2023

SRS for Advertisement Posting Service (APS).

Advertisement Posting website [aps]

TEAM APS

**Table of Contents**

[1. Software Requirement Analysis 6](#_Toc152186182)

[1.1 Introduction 6](#_Toc152186183)

[1.1.1 Purpose 6](#_Toc152186184)

[1.1.2. Scope 6](#_Toc152186185)

[1.1.3. Definitions, Acronyms, and Abbreviations 6](#_Toc152186186)

[1.1.4. References 6](#_Toc152186187)

[1.1.5. Overview 7](#_Toc152186188)

[1.2. Overall Description 7](#_Toc152186189)

[1.2.1. Product Perspective 7](#_Toc152186190)

[1.2.2. Product Features 7](#_Toc152186191)

[1.2.3. User Classes and Characteristics 7](#_Toc152186192)

[1.2.4. Operating Environment 8](#_Toc152186193)

[1.2.5. Design and Implementation Constraints 8](#_Toc152186194)

[1.2.6. User Documentation 8](#_Toc152186195)

[1.3. Specific Requirements 8](#_Toc152186196)

[1.3.1. Functional Requirements 8](#_Toc152186197)

[1.3.2. Non-Functional Requirements 9](#_Toc152186198)

[1.4. System Models 9](#_Toc152186199)

[1.4.1. Use Case Diagram 10](#_Toc152186200)

[1.4.2. Activity Diagram for User 10](#_Toc152186201)

[1.5. Estimate Timeframe: 11](#_Toc152186202)

[2. Software Development Life Cycle (SDLC). 12](#_Toc152186203)

[2.1 What is SDLC? 12](#_Toc152186204)

[2.1.1. Waterfall Method 13](#_Toc152186205)

[2.1.2. RAD Model 14](#_Toc152186206)

[2.1.3. Spiral Model 15](#_Toc152186207)

[2.1.4. V-Model 16](#_Toc152186208)

[2.1.5. Incremental Model 16](#_Toc152186209)

[17](#_Toc152186210)

[2.1.6. Agile Model 17](#_Toc152186211)

[18](#_Toc152186212)

[2.1.7. Iterative Model 18](#_Toc152186213)

[2.1.8. Big Bang Model 19](#_Toc152186214)

[2.2. Which One is Optimal for APS! 19](#_Toc152186215)

[2.2.2. Chosen SDLC Model. 19](#_Toc152186216)

[2.2.3. What is Agile Scrum Model? 20](#_Toc152186217)

[2.2.4. Agile Scrum in Software Development. 20](#_Toc152186218)

[2.2.5. Why Agile Scrum Model! 21](#_Toc152186219)

[3. Use Case Diagram & UML Diagram 22](#_Toc152186220)

[3.1. Use Case Diagram 22](#_Toc152186221)

[3.2. UML Diagram 23](#_Toc152186222)

[23](#_Toc152186223)

[3.3. Flow Chart 24](#_Toc152186224)

[4.Testing Approach. 25](#_Toc152186225)

[4.1. Testing Overview. 25](#_Toc152186226)

[4.2. Objectives. 25](#_Toc152186227)

[4.3. Types Of Testing. 25](#_Toc152186228)

[4.4. Testing Schedule. 26](#_Toc152186229)

[4.5. Testing Methodologies and Types. 26](#_Toc152186230)

[4.5.1. Black Box Testing. 26](#_Toc152186231)

[4.5.2. White Box Testing. 27](#_Toc152186232)

[4.6. Testing Strategies. 28](#_Toc152186233)

[4.6.1. Unit Testing. 29](#_Toc152186234)

[4.6.2. Integration Testing. 29](#_Toc152186235)

[4.6.3. System Testing. 30](#_Toc152186236)

[4.6.4. Acceptance testing 30](#_Toc152186237)

[5.Language, Methodology, Gantt Chart. 30](#_Toc152186238)

[5.1. Language and Frameworks. 30](#_Toc152186239)

[5.2 Methodology of Website. 31](#_Toc152186240)

[5.3 Estimated Timeframe (Gantt Chart). 32](#_Toc152186241)

[5.4. Version History. 32](#_Toc152186242)

[5.5. Project Links. 32](#_Toc152186243)

[6. Contact Us 33](#_Toc152186244)

### 1. Software Requirement Analysis

# 1.1 Introduction

## 1.1.1 Purpose

The purpose of this document is to specify the requirements for the Advertisement Posting Service, which allows users to submit and manage advertisements on various social media platforms.

## 1.1.2. Scope

The Advertisement Posting Service **(APS)** is a website which will provide a user-friendly platform for users to create, customize, schedule, and track their advertisements on social media platforms including but not limited to Facebook, Messenger, Instagram, and TikTok.

## 1.1.3. Definitions, Acronyms, and Abbreviations

- SRS: Software Requirements Specification.

- UI: User Interface.

## 1.1.4. References

* Saidur Rahman. (Electronics Shop Owner in Mirpur-1).
* Md. Monirul Islam. (Digital Marketer with 3 year of experience).
* AFM Shahriar Rahi. (Full Stack Developer with 3 year of experience).

