implement solutions to network latency problems and alert consumers to anticipated service outages.

17.7 Key QA Metrics

1. **Defect Density**
   It calculates the quantity of errors in a given unit of code, such as 1,000 lines of code or a feature.
   Frequency: During the testing phase, the data is gathered at the conclusion of each sprint and aggregated at the next release cycle.
   Rationale: A vital sign of the dependability and caliber of software.

Threshold: When the number of faults per 1,000 lines of code surpasses five, a programmed reaction is initiated.

Response: We need to carry out a root cause analysis (RCA) to pinpoint the origins of flaws (such as insufficient specifications or inadequate test coverage), adding more thorough unit and integration tests will be one way to do this.doing focused code reviews also helps and we can modify the method of development to reduce such problems.

2. **Mean Time to Resolution(MTTR)**

   It calculates the typical time needed to fix problems or flaws that have been found.

   Frequency: Taken from the bug-tracking system and averaged per week.

   Rationale: Shows how quickly the team fixes issues and lessens their impact on users.

   Threshold : A programmed response is initiated if the MTTR for critical defects above 48 hours.

   Response: Inform the management team of any outstanding problems, make sure developers allocate top priority resources to fixing important bugs, and provide patches as soon as possible.

3. **Test Coverage Ratio**

   It is the proportion of code that is subjected to automated testing.

   Frequency: Monitored cumulatively and assessed at the conclusion of each sprint.

   Rationale: Assures that crucial pathways and features (such as caching, filter updates, and decentralized file-sharing) are thoroughly tested.

   Threshold: Report the problem to the QA lead if test coverage is less than 80%.

   Response: Require more automated test authoring and concentrate on high-priority areas like cache management and file transfer.

18 System Level Test Plan

TC1.1:

Test Case: Login with user credentials		
Goal: Ensures that a user can successfully login with valid details. If invalid details are entered the system need to reject it.		
Impacted Modules: Login Module, Authentication Service		
Pre-Conditions/Dependencies: 1. User is on login page and has access to the internet. 2. User has an existing account.		
Test Sequence: Open the login page. Enter correct details such as username, password and email.		
Step	**Expected Result**	**Actual Result**
1. Go to the login page and enter a valid email and password.	User is redirected to the dashboard	PASS
2. Enter an invalid email that is not registered.	An error message is shown indicating an invalid email.	FAIL
3. The email entered is correct but the user enters an incorrect password.	An error message is shown saying that the password is incorrect.	FAIL
Final Conditions: If the entered details are correct, the user account is logged in and navigated to the dashboard. If the user details are invalid the user remains in the login page and is prompted to enter a valid email or password.		

TC1.2

Test Case: Register with user credentials		
Goal: Ensures that a user can successfully register with valid details. If invalid details are entered the system need to reject it.		
Impacted Modules: Registration Module, Email Validation		
Pre-Conditions/Dependencies: 1. User is on registration page and has access to the internet. 2. User does not have existing account.		
Test Sequence: Open the registration page. Enter details such as username, password and email.		
Step	**Expected Result**	**Actual Result**
1. Go to the registration page and enter a valid email and password.	User is redirected to the login page	PASS
2. In the registration page, enter an email in invalid format.	An error message is shown indicating an invalid email.	FAIL
Final Conditions: If the entered details are correct, the user account is created and stored in the database. If the user details are invalid the user remains in the registration page and is prompted to enter a valid email.		

TC 2

Test Case: Logout from the user account		
Goal: Ensures that a user can successfully logout of their account.		
Impacted Modules: Logout Module, Authentication Service, Session Management		
Pre-Conditions/Dependencies: 1. User is on login page and has the logout option. 2. User has a valid session running.		
Test Sequence: User is logged in to their account. The user clicks the logout option.		
Step	Expected Result	Actual Result
1. Go to the page where user is logged in and click the logout button.	User is redirected to the login page.	PASS
2. The user tries to open a page that needs authentication.	An error message is shown stating that the user cannot view this page.	FAIL
Final Conditions: The user is successfully logged out. The protected pages cannot be accessed after logging out.		

