Online Course Reservation System

1. A Meher Archana - 22BD1A0541
2. D Sreeja - 22BD1A052G
3. D Laxmi Sathwik - 22BD1A052E
4. A Hasini - 22BD1A0523
5. B Nithya Santhoshini -22BD1A052B

Problem Statement:

Online Course Reservation Systems provide a convenient and flexible way for students to explore and enroll in a wide variety of courses. However, learners often face challenges in finding courses that precisely match their interests and needs. These platforms must offer an efficient and user-friendly course search experience, enabling users to filter courses by factors such as subject, skill level, duration, language, instructor ratings, and other relevant criteria. As users apply more filters, the search should progressively narrow down to a refined list of suitable courses.

A key challenge lies in maintaining the accuracy of course categorization and details. Misrepresenting course information can lead to irrelevant or misleading search results, frustrating users. Additionally, the system should ensure that courses are promptly updated—new courses must be added as soon as they become available, while full or discontinued courses should be removed or marked as unavailable in real-time. This ensures that search results always reflect current course availability.

Moreover, ensuring data security is paramount. The platform must implement strong protections against data breaches and unauthorized access, safeguarding user information. Lastly, the user interface must be intuitive, responsive, and accessible, providing students with a smooth and efficient course discovery and enrollment experience.

Software Requirement Specification

For

Online Course Reservation System

Version 1.0 approved

Prepared by:

1. A Meher Archana - 22BD1A0541
2. D Sreeja - 22BD1A052G
3. D Laxmi Sathwik - 22BD1A052E
4. A Hasini - 22BD1A0523
5. B Nithya Santhoshini -22BD1A052B

Keshav Memorial Institute of Technology

19-10-2024.

Table of Contents

6. Other Requirements

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) outlines the requirements for the development of an Online Course Reservation System. The platform will enable students to search, filter, and enroll in courses based on various criteria such as subject, skill level, and instructor ratings. The document focuses on the core functionality of the platform, excluding the backend learning management system (LMS) and user content delivery aspects. This SRS covers Version 1.0 of the platform and details its search, filtering, course management, and user interface requirements.

1.2 Document Conventions

This document follows standard conventions for clarity and uniformity. Bold text indicates section headers or important terms, and italic text is used to emphasize key requirements. Priority levels are assigned as follows:

- Must Have: Critical features that are essential for product functionality.
- Should Have: Important features that add value but are not mission-critical.
- Could Have: Desirable features that may be considered if resources allow.

1.3 Intended Audience and Reading Suggestions

The intended audience for this SRS includes:

- Developers: To understand functional and non-functional requirements.
- Project Managers: To monitor progress and ensure all requirements are met.
- Marketing and Sales: To understand key features that appeal to users.
- Testers: To develop test plans based on specified requirements.

- End-users and Documentation Writers: For understanding platform features and user flows.

It is recommended to start with the Product Scope section for a high-level overview, followed by Functional Requirements for detailed feature descriptions.

1.4 Product Scope

The Online Course Reservation System aims to provide a seamless and intuitive experience for learners to search, filter, and enroll in courses. It ensures accuracy in course representation, real-time updates for course availability, and secure handling of user data. The platform will help users find relevant courses based on customizable criteria, enhancing the user experience and streamlining the course discovery process. This platform supports the educational goals of both individuals and educational institutions.

Key objectives:

- Efficient course search and filtering system.
- Real-time course availability updates.
- Secure user data protection.
- Intuitive and accessible user interface.

2. Overall Description

2.1 Product Perspective

The Online Course Reservation System is a new, standalone product designed to provide users with an enhanced course search, filtering, and enrollment experience. The platform is not part of a product family or replacing any existing system; it is a self-contained product that integrates with external systems, such as Learning Management Systems (LMS), for delivering courses. While the platform focuses on course discovery and user enrollment, it assumes that the LMS handles course delivery, assessments, and content management. The primary interface between this platform and the LMS is the course availability and enrollment data exchange.

2.2 Product Functions

The product will perform the following major functions:

- Course Search and Filtering: Users can search and filter courses based on various criteria such as subject, skill level, duration, language, instructor ratings, etc.
- Course Details and Enrollment: Display detailed course information (descriptions, ratings, availability) and allow users to enroll in selected courses.
- User Profile Management: Users can create and manage profiles, track course enrollments, and view progress.
- Real-time Course Availability Updates: Ensure courses are marked available or unavailable in real-time.
- Security and Authentication: Secure login, data protection, and user role management (students, instructors, admins).
- User Notifications: Send notifications to users about course enrollment confirmations, updates, or recommendations.