## 1.1.5. Overview

This document outlines the functional and non-functional requirements of the Advertisement Posting Service, detailing the features, user classes, and system constraints.

# 1.2. Overall Description

## 1.2.1. Product Perspective

The Advertisement Posting Service **(APS)** is a standalone system that interfaces with various social media platforms' APIs to post advertisements on behalf of users. It does not interact with external systems beyond the scope of advertisement posting.

## 1.2.2. Product Features

- User registration and authentication.

- Advertisement submission and customization.

- Selection of target social media platforms.

- Preview and editing of advertisements.

- Scheduling of advertisement posts.

- Real-time performance analytics.

- Customer support interface.

## 1.2.3. User Classes and Characteristics

-Advertisers: Individuals or businesses who wish to promote their content on social media.

-Administrators: System administrators who manage user accounts, monitor system health, and address technical issues.

## 1.2.4. Operating Environment

The Advertisement Posting Service will be accessible through standard web browsers on both desktop and mobile devices. It will utilize databases to store user information and advertisement data.

## 1.2.5. Design and Implementation Constraints

- The system must adhere to the APIs and guidelines of the integrated social media platforms.

- Data security and privacy regulations must be followed.

## 1.2.6. User Documentation

Comprehensive user guides and tutorials will be provided to help users navigate and effectively utilize the website's features.

# 1.3. Specific Requirements

## 1.3.1. Functional Requirements

1. Secure User Account Creation and Login: Users must be able to establish accounts and log in.

- There needs to be a password recovery system in place.

2. User-submitted advertisements can be customized by adding photos or videos and providing other information.

- Customization possibilities to meet the criteria of each social networking platform.

3. Target Social Media Selection: Users have the option of choosing one or more social media sites to run advertisements on.

4. Preview and Editing of Advertisements: Users can get a preview of their advertisements on the platforms they have chosen.

- Content editing options to improve it.

5. Scheduling of ads: Users can decide when adverts will be posted on particular platforms.

-Users can obtain data on ad reach, engagement, click-through rates, and conversions by using real-time performance analytics.

7. Customer Support Interface: Users can access a support system for assistance with issues.

## 1.3.2. Non-Functional Requirements

1. Performance: The website should be able to support several users at once and guarantee quick responses.

2. Security: User information and login information must be secured and encrypted.

3. Usability: The UI needs to be responsive, user-friendly, and intuitive.

4. Compatibility: The website needs to work on all popular web browsers and mobile platforms.

5. Scalability: The system needs to be able to expand as the user base does.

6. Availability: There should be less downtime for maintenance and a high uptime for the website.

User data should be treated in accordance with data protection laws.

7. Data Privacy: All of user’s data must be protected.

# 1.4. System Models

## 1.4.1. Use Case Diagram

**Advertisement Posting Service**

**(APS)**

**Administrator**

**Advertiser**

-User Authentication.

-Sign up/ Log in.

-Manage info Security.

-Authentication.

-Manage Secure payment.

-Change/Customize info.

-Review & check User Ads.

-Post Ads (in specific format).

-Allow Ads to publish.

-Review & customize Ads.

-Manage system Stability.

-Select Platforms & duration.

-Handle Ads of users.

-Complete Payment.

-Show Analytics.

-Submit Ads.

-Manage Spams & Delete.

-See Analytics and info.

-Edit Ads or Delete Ads.

## 1.4.2. Activity Diagram for User

**Submit Advertisement**

**Select Platforms Customize Content Schedule**

**Review and Confirm**

# 1.5. Estimate Timeframe:

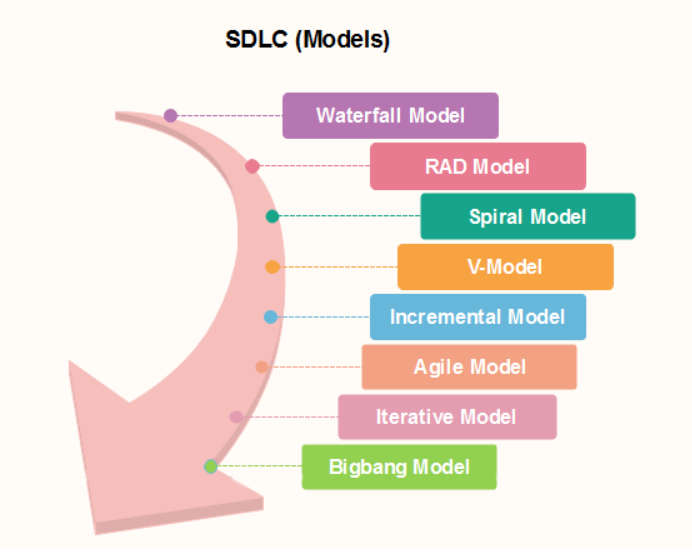
|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone** | **Tasks** | **Reporting** | **Time** |
| Analysis |  | Submit The Design | 3 days |
| Requirements Collection | Submit to Us All Data |  | 7 days |
| Development |  | Review The Work | 30 days |
| Testing |  |  | 10 days |
| Deployment | Must Ready the Server | Review Final Work | 5 days |
| Delivery |  | Live On Server | 5 days |

### 2. Software Development Life Cycle (SDLC).

# 2.1 What is SDLC?

#### The Software Development Life Cycle (SDLC) is a project management framework for organizing and carrying out software development projects. It describes the various phases of a project, starting from the initial feasibility assessment and ending with the upkeep of the finished application.

