ARCHITECTURE AND DATABASE DESIGN OF DIGITAL LEARNING PLATFORMS

Sh.Sh.Allamova
Acting Associate Professor, Department of Information Technologies and Exact Sciences, TIPI

https://doi.org/10.5281/zenodo.15430467

Abstract: This article explores the architecture and database design of a digital learning platform developed for primary school students. The study focuses on the modular structure of the platform built using the Django framework, its real-time communication capabilities, and the optimized design of the database architecture. It analyzes the ER models, table structures, and security measures implemented for the platform’s three core modules – students, teachers, and real-time communication. The research proposes approaches aimed at ensuring data security, applying normalization stages, and improving overall system efficiency. The results highlight the critical role of modular architecture and optimized databases in enhancing the effectiveness and adaptability of digital education platforms.

Keywords: Digital education, Django, modular architecture, database, ER model, REST API, WebSocket, security

INTRODUCTION

In today’s modern education system, digital platforms play a crucial role in simplifying the learning process, enhancing interactivity, and expanding the possibilities of distance learning. Platforms specifically designed for primary school students serve to make education more engaging, convenient, and adaptable. The success of such platforms largely depends on the careful design of their architecture and database structure.

The Django framework, with its robust structure, fast data exchange via REST APIs, and efficient integration with databases, is an ideal choice for developing educational platforms. This article focuses on designing the architecture and optimizing the database of a digital learning platform developed using Django.

LITERATURE REVIEW

Recent research on designing digital learning platforms emphasizes the modularity, security, and rapid development capabilities of the Django framework. The "Shreic" project developed by Adamya Shyam et al. demonstrates the design of an educational resource sharing platform based on the SDLC model, although real-time communication capabilities were limited [1].

Sameena et al. (2021) introduced an e-learning platform based on Django and SQLite, implementing student, teacher, and admin modules. However, technologies like WebSocket were not utilized [2].

In the article by ButterCMS (2023), the integration of Django with content management systems for building flexible platforms is discussed [3].

In the field of databases, the relational model and normalization principles (1NF, 2NF, 3NF) proposed by E.F. Codd (1970) are fundamental for ensuring data integrity [4].

Connolly and Begg (2014) present practical approaches to ER models, indexing, and query optimization in their book Database Systems [5].

Silberschatz et al. (2020) analyze transaction management and distributed systems, illustrating methods for handling large volumes of data [6].

Özsu and Valduriez (2011) offer solutions for distributed databases, emphasizing efficient load distribution [7].

Bhogekar et al. (2022) discuss modern approaches to query processing and security optimization [8].

The literature review indicates that modular architecture and optimized databases are crucial for digital learning platforms.

RESEARCH METHODOLOGY

The research employed an iterative SDLC (Software Development Life Cycle) model, which allows for the gradual identification of project requirements and the testing of features. In the iterative model, the project management list (PCL – Project Control List) is initially created. This list contains the tasks and functionalities to be implemented in the system, arranged in a specific order. In each iteration, one task is selected from the PCL, and the following steps are carried out sequentially: planning, analysis, design, development, testing, and evaluation. After this process, the functionality is removed from the PCL, and the next task is taken up. In cases where new requirements arise, they are added to the PCL and will be considered in subsequent iterations (see Figure 1).

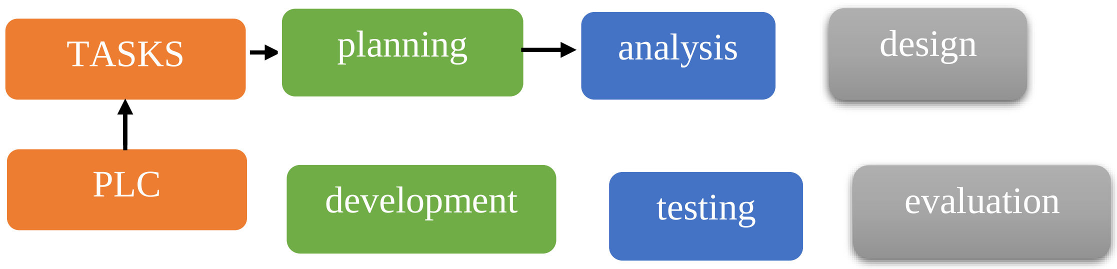

Figure 1: Stages of the Iterative Model

Technical Fundamentals

- **Django Framework**: Chosen for its REST API, security features, and extensibility. Django's architecture makes it a robust choice for building scalable and secure digital platforms.
- **PostgreSQL**: Selected for efficiently managing large volumes of data. It ensures data consistency, reliability, and performance, making it an ideal choice for database management in educational platforms.
- **Django Channels and WebSocket**: Implemented to support real-time communication. WebSocket ensures that students and teachers can interact in real-time, enhancing the platform's interactivity.