TC 3

Test Case: Search Content		
Goal: Ensures that the system returns valid results when the user searches for something		
Impacted Modules: Search Module, Content Database, Search Algorithms		
Pre-Conditions/Dependencies: 1. User is logged in and is on the dashboard. 2. The ability to search content is present in the dashboard.		
Test Sequence: User is on the search page. The user enters valid words to perform their search.		
Step	Expected Result	Actual Result
1. Go to the page where search option is enabled and enter valid keywords to search.	The system returns a list of relevant content.	PASS
2. User enters invalid keywords or special characters in the search bar.	An error message is shown stating that the user needs to enter the complete keyword.	FAIL
Final Conditions: The user can see the relevant content they searched for. The display shows a prompt explaining the special characters or shows slightly matching content.		

TC 4

Test Case: Record Content		
Goal: Ensures that the user can record content in the form of an article, blog or post.		
Impacted Modules: Content Creation Module, Text Editor, User Input Validator		
Pre-Conditions/Dependencies: 1. User is logged in and has access to the content creation page.		
Test Sequence: User is on the content creation page. The user enters valid text or incomplete input.		
Step	**Expected Result**	**Actual Result**
1. Enter valid text into the content creation field for the tile and body sections. Click Save and Publish to save the content.	The system saves the content and displays a success message.	PASS
2. The user enters random characters and invalid input into the content creation form.	An error message is shown stating that it is invalid input and the content cannot be saved.	FAIL
Final Conditions: The user can see if the content is saved or not based on the text they enter in the fields.		

TC 6

Test Case: Update Filters with specific criteria		
Goal: Ensures that the user can update filters with valid filter criteria.		
Impacted Modules: Filter Module, User Interface		
Pre-Conditions/Dependencies: 1. User is logged in and has access to the content creation page.		
Test Sequence: User is on the filters page. The user can update the filter criteria.		
Step	**Expected Result**	**Actual Result**
1. Choose a valid filter from the available options by looking into the data and price range. Click on apply button to enable to filter.	The content displayed while recording is updated with this new filter addition.	PASS
2. Select a filter that does not fit the criteria for suppose like a price range between $1000-$2000 which does not exist in our application.	The system displays an error message stating that "No results found"	FAIL
Final Conditions: The user can see the updated content or sees the error message based on the input they give.		

TC 7

Test Case: Share Content		
Goal: Ensures that the user can share content with various applications such as Facebook, Twitter.		
Impacted Modules: Sharing Functionality, Social Media Integration		
Pre-Conditions/Dependencies: 1. User is on the content sharing page. 2. The user can see the social media sharing options.		
Test Sequence: User views the content to share. The social media option is also selected by the user.		
Step	Expected Result	Actual Result
1. The user navigates to the content page and selects the social media applications such as Facebook, Twitter to share this content.	The content is displayed on social media in the form of a post or story.	PASS
2. The user selects the content and clicks on the share via email option.	The receiver will receive the content along with the shared link.	PASS
3. The user can share the content via a direct URL by clicking on the copy link option.	The link displays the content when it is clicked.	PASS
Final Conditions: The user can access the content shared to them by various available options.		

TC 8

Test Case: View all available stories		
Goal: Ensures that the user can view a list of available stories posted online.		
Impacted Modules: Story Listing Module, Content Display		
Pre-Conditions/Dependencies: 1. User is on the page where stories are listed.		
Test Sequence: User navigates to the stories section. User views the available stories.		
Step	Expected Result	Actual Result
1. The user navigates to the stories page and views all the available stories one by one.	All available stories are displayed along with proper metadata.	PASS
2. The user applies filters such as date and time to view the latest stories.	The updated list of stories will be displayed.	PASS
Final Conditions: All stories are displayed properly with every single detail.		

TC 9

Test Case: Download Stories in various formats.		
Goal: Ensures that the user can download the story in various formats like pdf, mp4 etc.		
Impacted Modules: Story Download Functionality, File Generation, Format Selection		
Pre-Conditions/Dependencies: 1. User is on the stories page and the story is available for download in multiple formats.		
Test Sequence: User selects the download format. User initiates the download for selected format.		
Step	Expected Result	Actual Result
1. The user clicks on the download option and selects the required format and wait for the download to complete.	The content is downloaded correctly in the appropriate format.	PASS
2. The user does not have access to internet and initiates the download.	An error message is displayed stating that there is no proper connection to perform download.	FAIL
Final Conditions: The story is either downloaded or interrupted based on the internet connection.		

