**LEARNHUB**

|  |  |
| --- | --- |
| 1.Siva Sai Maruthi Charan - 22BD1A051Z  2.P.Guru Prasad - 22BD1A051E  3.V.Nikhil Teja - 22BD1A051Y  4.G.Durga Prasad - 22BD1A050C  5.Sree Vibha.P - 22BD1A051Q |  |

Problem Statement:

In today's rapidly evolving educational landscape, creating engaging and accessible learning platforms is essential for both teachers and learners. When teachers join a learning platform, they expect a seamless process to upload, manage, and track their course content, while learners expect easy access to high-quality educational resources . A key challenge arises in facilitating smooth teacher verification, effective course creation, and tracking learner progress in real-time. The system must allow teachers to specify completion timelines, set user limits for courses, and provide diverse content formats, such as PDFs, YouTube video lectures, and quizzes, across multiple topics.

One primary challenge is managing content creation. Teachers must be able to upload, edit, and publish educational content in a flexible manner. PDF documents, video lectures, and quizzes must be tied to each course topic, with a limit on the number of enrolled users. The system should enable real-time tracking of learners' progress, including time spent on each resource, and allow teachers to remove users if necessary. Furthermore, generating and distributing certificates upon course completion, containing details like course name, teacher, website, and a QR code, adds another layer of complexity. Scanning the QR code must retrieve specific information about the user's completion status.

**Software Requirement specification**

For

LearnHub

Version 1.0 approved

Prepared by:

|  |  |  |
| --- | --- | --- |
| 1. Siva Sai Maruthi Charan | - | 22BD1A051Z |
| 2. Nikhil Teja | - | 22BD1A051Y |
| 3. Guru Prasad | - | 22BD1A051E |

4.Durga Prasad - 22BD1A050C

5.Sree Vibha.P - 22BD1A051Q

Keshav Memorial Institute of Technology

23-09-2024

## Table of Contents

1.Introduction.............................................................................................

1.1 Purpose........................................................................................................

1.2 DocumentConventions................................................................................

1.3 Intended Audience and Reading Suggestions...........................................

1.4 Product Scope............................................................................................

1.5 References.................................................................................................

### 2.Overall Description

2.1 Product Perspective………………………………………………..

2.2 Product Functions……………………………………………………

2.3 Operating Environment……………………………………………

2.4 User Characteristics………………………………………

2.5 Design and Implementation Constraints……………………………

2.6 User documentation…………………………………………

2.7 Assumptions and Dependencies…………………………………………

### 3.External Interface Requirements

3.1 User Interface…………………………………………………

3.2 Software Interface……………………………………….

3.3 Hardware Interface……………………………..

3.4 Communication Interface……………………………..

4. System Features.

4.1 System Feature 1

4.2 System Feature 2

5 Other Nonfunctional Requirements.......................................

5.1 Performance Requirements.........................................................

5.2 Safety Requirements.........................................

5.3 Security Requirements.......................................

5.4 Software Quality Attributes...........................

5.5 Business Rules………………………………

6 Other Requirements….

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason for changes | Version |
| Week-1 | 10-09-24 | SRS creation(Introduction) | 1.0 |
| Week-2 | 21-09-24 | SRS updation(External Interface Requirements) | 2.0 |
| Week-3 | 23-09-24 | SRS Updation(System features and other non functional requirements) | 3.0 |

## **1.Introduction**

The Software Requirement Specification is designed to document and describe the working of an Online Course reservation system, named Learning Hub. This documentation is done to provide a clear idea of customer requirements. This document can be used as reference in further development of the software system.

**1.1 Purpose:**

The purpose of our Learning Hub is to streamline and automate the process of course enrollment for students and faculty. It enables students to easily browse available courses, check seat availability, and register from anywhere, anytime. By reducing manual work, minimizing scheduling conflicts, and providing real-time updates, it improves efficiency and accuracy in managing academic schedules. Additionally, it provides a transparent and organized system for both students and teachers, ensuring a smooth enrollment experience.

**1.2 Document Convention**:

Heading:

Font-Size:16

Font-Style: Bold Font: Times New Roman Subheading:

Font-Size:14

Font-Style: Bold

Font: Times New Roman

Content:

Font-Size:12

Font: Times New Roman.

**1.3 Intended Audience and Reading Suggestions:**

The intended audience includes teachers, learners, and platform administrators. Teachers can use this guide to easily upload and manage content like PDFs, videos, and quizzes. Learners will find instructions for tracking their progress, while older learners or professionals looking to upskill will benefit from the platform’s flexibility. It’s best to start with the system overview before diving into advanced features like user limits, progress tracking, and certification generation.

