IT FOR SOCIETY, Vol. 09, No. 01
ISSN 2503-2224

Web-Based Realtime Course Platform With Integrated Live Coding Interface

Reinald Chenartha[1], Cutifa Safitri[2]
Faculty of Computing
President University
Cikarang, Indonesia
reinald.chenartha@student.president.ac.id

***Abstract*— This paper explores an innovative technological approach to a programming education platform. The platform aims to enhance the learning experience by integrating web-based resources with real-time video communication and live coding. Utilizing a peer-to-peer connection framework, this method facilitates efficient and direct communication between instructors and learners, optimizing performance and reducing latency. The primary goal is to create a dynamic and collaborative learning environment. Through the integration of peer-to-peer video chat and an Integrated Development Environment (IDE), the platform aims to improve real-time communication, code reviews, and interactive problem-solving, providing learners with a profound programming learning experience.**

Keywords—peer-to-peer communication, integrated development environment, electronic learning, collaboration

I. Introduction

Education in society continues to evolve with technological advancements, utilizing new methods to enhance efficiency and effectiveness. Despite the rapid progress in educational technology, traditional programming education still faces challenges in providing a deep and practical learning experience. The conventional model, which focuses on theoretical instruction, is less effective in equipping learners with direct skills and real-world problem-solving abilities. Therefore, students often struggle to transition from theoretical learning to practical applications in software development. In response to this change, this thesis proposes an innovative approach through the design and implementation of a social education platform that integrates real-time meeting capabilities and an Integrated Development Environment (IDE). The goal is to redefine the programming education experience by incorporating advanced features.

II. Literature Review

A. Related Applications

In terms of development aspects, this application can be considered inspired by a combination of several aspects from other existing applications. Fiverr is a digital platform where freelancers can sell their services or creative work. The offered services commonly posted on Fiverr vary among creative works, such as graphic design, animation, programming, and writing [1]. In addition to creative products and services, Fiverr is also commonly used by sellers as a platform to offer their knowledge through teaching services [2]. However, for this type of service, the implementation of the teaching process needs to be outsourced by other applications that enable optimal meeting sessions. Udemy is a dedicated online learning platform that serves as a virtual educational ecosystem for instructors and individuals looking to learn specific subjects [3]. Courses on Udemy are generally developed in the form of educational videos, meaning the learning process for each course is conducted in a one-way communication method [4]. Codecademy is an interactive online education platform specifically developed for learning programming [5]. The course content consists of hands-on practice, quizzes, and also source materials from course authors. The interactive programming education method used by Codecademy is considered highly engaging for learners, along with the gamification of the learning process [6].

B. Development Software

In addition to existing applications, there are also frameworks or tools whose functionalities provide inspiration for the conceptual ideas embraced. JavaScript is an object-oriented programming language commonly used asynchronously [7]. JavaScript can also be considered a functional programming language that utilizes many function calls, whether for callback purposes or other intentions [8]. Node.JS can be defined as a JavaScript runtime with access to computer I/O systems [9]. Apart from practical reasons, such as not needing to use multiple languages to build a full-stack web system, another key component surrounding Node.JS is its extensive ecosystem [10]. Express.JS can be regarded as a web framework for Node.JS that provides default configurations for standard http modules in Node.JS and inherits middleware properties from the Connect Node.JS library [11]. MongoDB is a document-oriented database system, as opposed to a relational database system that uses table-structured data [12]. On the other hand, Mongoose can be considered a highly practical Node.JS library used to communicate with the MongoDB server [13]. Child Process is a standard library in Node.JS used to access operating system functionality [14]. This module also enables the ability to receive command arguments as if using the command-line interface directly and access standard input/output from the process.

WebSocket API (Application Programming Interface) is an API used for event-driven communication methods and can be directed to clients connected to a party acting as a WebSocket server [15]. WebSocket connections are initiated using an HTTP request that will remain open after the initialization process. Because the connection stays open even after the initialization process, WebSocket clients and servers can exchange messages as long as the connection is not closed. WebRTC is a core JavaScript API capable of performing real-time media and data transfers using peer-to-peer communication method [16]. WebRTC uses signaling methods to initialize connections with remote peers. This signaling method requires a third-party component to send the necessary information to the remote peer. This third-party component can be an HTTP request or a WebSocket server. However, due to the complex network architecture nowadays, establishing peer connections requires a series of protocols called ICE (Interactive Connectivity Establishment) protocols. ICE protocols allow each peer to find the best connection path to reach each other. In addition to ICE protocols, modern network architecture also requires STUN (Session Traversal Utilities for NAT) and TURN (Traversal Using Relays around NAT) servers in WebRTC implementations.