2.3 User Classes and Characteristics

The platform is expected to cater to the following user classes:

- Students: The primary users who will search, filter, and enroll in courses. They may vary in technical expertise and will require an intuitive interface to assist with course discovery and registration.
- Instructors: Users who may upload and manage course content on connected LMS systems. They will need access to manage course availability and enrollment.
-Administrators: Users with higher privileges, responsible for managing the platform, troubleshooting, and overseeing system operations.
-Guest Users: Unregistered users who can browse through course catalogs but must register to enroll in courses.

2.4 Operating Environment

The platform will operate in the following environments:

- Web-based platform: Accessible through modern web browsers such as Chrome, Firefox, Safari, and Edge.
- Mobile Support: The platform should be mobile-friendly and optimized for access on mobile devices.

- Operating Systems: Compatible with major operating systems, including Windows, macOS, Linux, Android, and iOS.
- Server Environment: Backend infrastructure will be hosted on cloud platforms (e.g., AWS, Azure), with scalability and high availability considerations.
- Database: Relational or NoSQL databases for managing user data, course information, and transactional records.

2.5 Design and Implementation Constraints

Several constraints impact the design and development of the platform:
- Regulatory Compliance: The platform must comply with privacy laws like GDPR and CCPA to protect user data.
- Performance Constraints: Course search and filtering functions must be optimized to handle large data sets efficiently.
- Security Protocols: Encryption for sensitive data, SSL certificates for secure connections, and role-based access controls must be implemented.
- Third-party Integration: Integration with external LMS systems for course data synchronization must follow specific APIs and protocols, which may limit the design.
- Technology Stack: The platform must use a specific set of technologies, including front-end frameworks (React, Angular, Vue) and backend technologies (Node.js, Python, etc.).

2.6 Assumptions and Dependencies

The following assumptions and dependencies apply to the development and deployment of the platform:
- Third-party LMS Integration: The platform depends on external LMS systems for course content delivery and availability updates. These systems are assumed to provide reliable APIs for communication.
- Real-time Updates: It is assumed that the platform will have access to a high-availability infrastructure that supports real-time data updates for course availability.
- Browser Compatibility: Users are expected to access the platform through modern browsers that support JavaScript and HTML5, ensuring the platform's responsive design functions as intended.
- External Authentication: If an external authentication service (OAuth, SSO) is used, it is assumed to be reliable and secure.

3. External Interface Requirements

3.1 User Interfaces

The Online Course Reservation System will include several user interface components designed to provide an intuitive and accessible experience. The following details describe the logical characteristics of each interface between the platform and users:

- Login and Registration Interface: This interface will allow users to create accounts, log in, and manage their profiles. It will include fields for email, password, and basic profile information. Standard buttons like "Login", "Register", and "Forgot Password" will appear. A validation mechanism will ensure correct input formats.

- Course Search and Filtering Interface: A primary interface for users to search for and filter courses based on different criteria such as subject, skill level, instructor, and course duration. This page will feature a search bar, filter options (checkboxes, dropdowns), and pagination for navigating large sets of results. A clear and organized layout will help users refine search results with ease.

- Course Details Interface: When a course is selected, this interface will display all relevant details, including course description, instructor information, course rating, language, and schedule. Buttons for "Enroll", "Add to Wishlist", and "Share" will be prominently placed.

- User Profile Management Interface: This interface will allow users to manage their personal information, view their enrolled courses, and track their progress. Key elements include editable profile fields, notifications, and a dashboard view of enrolled and completed courses.

- Standard GUI Elements:
  - Navigation Bar: Present at the top of each page, providing quick access to Home, Courses, Profile, and Notifications.
  - Footer: A footer with links to help, contact support, and social media pages.

- Keyboard Shortcut: Common keyboard shortcuts will be available for ease of navigation, e.g., pressing "/" to focus on the search bar.
- Error Messages and Notifications: Clear, user-friendly error messages will be displayed in red for validation failures (e.g., invalid login, failed search query). Success notifications will appear in green.

- User Interface Design Specifications: Detailed design elements such as color schemes, fonts, and styles will adhere to the organization's branding guidelines. A separate user interface specification document will cover these elements in depth.

3.2 Hardware Interfaces

The platform will require interaction with various hardware components, specifically for accessing the service across multiple devices:

- Supported Devices:
  - Desktop computers (Windows, macOS, Linux)
  - Tablets and mobile phones (Android, iOS)

- Device Integration:
  - Mobile Responsiveness: The platform will be fully responsive, adjusting its layout to fit smaller screens on mobile devices.
  - Touch Interface: For mobile and tablet users, the platform will support touch-based interactions such as swiping, pinching to zoom, and tapping.
  - Camera Access: In certain cases, such as uploading profile pictures or completing identity verification (optional), the platform may require access to the user's camera.
  - Storage Access: Users will be able to download course materials and upload assignments, requiring access to local storage.