**1.4 Product Scope:**

The LearnHub is a web-based platform that allows users (students and professionals) to browse, select, and reserve seats for various online courses. Our project is similar to other online course reservation websites with some new advancements. Our site will be available for 24 hrs and 7 days in a week. The system will support multiple courses, instructors, time slots, and payment gateways, enabling a seamless and user-friendly experience for students and administrators. Our primary objective is to satisfy the customer. Instructors can create course schedules, set class timings, and manage their availability. A dynamic listing of available courses categorized by subject, level (beginner, intermediate, advanced), and instructor. Users can reserve a seat in a course by selecting an available time slot. Users can rate courses and leave reviews to help other users choose.

An administrative panel for managing courses, users, payments, and system settings. Admins can manage user accounts, track activity, and handle complaints or violations. Secure storage of personal and financial data with encryption. Regular backup of all user and course data to prevent loss.

**1.5 References:**

We took references from different websites like Udemy, Coursera, edX, and Khan acdemy etc.

<https://www.reqview.com/papers/ReqView-Example_Software_Requirements_Specification_SRS_Document.pdf>

**2. Overall Description**

### **2.1 Product Perspective**

LearnHub will provide a seamless platform for students and professionals to reserve online courses, while enabling instructors and institutions to manage course content and schedules. The system will offer key features like user registration, course search, secure payments, and notifications, alongside instructor tools for managing courses and student interactions.It is a Mobile-Friendly website with highly customizable themes. And it is produced with some new advancements like Summarizing the video and also the pdf which was uploaded by the instructor for a better understanding to the student.

**2.2 Product functions**

#### 2.2.1 Administrator

* Admins should be able to create,delete users.
* Can accept or reject teachers judging on their qualifications..
* Can delete courses judging on their content.
* Can be able to get the list of payments done on the website.

#### 2.2.2 User(Teacher)

* Can register his/her details to get approval from the admin.
* Can publish the course by creating content for the course.
* Enable the Chatbots for the content
* Provide time slots to the student for doubt clarification..

#### 2.2.2 User(Student)

* Can be able to enroll in the course and pay the amount.
* Can be able to clarify doubts using chatbots.
* Can receive a certificate after completion of course.
* Can book doubt sessions with the teacher.

### **2.3 Operating Environment**

* This e-commerce website shall operate in all famous browsers.
* It can be opened on Windows,Linuxand also on mac Os.
* The processor should be at least Pentium 3 or above.
* Ram should be greater than 512 mb.

### **2.4 User Characteristics**

Users of this website are students,teachers and administrators who maintain the website.We have system administrator who will maintain overall web store.there are teachers who upload their course on the website.Students who enroll into the course.So, customers and merchants assumed to have basic knowledge of computers and Internet browsing.Administrators of the system should have knowledge about the internal modules and are able to rectify problems arise.

### **2.5 Design and Implementation Constraints**

* Users (admins, teachers, and students) must have a device with a stable internet connection and internet browsing capabilities to access the website..
* User information, course content, and progress data will be stored in a secure database that can be accessed by the website.
* This website will be available 24/7 to ensure continuous access for users.
* The software does not require any specific operating system to run. It is designed to be platform-independent and can be accessed on any device, including mobile phones
* Teachers can create courses with multiple topics, including PDFs, video lectures (YouTube links), and quizzes. They can also edit and publish courses.
* Implement robust security measures to protect user data and ensure secure transactions and communications within the platform.

**2.6 User Documentation**

1. Overview: LearnHub is a user-friendly, mobile-responsive platform that enables students and professionals to search for and enroll in online courses. It also provides instructors with tools to manage course content, schedules, and student interactions. The platform's key features include course search, secure payments, and notifications, along with innovative video and PDF summarization for better content understanding.

2. Getting Started:

* Registration:
  + Users can register as students or instructors by filling out the required personal and account information.
  + Once registered, users will receive a confirmation email for verification purposes.
* Login:
  + After registration, users can log in using their email and password.
  + A "Forgot Password" option is available in case users need to reset their login credentials.

3. Student Guide:

* Browsing Courses:
  + Students can browse and search for courses using keywords, categories, and filters like course level and duration.
* Enrolling in a Course:
  + Upon selecting a course, students can view detailed course descriptions, instructor profiles, and available schedules.
  + After selecting a course schedule, students can proceed to secure payment to complete enrollment.
* Course Materials:
  + Course materials like videos, PDFs, and supplementary resources can be accessed from the course dashboard.
  + Summaries of videos and PDFs are automatically generated for easier comprehension.
* Notifications:
  + Students receive notifications regarding course updates, assignment deadlines, and any announcements made by instructors.