The platform is designed for learners in grades 1 to 4 and includes the following modules:

1. Learner Module: Personal accounts, educational materials, tests, and performance tracking.

2. Teacher Module: Lesson schedules, material uploads, test analysis, and communication with learners.

3. Real-time Communication Module: Chat and instant messaging features to facilitate communication between learners and teachers.

The primary modules designed for the platform are as follows:

✓ Learner Module: Provides personal account functionality for primary school learners. Through this module, users (learners) can view educational materials related to their subjects, take tests, and track their results.

✓ Teacher Module: Designed for primary school teachers, this module allows them to create lesson schedules, upload educational materials and assignments, analyze test results, and monitor learner activities. Teachers can also directly communicate with learners and exchange feedback through the platform.

✓ Real-time Communication Module: Powered by Django Channels and WebSocket technology. This module allows learners and teachers to chat in real-time and exchange instant messages. This communication feature enhances the interactivity of the learning process and ensures personalized approaches to teaching and learning (Figure 2).

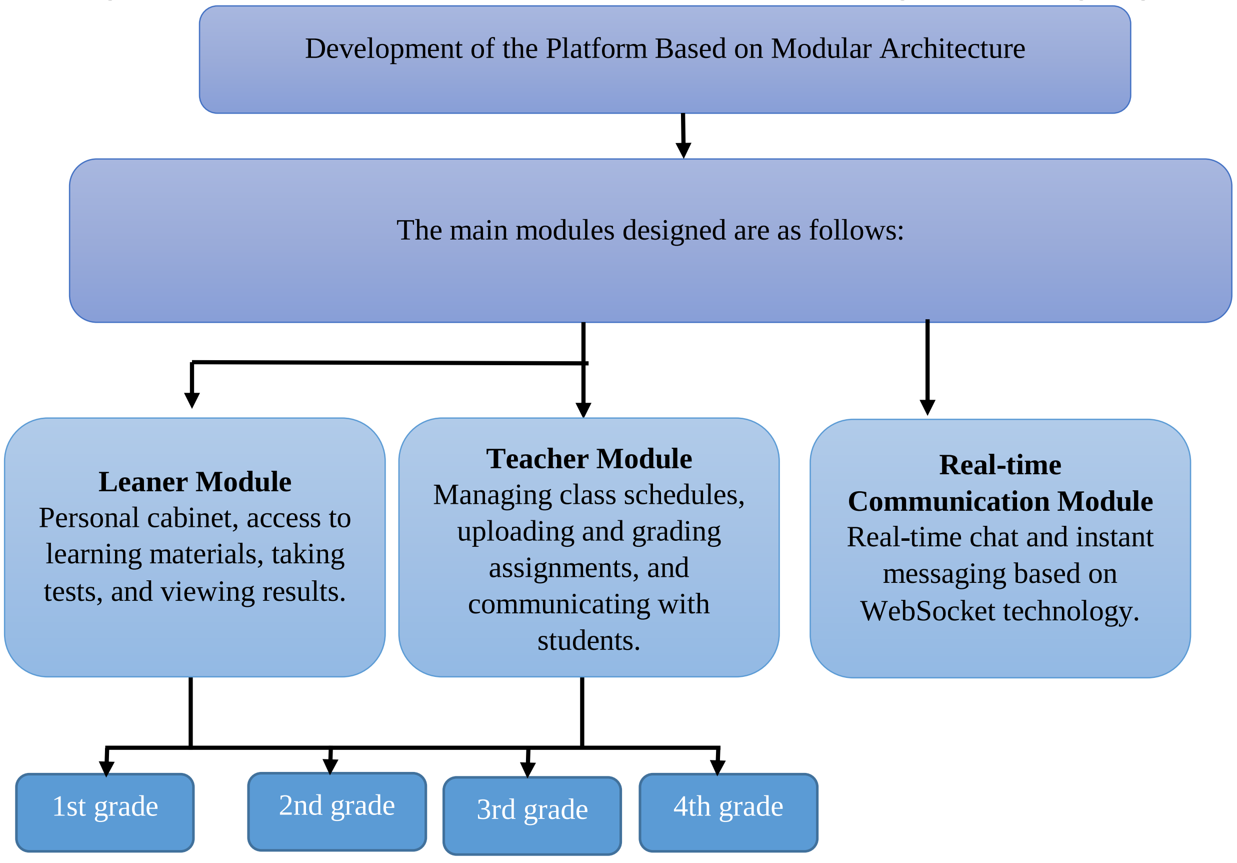
Development of the Platform Based on Modular Architecture
The main modules designed are as follows:
Leaner Module
Personal cabinet, access to learning materials, taking tests, and viewing results.
Teacher Module
Managing class schedules, uploading and grading assignments, and communicating with students.
Real-time Communication Module
Real-time chat and instant messaging based on WebSocket technology.
1st grade
2nd grade
3rd grade
4th grade

