Analysis

1. **Introduction to Analysis:**

The conversion method in which difficult task is split out into many more phases to simplify the task and make it understandable is known as Analysis. It can be regarded as one of the most important phase of software development. It is done to evaluate user demands and project goals. Firstly information is gathered, a feasibility study is carried out the user demands and project goals is maintained thoroughly. Also features of the system are estimated in this phase for both hardware and software component. In agile development methodology Analysis is done prior second to Planning project. Wholly speaking of Analysis there are still many processes to be carried out for the completion of analyzing phase in software development.

1. **Analysis methodology:**

Software development methodology can be described as the frame work which is used to plan, structure and control the process of developing system. My UMeChat project will follow the SDLC model throughout process of development. Software Development Life Cycle (SDLC) is a process used by the software organization in order to design, develop and test high quality soft-wares. Also I would be considering Soft System Approach. In this process engineering approach can be inappropriate with fuzzy requirements.

Advantage of Soft system Approaches:

1. Human factors important so given more priority.
2. It is a creative, intuitive approach to problem-solving.
3. Outcomes are learning, better understanding, rather than a ‘solution’.

Disadvantage of Soft system Approaches:

1. The organizational problems are ‘messy’ and poorly defined.
2. There are many problems to be solved.
3. It can have unpredictable, nondeterministic, non-definable complexity.

* **Data Flow Diagram:**

Data flow diagram is the graphical representation of the major phases of project process.