4. Instructor Guide:

* Creating Courses:
  + Instructors can create courses by providing a title, description, syllabus, and upload related content (videos, PDFs, etc.).
  + They can schedule course sessions and assign deadlines for assignments.
* Managing Students:
  + Instructors can track student progress, respond to questions, and provide feedback through the platform.
  + They can upload additional content or update course materials at any time.
* Course Summaries:
  + Uploaded PDFs and video content will have an automatic summary feature, making it easier for students to grasp key concepts quickly.

5. Payment and Security:

* Secure Payments:
  + LearnHub uses secure payment gateways to process transactions for course enrollment.
  + Students can pay using credit/debit cards, bank transfers, or supported online payment methods.
* Refund Policy:
  + The platform outlines clear refund policies, which are accessible in the user profile settings or during payment processing.

6. Customizing Themes:

* Users can personalize the appearance of their dashboards by selecting from customizable themes to suit their preferences.

7. Support:

* Help Center:
  + The platform provides a comprehensive Help Center with FAQs, tutorials, and step-by-step guides.
* Contact Support:
  + Users can reach out to LearnHub’s support team via email or live chat for any technical issues or queries.

### **2.7 Assumptions and Dependencies**

The website requires the following third party products.

* MongoDB to store the database, and Vercel for hosting.
* YouTube API: For embedding and managing video lectures.
* AZURE: For storing and serving PDF documents and other static content.
* Stripe: For handling payments if you plan to monetize courses.
* Express is used as backend server, also to generate QR.

The success of this E-Commerce website depends on

* Existence of an internet service to all people.
* Provide tools for teachers to easily create, edit, and publish courses with various content types (PDFs, videos, quizzes).
* Ensure the platform is intuitive and user-friendly for admins, teachers, and students.
* Ensure certificates are generated accurately with all necessary details and a verifiable QR code.
* Provide timely support to users for any technical issues or queries.

## **3.External Interface Requirements**

### 3.1 User Interfaces

* The Learnhub platform features a user-friendly interface, ensuring that users can easily navigate the website without any confusion.
* It is implemented using javascript.
* There are features typically seen in website.

3.1.1 Login Page:

* Allow users to sign in to their existing account or register as a new user.
* Username/Email ,Password (Text Field) ,Login Button,Forgot Password? (Link) ,Register as New User (Link)

3.1.2 Dashboard

* Home screen after logging in, showing user's personalized options.
* It mainly has Welcome Message ("Welcome, [User Name]")
* Navigation Menu: It consists of Course Catalog,My Registered Courses, Profile Settings ,Log Out

3.1.3 Course Detail page

* User can have details of every course.
* The courses can be filtered based on courseId , course name, lecture name.

3.1.4 Registration Confirm page.

* Confirm the course registration and provide important information.
* The user can have the payment to buy the course

3.1.5 My Registered Courses

* Display courses the user is registered for
* List of Registered Courses
* Scheduled Overview(Calender view of the courses).

3.1.6 Profile Settings Page

* The page which consists of the user details
* There can be also the option for editing it

### 3.1.7 Administrator/Instructor Dashboard

* For course administrators or instructors to manage courses.
* Course Management:
  + Add New Course
  + Edit Existing Courses
  + View Registered Students

### **3.2 Software Interfaces**

These refer to the various systems, applications, or services that the Learnhub will interact with.

#### 3.2.1 Database

* Store and manage all data related to students, courses, registration, schedules, etc. by using databases like MongoDB.
* Key components to store details like student id, course id, enrollment data.

3.2.2 Learning Management Systems

* Allow integration with Learning Management Systems for academic content, grades, and assessments.
* Key components are Syllabus and course content synchronization ,Grade reporting integration ,User authentication synchronization

#### 3.2.3 Payment Gateway

* For collecting any registration fees or tuition payments.
* Stripe, PayPal, Razorpay
* Secure payment processing
* Invoice and receipt generation

#### 3.2.4 Authorization System.

* Handle user login, access control, and session management.
* OAuth 2.0, JWT (JSON Web Tokens), LDAP (Lightweight Directory Access Protocol)
* User login
* Role-based access control (RBAC) (Student, Instructor, Admin)
* Password management and security

3.2.5 Email and Notification Service

* For sending registration confirmations, reminders, or alerts about course updates.
* SMTP servers, SendGrid, Amazon SES
* Automatic email generation for registration status
* Notification system for deadlines, registration changes, etc.

## **3.3 Hardware Interface**

These refer to the physical devices and systems that the Learnhub will interact with or require for proper functioning.