- Communication Protocols: Standard hardware interaction protocols such as USB, Bluetooth, or Wi-Fi for data synchronization between devices, if needed.

3.3 Software Interfaces

The platform will interact with various software components and external systems:

- Operating Systems: The platform will be compatible with major operating systems, including:
  - Windows (10 and above)
  - macOS (latest versions)
  - Linux distributions (e.g., Ubuntu, Fedora)
  - Android and iOS for mobile devices

- Database Interfaces:
  - The platform will interact with a relational database (e.g., MySQL or PostgreSQL) for managing user data, course catalog information, and transactional records.
  - Data Exchange: The system will use SQL queries for reading and writing data between the web server and the database.
  - APIs: An internal API layer will be exposed to interact with the database for actions like fetching course details or updating user profiles.

- Third-party LMS Integration:
  - The platform will interface with external Learning Management Systems (LMS) via REST APIs or GraphQL. These systems will provide course availability data, enrollment confirmation, and content management.
  - APIs : Specific endpoints will be defined for communication between the course search platform and the LMS to exchange course details and enrollment information.
  - OAuth may be used for secure authorization to access these external systems.

- Authentication:
  - Integration with third-party authentication services (e.g., Google, Facebook, LinkedIn) using OAuth 2.0 for user login and account management.
  - Support for JWT (JSON Web Token) for managing user sessions and API security.

- Payment Gateway :
  - The platform will interface with a third-party payment service (e.g., Stripe or PayPal) to handle secure payments for course enrollments.
  - APIs : Payment APIs will process transactions, handle refunds, and manage invoicing.

3.4 Communications Interfaces

The platform will require robust communication capabilities to interact with users and external systems:

- HTTP/HTTPS :
  - All communication between the client and server will be over HTTPS to ensure secure data transfer.
  - The system will use RESTful APIs for interactions between the frontend, backend, and external systems.

- Email Notifications :
  - The platform will send emails to users for enrollment confirmations, course updates, and system notifications.
  - SMTP or an email service provider like SendGrid or Mailchimp will be used to handle outgoing emails.
  - Emails will follow a standard format for transactional and promotional messages.

- WebSocket Protocol :
  - Real-time notifications, such as course availability updates or enrollment status changes, will be managed using WebSockets for instant updates.

- Data Encryption :
  - All data transmitted between users and the platform, especially sensitive information (e.g., payment details, passwords), will be encrypted using TLS (Transport Layer Security).

- APIs and Protocols :
  - REST or GraphQL will be used for fetching data between different software components.
  - FTP/SFTP may be used for bulk data transfers (e.g., uploading large course materials to the server).

- Security :

- The platform will use SSL certificates to ensure that all communications over the internet are encrypted and secure.

4. System Features

4.1 Course Search and Filtering

4.1.1 Description and Priority

This feature allows users to search for courses using various filters, such as subject, skill level, duration, language, and instructor rating. The platform must provide an efficient and accurate search mechanism. This is a High priority feature, as it directly impacts the user experience and ability to find relevant courses.

4.1.2 Stimulus/Response Sequences

- Stimulus : The user enters a keyword into the search bar or selects a combination of filters.
- Response : The system displays a refined list of courses that match the user's criteria in real-time.
- Stimulus : The user adds more filters to narrow down the search results.
- Response : The search results are updated based on the added filters, displaying only the most relevant courses.
- Stimulus : The user clears all filters.
- Response : The system resets and displays the entire list of available courses.

4.1.3 Functional Requirements

- REQ-1 : The system must provide a search bar for users to input course-related keywords.
- REQ-2 : The system must allow users to apply multiple filters, including subject, skill level, course duration, language, and instructor rating.
- REQ-3 : The system must update search results in real-time as filters are applied.
- REQ-4 : The system must display accurate course information, including title, description, duration, instructor, and price.

- REQ-5 : The system must handle invalid or non-existent search queries gracefully by showing a "no results found" message.
- REQ-6 : The system must allow users to clear all filters with a single action.

4.2 Course Enrollment

4.2.1 Description and Priority

This feature allows users to enroll in courses they are interested in. It is a High priority feature, as it directly enables the core functionality of the platform, which is course enrollment.