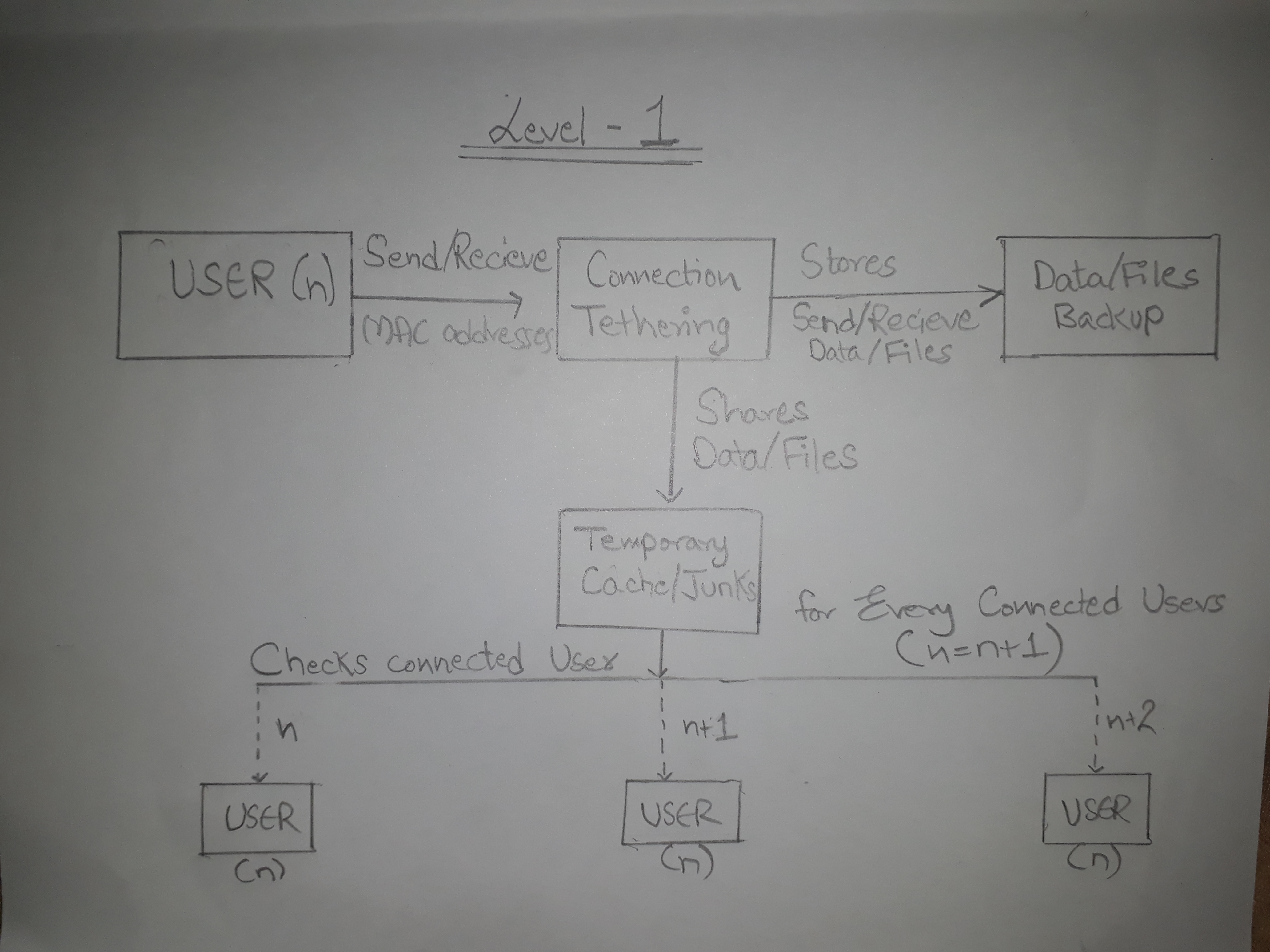
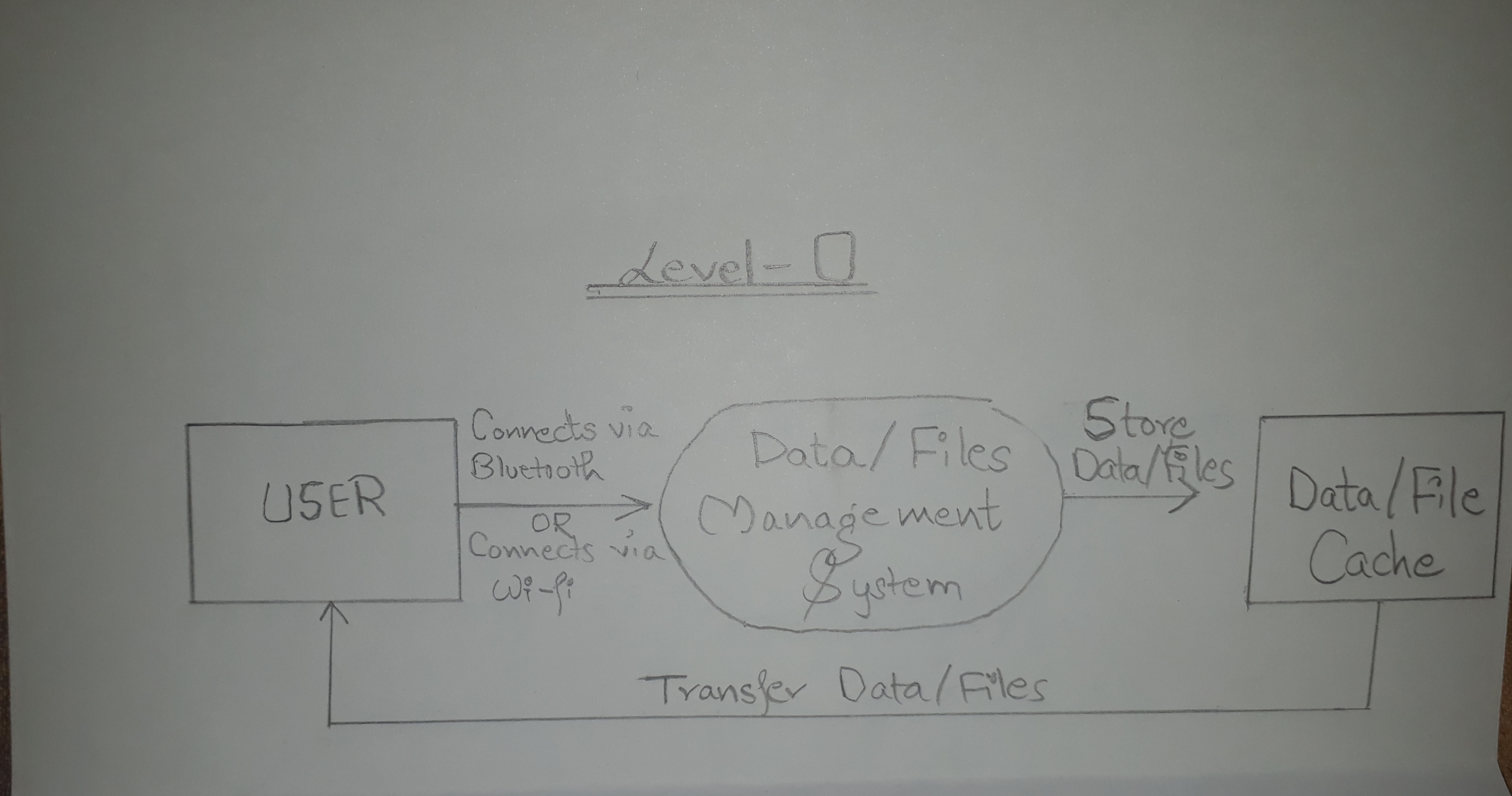