3.3.1 Server Hardware

* Run the system’s backend services, database, and application logic.
* CPU: Multi-core processors (Intel Xeon or AMD EPYC) for handling multiple simultaneous users.
* RAM: Minimum 16 GB
* Storage: High-speed SSD storage for quick data retrieval and storage (e.g., 1TB SSD, scalable)

3.3.2 Client Hardware

* Devices used by students, instructors, and administrators to access the system.
* Devices: Desktop, Laptop, Tablet, Smartphone
* Operating Systems: Windows, macOS, Linux, iOS, Android

3.3.3 Backup and Storage Devices

* Purpose: To store and back up the system’s critical data (e.g., student records, course information).
* Backup Servers: Separate hardware for database and system backup
* Storage Devices: Cloud-based or physical storage (e.g., AWS S3, NAS, or RAID systems)
* Data Retention: RAID configuration for fault tolerance and recovery

## **3.4 Communication Interfaces**

For the communication interfaces section of your SRS, based on your idea, here's what you can include:

Web Browser

The platform will be accessible via standard web browsers (e.g., Chrome, Firefox, Edge). All interactions between the users (admin, teacher, student) and the system will be facilitated through a web-based interface using HTTP/HTTPS protocols.

Network Protocols

HTTP/HTTPS: All communications between the client and the server will use secure HTTPS to ensure data integrity and encryption.

WebSocket: For real-time communication (e.g., chatbot interactions and doubt session booking), WebSocket may be employed to provide seamless, low-latency communication between users and the server.

REST APIs: The platform will expose RESTful APIs for various functionalities, including course management, user statistics tracking, and test generation.

Communication Security

Encryption: All sensitive information such as passwords, user progress, and course content will be encrypted using TLS (Transport Layer Security) for data in transit.

Authentication: OAuth 2.0 or JWT (JSON Web Tokens) will be used for securing user sessions and validating user roles (admin, teacher, user).

QR Codes: The certificate generated upon course completion will have a QR code, and when scanned, it will fetch details from a secure endpoint, adhering to the HTTPS protocol.

Messaging and Notifications

Email: Notifications such as course completion certificates, badges, and reminders will be sent via email. SMTP protocols will be used to communicate with email servers for dispatching notifications.

Push Notifications: The system will use push notifications (possibly via Firebase Cloud Messaging) to remind users about their pending courses, quizzes, and doubt sessions.

In-App Messaging: An internal messaging system will allow students to contact teachers for booking doubt sessions and for receiving responses to queries.

File Transfer Protocols

YouTube Integration: The system will handle the secure embedding of YouTube video links in the course content. The video link will be processed to provide AI summaries and chatbot interactions.

Document Handling: PDF documents uploaded as course content will be stored securely on the server and accessible to users over HTTPS. Encryption at rest (e.g., AES-256) will ensure document security.

Communication Standards

HTML5/CSS: All web-based interfaces will follow standard HTML5 and CSS3 protocols to ensure cross-browser compatibility.

AJAX: Asynchronous requests will be used to handle content loading without requiring full-page reloads, enhancing user experience.

Data Synchronization

Timers and Tracking: User progress (time spent on courses) will be synced with the server at regular intervals, possibly every 5 seconds, ensuring real-time tracking of activities.

Doubt Clarifications: Any doubts asked via the chatbot will be asynchronously stored and forwarded to the teacher dashboard for review and responses.

Message Formatting

JSON: For RESTful APIs, all messages and responses will be formatted in JSON to maintain consistency and ease of use across all platforms.

QR Codes: The QR code embedded in the certificate will point to a JSON-based endpoint detailing the course and user achievements.

Data Transfer Rates

Since the platform will handle video embedding, quizzes, and reading materials, standard data transfer rates will be sufficient. For smooth video playback, users will need an internet connection supporting at least 3-5 Mbps download speeds.

Based on your idea, here’s a similar format for your system features in the SRS document:

**4.System Features**

**4.1 Course Search and Enrollment**

4.1.1 Description and Priority

Description: Allows users to search for and enroll in courses based on criteria such as category, teacher, course completion time, and available slots.

Priority: High

Benefit: 9

Penalty: 4

Cost: 7

Risk: 4

4.1.2 Stimulus/Response Sequences

1. User Action: User enters search criteria (e.g., category, teacher) in search fields.

System Response: The system displays a list of courses matching the criteria.

2. User Action: User views course details and decides to enroll.

System Response: The system enrolls the user in the course and updates the available slots.

4.1.3 Functional Requirements

REQ-1: The system must provide search fields for entering criteria such as category, teacher, and completion time.

REQ-2: The system must display search results with relevant details (e.g., course title, teacher, duration, available slots).

REQ-3: The system must allow users to enroll in a course if slots are available.

REQ-4: The system must handle cases where no courses match the search criteria by displaying a suitable message.

REQ-5: The system must provide an option to cancel enrollment and free up slots.