TC 10

Test Case: Cache Stories		
Goal: Ensures that the user can cache their stories for offline viewing.		
Impacted Modules: Caching Functionality, Offline Story Access		
Pre-Conditions/Dependencies: 1. User has an active internet connection while caching stories.		
Test Sequence: User accesses the story when they have an active internet connection. They do not need a connection once the stories are cached.		
Step	Expected Result	Actual Result
1. The story to be cached needs to be properly loaded. Cache it while we have an internet connection. Restart the application and disconnect the internet. Check if the story is accessible or not.	The content that is cached can be seen even without an internet connection.	PASS
2. Check the cache expiry period. See if the cache space is full and whether the old stories are removed or not.	Stories are automatically removed based on the cache expiration policy.	PASS
Final Conditions: The cached stories are accesses online and some stories are deleted from time to time.		

TC 11

Test Case: Manage Friends		
Goal: Ensures that the user can add or remove friends, accept or ignore requests.		
Impacted Modules: User Management, Friend Request Functionality		
Pre-Conditions/Dependencies: 1. The user can search for people and request them in order to make friends.		
Test Sequence: The current user searches for another user and performs their required operation like adding, removing on them.		
Step	**Expected Result**	**Actual Result**
1. The user can search for another user and select the Add Friend or Remove Friend button and wait for the request to be sent.	The request is sent successfully if we want to add or the friend is removed.	PASS
2. In the friend requests section, we can see the user's profile and click on the Accept Request or Reject Request based on our interest.	The friend request is accepted successfully or rejected successfully based on the user's choice.	PASS
Final Conditions: The friends list is updated from time to time.		

TC 12

Test Case: Message		
Goal: Ensures that the user can send or receive a message in various formats.		
Impacted Modules: Message fucntionality		
Pre-Conditions/Dependencies: 1. The user needs to have a proper internet connection to send or receive messages.		
Test Sequence: The user types the message or selects and uploads the format they want to send.		
Step	**Expected Result**	**Actual Result**
1. The user types a message and clicks on the send option with a proper internet connection.	The message is successfully sent to the user and the same wise it can be received as well.	PASS
2. The user records a voice message they want to send but with no proper internet connection. The user now clicks on send button.	The message is not sent and the system displays an error message showing no proper connection.	FAIL
Final Conditions: The messages can be either successfully sent or received or not sent based on the internet connection.		

TC 13

Test Case: Change Avatar		
Goal: Ensures that the user can change their avatar on their profile.		
Impacted Modules: User Profile Management, Avatar Upload functionality		
Pre-Conditions/Dependencies: 1. User is logged in and has a valid image to upload as avatar.		
Test Sequence: The user selects an image to upload. The avatar gets modified.		
Step	Expected Result	Actual Result
1. The user selects the upload image option and selects an image from the device storage(JPG or PNG). This image is displayed as the new avatar.	The new avatar image is successfully shown in the profile.	PASS
2. The user selects an image in the BMP format. The user clicks on the upload option to modify the avatar.	The system does not upload the image and displays an error message.	FAIL
Final Conditions: The avatar is modified based on the format of the image selected.		

TC 14

Test Case: Earn Points		
Goal: Ensures that users can earn points when they complete a task or any activity in the system.		
Impacted Modules: Task Management, Point Calculation		
Pre-Conditions/Dependencies: 1. User is assigned a task and has access to the tasks and activities in the system.		
Test Sequence: User completes the tasks. The system rewards points to the user.		
Step	Expected Result	Actual Result
1. The user can access the activity section, complete the task and verify the system rewards them.	The user earns the points after completing the task.	PASS
2. The user can access the activity section without proper internet connection.	The system completes the task but does not reward the user due to a lack of internet connection.	FAIL
Final Conditions: The points are rewarded based on the activity and the internet connection.		

TC 14.1

Test Case: Buy Points		
Goal: Ensures that users can buy points by making a transaction using their card.		
Impacted Modules: Payment Integration, Point Calculation		
Pre-Conditions/Dependencies: 1. User is logged into the application and has a valid payment method linked to their account.		
Test Sequence: User selects the desired number of points to purchase. The system completes the transaction using the card.		
Step	**Expected Result**	**Actual Result**
1. The user navigates to the buy points section, the user can choose a payment method and the user's point balance is updated.	The points balance is increased after a successful transaction.	PASS
2. The user tries to make a payment but the card details are expired or invalid.	The system does not identify the payment and does not modify the points balance of the user.	FAIL
Final Conditions: The points are rewarded based on the activity and the payment method followed.		

