**CourseTide**

1. Mohammed Abdur Rasheed - 22BD1A0538
2. Nehith Sayini - 22BD1A053F
3. Laksh Vijayvargiya - 22BD1A050X
4. Jyothi Swaroop - 22BD1A050D

**Problem Statement:**

Online education has become a crucial part of learning, offering students flexibility and convenience. However, many course reservation platforms focus only on enabling users to enroll in courses, neglecting critical aspects such as peer collaboration and personalized support. This lack of engagement and guidance can hinder students' overall learning experience, as they often struggle to find study partners or receive real-time assistance with course-related queries. Course Tide aims to address these issues by providing a unique, feature-rich online course reservation system that emphasizes both collaborative learning and interactive support.

One of the primary challenges students face in existing systems is the inability to engage with peers who are enrolled in the same course. Collaborative learning is a proven method for enhancing knowledge retention and motivation, yet many platforms fail to facilitate group interaction. Course Tide’s Collaborative Learning Spaces provide a solution by allowing students to form study groups, participate in forums, and collaborate on course-related discussions. This feature encourages knowledge sharing, group problem-solving, and helps students stay connected throughout the course duration. It also enables the exchange of ideas and resources, making learning more dynamic and engaging.

Another key issue is the lack of personalized, real-time support for students navigating the platform or managing their course selections. Many users have difficulty choosing the right course or need quick help with specific queries, yet traditional course systems do not provide an efficient way to address these needs. To solve this, Course Tide will integrate an AI-powered Interactive Course Chatbot. This chatbot will assist users with personalized course recommendations based on their learning history, preferences, and career goals. Additionally, it will provide real-time support for course-related questions, technical issues, and scheduling, helping students make informed decisions and resolve issues quickly.

By addressing both collaboration and personalized assistance, Course Tide sets itself apart from existing platforms that focus solely on course enrolment. The combination of Collaborative Learning Spaces and the Interactive Course Chatbot ensures that students not only reserve courses but also engage deeply with the learning process. The platform is designed to be more than just a booking system; it offers a holistic learning experience that supports students from enrolment to course completion.

Moreover, Course Tide will offer a user-friendly, responsive interface that allows students to easily navigate through course options, interact with peers, and access chatbot assistance seamlessly. The system will also provide real-time notifications, ensuring that students are updated on their course progress, upcoming sessions, and any collaborative group activity. This will keep users engaged and help them stay on track throughout their educational journey.

In conclusion, Course Tide is designed to enhance the traditional online course reservation system by integrating essential features for collaboration and personalized support. The Collaborative Learning Spaces foster peer interaction and teamwork, while the Interactive Course Chatbot ensures students receive the guidance they need when navigating the platform. By offering these features, Course Tide aims to provide a more interactive, engaging, and supportive learning environment that caters to the needs of modern students.

**Software Requirement Specifications**

For

Course Tide Website

Version 1.0 approved

Prepared by:

1. Mohammed Abdur Rasheed - 22BD1A0538
2. Nehith Sayini - 22BD1A053F
3. Laksh Vijayvargiya - 22BD1A050X
4. Jyoti Swaroop - 22BD1A0553

**Keshav Memorial Institute of Technology**

**05-09-2024.**

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

**Appendix A: Glossary.................................................................................................. 5**

**Appendix B: Analysis Models......................................................................................5**

**Appendix C: To Be Determined List..........................................................................6**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
| Week-1 | 05-09-2024 | SRS creation(Introduction) | 1.0 |
| Week-2 | 06-09-2024 | SRS  updation(overall description) | 2.0 |
| Week-3 | 13-09-2024 | SRS updation (external interface requirements) | 3.0 |
| Week-4 | 20-09-2024 | SRS updation(system features and other non-functional requirements) | 4.0 |
| Week - 5 | 27-09-2024 | SRS updating (SDLC) | 5.0 |
| Week - 6 | 18-10-2024 | SRS updation (Use case diagrams) | 6.0 |

**1. Introduction**

The CourseTide project aims to streamline the process of enrolling in educational courses by providing a user-friendly platform. This system is designed to simplify course registration, manage scheduling conflicts, and facilitate communication between students and instructors. The introduction will cover the background of the project, its significance in the educational sector, and an overview of the features and functionalities of the reservation system.