Fig: DataFlow Diagram

1. **Feasibility study:**

Feasibility study is used in determining the viability of idea, such as ensuring a project is legally and technically feasible as well as economically justifiable.

1. Technical Feasibility: Rapid growth in progress of project can be expected. It would be reasonable and pretty much comparative to upcoming technologies.
2. Economic feasibility: It will reduce the cost within effective communication. It is also to bring benefits to the owner. It would be affordable and reliable and would be relevant to project scope.
3. Legal feasibility: Any change in legalities issued by government the product marked price may alter.
4. Operational Feasibility: It evaluates problem solving methods and takes advantages of opportunities provided with the represented course of project.
5. Scheduling Feasibility: Simply put it is the time management process of work completion where work are further sub divided and is regarded as one of the vital process of study.
6. **Software Requirement Specification(SRS):**

In software development SRS is also most important task. It can be defined as the process of gathering information from the user to solve problems. If the collected information is incorrect then whole project leads to failure. For this specification of such projects must be done carefully and effectively. There are two types of the requirements they are:

1. Functional Requirement
2. Non-functional Requirement
3. **Functional Requirement:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N | Title | Description | Rational | Dependencies |
| FR01 | Bluetooth Activation | User can activate Bluetooth. | Connectivity. | N/A |
| FR02 | Wi-Fi Activation | User can activate Wi-Fi. | Connectivity. | N/A |
| FR03 | Bluetooth Discoverable | User can enable Discoverability. | Discoverability Enabling. | FR01 |
| FR04 | Bluetooth Pairing | User can connect to other devices within its coverage area through Bluetooth, | Pairing. | FR01,FR02 |
| FR05 | Wi-Fi Pairing | User can connect to other devices within its coverage area through Wi-Fi hotspots, | Pairing. | FR02 |
| FR06 | Adding suitable nicknames and avatars | User can give connected devices suitable nicknames and avatars. | Connected Devices evaluation. | FR04,FR05 |
| FR07 | Creating a group | User can create a group. | User can name the group and choose a suitable image for the group image. | N/A |
| FR08 | Adding to a group | User can add people to a group. | User can add connected devices in a group. | FR04,FR05 |
| FR09 | Removing from a group | User can remove people from a group. | User can remove connected devices in a group. | FR08 |
| FR010 | Editing a group | User can edit a group. | User can rename the group and re-choose a suitable image for the group image. | FR07 |
| FR011 | Deleting a group | User can delete a group. | User can delete a created group. | FR07 |
| FR012 | Messaging | User can message to the connected devices within or excluded from a group. | User can send/receive text messages to the connected devices. | FR04,FR05 |
| FR013 | File Transfer | User can send or receive files with connected devices. | User can send/receive files to the connected devices. | FR04,FR05 |
| FR014 | Interactive UI | User can easily operate on the program with ease with better user interface. | User can use the program  Without facing any difficulties. | N/A |
| FR015 | Add themes to Chats | User can add themes on the chat background. | User can define his/her own chatting background. | N/A |
| FR016 | Send/Receive stickers to Chats | User can send/receive stickers. | User can make the chat interesting with cool new updated stickers. | N/A |
| FR017 | Listing Available Devices | User can be the available devices on the network. | User can see the device availability. | N/A |

1. Non-functional Requirement:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N | Title | Description | Rational | Dependencies |
| NFR01 | Security | User can maintain security through connections. | Effective Security | N/A |
| NFR02 | Availability | Program should be easily available to everyone. | Easily Available | N/A |
| NFR03 | Reliability | Program system should be reliable in terms of efficiency. | Most Reliable | N/A |
| NFR04 | Maintainability | System should be easier to handle. | Maintainable | N/A |
| NFR05 | Scalability | The system should be adapting changes per demands. | Expansion of system | N/A |
| NFR06 | Testability | System should be tested. | Well Tested | N/A |
| NFR07 | Interface | System should have better user interface (UI). | Well Interface | N/A |
| NFR08 | Performance | System should work without any obligation. | Best Performance | N/A |
| NFR09 | Manageability | System should have easy manageability. | Well Manageable | N/A |
| NFR010 | Recoverability | System should recover lost data or files. | Data Recovery Option | N/A |
| NFR011 | Data integration | System should provide security to data. | High Data Security | N/A |
| NFR012 | User friendly | System should be easy to use and well equipped. | Best equipped facilities | N/A |

1. **Moscow Prioritization:**

It helps to define the requirements prioritization. It also helps to assigning time and helps to development process of the system. Moscow also used to prioritize the requirement of system.