TC 15

Test Case: Display Advertisements		
Goal: Ensure that advertisements are displayed correctly throughout the platform at appropriate intervals and locations.		
Impacted Modules: Home Screen, Advertisement Display Module		
Pre-Conditions/Dependencies: 1. User is logged into the platform. 2. The system is connected to the advertisement service. 3. The platform has active advertisements.		
Test Sequence: Test that ads are displayed on the home screen, Test that ads remain visible as the user scrolls, Test user interaction with ads (clicking), Test that ads disappear when skipped.		
Step	**Expected Result**	**Actual Result**
1. User navigates to the Home Screen	Ads are displayed at appropriate intervals on the screen	PASS
2. User scrolls through content on the Home Screen	New ads appear as the user scrolls	PASS
3. User clicks on an ad	User is redirected to the destination of the ad	PASS
4. User skips an ad	The ad disappears, and the user continues interacting with the content	PASS
Final Conditions: Ads are displayed and intractable as expected, The user is redirected properly when they click on an ad, Ads disappear when skipped, and no errors are encountered.		

TC 16

Test Case: Display Copyright Warnings		
Goal: Ensure copyright warnings are displayed properly when users upload or play copyrighted content.		
Impacted Modules: Content Upload, Content Play Module, Copyright Warning System		
Pre-Conditions/Dependencies: 1. User is logged into the platform. 2. The system has a working copyright detection mechanism. 3. User uploads or plays content that is copyright-protected.		
Test Sequence: Test copyright warning when uploading content, Test copyright warning when playing content, Test user interaction with the warning, Test if the system blocks or allows content based on the warning.		
Step	**Expected Result**	**Actual Result**
1. User uploads copyrighted content	A copyright warning is displayed during the upload	PASS
2. User plays copyrighted content	A copyright warning is displayed before the content starts playing	PASS
3. User clicks on the copyright warning	A detailed explanation of the copyright issue is shown	PASS
4. User ignores or accepts the warning	User is either blocked from publishing the content or allowed to proceed	PASS
Final Conditions: Copyright warnings are shown in the appropriate scenarios, and users are informed about the status of the content, Content is blocked or allowed based on the copyright status.		

Test Case: Manage Advertisements		
Goal:Ensure that administrators can manage (create, edit, delete) advertisements on the platform.		
Impacted Modules: Admin Dashboard, Advertisement Management Module		
Pre-Conditions/Dependencies: 1. Admin is logged into the system. 2. The system has an active advertisement management interface.		
Test Sequence: Test admin login and access to the ad management interface, Test ad creation, Test ad editing, Test ad deletion.		
Step	**Expected Result**	**Actual Result**
1. Admin logs into the Admin Dashboard	Admin is redirected to the dashboard	PASS
2. Admin clicks on "Manage Advertisements"	Ad management interface loads successfully	PASS
3. Admin creates a new ad	The new ad is saved and displayed to users	PASS
4. Admin creates a new ad	The changes are saved, and the ad is updated accordingly	PASS
5. Admin deletes an ad	The ad is removed from the system and no longer displayed to users	PASS
Final Conditions: Admin can create, edit, and delete advertisements successfully, The system reflects changes made by the admin.		

TC 17

TC 18

Test Case: Add stories		
Goal: Ensure users can successfully add stories with media, text, and features.		
Impacted Modules: Story Upload, Media Processing, User Profile		
Pre-Conditions/Dependencies: 1. User is logged into the platform. 2. User has media (e.g., videos or images) available to upload		
Test Sequence: Test the story upload process. **Test media uploads (video/audio).** **Test adding text, filters, or stickers.** **Test story submission and publication.**		
Step	**Expected Result**	**Actual Result**
1. User clicks on "Add Story"	Story creation screen loads	PASS
2. User uploads media (video/image)	Media uploads without errors	PASS
3. User adds text or filters	Text and filters are applied and visible	PASS
4. User submits the story	Story is successfully published and visible to followers	PASS
Final Conditions: Stories are uploaded and displayed correctly. Users can add media, text, and apply filters or stickers as needed.		

TC 19

Test Case: Manipulation		
Goal: Ensure that users cannot manipulate or bypass the advertisement system.		
Impacted Modules: Advertisement Display, User Security and Integrity		
Pre-Conditions/Dependencies: 1. User is logged into the platform. 2. The platform is using a secure advertisement display system.		
Test Sequence: Test attempts to skip ads using developer tools. **Test manipulation of ad links.** **Test behavior with ad-blocking extensions.**		
Step	**Expected Result**	**Actual Result**
1. User tries to skip ads using developer tools	Ads cannot be skipped or bypassed	PASS
2. User tries to manipulate ad links	The system detects manipulation and blocks it	PASS
3. User uses an ad-blocking extension	Ads are still displayed or fallback ads appear	PASS
Final Conditions:The ad system prevents manipulation and bypassing. Users cannot interfere with ads through technical means.		