III. System Analysis

A. System Overview

This course platform is a web-based application that serves as a space for users to discover instructors and receive real-time guidance in various skills. The application offers a highly simple interface, allowing users to play a dual role as both instructors and students with a single account. In addition to booking courses from other instructors, users can also provide courses they want to teach. The real-time mechanism on this platform includes video chat, audio chat, and text chat between instructors and students. Although not limited to technology courses, each course session comes with a simple online compiler and live coding capabilities for sharing code in real-time between instructors and students.

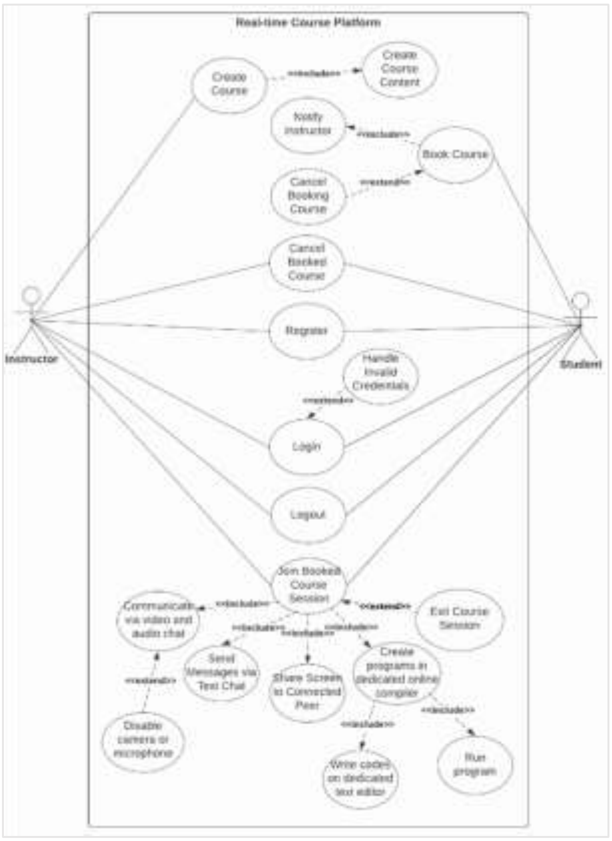

Figure 1: Application use case diagram

This application is built with various functionalities. These functionalities include features that allow each user to create an account, log into their account, create course instances where they can become instructors, book courses created by other users where they can participate as students, receive notifications whenever other users book their courses, easily join any course session with the ability to meet instructors or students in real-time using video and audio chat, share their screen to their peers to facilitate the course flow depending on external applications, send text messages through the text chat interface in each course session, run code on the dedicated online compiler interface in each course session, and share their code through a collaborative coding space on the dedicated online compiler interface.

B. Hardware and Software Requirements

This application has specific hardware requirements for both the server and client sides. The recommended server infrastructure utilizes cloud services, specifically the Amazon Web Services Elastic Compute Cloud (AWS EC2) instance, for flexibility and ease of setup. The T2 type of EC2 instance is chosen for this monolithic application, integrating both backend and frontend, providing suitable resources for lightweight performance. The application uses WebRTC for peer-to-peer connections on the client side, supported by STUN and TURN servers to overcome NAT traversal. Minimal hardware requirements are selected to ensure smooth performance on computers connected to the internet with appropriate software. The server side operates on Ubuntu Linux, leveraging Node.js for the backend with Express.js, WebSocket, and Mongoose for communication and interaction with MongoDB. MongoDB is chosen as the database due to its document format similar to JSON. WebRTC on the browser interface enables peer-to-peer connection features, such as video streaming and code sharing. Python on the server side runs as a parallel process, allowing users on the client side to write Python scripts with live-share code features for synchronized code execution across various Python processes.

IV. System design

A. *User Interface Design*

This web-based application focuses on user interface design with consideration for its usability across various screen widths. Primary considerations involve easy access to information, intuitive navigation, and a friendly user experience. The registration and login pages are designed simply to minimize confusion, while the main page is divided into three main sections with buttons and cards that facilitate user interaction. The course creation and course detail dialogs are designed with a user-friendly interface, simplifying the course booking and cancellation processes. The meeting session room, being the most complex dialog, displays dynamic components and buttons contributing to a feature-rich meeting experience. This meticulous user interface design ensures a cohesive and user-friendly experience across various pages and functionalities within the application.

B. *Class Diagram*

The application's database schema includes the User schema, representing each user with properties such as email address, username, and password. Users can have a one-to-many relationship with the Course schema and Booked Course schema, indicating their ability to create multiple courses and participate in various courses created by others. The Course schema consists of title, description, instructor, and a list of course content properties. It forms a one-to-many relationship with the User schema and the Booked Course schema. The Course Content schema, used as a sub-document for the Course schema, includes title, description, and duration. It forms a one-to-many relationship with the Course schema and a many-to-many relationship with the Booked Course schema. The Booked Course schema manages details such as participating users, start and end times, references to courses, and a list of references to course content. It serves as a transactional schema, specifying the date, time, and course content booked by users, with relationships indicating connections to users, courses, and course content.

