**Software Development Lifecycle (SDLC) Analysis of Meesho**

***A comparative study of different models in relation to Meesho’s software development***

**Dhanush D Shetty**

Nitte Mahalinga Adyantaya Memorial Institute of Technology, Nitte, Karnataka, India

{danukadamba6@gmail.com, [nnm23is048@nmamit.in](mailto:nnm23is048@nmamit.in)}  
https://github.com/Dhanush-dshetty/Software-Development-Lifecycle-SDLC-Analysis-of-Meesho

|  |  |
| --- | --- |
| Keywords: | SDLC, Meesho, AWS, Integration, Testing, Scalability, Requirements validation |
|  |  |
| Abstract: | Software development is an evolving discipline requiring structured approaches for building scalable, secure, and efficient systems. This report explores the Software Development Life Cycle (SDLC) models applicable to Meesho, a global leader in video streaming services. It provides a comparative analysis of SDLC methodologies, an overview of requirements engineering.  The study aims to offer insights into the selection of an appropriate SDLC model for large-scale cloud-based platform (Meesho for this report), emphasizing the waterfall, incremental development and spiral model approaches. The report also discusses challenges and strategies involved in requirements validation and software deployment at Meesho.  The findings in this document are based on extensive research, industry best practices, and insights from Meesho’s technology stack. I hope this report serves as a valuable resource for software engineers, architects, and researchers interested in the intersection of SDLC and requirements engineering methodologies and large-scale streaming platforms. This paper is followed by a conclusion and few considerations. |
| Publishing: | This paper was submitted to Dr. Jason Elroy Martis, Associate Professor, Department of Information Science and Technology, NMAM Institute of Technology. Nitte Karnataka, India. This paper is also hosted on a GitHub repository, along with the material used for preparing this research. The link to the GitHub Repository is given in the endnote.  . |

**Table of Contents**

1. Preface
2. Introduction
3. Overview of Meesho
   * System Overview
   * Technologies Used
4. Comparative Analysis of SDLC Models
   * Waterfall Model
   * Incremental Development Model
   * Spiral Model
   * Summary of Comparison
5. Requirements Engineering for Meesho
   * Functional Requirements
   * Non-Functional Requirements
   * Requirements Validation Strategy
   * Challenges in Requirements Validation
6. Conclusion
7. References

**1. Introduction**

Social commerce platform Meesho empowers Indian consumers to establish their online businesses through product resale operations. The system operates through a dependable cloud-based framework which runs mainly on Amazon Web Services (AWS). Meesho demands an effective software development lifecycle (SDLC) to oversee continuous updates along with new features and maintenance of system stability because of its extensive scale.

The report performs a comparative assessment of Incremental Development, Spiral Model and Waterfall Model regarding software development at Meesho. The essay examines requirements engineering along with the problems encountered during the process.