4.2.2 Stimulus/Response Sequences

- Stimulus : The user clicks on the "Enroll" button for a selected course.
- Response : The system processes the enrollment, deducts payment if applicable, and confirms the enrollment with a success message.
- Stimulus : The user cancels the enrollment.
- Response : The system cancels the enrollment, updates the user's course list, and refunds any applicable payments.

4.2.3 Functional Requirements

- REQ-1 : The system must allow users to enroll in a course by clicking an "Enroll" button.
- REQ-2 : The system must process payments securely using an integrated payment gateway.
- REQ-3 : The system must confirm successful enrollment by displaying a confirmation message.
- REQ-4 : The system must add the enrolled course to the user's profile and allow access to course materials.
- REQ-5 : The system must handle enrollment cancellations and issue refunds when applicable.
- REQ-6 : The system must restrict enrollment in full or unavailable courses.

4.3 User Profile Management

4.3.1 Description and Priority

This feature enables users to manage their profiles, update personal information, and view their enrolled courses. It is a Medium priority feature.

4.3.2 Stimulus/Response Sequences

- Stimulus : The user updates their personal information (e.g., name, email, password) in their profile settings.
- Response : The system validates and saves the changes.
- Stimulus : The user views their enrolled courses.
- Response : The system displays a list of courses the user has enrolled in, along with their progress.

4.3.3 Functional Requirements

- REQ-1 : The system must allow users to update their personal information in their profiles.
- REQ-2 : The system must display a list of courses the user is currently enrolled in.
- REQ-3 : The system must track and display the progress of each enrolled course.
- REQ-4 : The system must validate user input (e.g., email format, password strength) when updating profile details.
- REQ-5: The system must allow users to change their account password securely.

4.4 Payment Management

4.4.1 Description and Priority

This feature manages user payments for course enrollments and provides transaction history. It is a High priority feature as it ensures the financial operations of the platform.

4.4.2 Stimulus/Response Sequences

- Stimulus The user initiates a payment for a course enrollment.

- Response: The system processes the payment securely and updates the user's profile with the newly enrolled course.
- Stimulus: The user views their payment history.
- Response : The system displays a list of completed transactions, including the course title, payment date, and amount.

4.4.3 Functional Requirements

- REQ-1: The system must integrate with a third-party payment gateway (e.g., Stripe, PayPal) for processing transactions.
- REQ-: The system must display transaction details, including the course, date, and amount.
- REQ-3 : The system must handle and display error messages for failed transactions.
- REQ-4 : The system must ensure all transactions are encrypted and secure.
- REQ-5 : The system must support multiple payment methods (credit/debit cards, digital wallets, etc.).

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- System Response Time : The system must respond to user actions (e.g., course search, enrollment) within 2 seconds under normal operating conditions.

- Scalability : The platform must handle a minimum of 10,000 concurrent users without performance degradation.

- Data Processing Speed : Search results should be displayed within 3 seconds, even when filters are applied.

- Load Capacity : The system must support up to 500,000 courses and handle daily updates in real-time, reflecting changes (new courses, enrollment limits) within 10 minutes of the update.

- Uptime: The system must maintain 99.9% uptime, ensuring continuous availability for users across different time zones.

5.2 Safety Requirements

- Data Integrity: The platform must prevent loss or corruption of user data in the event of system crashes or unexpected shutdowns by employing regular backups and redundant systems.

- Safe Payment Processing: Users should be informed if any transactions fail, and the system must prevent double payments by ensuring transaction uniqueness.

- Error Handling: Critical system failures must be logged, and appropriate recovery actions should be initiated. For example, if course information is incorrectly displayed due to system malfunction, users should be alerted and the information corrected immediately.

5.3 Security Requirements

- Authentication: The platform must require secure user authentication using a combination of email/username and a strong password. Two-factor authentication (2FA) should be optional but recommended for all users.

- Data Encryption: All sensitive user data (e.g., personal information, payment details) must be encrypted during transmission (using HTTPS) and at rest in the database.

- Role-Based Access Control (RBAC): Different user classes (e.g., students, instructors, administrators) must have specific access privileges. For instance, only instructors and admins should be able to add, update, or remove course content.

- User Privacy: The platform must comply with global data privacy standards such as GDPR, ensuring that users have control over their data (e.g., account deletion, consent for data sharing).

- Security Certifications: The platform should aim to be compliant with ISO/IEC 27001 or equivalent security standards.

5.4 Software Quality Attributes

- Usability: The platform must be intuitive, with a simple and clean interface. Usability testing should be performed to ensure ease of use for all user types, including those with limited technical experience.

- Adaptability: The system should be adaptable to future updates, with new features being easy to integrate without major disruptions.