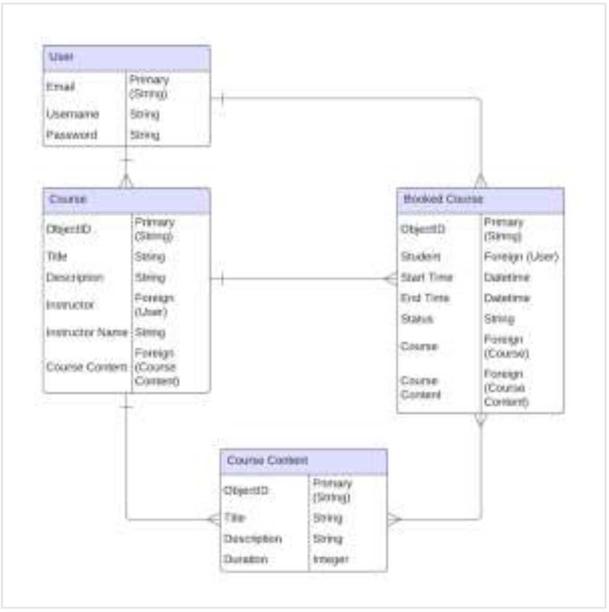

Figure 2: Application Entity relationship diagram (ERD)

V. System Implementation

A. *User Interface*

This web application utilizes HTML for interface structure, CSS for element design, and JavaScript for interface logic. The use of Embedded JavaScript (EJS) on the registration page enables dynamic rendering on the server, separating content into manageable components and enhancing modularity. The main page, the user interaction hub, incorporates Bootstrap 5, FontAwesome, and JQuery to create a dynamic experience. Dialog boxes, such as course creation and booking, leverage Bootstrap 5 modal API to present structured and visually appealing interactions. The login and registration pages are designed simply with EJS for dynamic rendering, pure CSS design, and Bootstrap 5 for interactive elements. The course creation dialog box uses Bootstrap 5 modal API to add dynamic rows to the course content table. The course detail and booking dialog boxes use Bootstrap 5 design and custom CSS, providing comprehensive information and organizing the booking process with responsive date and time inputs. The meeting session room, as a comprehensive feature, uses Bootstrap 5 and custom CSS for its layout and design. Divided into video, activities, and communication configuration sections, it provides an intuitive interface with visual and interactive elements. Overall, the application combines various technologies to create an engaging and functional user interface.

B. *Application Details*

This web-based application combines the power of JavaScript with Node.js as the runtime on both the client and server sides, leveraging various libraries to provide efficient execution. On the server side, initialization involves configuring the Express server to handle routes and WebSocket to support real-time communication. The use of MongoDB as a non-relational database system, accessed through Mongoose as an Object-Document Mapper (ODM), provides a solid structure for data storage and retrieval.

The user registration process begins by rendering the registration page using EJS templates and Express as the route. The submitted data is processed on the server using the

Mongoose User model, verifying the existence of the user and creating a new entry if the entered data is valid. The login process follows a similar step, with Mongoose and Express user session used for security and session management.

Rendering the list of courses involves an API route that returns course data, with client-side processing using jQuery to generate dynamic HTML elements. Course creation allows users to create courses through a dialog box with client-side validation, and the server handles the creation of a new course entry in the database. Booking a course involves submitting a form with details to the server via a POST request, and the server responds by creating a new Booked entry, notifying the course creator via WebSocket, and updating the booked courses section for the user. Canceling a booked course allows users to cancel their courses, triggering updates on the course creator's notifications and the user's booked courses.

The meeting session mechanism uses WebRTC for real-time communication, including video communication and data channels. The signaling process involves the exchange of offers and answers between peers through the WebSocket server, while ICE candidates are exchanged to ensure optimal network connectivity. The live coding mechanism relies on the Python runtime on the server, where the client sends Python scripts and input via AJAX and WebSocket, and the server executes the script in a secure environment, providing output back to the client through HTTP response. WebRTC is also used here to facilitate a collaborative experience with other peers in live coding. With these features, the application provides a comprehensive and dynamic web experience with user registration, course management, real-time communication, and live coding capabilities.

VI. System Testing

System testing for this application was conducted manually, avoiding the use of specific testing frameworks. The testing environment was set up on the client-side device with specific hardware and software configurations, including an AMD Ryzen 5 3550H CPU, AMD Radeon Vega Mobile Gfx GPU, 8GB + 8GB DDR4 Dual 2.4 GHz RAM, Windows 11 Home Operating System, and Google Chrome Internet Browser. This manual testing process was carried out following a test-driven development methodology, conducted iteratively during the development phase. This approach helps ensure systematic and smooth testing, with positive results associated with careful attention to hardware specifications and the selection of commonly used operating systems and internet browsers. Overall, this approach enhances the robustness and reliability of the application by creating a testing environment that reflects real-world scenarios.