#### The steps required for each stage are specified by several SDLC models. These models are frequently referred to as "software development process models." Each model has a distinct set of steps that must be followed for the software development process to be successful.

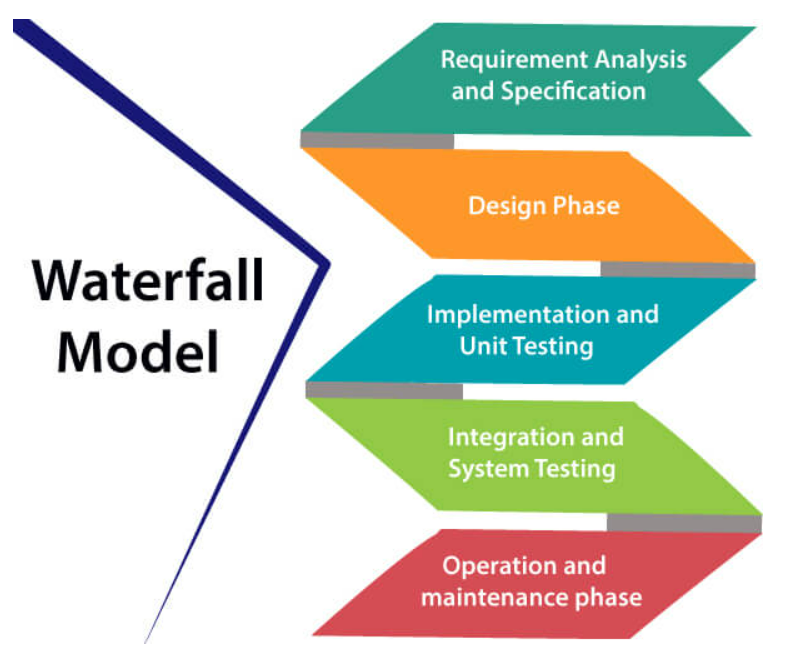


## 2.1.1. Waterfall Method

#### The SDLC model known as waterfall is widely used. This approach divides the entire software development process into a number of segments.

#### The requirements analysis, design, implementation, testing (validation), integration, and maintenance stages of the waterfall model are seen as flowing slowly downhill (like a waterfall) throughout continuous software development.

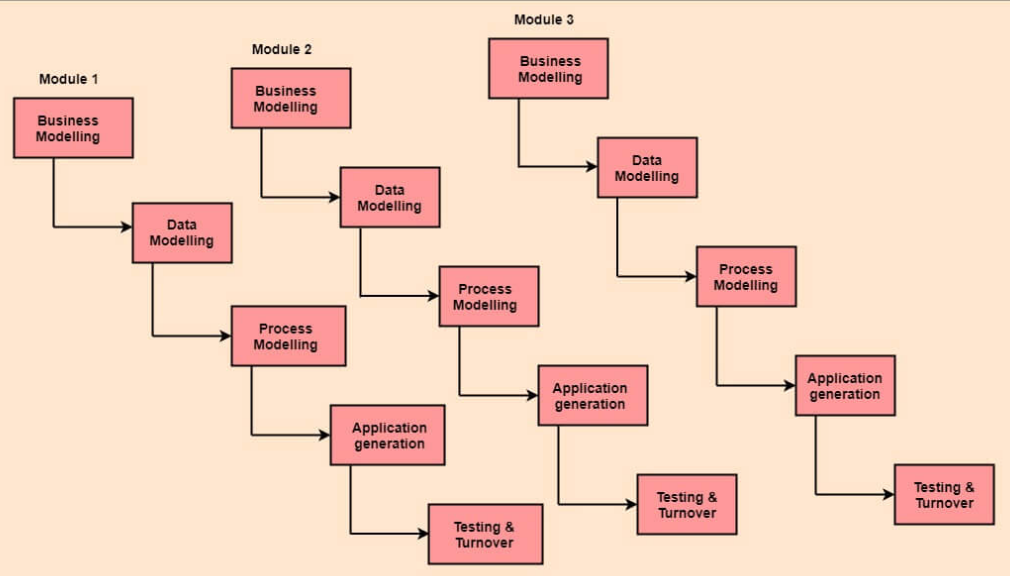
#### The linear arrangement of tasks has several important repercussions. First, some certification procedures must be used at the conclusion of each stage to signal the end of one phase and the start of the next. This is typically done by some verification and validation, which will make sure that the result of the stage is consistent with both its input (which is the output of the previous phase) and the system's overall needs.