- Maintainability: The system's codebase must be well-documented and follow standard coding practices to ensure it is easy to maintain, update, and debug by development teams.

- Interoperability: The system should integrate seamlessly with external tools (e.g., learning management systems, payment gateways) using standardized APIs.

- Portability: The platform should work on multiple devices (desktops, tablets, smartphones) and across major web browsers (Chrome, Firefox, Safari, Edge).

- Reliability: The system must be able to handle common errors gracefully (e.g., lost internet connection) and recover without data loss.

- Testability: All components of the system must be testable, with automated testing available for critical functionalities (e.g., user authentication, payment processing).

5.5 Business Rules

- User Roles: Users must be assigned roles such as student, instructor, or administrator, with permissions appropriate to their role. For example, only administrators can manage user accounts, and only instructors can create or modify course content.

- Refund Policies: Users must be able to cancel enrollments and request refunds within a defined time frame (e.g., within 14 days of enrollment), as per the platform's refund policy.

- Course Availability: Instructors must update the availability status of courses (e.g., full, discontinued) promptly, ensuring the system reflects real-time course availability.

-Enrollment Limits: The system must prevent users from enrolling in courses that have reached their maximum capacity.

- Compliance: The platform must comply with local education regulations (e.g., accreditation rules for courses) and ensure that content adheres to applicable laws.

Appendix A: Glossary

- SRS (Software Requirements Specification): A document that describes the software system to be developed, including its functionality, performance, and constraints.

- API (Application Programming Interface): A set of tools and protocols that allows different software applications to communicate with each other.

- GUI (Graphical User Interface): A user interface that includes graphical elements, such as windows, buttons, and icons, for user interaction.

- RBAC (Role-Based Access Control): A method of regulating access to computer resources based on the roles of individual users.

- 2FA (Two-Factor Authentication): A security process in which a user provides two different authentication factors to verify their identity.

- GDPR (General Data Protection Regulation): A regulation in EU law that focuses on data protection and privacy for individuals within the European Union.

- HTTPS (Hypertext Transfer Protocol Secure): A protocol for secure communication over a computer network, commonly used on the internet.

- ISO/IEC 27001: An international standard for managing information security.

- Scalability: The capability of a system to handle a growing amount of work, or its potential to accommodate growth.

- Usability: The ease of use and learnability of a system or product for users.