TC 20

Test Case: Unfiltered Content

Goal:Ensure the system identifies and blocks unfiltered or inappropriate content.

Impacted Modules: Content Moderation, Content Upload

Pre-Conditions/Dependencies:

1. User is logged into the platform.
2. The system has a content moderation system in place.
3. User uploads content that may be flagged as inappropriate.

Test Sequence: Test uploading unfiltered content.

Test system scanning and filtering content.

Test content rejection or approval.

Test modification or review of flagged content.

Step	Expected Result	Actual Result
1. User uploads unfiltered content (e.g., explicit material)	The system flags the content as inappropriate	PASS
2. The system scans content for inappropriate material	The filter system detects the unfiltered content	PASS
3. The system blocks or approves content based on the filtering mechanism	The content is either blocked or flagged for review	PASS
4. User modifies flagged content	A warning appears, and the content is either removed or reviewed	PASS

Final Conditions:Unfiltered or inappropriate content is detected and blocked. The system ensures the platform remains safe for users.

Trace table

Use Case ID	Use Case Name	Functional Requirement(s) Impacted	Non-Functional Requirement(s) Impacted	t Case(s) Refere
UC1	Register/Login	FR2.1 (Points system), FR5.1 (Search Content), FR3.5 (Notifications)	Performance, Usability, Security, Reliability	TC1.1, TC1.2
UC2	Logout	FR3.1 (Direct Messaging), FR4.3 (Ad Interactions)	Performance, Usability	TC2
UC3	Search Content	FR5.1 (Search content), FR5.2 (Filter search), FR5.3 (Categories search)	Performance, Usability	TC3
UC4	Record Content	FR1.1 (Uploading), FR1.5 (Notifications)	Performance, Security, Usability	TC4
UC5	Apply Filters	FR6.1 (Filters), FR5.2 (Filter search)	Performance, Usability	TC5
UC6	Update Filters	FR6.1 (Filters), FR5.2 (Filter search)	Performance, Usability	TC6
UC7	Share Content	FR1.2 (Downloading), FR1.4 (Privacy), FR2.1 (Points system)	Performance, Security, Usability	TC7
UC8	View Stories	FR6.2 (Order of stories), FR6.3 (Save stories)	Performance, Usability	TC8
UC9	Download Stories	FR1.2 (Downloading), FR3.4 (Secure chats)	Performance, Security	TC9
UC10	Cache Stories	FR6.5 (Track of stories), FR5.5 (Recommendations)	Performance, Availability, Usability	TC10

UC11	Manage Friends	FR3.1 (Direct Messaging), FR2.3 (User ranking)	Performance, Usability	TC11
UC12	Message	FR3.1 (Direct Messaging), FR3.4 (Secure chats)	Performance, Security	TC12
UC13	Change Avatar	FR2.2 (Redeem points), FR2.4 (Rewards)	Performance, Usability	TC13
UC14	Earn/Buy Points	FR2.1 (Points system), FR2.2 (Redeem points)	Performance, Usability	TC14, TC 14.1
UC15	Display Advertisements	FR4.1 (Ads), FR4.3 (Ad Interactions)	Performance, Usability	TC15
UC16	Display Copyright Warnings	FR4.4 (Ads relevance)	Usability, Security	TC16
UC17	Manage Advertisements	FR4.2 (Skip ads), FR4.3 (Ad Interactions)	Performance, Usability	TC17
UC18	Add Stories	FR1.1 (Uploading), FR5.1 (Search content)	Performance, Usability	TC18
UC19	Advertising Manipulation	FR4.1 (Ads), FR4.4 (Ads relevance)	Performance, Security	TC19
UC20	Unfiltered Content	FR1.3 (Parallel downloads), FR1.4 (Privacy)	Performance, Security	TC20

19. Glossary

Decentralized Architecture : A system design in which data and functionality are distributed across multiple nodes, enabling peer-to-peer (P2P) interactions without reliance on a central server.

Asynchronous Communication **:** Communication where messages or data exchanges occur independently of the sender's or receiver's time frame, commonly used in direct messaging (DM) or content sharing.