**1.1 Purpose**

The primary purpose of this project is to create a comprehensive, efficient, and reliable course reservation system that enhances the enrolment experience for both students and administrators. It will address the challenges associated with manual registration processes, such as scheduling conflicts, overbooked classes, and administrative inefficiencies. By automating these processes, the system aims to reduce errors, save time, and provide a seamless user experience.

**1.2 Document Convention**

This document adheres to standard conventions to ensure clarity and consistency. Key conventions include:

**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:**

This document is intended for a diverse audience including:

Project Stakeholders: Individuals involved in funding, approving, or overseeing the project.

Developers: Technical teams responsible for designing, coding, and implementing the system.

End Users: Students and administrative staff who will interact with the reservation system.

Documentation Team: Authors and reviewers who are responsible for maintaining and updating the project documentation.

For effective reading: Project Stakeholders should focus on sections detailing the purpose and product scope to understand the project's value and objectives. Developers should pay close attention to the technical requirements and system specifications. End Users might benefit from sections describing system features and user interface guidelines. Documentation Team should review the document conventions and references to ensure consistency and accuracy.

**1.4 Product Scope**

The product scope outlines the boundaries and deliverables of the course reservation system. This includes:

Core Features: Online course search, enrolment, scheduling, and payment processing.

User Roles: Different functionalities for students, instructors, and administrative staff.

System Requirements: Technical specifications, hardware, and software prerequisites.

Limitations: Any constraints or exclusions, such as integration with third-party systems or support for specific types of courses.

The scope defines what will be included in the project and helps manage expectations by specifying what is outside the project’s boundaries.

**1.5 References:**

We took references from various websites related to online learning platforms like Coursera, Udemy, edX, Khan Academy, and LinkedIn Learning.

* Udemy: <https://www.udemy.com/blog/write-software-requirements-specifications/>
* edX: <https://www.edx.org/course/software-requirements-analysis>
* Khan Academy: <https://www.khanacademy.org/computing/computer-programming>
* LinkedIn Learning: <https://www.linkedin.com/learning/topics/software-requirements>
* Coursera: <https://www.coursera.org/articles/how-to-write-software-requirements-specifications>

Additionally, we referred to relevant documents and software requirements guidelines:

* Ch9: Software Requirements Document Guidelines: <http://users.csc.calpoly.edu/~csturner/courses/308w09/Ch9_DocReqts.pdf>
* Software Requirements Specification for FSoft\_D: <https://www.academia.edu/8831452/Software_Requirements_Specification_for_FSoft_D>

References of articles:

* Udemy vs Coursera vs edX: The Ultimate Comparison (2023): <https://www.transizion.com/udemy-vs-coursera-vs-edx/>
* SRS Documents: Requirements and Diagrammatic Notations on Coursera (2023): <https://www.coursera.org/articles/software-requirements-specifications-srs-diagrams>

**2. Overall Description**

**2.1 Product Perspective**

Course Tide is an innovative online course reservation system designed to improve upon existing online learning platforms like Coursera, Udemy, and edX. While it incorporates many of the standard features of these platforms, such as course browsing, enrolment and user account management, Course Tide stands out with its Collaborative Learning Spaces and Interactive Course Chatbots. These features encourage peer interaction and offer personalized support, enhancing the overall learning experience.

The platform is user-friendly and mobile-responsive, allowing students to access courses and engage with peers from any device. It integrates modern design with intuitive navigation and provides personalized learning suggestions, real-time support, and collaboration tools. Course Tide seeks to bridge the gap between passive course registration and an engaging, interactive learning environment.