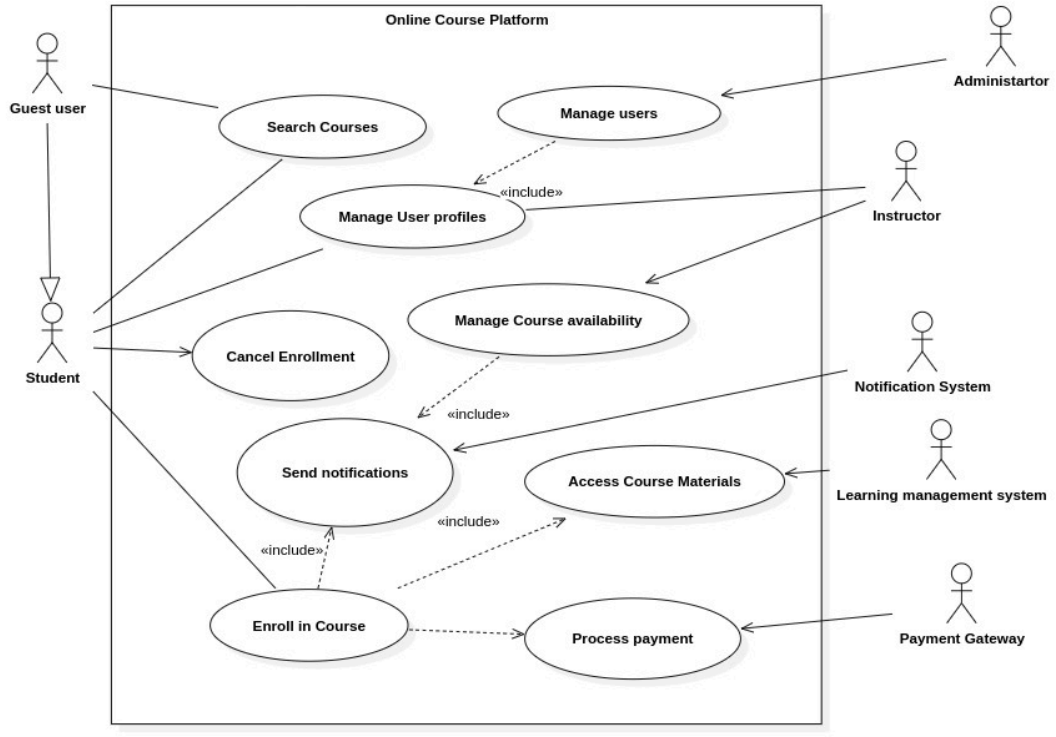
Appendix B: Analysis Models

- Use Case template

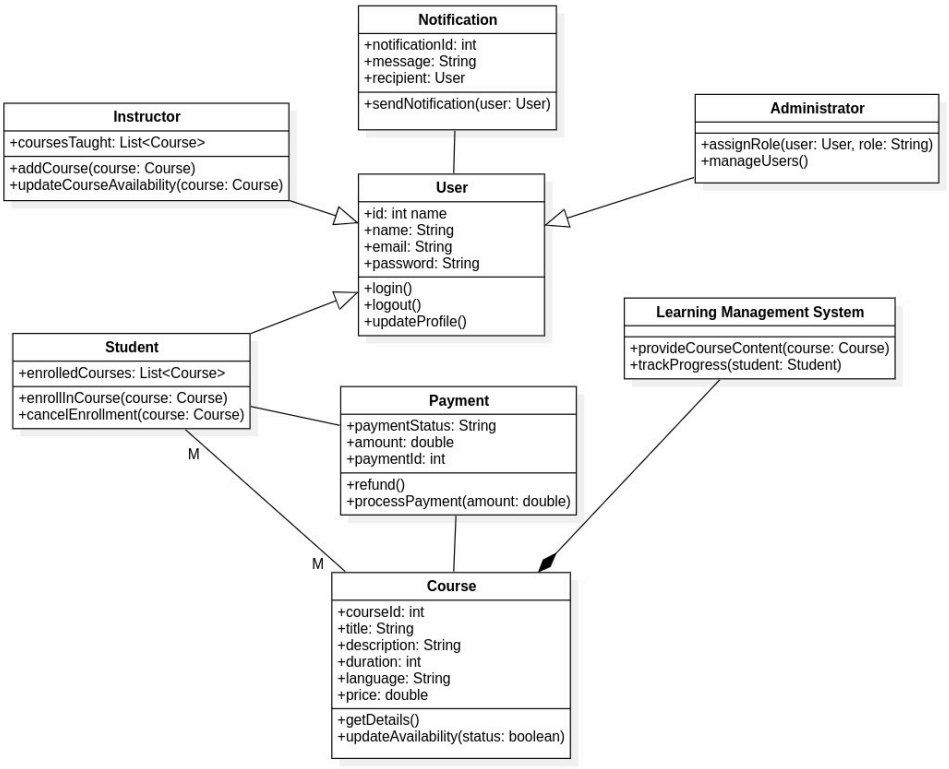
Use Case ID	UC-05
Use Case Name	Search Courses
Description	Allows users to search for available courses based on various filters such as category, duration, and language.
Actors	Guest User, Student
Preconditions	User must have access to the online course platform.
Postconditions	The user receives a list of courses matching their search criteria.
Basic Flow (Main Success Scenario)	1. User navigates to the search page. 2. User enters search criteria. 3. System displays a list of courses.
Alternative Flows (Extensions)	1. No courses found: - System displays a message indicating no results. 2. User refines search criteria.
Exceptions	1. System error during search: - System displays an error message.
Priority	High
Frequency of Use	Frequently (daily)
Assumptions	Users are familiar with the search functionality and criteria.
Notes	This use case is essential for user engagement and

	course enrollment.

