Requirements

1. **Requirements Engineering Process**
2. **Requirements Elicitation**: This is the process of gathering information about the needs and expectations of stakeholders for the software system. This step involves interviews, surveys, focus groups, and other techniques to gather information from stakeholders.
3. **Requirements Analysis**:This step involves analyzing the information gathered in the requirements elicitation step to identify the high-level goals and objectives of the software system. It also involves identifying any constraints or limitations that may affect the development of the software system.
4. **Requirements Specifications**:This step involves documenting the requirements identified in the analysis step in a clear, consistent, and unambiguous manner. This step also involves prioritizing and grouping the requirements into manageable chunks.
5. **Requirements Validation**:This step involves checking that the requirements are complete, consistent, and accurate. It also involves checking that the requirements are testable and that they meet the needs and expectations of stakeholders.
6. **Requirements Management**: This step involves managing the requirements throughout the software development life cycle, including tracking and controlling changes, and ensuring that the requirements are still valid and relevant.

The Requirements Engineering process is a **critical** step in the software development life cycle as it helps to ensure that the software system being developed meets the needs and expectations of stakeholders, and that it is developed on time, within budget, and to the required quality.

1. **Types of Requirements**
2. **Functional (Behavioral) Requirements**: These are the requirements that define the functions and features of the software system. They describe what the software should do, and how it should behave when specific user actions or inputs are provided. Functional requirements are often documented as use cases or user stories.
3. **Non-functional Requirements**: These requirements define the quality attributes of the software system, such as performance, reliability, scalability, usability, security, and compatibility. Non-functional requirements are critical for ensuring that the software meets the user’s needs and expectations.

* Interface constraints
* Performance constraints: response time, security, storage space, etc.
* Operating constraints
* Life cycle constraints: maintainability, portability, etc.
* Economic constraints

1. **Domain requirements**: Domain requirements are the requirements which are characteristic of a particular category or domain of projects. Domain requirements can be functional or nonfunctional. Domain requirements engineering is a continuous process of proactively defining the requirements for all foreseeable applications to be developed in the software product line. The basic functions that a system of a specific domain must necessarily exhibit come under this category. For instance, in an academic software that maintains records of a school or college, the functionality of being able to access the list of faculty and list of students of each grade is a domain requirement. These requirements are therefore identified from that domain model and are not user specific.
2. **Business requirements**: These requirements describe the business needs and objectives that the software system is intended to fulfill. They define the problem or opportunity that the software is addressing, and the benefits that it is expected to provide.
3. **User requirements**: These requirements describe the needs and expectations of the end-users of the software system. They include information about the users’ *tasks, workflows, and preferences*, and how the software should support them.
4. **Design requirements**: These requirements describe the *technical design* of the software system. They include information about the software architecture, data structures, algorithms, and other technical aspects of the software.
5. **Interface requirements**:These requirements describe the *interfaces between the software system and other systems, devices, or services*. They include information about the data formats, protocols, and communication channels that the software should use to integrate with other systems. (**DO NOT** mess up with UI)
6. **Constraints**: These describe any limitations or restrictions that must be considered when developing the software system.
7. **Acceptance Criteria**: These describe the conditions that must be met for the software system to be considered complete and ready for release.
8. **Software Requirements Specification (SRS)**
9. **SRS** – description of a software system to be developed. It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide.
10. SRS must be written in to steps: first is written by a customer to introduce his requirements and second is written by developer as a contract between them.
11. In order to fully understand one’s project, it is very important that they come up with an SRS listing out their requirements, how are they going to meet them and how will they complete the project. It helps the team to save upon their time as they are able to comprehend how are going to go about the project. Doing this also enables the team to find out about the limitations and risks early on.
12. SRS should be:

* **Correct**: user review is used to ensure the correctness of requirements stated in the SRS. SRS is said to be correct if it covers all the requirements that are actually expected from the system.
* **Unambiguous**: clear and specific, and should avoid using vague or imprecise language.
* **Complete**: include all the requirements for the software system, including both functional and non-functional requirements.
* **Consistent**: be consistent in its use of terminology and formatting, and should be free of contradictions.
* **Ranked for importance and/or stability**: there should a criterion to classify the requirements as less or more important or more specifically as desirable or essential. An identifier mark can be used with every requirement to indicate its rank or stability.
* **Verifiable**: the requirements can be tested and validated to ensure that they are being met.
* **Modifiable**: can be updated and changed as the software development process progresses.
* **Traceable**: traceable to other documents and artifacts, such as use cases and user stories, to ensure that all requirements are being met.