**2.2 Product Functions**

**2.2.1 Administrator**

* Administrators have the ability to add, modify, and delete courses.
* Manage user accounts by approving or rejecting instructor or student registrations based on platform policies.
* Administrators can organize and categorize courses, ensuring they are easy to browse.
* Set up system notifications for course discounts, promotions, or group learning sessions.
* Monitor course completion rates and handle feedback or issues reported by users.

**2.2.2 Instructors**

* Instructors can create, update, and delete courses, manage course content, and upload learning materials.
* They can interact with students through forums and collaborative learning groups, enhancing engagement.
* Can track student progress and provide feedback on assessments or assignments.

**2.2.3 Students/Users**

* Students can search and enroll in courses based on subject, difficulty level, instructor, or specific skills.
* They have the ability to form study groups and engage in CollaborativeLearning Spaces with peers enrolled in the same course.
* Can access personalized course recommendations and real-time assistance through Interactive Course Chatbots.
* Students are provided with account management features, such as tracking course progress, earning certificates, and participating in forums or discussions.

**2.3 Operating Environment**

* Course Tide is a web-based platform compatible with all modern browsers such as Google Chrome, Firefox, Safari, and Edge.
* The platform is fully responsive and optimized for both desktop and mobile devices.
* It requires a minimum 1 GHz processor, 2 GB RAM, and 512 MB available disk space for smooth operation on desktop devices.
* For mobile users, it supports iOS, Android, and Windows operating systems.

**2.4 User Characteristics**

The primary users of Course Tide are students, instructors, and administrators.

* Students are expected to have a basic understanding of computers and mobile devices to browse, enrol in, and complete courses.
* Instructors should have proficiency in managing course materials, interacting with students through the collaborative tools, and monitoring student progress.
* Administrators must be well-versed in platform operations, troubleshooting, and course or user management.

**2.5 Design and Implementation Constraints**

* Users must have a stable internet connection to access the platform and participate in live group discussions or collaborative spaces.
* All user data, course progress, and collaboration history will be stored securely in a cloud-based database.
* Course Tide will be available 24/7, ensuring students can access their enrolled courses and interactive features at any time.
* No specific operating system is required; the platform is platform-agnostic, running on any device with internet browsing capabilities.
* Users must use valid credentials to access their accounts, and two-factor authentication is recommended for additional security.

**2.6 Assumptions and Dependencies**

* Course Tide requires integration with third-party services such as video streaming (e.g., YouTube, Vimeo) for course content delivery and cloud storage solutions for storing course materials.
* The success of Course Tide depends on the availability of internet access and the familiarity of users with online platforms.
* The platform assumes instructors and students are comfortable using discussion forums, collaborative tools, and chatbot interactions for enhanced learning experiences.
* Continuous internet access and updated browsers are required for seamless platform use.

**3. External Interface Requirements**

**3.1 User Interfaces**

1. **Course Search and Filtering UI**
   * **Goal**: Create a straightforward, intuitive search function.
   * **Features**:
     + **Search by Keyword**: A search bar for entering course-related keywords.
     + **Filter Options**: Drop-down menus or checkboxes for:
       - Difficulty Level (e.g., Beginner, Intermediate, Advanced)
       - Course Type (e.g., Online, Onsite, Hybrid)
       - Certification (e.g., with or without certificate)
       - Start Date (selectable date ranges)
     + **Responsiveness**: Ensure the interface works smoothly on both desktop and mobile devices.
2. **Course Details Page**
   * **Goal**: Provide comprehensive course information to the user.
   * **Features**:
     + Title, Instructor Name, Duration, Delivery Format (e.g., video, written), Difficulty Level
     + Pricing details
     + Additional course information (e.g., syllabus, prerequisites, reviews)
     + Buttons for enrolling in or saving the course for later.
3. **Course Summarization and Celebrity Voice Features**
   * **Goal**: Allow users to customize the course experience through AI summarization and voice narration.
   * **Features**:
     + **Summarization Options**: Users can select specific sections of the course to summarize.
     + **Celebrity Voice Selection**: A drop-down list of available celebrity voices for text-to-speech (TTS) narration.
     + **Audio Playback**: An embedded audio player for listening to the summarized content.
4. **User Registration/Login Interface**
   * **Goal**: Provide secure and seamless user authentication.
   * **Features**:
     + **Registration**: Fields for name, email, password, and possibly two-factor authentication (2FA) setup.
     + **Login**: Secure authentication with password or social media login options.
     + **Profile Management**: Manage enrolled courses, personalized recommendations, and track learning progress.