Figure 2. Development of the Platform Based on Modular Architecture

The modular architecture of the digital learning platform is developed based on Django, with each module designed and tested separately. The process of installing Django and creating modules is carried out in a structured, step-by-step manner, just like the platform itself (Figure 3).

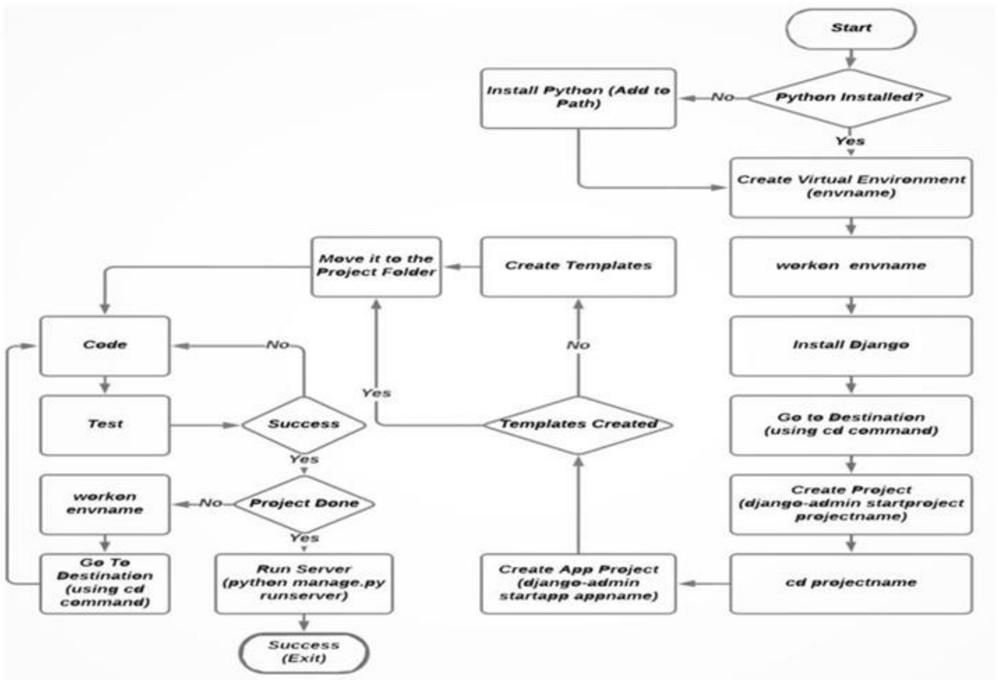
Start
Python Installed?
No
Install Python (Add to Path)
Yes
Create Virtual Environment (envname)
workon envname
Install Django
Go to Destination (using cd command)
Create Project (django-admin startproject projectname)
cd projectname
Create App Project (django-admin startapp appname)
Templates Created
No
Create Templates
Move it to the Project Folder
Yes
Code
Test
Success
No
Yes
Project Done
No
workon envname
Go To Destination (using cd command)
Yes
Run Server (python manage.py runserver)
Success (Exit)

Figure 3. The process of installing Django and creating modules
All modules are interconnected through REST APIs built using the Django REST Framework, ensuring fast and secure communication between users.

The database is designed based on the relational model:

✓ Normalization: The 1NF, 2NF, and 3NF stages were applied to reduce data redundancy.

✓ ER Model: The entities for students, teachers, classes, subjects, and communication are interconnected.

Table structure:

1. Pupils Module:

✓ leaners: leaner_id, first_name, last_name, class_id, username, password_hash.
✓ Classes: class_id, class_name (e.g., “Grade 1”).
✓ Subjects: subject_id, subject_name, class_id.