**4.2 Course Content Management**

4.2.1 Description and Priority

Description: Enables teachers to create, edit, and manage course content, including PDFs, videos, and quizzes.

Priority: High

Benefit: 8

Penalty: 5

Cost: 6

Risk: 4

4.2.2 Stimulus/Response Sequences

1.User Action: Teacher creates a new course and uploads content such as PDFs, YouTube video links, and quizzes.

System Response: The system saves the course and its content for publishing.

2.User Action: Teacher edits the course by adding more topics or adjusting the course structure.

System Response: The system updates the course and notifies enrolled users.

4.2.3 Functional Requirements

REQ-1: The system must allow teachers to create new courses and upload PDFs, video links, and quizzes.

REQ-2: The system must allow teachers to edit or update course content at any time before or after publishing.

REQ-3: The system must ensure that the course contains multiple topics with structured content (PDF, video, quiz).

REQ-4: The system must handle cases where the uploaded content format is invalid and provide appropriate error messages.

REQ-5: The system must track and update user progress in real-time as they view the course content.

4.3 Progress Tracking and Certification

4.3.1 Description and Priority

Description: Tracks user progress in real-time and generates badges and certificates upon course completion.

Priority: High

Benefit: 9

Penalty: 6

Cost: 7

Risk: 5

4.3.2 Stimulus/Response Sequences

1.User Action: User completes a course by viewing all content and passing quizzes.

System Response: The system generates a badge and certificate with the course details.

2. User Action: Teacher reviews user progress and notices a user is struggling.

System Response: The system allows the teacher to send reminders or provide extra materials for slow learners.

4.3.3 Functional Requirements

REQ-1: The system must track the time spent on each topic by the user and log completion dates.

REQ-2: The system must generate a badge upon course completion and store it in the user’s profile.

REQ-3: The system must create a certificate including course details, teacher name, website name, and a QR code.

REQ-4: The system must allow users to view their badges and download certificates.

REQ-5: The system must handle errors where progress tracking fails and notify both users and teachers.

**4.4 AI-Powered Summarization and Chatbot**

4.4.1 Description and Priority

Description: Provides AI-powered summarization for PDFs and videos and a chatbot for real-time doubt clarification.

Priority: High

Benefit: 9

Penalty: 6

Cost: 8

Risk: 7

4.4.2 Stimulus/Response Sequences

1. User Action: Teacher uploads a YouTube video or PDF for a course.

System Response: The AI generates a summary for the content.

2. User Action: User asks a question via the chatbot related to the video or PDF.

System Response: The AI chatbot provides an accurate answer based on the course content.

4.4.3 Functional Requirements

REQ-1: The system must use AI to generate summaries for PDFs and video content.

REQ-2: The chatbot must respond to user queries regarding course content, powered by AI.

REQ-3: The system must ensure that AI-generated summaries are accurate and accessible to users.

REQ-4: The system must integrate the chatbot seamlessly into the user interface for easy access.

REQ-5: The system must handle cases where the AI or chatbot encounters an error and provide fallback options (e.g., redirecting to a teacher).

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

* The platform must support simultaneous access by up to 10,000 users, ensuring smooth content browsing and course progress tracking without noticeable delays.AI-driven video/pdf summarization must generate results within 5 seconds.
* Course searches should return results in under 2 seconds, even under high-load conditions.
* The platform must process at least 500 concurrent quizzes or tests during peak hours to ensure seamless mock test-taking experiences.
* GenAI-powered responses (e.g., doubt clarification) should be generated in less than 1 second.
* System uptime should be maintained at 99.9%, with downtime limited to scheduled maintenance during non-peak hours.

**5.2 Safety Requirements**

* The platform must have safeguards in place to prevent accidental deletion of course materials, user progress data, and assessment results.
* Regular automated backups must be performed daily to prevent data loss, ensuring quick recovery in case of system failures or breaches.
* The platform must comply with relevant education technology safety standards (such as FERPA) and adhere to local and international data protection laws.
* Emergency shutdown procedures must ensure that in-progress mock tests or doubt clarification sessions are recoverable when the system comes back online.
* Regular security audits should be performed to ensure the platform complies with educational standards, such as ISO 27701 for privacy management.

**5.3 Security Requirements**

* All user data, including personal details and sensitive information (such as certificates, badges, and performance data), must be encrypted at rest and in transit using SSL/TLS protocols.
* Multi-factor authentication (MFA) should be enforced for admins and teachers, with optional MFA available for regular users.
* Role-based access control (RBAC) must restrict access to sensitive features (e.g., course creation, certificate management) based on user roles (admin, teacher, student).
* Regular penetration testing should be conducted to identify vulnerabilities and ensure data integrity.
* In the event of a breach or suspected data compromise, incidents must be reported immediately, and actions must be taken in line with predefined protocols (including notifying affected users).