* Must have: important requirements of system that is very essential.
* Should have: it is not much important to system but it expand its scope.
* Could have: it is not essential to the system.
* Won’t have: the requirements that is not necessary to the system.

The prioritization of requirement for my project given below:

For Users

|  |  |  |
| --- | --- | --- |
| S.N | Functional requirement | Moscow |
| FR01 | Bluetooth Activation | Must have |
| FR02 | Wi-Fi Activation | Must have |
| FR03 | Bluetooth Discoverable | Must have |
| FR04 | Bluetooth Pairing | Must have |
| FR05 | Wi-Fi Pairing | Must have |
| FR06 | Adding suitable nicknames and avatars | Could have |
| FR07 | Creating a group | Could have |
| FR08 | Adding to a group | Should have |
| FR09 | Removing from a group | Could have |
| FR010 | Editing a group | Should have |
| FR011 | Deleting a group | Should have |
| FR012 | Messaging | Must have |
| FR013 | File Transfer | Must have |
| FR014 | Interactive UI | Must have |
| FR015 | Add themes to Chats | Could have |
| FR016 | Send/Receive stickers to Chats | Could have |
| FR017 | Listing Available Devices | Should have |

Non-functional requirements

|  |  |  |
| --- | --- | --- |
| S.N | Non-functional requirements | Moscow |
| NFR01 | Security | Must have |
| NFR02 | Availability | Should have |
| NFR03 | Reliability | Must have |
| NFR04 | Maintainability | Should have |
| NFR05 | Scalability | Must have |
| NFR06 | Testability | Should have |
| NFR07 | Interface | Must have |
| NFR08 | Performance | Should have |
| NFR09 | Manageability | Should have |
| NFR010 | Recoverability | Should have |
| NFR011 | Data integration | Won’t have |
| NFR012 | User friendly | Must have |

1. **Hardware/Software specification:**

Hardware Specification

 Processor: Intel(R) Core (TM) i5-5200U CPU @ 2.20GHz 2.20

 Ram: 4GB or higher

 Hard Disk: 100 GB or higher

 Display Type: Standard VGA or SVGA card

 Peripherals: Keyboard, Mouse

Software Specification

 Operating system: Windows 7 or higher, Linux

 Front-end: Android Studio

 Back-end: None.