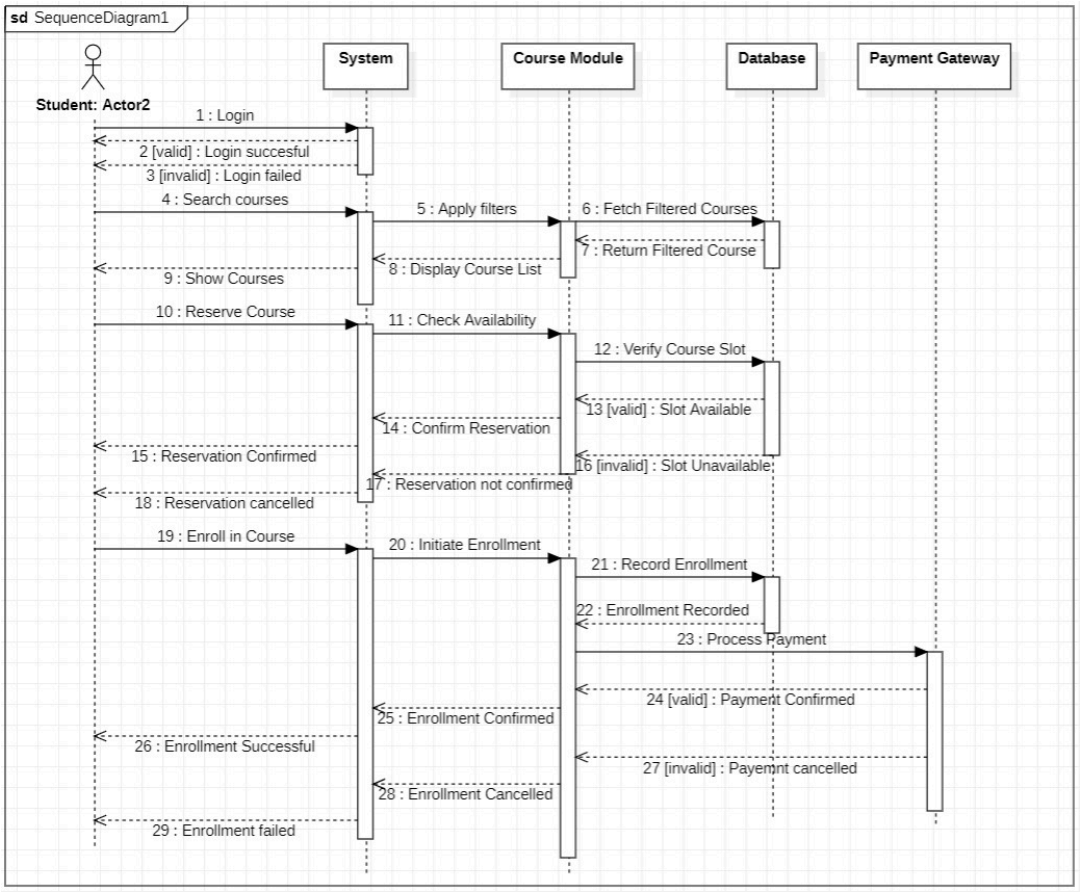
- Use Case Diagram:



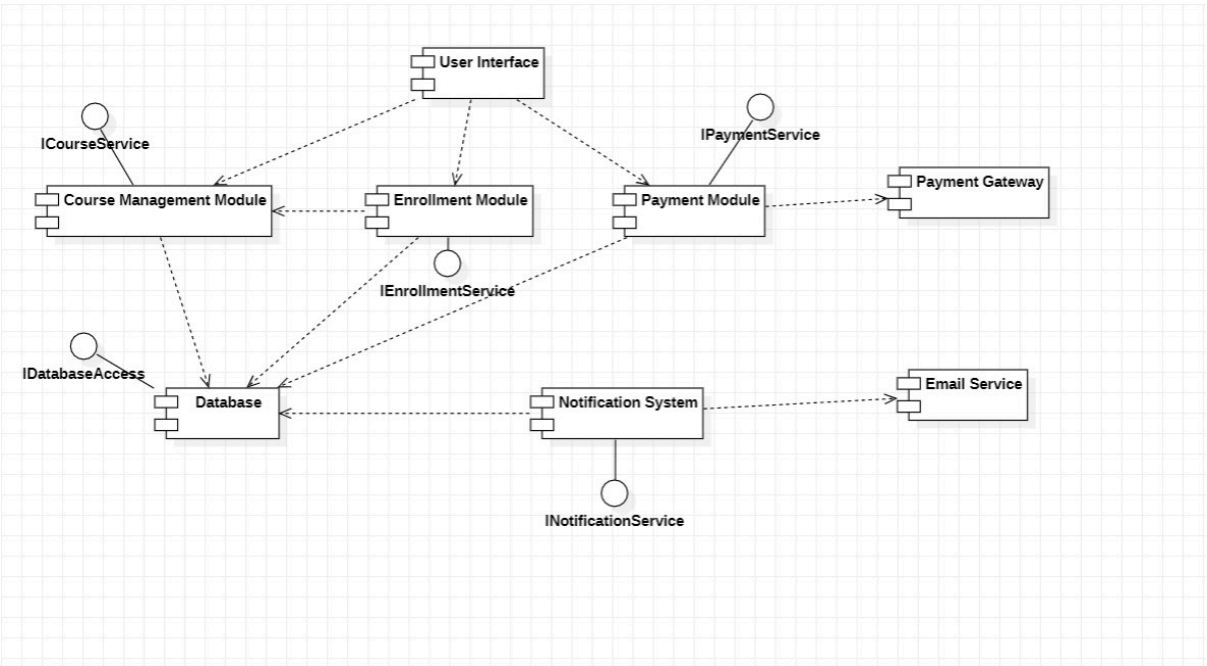
- Class Diagram:



- Sequence Diagram:



- Component Diagram:



Appendix C: To Be Determined List

1. TBD-1: Finalization of user interface design and layout specifications for the course search page.

2. TBD-2: Specific encryption standards and protocols to be used for data at rest.

3. TBD-3: Definition of performance benchmarks for different user roles under varying system loads.

4. TBD-4: Decision on external payment gateway to be integrated with the platform.

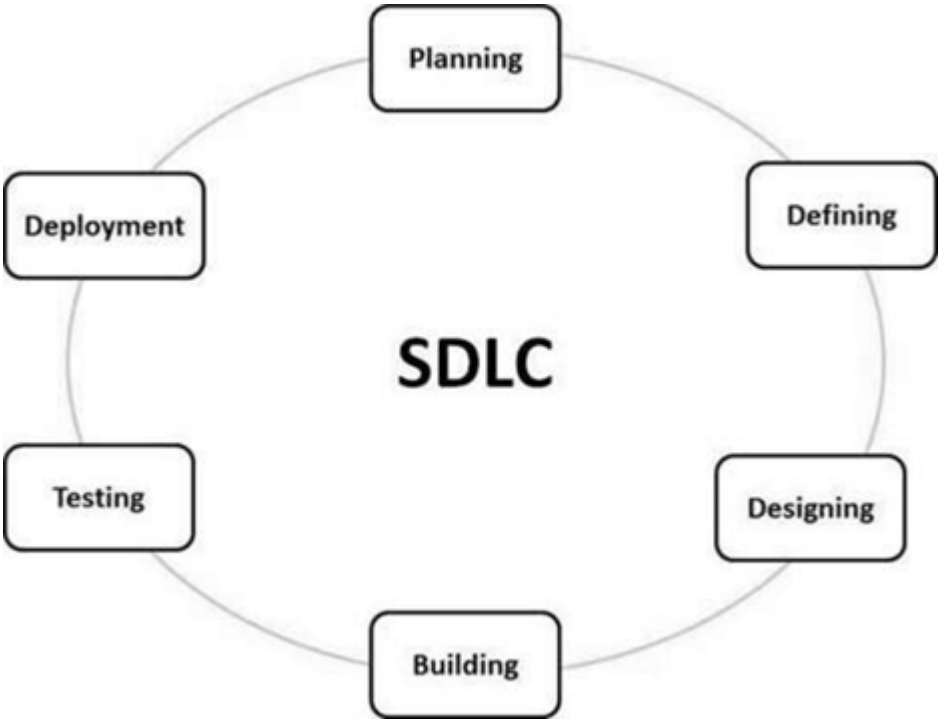
5. TBD-5: Selection of tools for automated testing of key system functions (e.g., course enrollment, payment processing).

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 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. Functional requirements are supported by non-functional requirements (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 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 or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a 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 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.

- 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

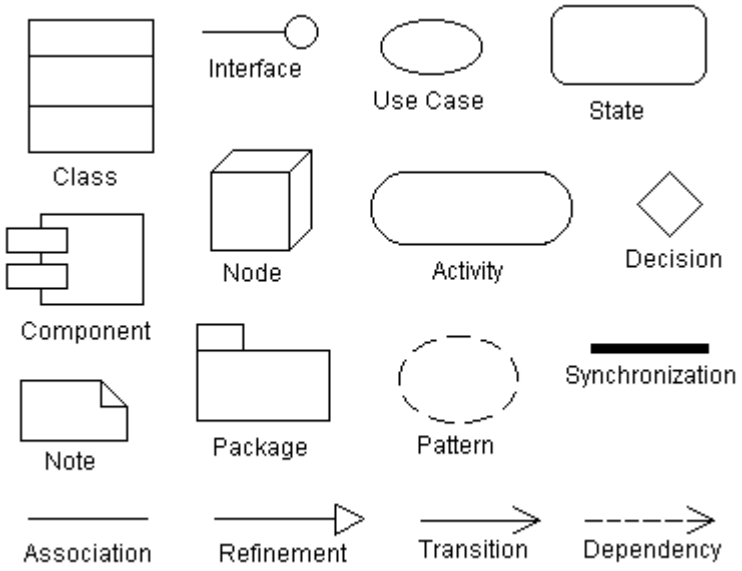
- Name: A string that uniquely identifies some model element within some scope

- 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:


Class
Interface
Use Case
State
Node
Activity
Decision
Component
Synchronization
Note
Package
Pattern
Association
Refinement
Transition
Dependency