2. Teacher Module:

✓ Teachers: teacher_id, first_name, last_name, username, password_hash.
✓ Schedules: schedule_id, teacher_id, subject_id, class_id, time.
✓ Assignments: assignment_id, teacher_id, subject_id, content.

3. Real-Time Communication Module:

✓ Chat_Messages: message_id, sender_id, receiver_id, message_text, timestamp.
✓ **Relationships:**

- Learners.class_id → Classes.class_id (learners are linked to classes).
- Subjects.class_id → Classes.class_id (subjects are assigned to classes).
- Assignments.teacher_id → Teachers.teacher_id (assignments are linked to teachers).
- Chat_Messages.sender_id, receiver_id → general user tables.

✓ **Optimization methods:**

- Indexes: Indexes were added to the fields learner_id, teacher_id, subject_id, and class_id.
- Caching: Learning materials and schedules are cached.
- WebSocket: Real-time communication is optimized through Django Channels.

ANALYSIS AND RESULTS

The platform's modular architecture and database design were analyzed based on the principles of relational algebra and relational calculus. The following outcomes were identified:

1. **Learner Module:**

- Relational Algebra:

StudentSubjects ← π first_name, last_name, subject_name ((Learners ⋈ Learners.class_id = Classes.class_id) ⋈ Classes.class_id = Subjects.class_id)

This query identifies the subjects taken by learners.

- Relational Calculus:

{ s.first_name, s.last_name | Learners(s) ∧ Classes(c) ∧ s.class_id = c.class_id ∧ c.class_name = 'Grade 3' }

This expression filters learners by class name.

2. **Teacher Module:**

- Relational Algebra:

π first_name, last_name, content (Teachers ⋈ Teachers.teacher_id = Assignments.teacher_id)

This query shows the assignments created by teachers.

- Relational Calculus:

{ t.first_name, t.last_name, a.content | Teachers(t) ∧ Assignments(a) ∧ t.teacher_id = a.teacher_id }

3. **Real-time Communication Module:**

- Relational Algebra:

π sender_id, receiver_id, message_text, timestamp (Chat_Messages)

This query returns messages.

- Relational Calculus:

{ m.sender_id, m.receiver_id, m.message_text, m.timestamp | Chat_Messages(m) }

To ensure data security, JWT authentication and password hashing (via Django's PasswordHasher) were implemented. System performance was enhanced through indexing and caching, which helped reduce query execution time. The modular architecture simplified the integration of new features, such as a dedicated module for parents.

CONCLUSION AND RECOMMENDATIONS

This research demonstrated that a modular architecture and optimized database, developed using Django, can provide an effective digital learning environment. Real-time communication was enabled via REST APIs and WebSocket technology, while a normalized relational database ensured data integrity and high performance. JWT-based authentication and password hashing significantly improved platform security.

Recommendations:

1. Scalability: Increase the system's capacity to serve more users by implementing distributed architecture and database partitioning.
2. Security: Introduce multi-factor authentication (MFA) and data encryption mechanisms to further enhance data protection.
3. Integration: Align the platform with national educational standards to ensure curricular compliance.
4. Artificial Intelligence: Incorporate AI-based personalized learning and analytics

IBAST | Volume 5, Issue 05, May

functionalities to adapt to individual learners' needs.
5. Monitoring: Continuously analyze system metrics and user feedback to improve performance and user experience.

This platform contributes to the advancement of digital education by offering an interactive, secure, and adaptable learning environment tailored for primary school learners.

References:

1.Adamya Shyam, Nitin Mukesh. "A Django Based Educational Resource Sharing Website: Shreic." Journal of Scientific Research, vol. 64, no. 01, 2020, pp. 238-252.
2.Sameena, Dua, K., Gaur, M., Bhatia, S., & Gopavaram, S. (2021). Django Based E-Learning Website. International Journal for Research in Applied Science & Engineering Technology (IJRASET), 9(12), 2266–2272.
3.ButterCMS. (2023). Building an Online Learning Platform with ButterCMS and Django. ButterCMS Blog.
4.Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 13(6), 377–387.
5.Connolly, T., & Begg, C. (2014). Database Systems: A Practical Approach to Design, Implementation, and Management. Pearson.
6.Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). Database System Concepts. McGraw-Hill Education.
7.M.T. Özsu, P. Valduriez. "Principles of Distributed Database Systems". Springer Science+Business Media, 2011.
8.Monali Bhogekar, Neehal B. Jiwane, Lovelesh N. Yadav. Database Architecture and Management System. International Journal of Advanced Research in Computer and Communication Engineering, vol. 11, issue 11, 2022.