## 2.1.2. RAD Model

#### Now we will see the RAD Model. The waterfall approach is adopted by the RAD or Rapid Application Development process, which aims to produce software quickly. The RAD methodology is founded on the idea that by using focus groups to gather system requirements, a better system may be produced in less time. Such as,

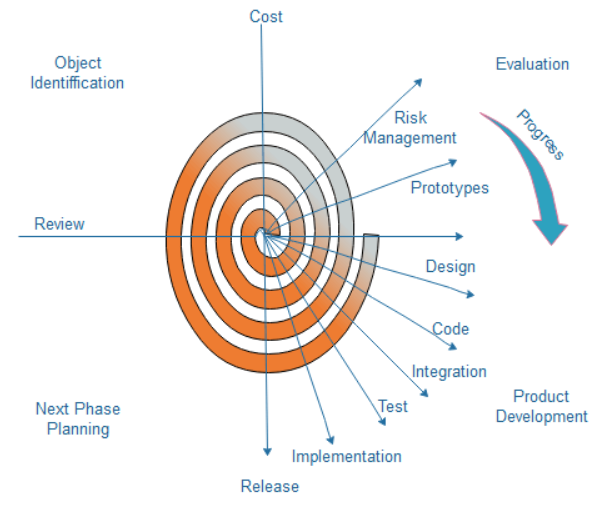
* Business Modelling
* Data Modeling
* Process Modeling
* Application Generation
* Testing and Turnover



## 2.1.3. Spiral Model

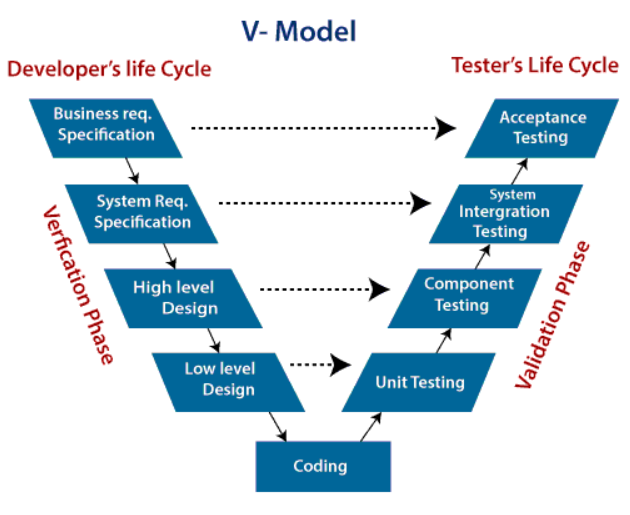
#### A process model driven by risk is the spiral model. The group can adopt parts of one or more process models, such as waterfall, incremental, waterfall, etc., with the use of this SDLC model. Rapid prototyping and concurrent design and development operations are combined in the spiral technique. The spiral's cycles each start with the identification of the cycle's goals, potential means of achieving those goals, and any limits that may apply. The cycle's initial quadrant (upper-left quadrant) is this one.

#### The following cycle phase is to assess these various options in light of the goals and limitations. This step's evaluation is centered on the perceived risk to the project. Creating plans to address risks and uncertainties is the next step. Activities like benchmarking, simulation, and prototyping could be a part of this level.



## 2.1.4. V-Model

#### Also known as the Verification and Validation Model, the V-Model. This requires that each stage of the SDLC be finished before moving on to the next. The waterfall model's sequential design approach is also followed. The device's testing is scheduled concurrently with the relevant stage of development.



## 2.1.5. Incremental Model

#### There is no distinct model from the incremental model. It must consist of several waterfall cycles. At the beginning of the project, the needs are split into groups. The SDLC paradigm is used to the software development process for each group. Every release adds more functionality as the SDLC process is repeated until all requirements are satisfied. Each cycle in this strategy serves as the upkeep phase for the preceding software release. The incremental model has been modified to allow for overlapping development cycles. After that, the following cycle can start before the preceding one is finished.

# 

## 2.1.6. Agile Model

#### Now we will see the Agile Model. Agile methodology is a practice which promotes continues interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.

#### -It might be challenging to predict which software requirements will change and which ones will last. Equally challenging is predicting how user priorities will evolve over the course of the project.

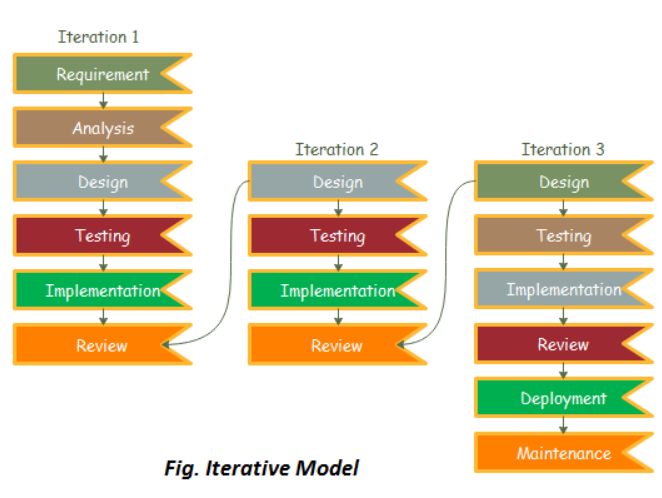
#### -Design and development are often integrated in many different forms of software. In other words, both tasks should be carried out simultaneously to allow for the validation of design models as they are developed. How much design must be done before building is utilized to evaluate the arrangement is a challenging concept to grasp.

#### -From a planning perspective, analysis, design, development, and testing are not as predictable as we would want.

# 

## 2.1.7. Iterative Model

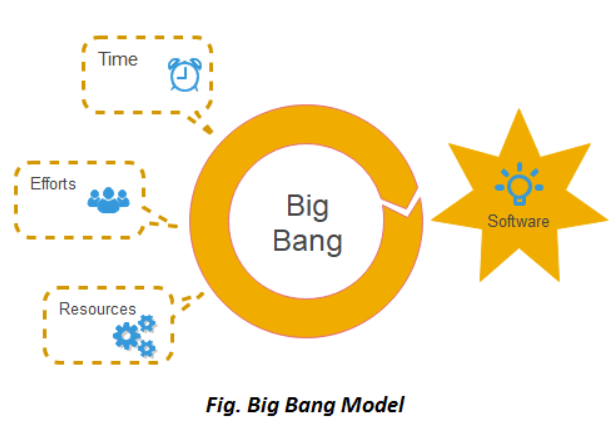
#### It is a specific application of a software development life cycle that emphasizes an initial, basic implementation before steadily gaining complexity and a wider feature set until the final system is finished. Iterative development is a technique for dividing the software development of a big application into smaller segments.



## 2.1.8. Big Bang Model

#### In this model, developers do not follow any specific process. Development begins with the necessary funds and efforts in the form of inputs. And the result may or may not be as per the customer's requirement, because in this model, even the customer requirements are not defined.

#### This model is ideal for small projects like academic projects or practical projects. One or two developers can work together on this model.



# 2.2. Which One is Optimal for APS!

## 2.2.2. Chosen SDLC Model.

#### Various criteria, including the project's size, complexity, needs, and team expertise, determine which Software Development Life Cycle (SDLC) model is best for a given project. The needs we've outlined in our scenario point to the necessity for an adaptable, iterative strategy that prioritizes security, usability, and performance. Given the specifications and non-functional components, an agile model or an agile variant like scrum would be appropriate.

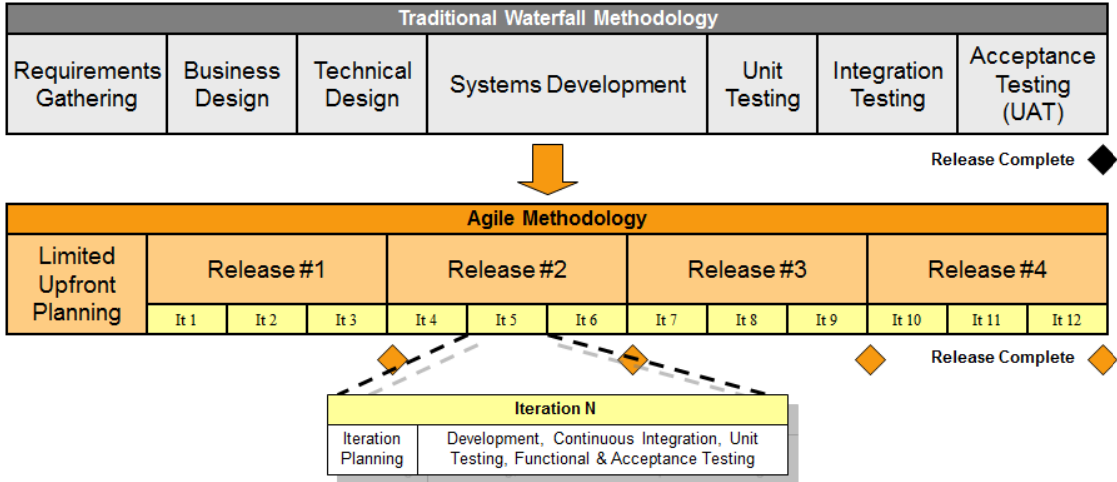
## 2.2.3. What is Agile Scrum Model?

#### Agile scrum methodology is an incremental development-based approach to project management. The purpose of each sprint, which lasts two to four weeks during each iteration, is to implement the most crucial features first and produce a Potentially Shippable Product.

## 2.2.4. Agile Scrum in Software Development.

#### Scrum, one of the most widely used agile development approaches today, focuses on having brief, time-boxed sprints of new functionality that are merged into a baseline integrated product. Scrum prioritizes open communication with customers and their comments and adjustments over paperwork and forecasting.

#### Scrum projects are divided into releases and sprints rather than phases. You have a fully operational system that is ready for release at the conclusion of each sprint:



## 2.2.5. Why Agile Scrum Model!

#### **- Flexibility:** Agile techniques are renowned for their adaptability and flexibility. The needs for your project may change or evolve over time, therefore an agile strategy enables constant adaptability to new requirements.

#### **- Iterative development:** Agile employs iterative development and frequent releases. This is in line with your desire for features that may be developed and distributed gradually, such as secure user account creation, customization, and real-time performance metrics.

#### **-Consumer Involvement:** Agile promotes consumer input and collaboration. Agile provides for ongoing consumer interaction in the development process because it has features like the ability to preview and change adverts and a customer support interface.

#### **-Security and Data Privacy:** By incorporating security principles and conducting regular security assessments throughout the development process, the Agile methodology may meet security and data privacy standards.

#### **-Performance and Usability:** Based on user feedback, Agile's iterative development approach can enable continual improvement in performance and usability.

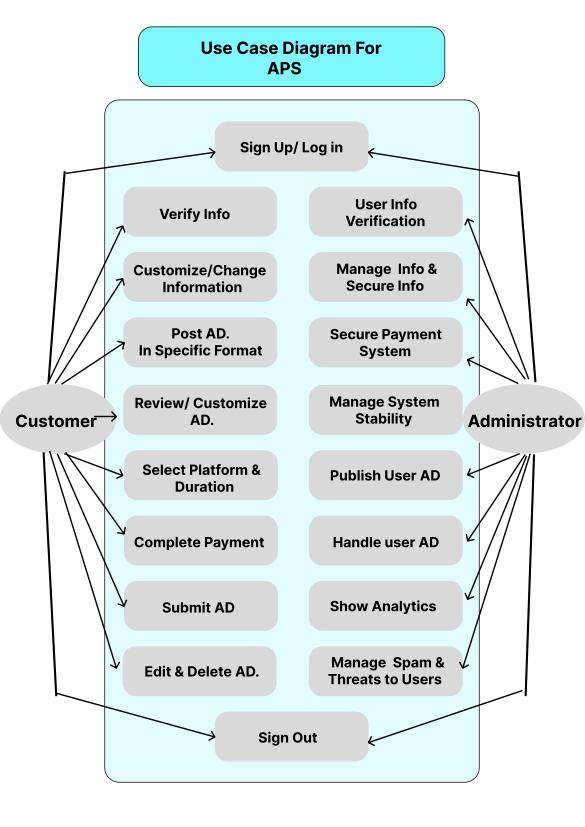
#### **-Scalability and compatibility:** Agile can help with these concerns by allowing teams to handle problems as they come up and scale the system as necessary.

#### **-Availability:** Agile's iterative and incremental development can assist reduce maintenance outages while maintaining a high level of website availability.

#### **-Regulatory Compliance:** The development process can be adjusted to include regulatory compliance needs, such as data protection laws.

### 3. Use Case Diagram & UML Diagram

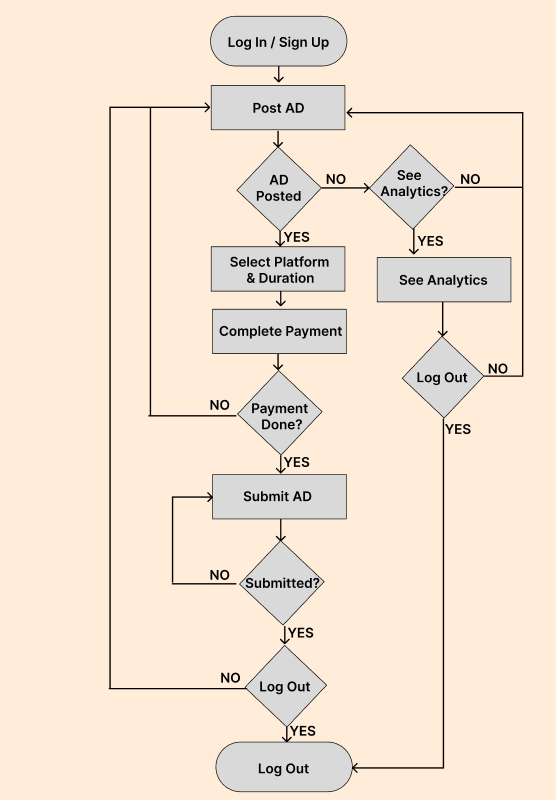
# 3.1. Use Case Diagram



# 3.2. UML Diagram

# 

# 3.3. Flow Chart



### 4.Testing Approach.

# 4.1. Testing Overview.

The testing phase is a critical component of the AI Financial Advisor software development process. It ensures that the software functions as intended, meets performance standards, and complies with regulatory requirements.

# 4.2. Objectives.

The primary objectives of testing for the AI Financial Advisor include:

* Verify Functionality: Ensure that all specified features and functionalities work as intended.
* Performance Testing: Evaluate the system's responsiveness, real-time data analysis, and overall performance.
* Security Testing: Identify and address any potential security vulnerabilities.
* Regulatory Compliance Testing: Validate compliance with relevant financial regulations and data protection standards.
* Scalability Testing: Assess the system's ability to handle increased user loads and data sources.

# 4.3. Types Of Testing.

Various testing types will be employed, including:

- Unit Testing: Test individual components to ensure they function correctly.

- Integration Testing: Validate the interaction between integrated components.