**2. Overview of Meesho**

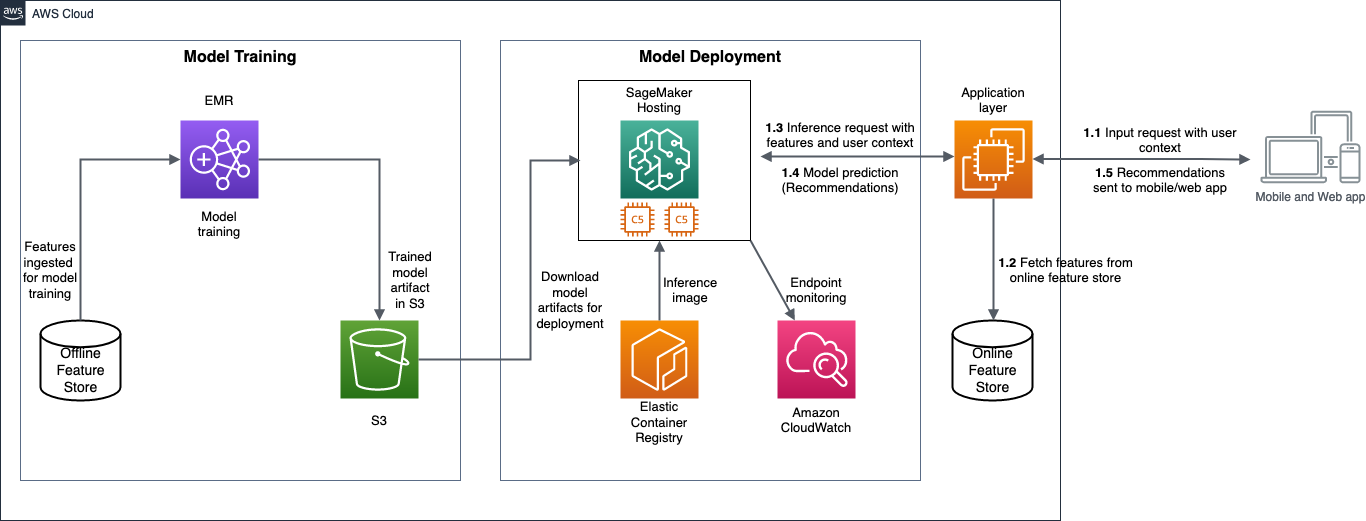
**2.1 System Overview**

Through its platform Meesho enables suppliers to link up with resellers thereby allowing them to sell products to end customers. The system enables several device usage and personalized suggestion features along with seamless transaction processes. Meesho serves as an Indian social commerce platform which lets small businesses and independent sellers launch their online stores at no cost. Across India customers can access the service which distributes fashion items as well as home decor along with electronics and additional product categories.

**2.2 Technologies Used**

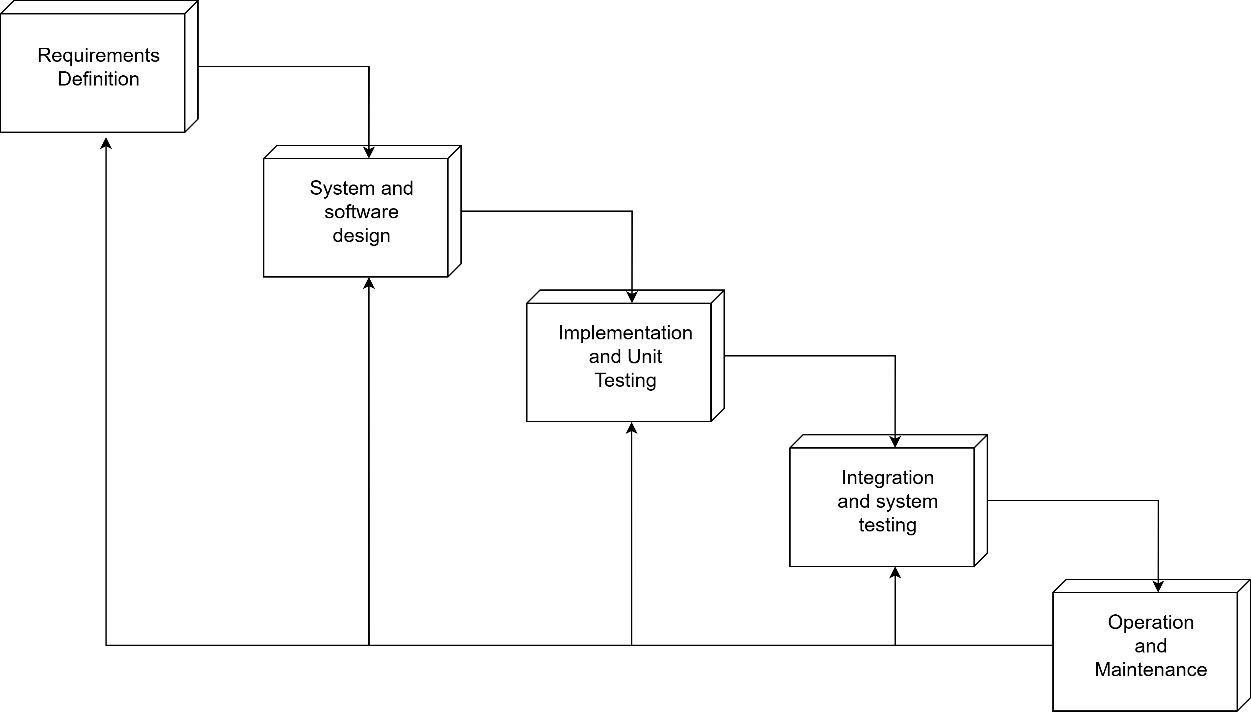
* The organization utilizes AWS (Amazon Web Services) as its cloud platform to handle its operations.
* Architecture: Microservices-based
* Database: NoSQL (DynamoDB, Cassandra), MySQL
* Content Delivery: AWS CloudFront, CDN
* Programming Languages: Java, Python, Node.js
* DevOps: Continuous Integration & Continuous Deployment (CI/CD)

The Meesho platform operates through different technological solutions. Only the major technologies discussed above maintain operations that drive the e-commerce platform through individual components. The individual parts of the platform merge seamlessly to deliver dependable performance for users throughout the system.



**3. Comparative Analysis of SDLC Models**

**3.1 Waterfall Model**

****

*Figure 2- Depicts the Waterfall Model*

**How Meesho Would Be Developed Using Waterfall:**

* **Phase 1: Requirements Definition** -- All system requirements are defined at once in this phase. This includes defining user authentication, product listing capabilities, payment gateway integration, and order management. Since changes are difficult to implement later, this is very important and exhaustive documentation is required at this stage.
* **Phase 2: System & Software Design** -- A complete architecture is developed or produced in this phase. Including all the database structures and whatever big data models that are to be applied. API endpoints, and server infrastructure on AWS is developed all at once in this stage, again since further changes are very difficult in the waterfall model. Every aspect of the system is carefully mapped out before development begins.
* **Phase 3: Implementation & Unit testing**-- Developers begin coding the entire system in one go, this has to be done since changes are tough to make, so they follow the previously defined architecture. No changes to the requirements are permitted, and development follows a sort of linear path, as shown in *Figure 2*.
* **Phase 4: Integration and System Testing** -- After development is completed, the entire system undergoes integration and rigorous testing. Integration becomes a crucial part in this process since, they have to be very fluid with integrating it properly so it runs everywhere without any hassle, otherwise the millions of users using it will be affected. Now the testing phase includes functional testing, performance testing, and security testing. Since all components are built at once, identifying and fixing bugs can be time-consuming.
* **Phase 5: Operation and Maintenance** -- The fully developed e-commerce platform is deployed to production. Now this can be localized, for example to a specific region or the whole country. This marks the system's release, and users can now access Meesho. Any bugs or issues discovered post-launch are addressed during the maintenance. However, because in this model new changes require extensive planning and reimplementation, updates take a long time to roll out and overall, we can see how this is very inefficient, especially for a data-driven and dynamic organization such as Meesho.

**Suitability for Meesho:**

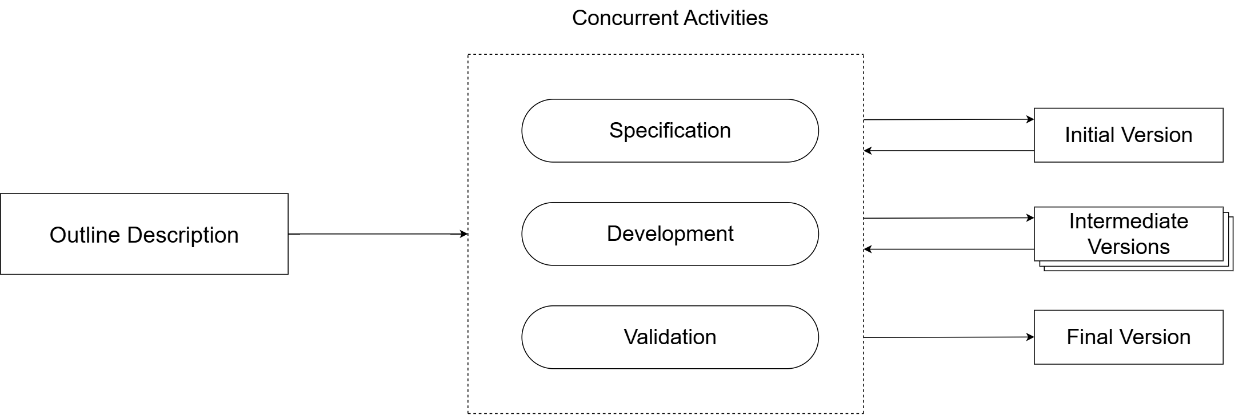
**Pros:**

* Well-documented and structured approach. The structure provides developers with cleaner development methods.
* The project management benefits from separate structured phases because of proper definition. It sort of Distributes the management on each step.
* Suitable for smaller, well-defined projects with minimal expected changes

**Cons:**

* The complete system displays minimal adaptability when dealing with changing user needs.
* Discovering problems after the initial completion could lead to significant time addictions.
* The extensive development period discussed earlier makes continuous updates impossible.
* The production method does not work because Meesho demands quick updates of features and needs to adapt quickly to shifting user preferences.

**3.2 Incremental Development Model**

****

*Figure 3- represents the Incremental Development Model*

**How Meesho Uses Incremental Development:**

* **Phase 1: Planning and Initial Requirements** -- This model is a priority-based model. It is variable and can be revised multiple times. It is clearly represented in *Figure 3*. The development team identifies high-priority features, such as user authentication, product listing, and payment gateway integration. These are the core components for Meesho especially. Only essential requirements for the first iteration are finalized, leaving room for future updates. This causes less complicated and piled development. I also feel it gives more head room to developers.
* **Phase 2: Feature-wise Development** -- Instead of developing the entire system at once, Meesho will be able to continuously releases new features. Now for example, it can improve the product recommendation algorithms, have new product categories, and enhanced seller tools. Each feature undergoes design, development, testing, and deployment in separate cycles, and each of it can be visited again, changes can be made. It makes it very sophisticated, speaking in terms of development.
* **Phase 3: Continuous User Feedback Integration** -- Each increment is deployed to beta users who provide feedback through usage patterns and explicit ratings. This becomes very helpful for Meesho since, if there are any bugs or inconveniences, it won't affect its actual users. Developers analyse this feedback and make necessary adjustments before launching the next increment. Now, feedback is also taken through its regular users too. But the idea is to reduce the inconvenience caused to them and to provide them with a reliable platform.
* **Phase 4: System Testing and Refinement** -- Every new feature or update undergoes extensive testing. Each development phase can be visited again, developers can add as many changes as they want, like we have already discussed. This including performance analysis, bug fixes, and security assessments. Automated pipelines are used, which is an awesome way to ensure continuous integration and testing.
* **Phase 5: Deployment and Monitoring** -- Features are released to production in batches. By batches, I mean it can be released to a specific localized zone, a region, etc. This ensures minimal downtime. Something called an A/B testing is often used to compare different versions of a feature before full-scale deployment. This can really help Meesho with seamless user experience.
* **Phase 6: Iteration and Maintenance** -- Now, as we discussed, this is an iterative process. It's like a cycle that repeats as developers refine existing features, resolve any user-reported issues, and also when they introduce new functionalities to the platform, which is done based on the ever-changing user needs.

**Suitability for Meesho:**

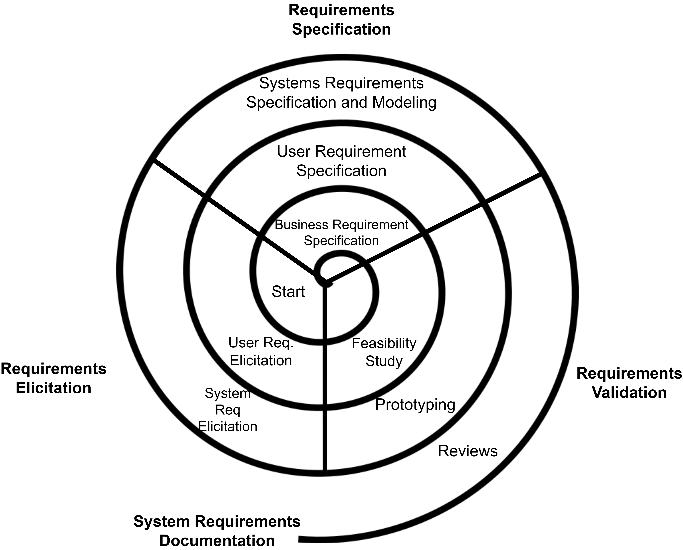
**Pros:**

* The development of logical code blocks enables the platform to deliver new features to customers faster since the development process remains efficient.
* The system enables fast adjustments to newer user requirements during operation.
* Continuous testing described earlier enables Meesho to achieve higher reliability and perform better on a wider scale overall.
* Meesho benefits from this model because it offers fast scaling potential specifically designed to meet their platform requirements while using their current cloud-based microservices framework.

**Cons:**

* Such integration requires proper strategies to prevent system conflicts thus becoming slightly more challenging.
* The system requires robust version control systems to combine different development branches into one integral framework.
* The system complexity rises when too many development features attempt simultaneous work.
* The platform parameters of Meesho fit well with this system because it enables smooth upgrades along with features that emerge from user-driven development.

**3.3 Spiral Model**

****

*Figure 4- Represents the Spiral Model*

**How Meesho Would Be Developed When Using Spiral**

* **Phase 1: Risk Analysis and Prototyping** -- Prior to deploying a significant feature maybe like AI-powered recommendations, etc, Meesho analyses the risks surrounding it. This includes algorithm biases, data security threats, and performance overhead. In a spiral model it is feasible to develop small scale prototypes first, before proceeding to develop on a full scale. This makes it separated and better maintained, and is efficient when it comes to testing.
* **Phase 2: Concept Validation and Refinement** -- Once they test the prototype, Meesho takes in preliminary data and refines the design of the feature. This is generally done for many times, until a satisfactory and stable version can be launched. This iterative process helps validate the feature against technical limitations and business objectives. Overall, it makes the process provide quality outputs.
* **Phase 3: Iterative Development and Testing** -- The features are built and tested in multiple cycles. Developers follow this iterative approach; they make changes based on risk assessment and the early feedback that they get from the beta users before the feature actually reaches full-scale deployment.
* **Phase 4: Broad Validation and Security Testing** -- The features are developed and tested in many iterations based on risk assessment and the initial feedback, changes are made, and the feature is gradually rolled out to higher-scale environments until full deployment. This is important for all of the complex features such as the machine learning models, personalization features, and data privacy regulations. Some of which are core components for Meesho.
* **Phase 5: Gradual Deployment and Performance Monitoring** -- The developed feature is initially deployed to few users, as discussed previously, some software use something like a beta version of their application, so that some people can now test their updates before it goes to all users. Real-world performance is monitored, and adjustments will be made before scaling the deployment to a larger audience.
* **Phase 6: Refinement and Continuous Improvement** -- By having a strong network of beta testers, as discussed in the previous phase, developers use real-time analytics and user insights to refine the feature further. This cycle will repeat for future improvements, ensuring a sort of innovation that's happening while the app is running live and used by millions around the world, while minimizing the risks.

**Suitability for Meesho:**

Pros:

* The multiple redesigns lead to professional risk control which enables the development of sturdy systems.
* Through repetitive enhancement processes developers succeed in enhancing complex systems and features such as artificial intelligence-driven personalization elements.
* Suitable for the large-scale, high-risk functionalities such as the security upgrades and algorithm-driven recommendations.

Cons:

* The development of basic features using this method proves expensive and time-consuming so it becomes less efficient for straightforward elements.
* The development model requires experienced professionals together with careful risk evaluation procedures.
* The security updates and AI-based recommendations and cloud infrastructure upgrades qualify as high-priority elements whereas normal feature development does not require this approach.
* The approach proves ineffective when handling routine updates unless these updates have high-risk factors or substantial alterations.

**3.4 Summary of Comparison**

| **SDLC Model** | **Flexibility** | **Risk Management** | **Time-to-Market** | **Cost** | **Suitability for Meesho** |
| --- | --- | --- | --- | --- | --- |
| Waterfall | Low | Low | Slow | Medium | Not Suitable |
| Incremental | High | Medium | Fast | High | Highly Suitable |
| Spiral | High | High | Moderate | Fast | Suitablefor High-Risk Features |

**SDLC Flexibility Risk Time-to-Market Cost Suitability for  
Model Management Meesho**

Waterfall Low Slow Medium Not Suitable

Incremental High Medium Fast High Highly Suitable

The Spiral High School presents a highly difficult environment to navigate with moderate standout highs and severe recurring challenges.

High-Risk Features

**4. Requirements Engineering for Meesho**

The development of requirements assumes a pivotal position at Meesho. Users demand from Meesho that it maintains consistently high delivery of operational performance alongside scalable functionality while providing absolute protection and exceptional user experiences. Meesho warrants Software Development Lifecycle (SDLC) to fulfill this requirement. We will examine the essential practices of requirements engineering at Meesho as well as their vital role and management hurdles when operating a massive high-traffic system.

**4.1 Functional Requirements**

* Specific behaviours along with functionalities make up Functional requirements which must be supported to fulfill user expectations. Now, these will include:
* The system implements OAuth as a secure platform which handles user authentication alongside account management functionalities. Users require multi-factor authentication or MFA and social media sign-in for the current era because they need security together with convenience. Account holders should have support for running different user profiles while getting personalized account settings under one master account. The system requires added security through mechanisms that manage sessions securely while enabling proper logout functions to stop unauthorized accesses.
* Sellers need access to functionality that enables them to upload products together with complete information about pricing along with descriptions and images. Users must be able to upload their products in ease while the system automatically assigns proper categories.
* The system integrates various payment gateways to process transactions through multiple payment options including credit/debit cards or UPI and PayPal and features automated financial matching as well as fraud prevention modules.
* Order Management system allows contact parties to track orders and obtain status updates through notification services. Integration with logistics partners for seamless delivery.
* Products are recommended through an AI-powered system which depends on user participation data along with their purchasing activities and individual preferences. Dynamic algorithm modifications occur through A/B testing together with ML models.
* The search function includes multiple filters that let users seek specific items according to chosen categories or set price amounts then personalize their search through preferences. The system will customize its user interface and user experience to improve customer engagement.
* Real-time customer support is enabled through combined chatbot and ticketing system presentations. Feedback collection and resolution tracking.
* The collection of user engagement information serves dual purposes for marketing purposes and strategic personalization through analytics and reporting systems. All types of customer data including purchase records as well as abandonment levels and product regularity statistics qualify for this system. Predictive analytics solutions exist for improving user experience when integrated with systems.

**4.2 Non-Functional Requirements**

* Non-functional requirements establish the operational limitations along with quality characteristics of a system.
* The system requires the capacity to operate with millions of simultaneous users without any decrease in operational speed. AWS Auto Scaling along with Load balancing allows users to automatically distribute resources based on recognizing real-time demand.
* The system can achieve 99.99% uptime through multi-region AWS deployment. Regional duplicate servers should be deployed to stop service disruptions and other maintenance-related outages.
* The system requires end-to-end encryption through TSL/SSL protocols together with authentication systems utilizing biometric scan technology and CAPTCHA authorization and preventive security measures from penetration testers.
* The website should perform transactions swiftly and display web pages rapidly. AWS CloudFront operates as a Content Delivery Network (CDN) to deliver content with higher speed.
* Compliance & Legal Considerations: Adherence to GDPR, PCI DSS, and local e-commerce regulations. Digital Rights Management (DRM) implements measures for stopping unauthorized financial activities.
* Infrastructure upgrades along with support for continuous deployment require implementation in the system.

**4.3 Requirements Validation Strategy**

* Functional and non-functional specifications need to confirm their alignment with Meesho's strategic goals. A set of different procedures helps validate requirements in software development.
* The validation process becomes more accurate by aligning requirements through meetings with developers and business teams and end-users.
* The early testing of UI/UX through prototyping allows company representatives to conduct A/B testing which guides data-driven user behaviour decisions.
* Testing and CI/CD pipelines with automated systems include functional testing alongside unit and regression methods to fulfill multiple regulatory standards.
* The system remains secure by conducting continuous security audits that test both vulnerability exposure and penetration abilities in order to comply with all security regulations.
* The system benefits greatly from user feedback loops because live analytics provide real-time data for evolving customer needs.

**4.4 Challenges in Requirements Validation**

Despite a good structured validation approach, Meesho faces challenges in requirements validation:

* **Dynamic User Expectations**: Trends tend to change constantly in e-commerce as we all know.
* **Global Compliance Issues**: Adapting to different regional e-commerce laws and payment regulations becomes a tedious job.
* **Scalability & Performance Bottlenecks**: Handling peak traffic loads and maintaining seamless service across diverse network conditions is difficult speaking of the infrastructure.
* **AI Bias in Personalization**: Ensuring fairness and inclusivity in recommendations. Sometimes it can over-prioritize specific products or demographics.
* **Fraud & Security Threats**: There has been a rising threat of fraudulent transactions and constantly evolving risks in digital payments.

**5. Conclusion**

The platform which we know as Meesho continues to evolve on a regular basis. This paper shows Incremental Development produces the best results for fast feature implementation and Spiral Model works best for dealing with high-risk features. Due to its strict framework the Waterfall model should not be adopted.

Meesho implements a requirements engineering procedure which effectively targets functional requirements and non-functional needs at the same time.

**7. References**

1. Sommerville, I. (2015). Software Engineering (10th Edition).
2. Wikipedia – Meesho- https://en.wikipedia.org/wiki/Meesho
3. Meesho Technology Blog -https://www.meesho.io/blog