1. **Use Case Diagram:**

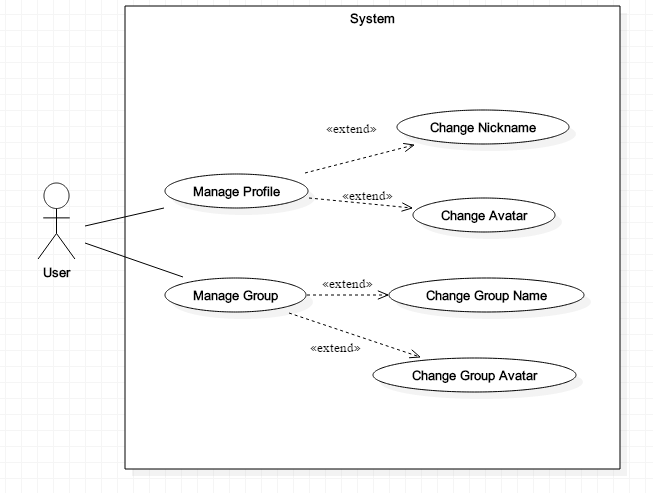
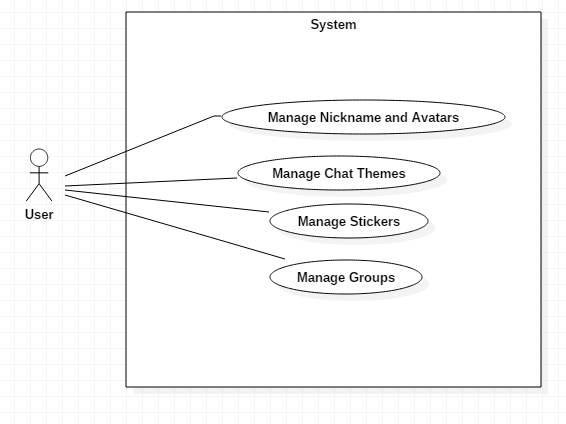
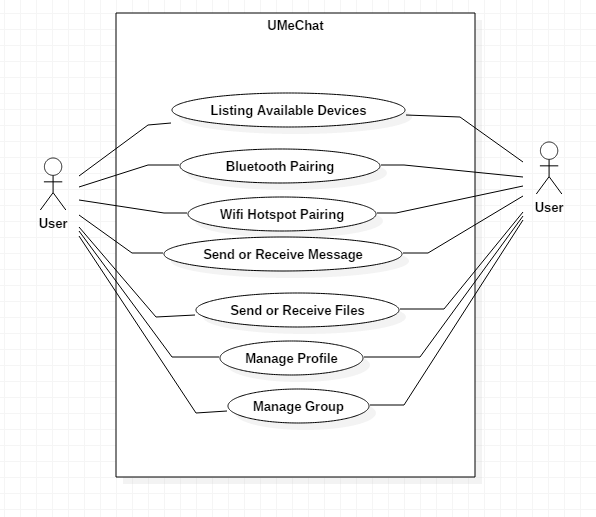
Use case diagram is a diagrammatic interaction between actor and system drawn in UML. It represents user interaction in diagram and shows relation between system and user. It is very effective in developmental process. 

Fig: Use case diagram

Title: “UMeChat”:

|  |  |
| --- | --- |
| Id | U1 |
| Primary actor | Users |
| Supporting actor | N/A |
| Primary flow | 1. User can list out available users. 2. User can pair with each another. 3. User can send or receive messages. 4. User can send or receive files. 5. User can manage each profile with nickname and avatar. 6. User can manage group name and avatars. |
| justification | Shows the system working mechanics. |
| Alternative flow | N/A |

Title: “System-1”:

|  |  |
| --- | --- |
| Id | U2 |
| Primary actor | Users |
| Supporting actor | N/A |
| Primary flow | 1. User manage nickname and avatars for connected devices. 2. User manage chat themes available. 3. User manage chat stickers. 4. User manage group name and avatars. |
| justification | Shows basic necessary system data to be included. |
| Alternative flow | N/A |

Title: “System-2”:

|  |  |
| --- | --- |
| Id | U3 |
| Primary actor | Users |
| Supporting actor | N/A |
| Primary flow | 1. User can change the nickname and avatars. 2. User can change the group name and avatars. |
| Justification | Shows how users can manage connected devices within or excluded from a group. |
| Alternative flow | N/A |

**NLA (Natural Language Analysis)**

Natural Language Analysis can be defined as the use of ability of systems to process sentences in a natural language rather than in artificial computer language such as C++ or C# and separating out nouns and adjectives.

|  |  |  |
| --- | --- | --- |
| Nouns: classes | Verbs: functions | Adjective : attributes |
| Users | Connect | Detail |
| Application | Send | Color |
| Chat | Receive | Theme |
| Bluetooth | Change | User\_Id |
| Wi-Fi | Manage | Files |
| Stickers | Edit |  |
| Avatar | Recover |  |

**Initial class diagram:**

Initial class diagram is the diagrammatic representation of static view of application. It is further classified into attributes and operations breaking them into a singular tabular format for easy configuration and construction of codes. Thus it provides a static representation of application. Initial class diagram consists of three types of relationship which are associations, compositions and dependency. Initial diagram of UMeChat is illustrated below:

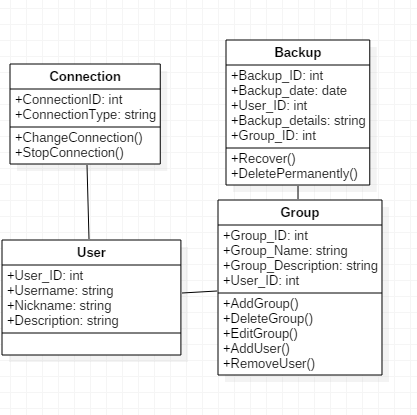


Fig: Initial Class Diagram