1. See example at the bottom of the document.

**Project Plan:  MeetUrMate**

**1. Introduction**

This document lays out a project plan for the development of the “MeetUrMate” open source repository system by Anurag Mishra.

The intended readers of this document are current and future developers working on “MeetUrMate” and the sponsors of the project. The plan will include, but is not restricted to, a summary of the system functionality, the scope of the project from the perspective of the “MeetUrMate” team (me and my mentors), scheduling and delivery estimates, project risks, and how those risks will be mitigated, the process by which I will develop the project, and metrics and measurements that will be recorded throughout the project.

**2. Overview**

In today’s world, owning to the heavy workload on the employees, they are having a huge amount of stress in their lives. Even with the presence of so many gadgets in and around them, they are not able to relieve their stress. I aim to develop an application that would enable them to share the thing of their liking and meet the person who has the same passion as theirs. For eg. If someone wants to share their art, they can share it through the platform, if someone wants to sing any song, they can record it and share the same. They can also share videos (with some funny commentary in the background), share mysteries that other people can solve, post any question. Through my platform, I’ll enable them to meet people who share common interests and passions, chat with them, and have some fun.

**2.1 Customers**

Everyone. Anyone can use this application ranging from a child to an old-age person.

**2.2 Functionality**

* Users should be able to register through their already existing accounts.
* They should be able to share snaps/videos/snaps.
* People should be able to like and comment on any post. One person can follow another person who shares common interests and likings which would enable them to find mates apart from their usual friend circle.
* Each user can have his/her profile picture, status
* People can post mysteries and other people can solve the mysteries.
* Users will get points for the popularity of their posts/the number of mysteries they solve.
* *Add own funny commentary on any video*
* *Post any questions regarding their interests and people can answer.*

P.S.  Italic points features can be inculcated later.

**2.3 Platform**

It will be launched both as a Web-based application and a Mobile app for Android.

**2.4 Development Responsibility**

I, Anurag Mishra, would be developing the software and I am responsible for the creation of the Database and all the other related kinds of stuff.

**3. Goals and Scopes**

* Users should be able to register through their already existing accounts.
* They should be able to share snaps/videos/snaps.
* People should be able to like and comment on any post.
* One person can follow another person who shares common interests and likings which would enable them to find mates apart from their usual friend circle.
* Each user can have his/her profile picture, status.
* People can post mysteries and other people can solve the mysteries.
* Users will get points for the popularity of their posts/the number of mysteries they solve.

**4. Deliverables**

I’ll deliver the following during the course of development:

* Feature specification
* Product design
* Test plan
* Development document
* Source code

**5. Risk Management**

**5.1 Risk Identification**

Following will be the risk involved in my project:

1) People are already using Facebook to find friends. So, what would be the real cause that would motivate them to join my application?

**5.2 Risk Mitigation**

Even though most of the users would already be using Facebook, our platform would still offer them many things that are not there on Facebook. For eg.

1. They don’t meet people who share common interests and passions as much. Our application would enable them to meet people (apart from usual friends) who share common interests and passions on a more frequent basis.
2. Users of FB cannot share songs on the go that they have sung whereas on our app they can do that on the go.
3. People can post mysteries/cases and other people can solve them. Moreover, people will get points in case they solve the mysteries or on the basis of the popularity of their posts.
4. More importantly, people need not register for my application, instead, they can log in using their already existing accounts of Google/Facebook.

Thus, I think that there is a considerable amount of difference between Facebook/Instagram/Twitter and my application and it would attract many people.

**6. Scheduling and Estimates**

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone** | **Description** | **Release Date** | **Release Iteration** |
| M1 | Application view and Design | October 5, 2015 | R1 |
|  | (Front-end development) |  |  |
| M2 | Database for my application | October 17, 2015 | R1 |
|  | (Back-end) |  |  |
| M3 | Integrating views and designs | November 12, 2015 | R1 |
|  | (Integrating front-end and |  |  |
|  | back-end) |  |  |
| M4 | Testing for initial release | November 20, 20015 | R2 |
| M5 | Issue tracker, user reviews, | December 1, 2015 | R2 |
|  | web design integration |  |  |
| M6 | Final release | December 23, 2015 | R2 |

**7. Technical Process**

Following would be the languages I would use to develop my application within the stipulated time period:

Front-end development: Jquery, HTML, CSS, PHP.

Back-end development: PHP, MySQL.

For Android app: Java on Android SDK.