VII. Conclusion and Future Works

A. Conclusion

This web-based application serves as a versatile platform facilitating the creation of personalized learning journeys in real-time interactions between instructors and students. The development of the application is grounded in JavaScript, utilized for both the frontend and backend components. Notably, the application integrates a non-relational database system and leverages several libraries, including an Object-Document Mapper (ODM), to establish a seamless connection with the database. Drawing inspiration from features found in prior applications, such as real-time video chat and programming functionalities, the final product has successfully materialized, staying true to its initial conceptual foundations. The development process adhered to the outlined plan, incorporating essential pre-planned tools and technologies, resulting in a well-rounded application that aligns with its intended purpose of fostering dynamic and customized learning experiences.

B. Future Works

The initial release of the application achieved its technical goals, but there is room for improvement and additional features to enhance user experience and marketability. Notably, the user interface design needs refinement, and introducing a payment method for course content could boost the application's appeal. Considering potential use in educational institutions, modifications are needed to adapt to one-to-many learning settings. Scaling up the meeting room functionality, adjusting peer connection initiation, and refining the video chat interface would be essential. Integration with existing educational institution systems is another possibility, requiring the development of new REST API routes for seamless data exchange.

To enhance the Integrated Development Environment (IDE) feature, a novel approach involves creating separate containers or virtual spaces for each IDE session, granting users access to isolated terminals with distinct root directories. This not only allows for greater program variety and user security by avoiding program execution on the server but also enables the installation of various programming languages within each container. Future enhancements could involve incorporating more advanced development interfaces like Jupyter Notebook and web programming, providing users with a broader understanding of programming languages and associated tools, extending beyond basic terminal-based applications to deliver more value in diverse programming scenarios.

VIII. References

[1] D. D. Green, J. McCann, T. Vu, N. Lopez and S. Ouattara, "Gig Economy and the Future of Work: A Fiverr.com Case Study," Management and Economics Research Journal, Vol. 4, pp. 281-288, 2018.

[2] B. Maina, "How to Teach On Fiverr – A Step by Step Guide," 16 June 2022. [Online]. Available: https://www.linkedin.com/pulse/how-teach-fiverr-step-guide-belinda-maina/. [Accessed 28 November 2023].

[3] B. Panth and R. Maclean, Anticipating and Preparing for Emerging Skills and Jobs, Asian Development Bank, 2020.

[4] R. J. Eusebio Jr., "Computer Engineering Students Performance Using the Udemy and Khan Academy Videos in Learning Differential Equations," in 4th INTERNATIONAL CONGRESS ON ACTION RESEARCH, ACTION LEARNING, Manila, 2019.

[5] J. Sharp, "Using Codecademy Interactive Lessons as an Instructional Supplement in a Python Programming Course," Information Systems Education Journal, vol. 17, pp. 20-28, 2019.

[6] R. Shen and M. J. Lee, "Learners' Perspectives on Learning Programming from Interactive Computer Tutors in a MOOC," 2020 IEEE Symposium on Visual

Languages and Human-Centric Computing (VL/HCC), 2020.

[7] D. Flanagan, JavaScript: The Definitive Guide, Sixth Edition, M. Loukides, Ed., O'Reilly Media, Inc., 2011, pp. 1-8.

[8] J. Resig and B. Bibeault, "Enter the ninja," in Secrets of the JavaScript Ninja, Manning Publications Co., 2013, pp. 3-12.

[9] M. Cantelon, M. Harter, T. Holowaychuk and N. Rajlich, Node.js in Action, Shelter Island: Manning Publications Co., 2014.

[10] M. Casciaro, Node.js Design Patterns, Birmingham: Packt Publishing Ltd., 2014.

[11] A. Mardan, Pro Express.js, Springer Science+Business Media New York, 2014.

[12] K. Chodorow and M. Dirolf, MongoDB: The Definitive Guide, O'Reilly Media, Inc., 2010.

[13] E. M. Hahn, Express in Action, Shelter Island: Manning Publications Co., 2016.

[14] S. Buna, "Node.js Child Processes: Everything you need to know," 9 June 2017. [Online]. Available: https://medium.com/edge-coders/node-js-child-processes-everything-you-need-to-know-e69498fe970a. [Accessed 28 November 2023].

[15] A. Lombardi, WebSocket, y O'Reilly Media, Inc., 2015.

[16] S. Loreto and S. P. Romano, Real-Time Communication with WebRTC, O'Reilly Media, Inc., 2014.