**5.4 Software Quality Attributes**

* Reliability: The system should have an uptime of 99.9%, and any failure in course submission or test-taking should be recoverable within 2 minutes.
* Usability: The platform’s interface must be user-friendly, requiring no more than 3 clicks to access a course, submit a test, or ask for doubt clarification using the chatbot.
* Maintainability: The codebase should be modular, allowing for easy updates, with clear documentation for future developers to maintain or extend the platform.
* Scalability: The platform must be scalable both vertically (by improving server capabilities) and horizontally (by adding new servers) to accommodate a growing user base and future features.
* Portability: The platform must be deployable on cloud services (e.g., AWS, Azure, GCP) and be accessible across mobile, tablet, and desktop devices.
* Testability: Automated testing must cover at least 90% of the codebase to ensure efficient identification and resolution of bugs during development and after deployment.

**5.5 Business Rules**

* Only verified teachers and admins can create and manage courses, upload content, and manage enrolled users.
* Admins must review and verify teachers before they are allowed to manage their courses.
* Teachers can track the progress of students enrolled in their courses and award badges or certificates based on predefined completion criteria.
* Refunds for courses (if applicable) must be processed through an admin approval system.
* The system must provide automatic eligibility checks for badges and certificates based on course completion and performance.
* GenAI-powered doubt resolution and mock tests must be made available only to users enrolled in specific courses.
* Admins can oversee all activities on the platform (e.g., teacher verification, course content management), but they cannot directly modify student assessment data or bypass system protocols.

# **Software Development Life Cycle (SDLC)**

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

● SDLC is the acronym of Software Development Life Cycle.

● It is also called the Software Development Process.

● SDLC is a framework defining tasks performed at each step in the software development process.

● ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

## **What is SDLC?**

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.

**A typical Software Development Life Cycle consists of the following stages**

### **Stage 1: Planning and Requirement Analysis**

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

### **Stage 2: Defining Requirements**

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

### **Stage 3: Designing the Product Architecture**

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters such as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

### **Stage 4: Building or Developing the Product**

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

### **Stage 5: Testing the Product**

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

### **Stage 6: Deployment in the Market and Maintenance**

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

## **SDLC Models**

There is various software development life cycle models defined and designed which are followed during the software development process. These models are also referred to as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry −

● Waterfall Model

● Iterative Model

● Spiral Model

● V-Model

● Big Bang Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

**SRS Document**

**1.Introduction**

**1.1 Purpose of Document**

Provide an introductory paragraph explaining the purpose of this document. Its purpose is to explicitly cite all functions that the project shall do. This document is the primary document, upon which the design, source code, and test plan all base their content. This document is used to determine if the final delivered product provides everything that it was supposed to. The Client, User, and Software Engineering representatives often negotiate the content of this document.

**1.2 Scope**

Provide two paragraphs, the first describing the scope of the product, with the second describing the scope of this document. Remember that "scope" basically means the extent of activity or influence, or range of operation. Be sure that the two paragraphs in this section distinguish between the scope of the product, versus the scope of this document.

You will probably find that in most of the Software Engineering documents that you create in this course, the paragraph for scope of product will be identical (as expected). Specifically for this document, the scope includes all team members and their responsibilities for specifying the product's requirements.

**1.3 Objective**

A project objective describes the desired results of a project, which often includes a tangible item. An objective is specific and measurable, and must meet time, budget, and quality constraints. ... A project may have one objective, many parallel objectives, or several objectives that must be achieved sequentially.

**1.4 Proposed System**

The proposed system should have the following features. The transactions should take place in a secured format between various clients in the network. It provides flexibility to the user to transfer the data through the network very easily by compressing the large amount of file.

**2. Requirements Specifications**

**2.1 Functional Requirements**

functional requirement defines a function of a [system](https://en.wikipedia.org/wiki/System) or its component, where a function is described as a specification of behavior between outputs and inputs.

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in [use cases](https://en.wikipedia.org/wiki/Use_case). Functional requirements are supported by [non-functional requirements](https://en.wikipedia.org/wiki/Non-functional_requirement) (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>," while non-functional requirements take the form "system shall be <requirement>."The plan for implementing functional requirements is detailed in the system design, whereas *non-functional* requirements are detailed in the system architecture.

**2.2 Non-Functional Requirements**

Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs. Also known as system qualities, nonfunctional requirements are just as critical as functional Epics, Capabilities, Features, and Stories. They ensure the usability and effectiveness of the entire system. Failing to meet any one of them can result in systems that fail to satisfy internal business, user, or market needs, or that do not fulfill mandatory requirements imposed by regulatory or standards agencies. In some cases, non-compliance can cause significant legal issues (privacy, security, safety, to name a few).