- System Testing: Verify the system as a whole to ensure it meets the specified requirements.

- Performance Testing: Evaluate the system's speed, responsiveness, and resource utilization.

- Security Testing: Identify and address security vulnerabilities.

- User Acceptance Testing (UAT): Obtain feedback from users to ensure the system meets their expectations.

- Compatibility Testing: Ensure the software functions correctly across different browsers and devices.

- Usability Testing: Evaluate the user interface and overall user experience.

- Load Testing: The system's performance under expected user loads is tested.

# 4.4. Testing Schedule.

|  |  |
| --- | --- |
| Testing Phase | Time |
| Unit Testing | 3 Days |
| Integration Testing | 3 Days |
| Performance Testing | 4 Days |
| Security Testing | 7 Days |
| User Acceptance Testing & Regulatory Compliance | 7 Days |
| Load, Scalability & Usability Testing | 10 Days |

# 4.5. Testing Methodologies and Types.

Here, I have considered the two testing methodology and types that is mention above:

1. Black box testing.

2. White box testing.

## 4.5.1. Black Box Testing.

Black box testing is testing software based on output requirements and without any knowledge of the internal structure or coding in the program. The goal is to test how well the component conforms to the published requirement for the component. Black box testing have little or no regard to the internal logical structure of the system, it only examines the fundamental aspect of the system. It makes sure that input is properly accepted and output is correctly produced.

Some Different types of Black box testing techniques are as follows:-

1) Equivalent Partitioning;

2) Boundary value Analysis;

3) Cause-Effect Graphing Techniques;

4) Comparison Testing.

***Advantages:***

1) The number of test cases are reduced to achieve reasonable testing

2) The test cases can show presence or absence of classes of errors.

3) Black box tester has no “bond” with the code.

4) Programmer and tester both are independent of

each other.

5) More effective on larger units of code than clear

box testing.

***Disadvantages:***

1) Test cases are hard to design without clear specifications.

2) Only small numbers of possible input can actually be tested.

3) Some parts of the back end are not tested at all.

4) Chances of having unidentified paths during this

testing

5) Chances of having repetition of tests that are

already done by programmer.

## 4.5.2. White Box Testing.

White box testing is highly effective in detecting and resolving problems, because bugs can often be found before they cause trouble. White box testing is the process of giving the input to the system and checking how the system processes that input to generate the required output. White box testing is also called white box analysis, clear box testing or clear box analysis. White box testing is applicable at integration, unit and system levels of the software testing process. White box testing is considered as a security testing method that can be used to validate whether code implementation follows intended design, to validate implemented security functionality, and to uncover exploitable vulnerabilities.

**Advantages:**

1) All independent paths in a module will be exercised at least once.

2) All logical decisions will be exercised.

3) All loops at their boundaries will be executed.

4) Internal data structures will be exercised to

maintain their validity.

5) Errors in hidden codes are revealed.

6) Approximate the partitioning done by execution

equivalence.

**Disadvantages:**

1) Missed out the cases omitted in the code.

2) As knowledge of code and internal structure is a prerequisite, a skilled tester is needed to carry

out this type of testing, which increases the cost.

3) And it is nearly impossible to look into every bit of code to find out hidden errors, which may create

problems, resulting in failure of the applications.

# 4.6. Testing Strategies.

A strategy for software Testing integrates software test case design methods into a well planned Series of steps that result in successful Construction of software that result in successful construction of software. Software testing Strategies gives the road map for testing. A software testing Strategy should be flexible enough to promote a customized testing approach at same time it must be right enough. Strategy is generally developed by project managers, software engineer and testing specialist.

There are four different software testing strategies:

1) Unit testing.

2) Integration testing.

3) Acceptance/Validation testing.

4) System testing.

## 4.6.1. Unit Testing.

Unit is the smallest module i.e. smallest collection of lines of code which can be tested. Unit testing is just one of the levels of testing which go together to make the big picture of testing a system. IT complements integration and system level testing. It should also complement code reviews and walkthroughs. Unit testing is generally seen as a white box test class. That is it is biased to looking at and evaluating the code as implemented. Rather than evaluating conformance to some set of requirements.

**Advantages:**

1) Unit level testing is very cost effective.

2) It provides a much greater reliability improvement for resources expanded than system level testing. In Particular, it tends to reveal bugs which are otherwise insidious and are often catastrophic like the strange system crashes that occur in the field when something unusual happens.

3) Be able to test parts of a project without waiting for the other parts to be available, be able to detect and remove defects at a much less cost compared to other later stages of testing.

## 4.6.2. Integration Testing.

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design.

Different Integration testing Strategies are discussed below:-

1) Top down Integration testing. 2) Bottom up Integration testing.

Top down Integration Top-down integration testing is an incremental approach to construct program structure. Modules are integrated by moving downward through the structure, beginning with the main control module. Modules subordinate to the main control module are incorporated into the structure in either a depth-first or breadthfirst manner. Bottom up Integration Bottom-up integration testing, as its name implies, begins construction and testing with atomic modules. Because components are integrated from the bottom up, processing required for components subordinate to a given level is always available and the need for stubs is eliminated.