**3.2 Hardware Interfaces**

1. **Client Devices**
   * The platform should be optimized to run smoothly on various devices, including:
     + **Desktops and Laptops**: Fully responsive web interface.
     + **Tablets and Smartphones**: A mobile-friendly or adaptive design for smaller screens.
2. **Servers**
   * The platform will need cloud-based servers to handle:
     + **Web Hosting**: The platform’s frontend and backend services.
     + **Databases**: Storing user data, course information, and transaction records.
     + **Course Recommendation Engine**: AI-based algorithm for course suggestions.
     + **AI-Powered Summarization**: Hosting and executing machine learning models.
     + **Text-to-Speech (TTS)**: Servers capable of processing and delivering narrated content.

**3.3 Software Interfaces**

1. **Text-to-Speech (TTS) API**
   * Integration with services like Google Cloud TTS, Amazon Polly, or Eleven Labs to generate course audio using selected celebrity voices.
2. **AI Summarization API**
   * Use AI models from providers like OpenAI or Hugging Face to summarize course materials, such as PDFs and video content.
3. **Payment Gateway API**
   * Implement services like Stripe or PayPal to handle secure payments for course enrolments.
4. **Email and Notification API**
   * Connect with an email service to send confirmations, reminders, and summary notifications to users.

**3.4 Communications Interfaces**

1. **HTTPS**
   * Secure communication between the platform and client devices via HTTPS for encrypted data transfer.
2. **REST API**
   * REST APIs will handle data exchanges between the frontend, backend, and third-party services like TTS and summarization.

### 4. System Features

**4.1 System 1: Course Search, Filter, and Enrolment**

* **Description**: Users can search for courses using keywords and apply filters based on course type, difficulty level, certification, and pricing. The system updates results dynamically based on user selections.
* **Priority**: High
* **Stimulus/Response Sequences**:
  + User enters a keyword and applies filters.
  + The system displays relevant courses.
  + User selects a course to view details and can proceed with enrollment.
* **Functional Requirements**:
  + **F1.1**: Support keyword-based searches.
  + **F1.2**: Implement filtering options.
  + **F1.3**: Provide course details and enrolment functionality.

**4.2 System Feature 2: Celebrity Voice Course Narration and AI Summarization**

* **Description**: Users can select a celebrity voice for course narration and request AI-generated summaries of specific course sections, whether PDF documents or videos.
* **Priority**: High
* **Stimulus/Response Sequences**:
  + User chooses a course section, selects a celebrity voice, and the system generates an audio narration.
  + User uploads a PDF or selects a video, and the system provides a summary.
* **Functional Requirements**:
  + **F2.1**: Integrate with TTS service to offer celebrity voices.
  + **F2.2**: Enable AI-powered summarization for course materials.

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

* The system must handle large numbers of concurrent users.
* Search results and summaries should be delivered within 2-3 seconds.
* TTS generation should complete within 5-10 seconds for standard-length course sections.

**5.2 Safety Requirements**

* Ensure data integrity, protecting user data such as course history and payment details.
* Users should be able to retrieve their progress in case of system failure.

**5.3 Security Requirements**

* Secure user authentication using encryption (SSL/TLS) and multi-factor authentication (MFA).
* Payment processing must comply with PCI DSS standards.

**5.4 Software Quality Attributes**

* **Usability**: Intuitive, engaging user experience.
* **Scalability**: Support growing user demand, especially during peak periods.
* **Reliability**: Ensure 99.9% uptime.
* **Maintainability**: Modular, well-documented codebase for future updates.

**5.5 Business Rules**

* The platform will charge a commission fee for each course enrolment.

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

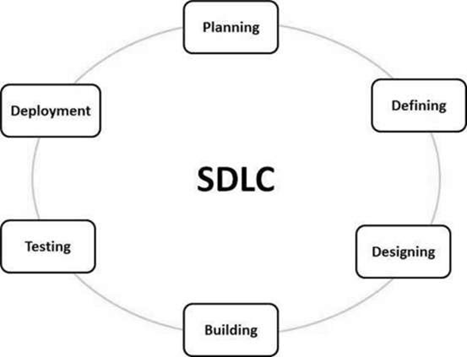
Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

* 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**

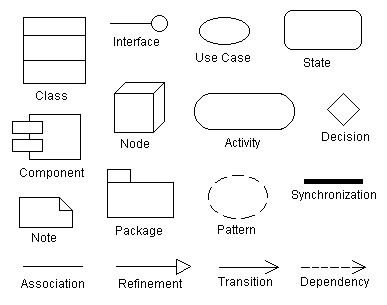
Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

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:



**6. Other Requirements**

* The system must comply with data privacy regulations, including **GDPR**, for managing user data.
* The platform should be localized to support **multiple languages** for a global user base.

**Appendix A: Glossary**

* **TTS (Text-to-Speech)**: Technology that converts text into human-like speech.
* **API (Application Programming Interface)**: A set of protocols for building and interacting with software applications.
* **MFA (Multi-Factor Authentication)**: A security process requiring users to verify their identity using two or more verification methods.

**Appendix B: Analysis Models**

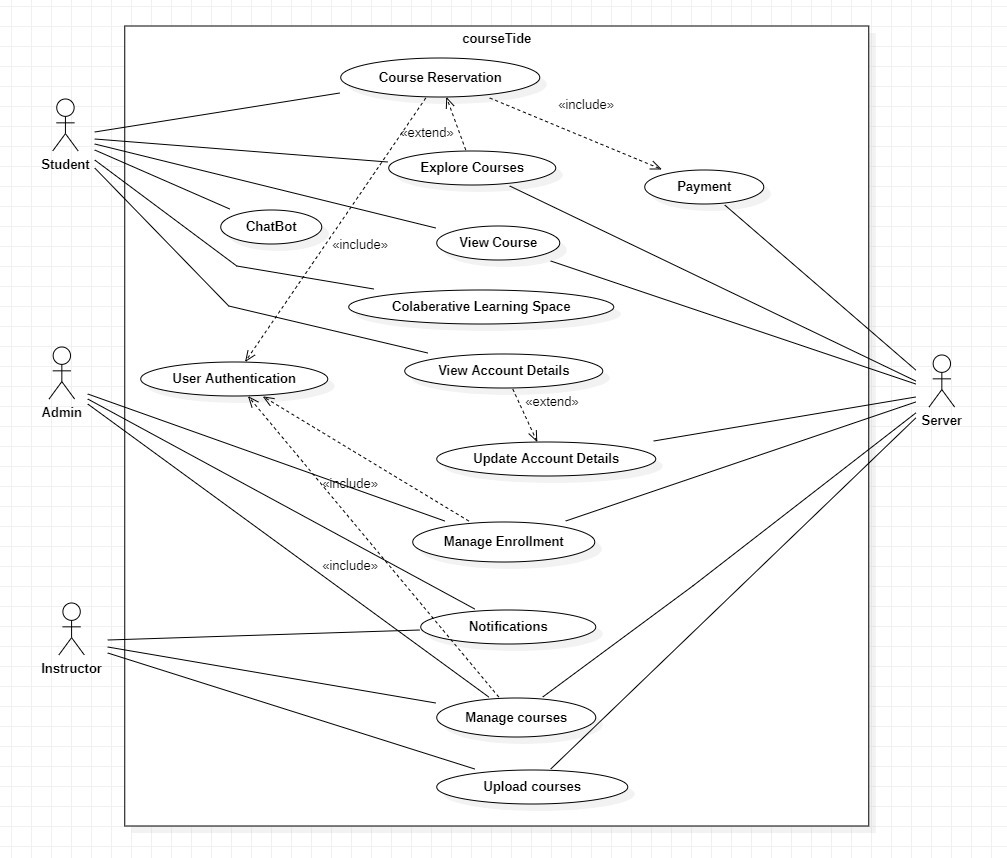
* **UML Diagrams**: Use case diagrams to visualize user interactions with search, filter, celebrity voice, and summarization features.
* **ER Diagrams**: Entity-Relationship diagrams representing data relationships between courses, users, and enrolments.

**Appendix C: To Be Determined List**

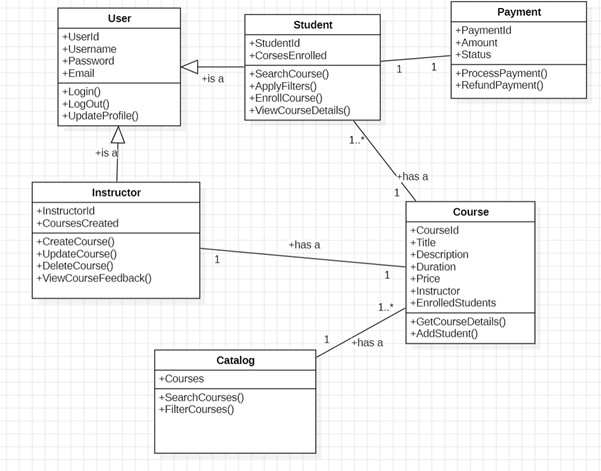
* **Integration Partner for TTS API**: Final choice of TTS provider (Google, Amazon, or Eleven Labs) is pending.
* **AI Summarization Model**: Decision between using OpenAI models or developing a custom summarization model is yet to be made.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | 04566789999 | | | |
| **Use Case Name:** | CourseTide | | | |
| **End Objective:** | Discovering,analysing and recomending courses based on user preferences | | | |
| **Created by:** | 1.MohammedAbdurRasheed 2. Nehith S 3. Laksh  4.Jyothi Swaroop | | **On (date):** |  |
| **User/Actor:** | Administator,student,instructor,server | | | |
| Trigger: | The user (admin, student,instructor) attempts to access the system | | | |
| **Basic/Normal Flows** | | | | |
| **User Actions** | | **System Actions** | | |
| The user logs into the online course platform by entering valid credentials  The user views their dashboard (Admin views course management, Instructor views enrolled students, Students view their enrolled courses).  The user selects the View Course Content option  The user edits their personal details or course information (if Admin or Instructor).    The user enrolls in a course or creates a course (based on role).    The user checks out and makes payments for course fees. | | The system displays a login page prompting the user to provide the correct username and password.    The system checks the user's role (Admin, Instructor, or Student) and displays relevant features (e.g., course management, student lists, or enrolled courses).  The system retrieves and displays the course content associated with the user’s enrolled courses.  The system provides editable fields for personal or course information and updates the database with the new details after confirming the changes.  The system presents enrollment or course creation options and confirms the action once submitted.  The system processes the payment (using online methods) and updates the billing records, providing a receipt for the course enrollment or creation fees. | | |
| **Exception Flows** | | | | |
| **User Actions** | | **System Actions** | | |
| User tries to login but doesn't have an account    User tries to login by entering incorrect details      User tries to view course content but lacks permission  Instructor tries to create a course, but it has already been created.      The student tries to enroll in a course, but no available seats      Admin tries to update user data, but the data is corrupted | | The system requests the user to register an account on the registration page before allowing the login process.    The system displays an error message: “Please check the username or password entered” and prompts the user to re-enter correct login credentials.    The system shows a message: “Access denied. You do not have permission to view this data,” and restricts access.    The system displays a message: “Restocking request already made. Please wait for the admin to process the current request.”  The system displays a message: No available seats. Please try again later or choose a different course  The system displays an error message: “Unable to update data. Database error encountered. Please get in touch with support.” | | |
|  |  |  |  |  |

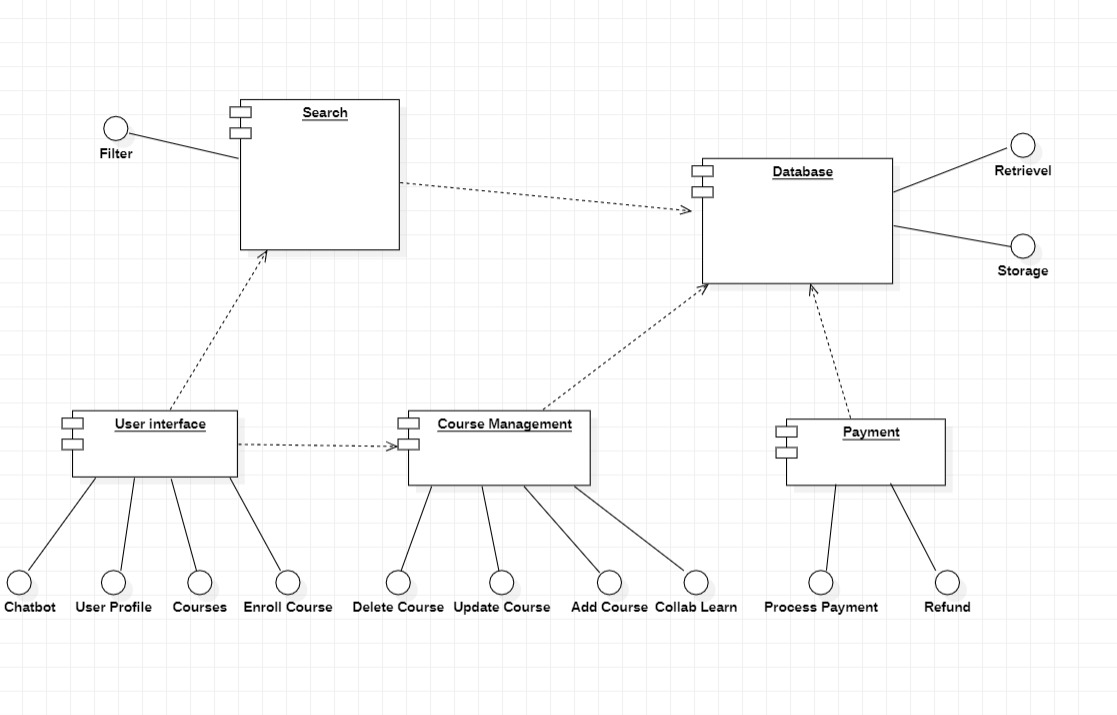
**Use case diagram:**



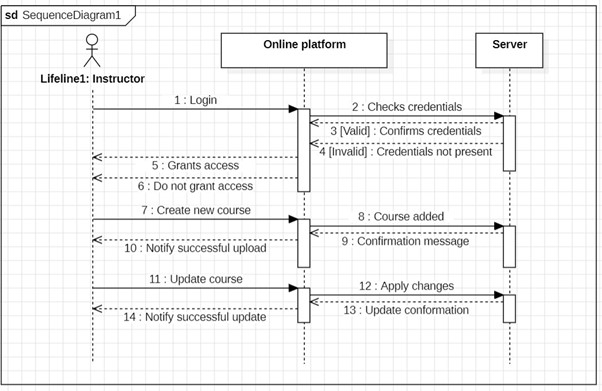
**Class diagram:**



**Component diagram:**



**Sequence diagram:**



1. 5