User Interface (UI) : User Interface (UI) is usually concerned with the interactive pieces that a user is able to manipulate in any application. It refers to all those things that are visible on a screen of a mobile application and which help the users to interact the app. These are real-time areas throughout which users interact, for example, through the “Home Screen” and the “Profile Screen,” whereby they view content and update their details, respectively.

User Experience (UX) : User Experience (UX) is defined in overall experience that a user has while using the app with emphasis on usability, and utility. It is very important that the next users of SNAPSTER find it easy to use and have fun while doing so, for instance when sliding from the "Feed" to the "Search Screen."

Points : It is similar to coins, levels, or-stars where the application offers them users as a way of encouraging the use of the application. They are awarded points depending on what they do like uploading contents, contributing to the application or achieving certain landmark. In SNAPSTER, it is possible to use points in a way that they let you gain or purchase self-avatars and other special options that can only be unlocked with these points, thereby extending the time users will spend with the application on their mobile phones.

Terms of Use License (TULA) : An EULA is a legal agreement between the vendor and the end-users of the application that state and frame the application usage and limitations. Like many similar programs, people will have to agree to comply with the EULA during the registration into SNAPSTER.

Application Programming Interface or more commonly known as API : Application Programming Interface (API) refers to a collection of specifications that define how computer applications should interact with one another. In SNAPSTER, API is also a tool for sharing various components such as implementing the user sign-up and login from the application, streaming content, and data management between the application and other services.

20. Bibliography

1. **"Designing Social Interfaces: Principles, Patterns, and Practices for Improving the User Experience" by Christian Crumlish and Erin Malone**
   Crumlish, Christian, and Erin Malone. *Designing Social Interfaces: Principles, Patterns, and Practices for Improving the User Experience.* O'Reilly Media, 2009.
   This book provided key insights into designing user interfaces for social media applications, focusing on the best practices for fostering user interaction, engagement, and community-building, which are integral to the SNAPSTER project.

2. **"Social Media Design For Dummies" by Janice S. Sutherland**
   Sutherland, Janice S. *Social Media Design For Dummies.* Wiley, 2012.
   This resource provided practical tips on designing interfaces for social media platforms, offering examples of how to structure user profiles, feeds, messaging systems, and interactive features, which were crucial in designing SNAPSTER's core features.

3. **"User-Centered Design: A Designer's Guide to Building Usability into Software" by Travis Lowdermilk**
   Lowdermilk, Travis. *User-Centered Design: A Designer's Guide to Building Usability into Software.* O'Reilly Media, 2013.
   This book provided essential information on designing intuitive, user-centered software, which was applied to SNAPSTER's interface, ensuring an easy-to-use experience for users managing content, profiles, and interactions.

4. **"The Art of Social Media: Power Tips for Power Users" by Guy Kawasaki and Peg Fitzpatrick**
   Kawasaki, Guy, and Peg Fitzpatrick. *The Art of Social Media: Power Tips for Power Users.* Penguin, 2014.
   This resource informed the social media features of SNAPSTER, offering strategies for promoting user engagement, building a content feed, and fostering community interaction, all of which were incorporated into the app's design.

5. **"Mobile App Development with Ionic Framework: Build Android and iOS Apps with Web Technologies" by Chris Griffith**
   Griffith, Chris. *Mobile App Development with Ionic Framework: Build Android and iOS Apps with Web Technologies.* O'Reilly Media, 2015.
   This reference was considered for integrating web technologies into SNAPSTER's mobile app, providing insights into hybrid app development and creating mobile applications that work across different devices and platforms.

6. **"Designing Mobile Interfaces: Patterns for Interaction Design" by Steven Hoober and Eric Berkman**
   Hoober, Steven, and Eric Berkman. *Designing Mobile Interfaces: Patterns for Interaction Design.* O'Reilly Media, 2011.

This book provided valuable patterns and guidelines for designing effective mobile user interfaces, particularly focusing on user touch interactions and designing easy-to-navigate layouts, which were applied in SNAPSTER's mobile user experience.

7. **"Understanding the User: Designing for Usability" by Jim Ross**
   Ross, Jim. *Understanding the User: Designing for Usability.* Wiley, 2015.
   This resource contributed to the SNAPSTER project by focusing on how to design with the user's needs in mind. It emphasized the importance of usability testing and user feedback, which were integral to the iterative development process of SNAPSTER.

8. **Lecture notes.**