## 4.6.3. System Testing.

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that system elements have been properly integrated and perform allocated functions.

Some of Different types of system testing are as follows:-

1. Recovery testing.

2. Security testing.

3. Graphical user interface testing.

4. Compatibility testing.

## 4.6.4. Acceptance testing

Acceptance testing is a crucial evaluation phase ensuring a product meets specified criteria and customer requirements. Typically performed by the user/customer, it validates the system's overall functionality rather than its internal workings. This testing, falling under black box methodology, assesses the system against outlined requirements. Referred to as validation, final, QA, or application testing, it confirms that the system aligns with user expectations before final delivery. It can occur at both the system provider and end-user levels, serving as a pivotal step before system handover.

### 5.Language, Methodology, Gantt Chart.

# 5.1. Language and Frameworks.

|  |  |
| --- | --- |
| *Front End* | *HTML5.0, CSS, JavaScript, Bootstrap5.4,*  *React.js* |
| *Back End* | *Python 3.11, Django 4.2.6* |
| *Database* | *MYSQL* |
| *Platform* | *AWS (Amazon Web Service)* |

# 5.2 Methodology of Website.

Here are the seven stages of web development in bullet points:

**1. Planning and Discovery:**

- Define goals, audience, and scope.

- Research and analysis.

- Create project roadmap and strategy.

**2. Design:**

- Wireframing and prototyping.

- UI design (aesthetics and usability).

- UX design (user interaction and experience).

**3. Content Creation:**

- Develop or gather engaging content.

- Ensure content aligns with design and SEO.

**4. Development:**

- Frontend Development (HTML, CSS, JavaScript).

- Backend Development (server-side logic, databases).

**5. Testing:**

- Quality Assurance (QA).

- User Testing for feedback.

**6. Deployment:**

- Upload to web server or hosting platform.

- Configure domain and server settings.

**7. Maintenance and Optimization:**

- Regular updates and bug fixes.

- Performance monitoring and improvements.

# 5.3 Estimated Timeframe (Gantt Chart).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Milestone Name*** | ***Description*** | ***Start Date*** | ***End Date*** | ***Total Days*** |
| *Project Planning* | *Design, Team Selection, Financial Planning.* | *15 – 09 – 2023* | *21 – 09 – 2023* | *7 Days* |
| *Project Initiation* | *Establish Project Teams* | *22 – 09 – 2023* | *24 – 09 – 2023* | *3 Days* |
| *Designing* | *Submit Final Design* | *25 – 09 – 2023* | *05 – 10 – 2023* | *12 Days* |
| *Development* | *Finish Developing the Project* | *06 – 10 – 2023* | *10 – 11 – 2023* | *40 Days* |
| *Testing* | *Testing Phase and Improvement* | *11 – 11 – 2023* | *24 – 11 – 2023* | *14 Days* |
| *Deployment* | *Ready Server and Make It Live on the Server* | *25 – 11 – 2023* | *01 – 12 – 2023* | *7 Days* |
| *Delivery* | *Hand Over the Project* | *02 – 12 – 2023* | *04 – 12 – 2023* | *3 Days* |

# 5.4. Version History.

|  |  |  |
| --- | --- | --- |
| ***Version*** | ***Changelog*** | ***Date*** |
| *1.0* | *SRS* | *25 – 08 – 2023* |
| *2.0* | *SRS+SDLC* | *25 – 09 – 2023* |
| *3.0* | *Design Approach* | *10 – 10 – 2023* |
| *4.0* | *Testing Approach* | *25 – 11 – 2023* |
| *5.0* | *Final Version* | *29 – 11 – 2023* |

# 5.5. Project Links.

**Design Link:** [**https://www.figma.com/file/u3qNwve6XgCHgiWQTEJEns/APS-prototype?type=design&node-id=0%3A1&mode=design&t=TMMgNxHZPjIP3Vfc-1**](https://www.figma.com/file/u3qNwve6XgCHgiWQTEJEns/APS-prototype?type=design&node-id=0%3A1&mode=design&t=TMMgNxHZPjIP3Vfc-1)

**Project Link:** [**https://github.com/masumbillah06/APS\_Project**](https://github.com/masumbillah06/APS_Project)

### 6. Contact Us

Name: Masum Billah.

ID: 221-15-5571.

Mobile: 01774-105413.

Email: [mbmasum06@gmail.com](mailto:mbmasum06@gmail.com)

Diu Mail: [billah15-5571@diu.edu.bd](mailto:billah15-5571@diu.edu.bd)

Address: YKSG-2, Daffodil International University, Ashulia, Savar, Dhaka.

Name : Tanha Tul Tasnim.

ID: 221-15- 5626.

Mobile:

Email: tasnim15-5626@diu.edu.bd

Address: RASG-girls Hall, Daffodil International University, Ashulia, Savar, Dhaka.

Name : Mst. Kanij Fatima.

ID: 221-15-5702.

Mobile:

Email: bmkanij26867@gmail.

Address: RASG-girls Hall, Daffodil International University, Ashulia, Savar, Dhaka.