**2.3 Software Requirements**

[Software requirements](https://en.wikipedia.org/wiki/Software_requirements) deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

**2.4 Hardware Requirements**

The most common set of requirements defined by any [operating system](https://en.wikipedia.org/wiki/Operating_system) or [software application](https://en.wikipedia.org/wiki/Software_application) is the physical computer resources, also known as [hardware](https://en.wikipedia.org/wiki/Computer_hardware), A hardware requirements list is often accompanied by a [hardware compatibility list](https://en.wikipedia.org/wiki/Hardware_compatibility_list) (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following subsections discuss the various aspects of hardware requirements.

**3. Literature Survey**

A literature survey or a literature review in a project report is that section which shows the various analyses and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the extent of the project.

It is the most important part of your report as it gives you a direction in the area of your research. It helps you set a goal for your analysis - thus giving you your problem statement.

When you write a literature review in respect of your project, you have to write the researches made by various analysts - their methodology (which is basically their abstract) and the conclusions they have arrived at. You should also give an account of how this research has influenced your thesis.

Descriptive papers may or may not contain reviews, but analytical papers will contain reviews. A literature review must contain at least 5 - 7 published researches in your field of interest.

**4.System Designing**

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

# **Diagrams in the UML**

We prepare UML diagrams to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system.

**1.** **Activity Diagrams –** We use Activity Diagrams to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on the condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.

**2.** **Use Case Diagrams –** Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents(actors). A use case is basically a diagram representing different scenarios where the system can be used. A use case diagram gives us a high-level view of what the system or a part of the system does without going into implementation details.

**3.** **Sequence Diagram –** A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

**4.** **Class Diagram –** The most widely used UML diagram is the class diagram. It is the building block of all object-oriented software systems. We use class diagrams to depict the static structure of a system by showing the system's classes, their methods and attributes. Class diagrams also help us identify relationships between different classes or objects.

**5. Implementation**

The software implementation stage involves the transformation of the software technical data package (TDP) into one or more fabricated, integrated, and tested [software configuration](https://www.sciencedirect.com/topics/computer-science/software-configuration) items that are ready for software acceptance testing. The primary activities of software implementation include the:

● Fabrication of software units to satisfy structural unit specifications.

● Assembly, integration, and testing of software components into a [software configuration item](https://www.sciencedirect.com/topics/computer-science/software-configuration-item).

● Prototyping challenging software components to resolve implementation risks or establish a fabrication proof of concept.

● Dry-run acceptance testing procedures to ensure that the procedures are properly delineated and that the software product (software configuration items (CIs and computing environment) is ready for acceptance testing.

**6. Testing**

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises Validation and Verification.

## **Software Validation**

Validation is the process of examining whether or not the software satisfies the user requirements. It is carried out at the end of the SDLC. If the software matches requirements for which it was made, it is validated.

● Validation ensures the product under development is as per the user requirements.

● Validation answers the question – "Are we developing the product which attempts all that user needs from this software?".

● Validation emphasizes on user requirements.

## **Software Verification**

Verification is the process of confirming if the software is meeting the business requirements, and is developed adhering to the proper specifications and methodologies.

● Verification ensures the product being developed is according to design specifications.

● Verification answers the question– "Are we developing this product by firmly following all design specifications?"

● Verifications concentrate on the design and system specifications.

**7.Conclusion**

SRS helps the customers to define their needs with accuracy, while it helps the development team understand what the customers need in terms of development. Investing time in writing the SRS document will lead to the successful development of the software the customer needs.

**SOFTWARE REQUIREMENTS**

**Functional Requirements:**

● These are statements of services the system should provide

=>how the system should react to particular inputs and

=>how the system should behave in particular situations

● In some cases, the functional requirements may also explicitly state

=> What the system should not do

● The functional requirements definition of a system should be both

=> Complete [i.e. It means that all services required by the user should be defined]

=> Consistent [i.e. it means that requirements should not has contradictory definitions]

**Non- Functional Requirements:**

● These are constraints on the services (Or) functions offered by the system

● They include

=> Timing Constraints

=> Constraint on development process

=> Standards and so on…

● Some non-functional requirements may be process rather than product requirements

● Customer imposes these process requirements for two reasons:

=> System Quality

=> System Maintainability

**Non-Functional Requirements Types:**

Product Requirements Process Requirements External Requirements

**(i) Product Requirements:**

These requirements result from the need for the delivered product, to behave in a particular way

Example:

● Requirements on how fast the system must execute and how much memory it

requires

● Reliability Requirements [i.e, acceptable failure rate]

● Portability Requirements

**(ii) Organizational Requirements:**

● These requirements are consequence of organizational policies and procedures

Example:

Implementation requirements such as programming language (Or) design method

used

● Delivery Requirements which specify when the product and its documentation to be

Delivered

**(iii) External Requirements:**

● These requirements arise from factors external to the system and its development

process

Example:

● Interoperability Requirements which specify how the system interacts with systems in

other organizations

● Legislative Requirements, which ensure that the system operates within the law

**An Overview of UML**

Unified Modeling Language (UML) is a general-purpose modelling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language; it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system. UML helps software engineers, businessmen and system architects with modelling, design and analysis. The Object Management Group (OMG) adopted Unified Modelling Language as a standard in 1997. It’s been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

## **A Conceptual Model of UML**

● A conceptual model can be defined as a model which is made of concepts and their relationships.

● A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other.

As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. The conceptual model of UML can be mastered by learning the following three major elements −

● UML building blocks

● Rules to connect the building blocks

● Common mechanisms of UML

### 

### 

### **Object Oriented Concepts Used in UML –**

**1. Class –** A class defines the blueprint i.e. structure and functions of an object.

**2.Objects –** Objects help us to decompose large systems and help us to modularize our system. Modularity helps to divide our system into understandable components so that we can build our system piece by piece. An object is the fundamental unit (building block) of a system which is used to depict an entity.

**3.Inheritance –** Inheritance is a mechanism by which child classes inherit the properties of their parent classes.

**4.Abstraction –** Mechanism by which implementation details are hidden from the user.

**5.Encapsulation –** Binding data together and protecting it from the outer world is referred to as encapsulation.

**6.Polymorphism –** Mechanism by which functions or entities are able to exist in different forms.

# **Diagrams in the UML**

We prepare UML diagrams to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system.

There are two broad categories of diagrams and they are again divided into subcategories −

**1.Structural Diagrams –** Capture static aspects or structure of a system. Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.

**2.Behavior Diagrams –** Capture dynamic aspects or behavior of the system. Behavior diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

## **Structural Diagrams**

The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable.

These static parts are represented by classes, interfaces, objects, components, and nodes. The four structural diagrams are −

● Class diagram

● Object diagram

● Component diagram

● Deployment diagram

**1.Class Diagram**

Class diagrams are the most common diagrams used in UML. Class diagrams consist of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature.

Active class is used in a class diagram to represent the concurrency of the system.

Class diagrams represent the object orientation of a system. Hence, it is generally used for development purposes. This is the most widely used diagram at the time of system construction.

**2.Object Diagram**

Object diagrams can be described as an instance of class diagrams. Thus, these diagrams are closer to real-life scenarios where we implement a system. Object diagrams are a set of objects and their relationship is just like class diagrams. They also represent the static view of the system. The usage of object diagrams is similar to class diagrams but they are used to build a prototype of a system from a practical perspective.

**3.Component Diagram**

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system.

During the design phase, software artifacts (classes, interfaces, etc.) of a system are arranged in different groups depending upon their relationship. Now, these groups are known as components. Finally, it can be said component diagrams are used to visualize the implementation.

**4.Deployment Diagram**

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed. Deployment diagrams are used for visualizing the deployment view of a system. This is generally used by the deployment team.

## **Behavioral Diagrams**

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered. Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

UML has the following five types of behavioral diagrams −

● Use case diagram

● Sequence diagram

● Collaboration diagram

● Statechart diagram

● Activity diagram

**1.Use Case Diagram**

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. Hence, a use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

**2.Sequence Diagram**

A sequence diagram is an interaction diagram. From the name, it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another.

Interaction among the components of a system is very important from implementation and execution perspective. Sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality.

**3.Collaboration Diagram**

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links.

The purpose of the collaboration diagram is similar to a sequence diagram. However, the specific purpose of collaboration diagrams is to visualize the organization of objects and their interaction.

**4.Statechart Diagram**

Any real-time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system.

Statechart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface, etc. State chart diagram is used to visualize the reaction of a system by internal/external factors.

**5.Activity Diagram**

Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.

**Diagram Elements**

Some of the graphical constructs from which diagrams are made are:

● Icon: graphical symbol of fixed size and shape (doesn't hold contents)

● Two-dimensional symbols: have variable size and can expand to hold contents, may be divided into compartments

● Paths: sequences of line segments with attached endpoints. The endpoints are always symbols (no dangling paths). May also have icons at the end to qualify the meaning of the path symbol.

● Strings: text

● Name: A string that uniquely identifies some model element within some scope

● Label: A string attached to a graphic symbol

● Keyword: Text enclosed within "«" and "»" to convey some concept. There are many keywords so we don't need zillions of specialized graphical symbols.

● Expression: A linguistic formula that yields a value

● Some model elements:

