# Chapter 3: System Design

## 3.1 System Design Overview

### 3.1.1 Activity Diagram

### 3.1.2 Sequence Diagram

## 3.2 Interface Design

### 3.2.1 Logo

### 3.2.2 Style Guide (Typography and Color Palette)

### 3.2.3 Prototype

## 3.3 Database Design

### 3.3.1 Entity Relationship Diagram (ERD)

### 3.3.2 Unified Modeling Language (UML)

System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. It is a crucial phase in the software development lifecycle that bridges the gap between system analysis (what the system should do) and system implementation (how the system will be built). The primary goal of system design is to create a robust, scalable, maintainable, and efficient solution that meets the functional and non-functional requirements identified during the analysis phase [1].

For the Nomo Platform, system design involves translating the user and system requirements into a detailed blueprint for development. This includes making decisions about the overall system architecture, selecting appropriate technologies, designing the database structure, defining user interfaces, and outlining the interactions between different system components. A well-designed system ensures that the development process is streamlined, risks are minimized, and the final product is of high quality and performs as expected.

This chapter will delve into the various aspects of the Nomo Platform’s system design, starting with an overview of the system’s high-level processes and interactions through activity and sequence diagrams. Subsequently, it will cover the interface design, including the logo, style guide, and prototype, which are essential for user experience. Finally, the chapter will detail the database design, encompassing the Entity-Relationship Diagram (ERD) and Unified Modeling Language (UML) diagrams, to illustrate the data structure and relationships within the system.

### 3.1.1 Activity Diagram

An Activity Diagram visually represents the flow of activities and actions within a system or a specific process. It illustrates the sequence of steps, decision points, and parallel paths involved in a workflow, providing a clear understanding of the system’s dynamic behavior. For the Nomo Platform, the Activity Diagram helps in visualizing complex processes such as resource uploading, book exchange, or user registration, ensuring that all steps and their logical order are well-defined [2].

Below is the Activity Diagram for the “Upload Academic Resource” process, which was previously detailed in Chapter 2. This diagram shows the interactions between the user, the Nomo Platform, the Resource Management Module, and the Database, from accessing the upload page to the final confirmation of the resource up```plantuml @startuml start

partition “User” { :Access Upload Resource Page; :Fill Form & Select File; :Add Tags/Categories (Optional); }

partition “Nomo Platform” { :Display Upload Form; :Validate Input & File Type; if (Input Valid?) then (yes) :Process File Upload; :Store Resource Metadata & File Path; :Display Success Message; else (no) :Display Error Message; endif :Update Resource Metadata; :Save Tags/Categories; :Display Update Confirmation; }

stop @enduml User: Display Update Confirmation @enduml

*Figure 3.1: Activity Diagram for Upload Academic Resource Process*

This diagram highlights the sequential nature of the upload process, including validation steps and alternative flows for invalid input. It provides a clear visual guide for developers to implement the functionality and for stakeholders to understand the user journey for resource contribution.

### 3.1.2 Sequence Diagram

A Sequence Diagram illustrates the interactions between objects or components in a sequential order, focusing on the messages exchanged and the order in which they occur. It is particularly useful for understanding the flow of control and data within a specific use case, showing how different parts of the system collaborate to achieve a particular functionality. For the Nomo Platform, the Sequence Diagram helps in visualizing the dynamic behavior of key interactions, such as the “Borrow Book” process [3].

Below is the Sequence Diagram for the “Borrow Book” use case, which was previously detailed in Chapter 2. This diagram depicts the interactions between a Student, the Nomo Platform, the Book Exchange Module, the Database, and the Book Lender, from searching for a book to the approval or rejection of a borrow request.

@startuml  
actor Student  
participant NomoPlatform as "Nomo Platform"  
participant BookExchangeModule as "Book Exchange Module"  
participant Database  
participant Lender as "Book Lender"  
  
Student->NomoPlatform: Search for Book  
NomoPlatform->BookExchangeModule: Request Book Search  
BookExchangeModule->Database: Query Available Books  
Database-->BookExchangeModule: Return Book List  
BookExchangeModule-->NomoPlatform: Display Book List  
NomoPlatform-->Student: Show Search Results  
  
Student->NomoPlatform: Select Book & Request Borrow  
NomoPlatform->BookExchangeModule: Process Borrow Request  
BookExchangeModule->Database: Check Book Availability & Lender Info  
Database-->BookExchangeModule: Return Availability & Lender Info  
BookExchangeModule->Lender: Notify of Borrow Request  
Lender->BookExchangeModule: Approve/Reject Request  
BookExchangeModule-->NomoPlatform: Borrow Request Status  
NomoPlatform-->Student: Display Borrow Status  
@enduml

*Figure 3.2: Sequence Diagram for Borrow Book Use Case*

This diagram provides a clear, step-by-step representation of the interactions involved in borrowing a book, highlighting the responsibilities of each component and the flow of information. It is instrumental in ensuring that the system’s various modules communicate effectively and that the user experience for this critical function is seamless.

## 3.2 Interface Design

Interface design is a critical aspect of system development, as it directly impacts user experience and the overall usability of the Nomo Platform. A well-designed interface is intuitive, aesthetically pleasing, and efficient, enabling users to interact with the system seamlessly and effectively. This section outlines the key elements of the Nomo Platform’s interface design, including its logo, style guide (typography and color palette), and the approach to prototyping [4].

### 3.2.1 Logo

The logo serves as the primary visual identity of the Nomo Platform, embodying its mission and values. It is designed to be modern, minimalist, and reflective of the platform’s core functionalities: digital solutions, education, and connectivity. The logo aims to convey knowledge sharing, community building, and technological innovation. The chosen design incorporates elements that subtly suggest these concepts, utilizing a clean aesthetic and a thoughtful color palette to ensure memorability and professionalism.

[**Image of Nomo Platform Logo**](/home/ubuntu/nomo_platform_new_logo.png)

*Figure 3.3: Nomo Platform Logo*

This logo will be consistently used across all platform touchpoints, including the website, mobile application, and promotional materials, to establish a strong and recognizable brand presence.

### 3.2.2 Style Guide (Typography and Color Palette)

A comprehensive style guide is essential for maintaining visual consistency and brand identity throughout the Nomo Platform. It defines the typography, color palette, and other visual elements, ensuring a cohesive and professional user interface. Adhering to a style guide streamlines the design process, enhances usability, and reinforces the platform’s brand image [5].

#### 3.2.2.1 Typography

Typography plays a crucial role in readability and user experience. For the Nomo Platform, a combination of fonts will be selected to ensure clarity, hierarchy, and aesthetic appeal. A primary font will be used for headings and prominent text, while a secondary font will be designated for body text and smaller elements. The selection will prioritize legibility across various devices and screen sizes.

* **Primary Font (Headings):** [Suggest a modern, clean sans-serif font, e.g., ‘Montserrat’ or ‘Open Sans Bold’]
* **Secondary Font (Body Text):** [Suggest a highly readable sans-serif font, e.g., ‘Open Sans Regular’ or ‘Roboto Regular’]

These fonts will be chosen for their versatility, web-friendliness, and ability to convey a sense of professionalism and approachability.

#### 3.2.2.2 Color Palette

The color palette is carefully chosen to evoke a sense of trust, innovation, and community, aligning with the Nomo Platform’s objectives. The palette will consist of primary, secondary, and accent colors, each serving a specific purpose in the user interface. Blues and greens are often associated with technology, growth, and reliability, making them suitable choices for a platform focused on education and digital solutions.

* **Primary Colors:**
  + **Main Blue:** #2A52BE (Represents trust, stability, and professionalism)
  + **Accent Green:** #4CAF50 (Symbolizes growth, vitality, and community)
* **Secondary Colors:**
  + **Light Gray:** #F5F5F5 (Used for backgrounds and subtle distinctions)
  + **Dark Gray:** #333333 (For text and important elements to ensure readability)
* **Accent Colors:**
  + **Bright Orange:** #FF9800 (For calls to action and highlights)

This color palette will be applied consistently to all UI elements, including buttons, icons, backgrounds, and text, to create a harmonious and visually appealing experience.

### 3.2.3 Prototype

Prototyping is an iterative process of creating preliminary versions of the Nomo Platform’s interface to visualize and test design concepts before full-scale development. Prototypes range from low-fidelity wireframes to high-fidelity interactive models, allowing stakeholders to experience the user flow and provide feedback early in the design phase [6].

For the Nomo Platform, prototyping will involve:

* **Wireframes:** Basic visual guides that represent the skeletal framework of the website and mobile application. These will focus on layout, content placement, and user flow, without detailed visual design.
* **Mockups:** Static, high-fidelity representations of the user interface, incorporating the chosen typography, color palette, and visual elements. Mockups will provide a realistic preview of the platform’s appearance.
* **Interactive Prototypes:** Clickable and navigable versions of the interface, simulating the user experience. These prototypes will allow for testing of user flows, usability, and overall interaction design, enabling early identification and resolution of design flaws.

Prototyping will facilitate continuous feedback from target users (students, alumni, faculty) and stakeholders, ensuring that the final interface is user-centric, intuitive, and meets their expectations effectively. This iterative approach minimizes rework and ensures a more efficient development process.

## 3.3 Database Design

Database design is a fundamental component of system architecture, focusing on structuring the data to support the application’s functionalities efficiently and reliably. For the Nomo Platform, a well-designed database is crucial for managing diverse information, including user profiles, academic resources, book listings, forum posts, and mentorship records. The goal is to create a robust, scalable, and secure data model that ensures data integrity, minimizes redundancy, and optimizes data retrieval and storage operations [7].

This section will detail the database design using two primary modeling techniques: the Entity-Relationship Diagram (ERD) and the Unified Modeling Language (UML) Class Diagram. These diagrams provide a visual representation of the database structure, illustrating entities, their attributes, and the relationships between them, which are essential for guiding the database implementation and ensuring a coherent data management strategy.

### 3.3.1 Entity Relationship Diagram (ERD)

An Entity-Relationship Diagram (ERD) is a high-level conceptual data model that illustrates the relationships between different entities within a system. It helps in understanding the logical structure of the database by representing entities (objects or concepts), attributes (properties of entities), and relationships (associations between entities). For the Nomo Platform, the ERD will capture the key data elements and their interconnections, providing a clear blueprint for the relational database [8].

Key entities identified for the Nomo Platform include:

* **User:** Represents all individuals interacting with the platform (Students, Alumni, Faculty, Administrators).
* **AcademicResource:** Represents educational materials shared on the platform.
* **Book:** Represents books available for exchange.
* **BookExchange:** Represents a transaction or request for book exchange.
* **ForumPost:** Represents a post or question in the discussion forums.
* **Comment:** Represents replies to forum posts.
* **Mentorship:** Represents a mentorship relationship between an alumnus and a student.

Below is the Entity-Relationship Diagram for the Nomo Platform, illustrating these entities and their relationships.

@startuml  
skinparam linetype ortho  
  
entity User {  
 \*userID : INT (PK)  
 --  
 username : VARCHAR  
 email : VARCHAR  
 passwordHash : VARCHAR  
 userType : ENUM (Student, Alumni, Faculty, Admin)  
 registrationDate : DATETIME  
 major : VARCHAR  
 academicLevel : VARCHAR  
 contactInfo : VARCHAR  
}  
  
entity AcademicResource {  
 \*resourceID : INT (PK)  
 --  
 title : VARCHAR  
 description : TEXT  
 uploadDate : DATETIME  
 fileURL : VARCHAR  
 resourceType : ENUM (Lecture, Notes, Exam, Article, Other)  
 subject : VARCHAR  
 uploadedBy : INT (FK)  
}  
  
entity Book {  
 \*bookID : INT (PK)  
 --  
 title : VARCHAR  
 author : VARCHAR  
 isbn : VARCHAR  
 condition : ENUM (New, Good, Fair, Worn)  
 ownerID : INT (FK)  
 status : ENUM (Available, Borrowed, Exchanged)  
}  
  
entity BookExchange {  
 \*exchangeID : INT (PK)  
 --  
 requestDate : DATETIME  
 status : ENUM (Pending, Approved, Rejected, Completed)  
 borrowerID : INT (FK)  
 lenderID : INT (FK)  
 bookID : INT (FK)  
}  
  
entity ForumPost {  
 \*postID : INT (PK)  
 --  
 title : VARCHAR  
 content : TEXT  
 postDate : DATETIME  
 postedBy : INT (FK)  
}  
  
entity Comment {  
 \*commentID : INT (PK)  
 --  
 content : TEXT  
 commentDate : DATETIME  
 commentedBy : INT (FK)  
 postID : INT (FK)  
}  
  
entity Mentorship {  
 \*mentorshipID : INT (PK)  
 --  
 startDate : DATETIME  
 endDate : DATETIME  
 status : ENUM (Active, Completed, Cancelled)  
 mentorID : INT (FK)  
 menteeID : INT (FK)  
}  
  
User ||--o{ AcademicResource : uploads  
User ||--o{ Book : owns  
User ||--o{ BookExchange : participates\_as\_borrower  
User ||--o{ BookExchange : participates\_as\_lender  
User ||--o{ ForumPost : posts  
User ||--o{ Comment : comments  
User ||--o{ Mentorship : mentors  
User ||--o{ Mentorship : mentees  
ForumPost ||--o{ Comment : has  
Book ||--o{ BookExchange : is\_part\_of  
  
@enduml

*Figure 3.4: Entity-Relationship Diagram (ERD) for Nomo Platform*

This ERD provides a clear visual representation of the database schema, showing how different pieces of information are related. It serves as a foundational document for database developers to implement the physical database and ensures that data integrity and relationships are maintained across the system.

### 3.3.2 Unified Modeling Language (UML) Class Diagram

A UML Class Diagram is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods), and the relationships among objects. It provides a more detailed, object-oriented view of the system’s components compared to an ERD, making it particularly useful for software development. For the Nomo Platform, the UML Class Diagram will illustrate the classes that represent the system’s entities, their properties, and the methods that define their behavior [9].

Below is the UML Class Diagram for the Nomo Platform, which expands on the ERD by including the operations (methods) for each class. This diagram provides a more comprehensive view of the system’s object-oriented design, guiding developers in implementing the classes and their interactions.

@startuml  
class User {  
 - userID : INT  
 - username : VARCHAR  
 - email : VARCHAR  
 - passwordHash : VARCHAR  
 - userType : ENUM  
 - registrationDate : DATETIME  
 - major : VARCHAR  
 - academicLevel : VARCHAR  
 - contactInfo : VARCHAR  
 + register()  
 + login()  
 + logout()  
 + updateProfile()  
 + searchResources()  
 + requestBookExchange()  
 + postToForum()  
 + commentOnPost()  
 + requestMentorship()  
}  
  
class AcademicResource {  
 - resourceID : INT  
 - title : VARCHAR  
 - description : TEXT  
 - uploadDate : DATETIME  
 - fileURL : VARCHAR  
 - resourceType : ENUM  
 - subject : VARCHAR  
 - uploadedBy : User  
 + uploadResource()  
 + downloadResource()  
 + updateResource()  
 + deleteResource()  
}  
  
class Book {  
 - bookID : INT  
 - title : VARCHAR  
 - author : VARCHAR  
 - isbn : VARCHAR  
 - condition : ENUM  
 - owner : User  
 - status : ENUM  
 + addBook()  
 + updateBook()  
 + removeBook()  
 + searchBook()  
}  
  
class BookExchange {  
 - exchangeID : INT  
 - requestDate : DATETIME  
 - status : ENUM  
 - borrower : User  
 - lender : User  
 - book : Book  
 + createRequest()  
 + approveRequest()  
 + rejectRequest()  
 + completeExchange()  
}  
  
class ForumPost {  
 - postID : INT  
 - title : VARCHAR  
 - content : TEXT  
 - postDate : DATETIME  
 - postedBy : User  
 + createPost()  
 + updatePost()  
 + deletePost()  
}  
  
class Comment {  
 - commentID : INT  
 - content : TEXT  
 - commentDate : DATETIME  
 - commentedBy : User  
 - post : ForumPost  
 + createComment()  
 + updateComment()  
 + deleteComment()  
}  
  
class Mentorship {  
 - mentorshipID : INT  
 - startDate : DATETIME  
 - endDate : DATETIME  
 - status : ENUM  
 - mentor : User  
 - mentee : User  
 + requestMentorship()  
 + acceptMentorship()  
 + endMentorship()  
}  
  
User "1" -- "0..\*" AcademicResource : uploads  
User "1" -- "0..\*" Book : owns  
User "1" -- "0..\*" BookExchange : as borrower  
User "1" -- "0..\*" BookExchange : as lender  
User "1" -- "0..\*" ForumPost : posts  
User "1" -- "0..\*" Comment : comments  
User "1" -- "0..\*" Mentorship : as mentor  
User "1" -- "0..\*" Mentorship : as mentee  
ForumPost "1" -- "0..\*" Comment : has  
Book "1" -- "0..\*" BookExchange : is part of  
  
@enduml

*Figure 3.5: UML Class Diagram for Nomo Platform*

This UML Class Diagram provides a detailed blueprint for the system’s object-oriented structure, defining the classes, their attributes, and their methods. It is an essential tool for developers, ensuring that the system is built with a clear, consistent, and well-organized architecture that aligns with the project’s requirements.

## References

[1] Techopedia. (n.d.). *System Design*. Retrieved from <https://www.techopedia.com/definition/27907/system-design>

[2] Visual Paradigm. (n.d.). *What is Activity Diagram?*. Retrieved from <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>

[3] Visual Paradigm. (n.d.). *What is Sequence Diagram?*. Retrieved from <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>

[4] Interaction Design Foundation. (n.d.). *What is Interface Design?*. Retrieved from <https://www.interaction-design.org/literature/topics/interface-design>

[5] Adobe. (n.d.). *What is a style guide?*. Retrieved from <https://www.adobe.com/creativecloud/design/discover/style-guide.html>

[6] UXPin. (n.d.). *What is Prototyping in UX?*. Retrieved from <https://www.uxpin.com/studio/blog/what-is-prototyping-in-ux/>

[7] Oracle. (n.d.). *Database Design Concepts*. Retrieved from <https://docs.oracle.com/cd/B19306_01/server.102/b14220/logical.htm>

[8] Visual Paradigm. (n.d.). *What is Entity Relationship Diagram (ERD)?*. Retrieved from <https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/>

[9] Visual Paradigm. (n.d.). *What is Class Diagram?*. Retrieved from <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/>