JURNAL JTIK
(JURNAL TEKNOLOGI INFORMASI DAN KOMUNIKASI)
KITA Komunitas Informasi Teknologi Aceh
OPEN ACCESS
E-ISSN: 2580-1643
Volume 11, Nomor 1, Januari 2027 | Pages: 209-216 | DOI: https://doi.org/10.35870/jtik.v11i1.7567

Containerized Runtime Berbasis Docker untuk Lingkungan Praktikum Pemrograman yang Seragam dan Terisolasi

Bima Cakra Bara Karebet [1], Angger Binuko Paksi [2*], Tri Septianto [3]

[1,2*,3] Program Studi D-III Teknologi Informasi, Politeknik Negeri Madiun, Kota Madiun, Provinsi Jawa Timur, Indonesia.

Corresponding Email: [2*]angger.binuko@pnm.ac.id.

Article History:
Received: 8 June 2026.
Revised: 8 July 2026.
Accepted: 14 July 2026.
Available Online: 1 November 2026.
Published: 1 January 2027.

Abstract

Algorithm and Programming practicum requires a uniform execution environment so that students can run code and participate in practicum activities consistently. Differences in devices, operating systems, and software versions often lead to installation issues and inconsistent execution results. This study implements a Docker-based containerized runtime in a web-based interactive practicum platform using the Rapid Application Development (RAD) method, encompassing requirements planning, system design, development, implementation, and testing. The developed platform provides features for registration, login, content management, practicum work through a web-based editor, real-time answer checking, and progress reports. Testing results using black box testing show that all 13 core features, verified through 63 test scenarios covering both student and lecturer roles, met the defined success criteria (100% compliant). These results indicate that Docker can provide an isolated and stable practicum environment, supporting a more consistent programming practicum process.

Keywords: Containerized; Runtime; Docker; Practical Platform; Interactive Programming Algorithm.

Abstrak

Praktikum Algoritma dan Pemrograman memerlukan lingkungan eksekusi yang seragam agar mahasiswa dapat menjalankan kode dan mengikuti kegiatan praktikum secara konsisten. Perbedaan perangkat, sistem operasi, dan versi perangkat lunak sering mengakibatkan kendala instalasi dan perbedaan hasil eksekusi. Penelitian ini menerapkan containerized runtime berbasis Docker pada platform praktikum interaktif berbasis web menggunakan metode Rapid Application Development (RAD), mencakup perencanaan kebutuhan, perancangan sistem, pengembangan, implementasi, dan pengujian. Platform yang dikembangkan menyediakan fitur registrasi, login, manajemen konten, pengerjaan praktikum melalui editor berbasis web, pengecekan jawaban secara langsung, dan laporan progres pembelajaran. Hasil pengujian menggunakan black box testing menunjukkan bahwa seluruh 13 fitur utama, yang diverifikasi melalui 63 skenario pengujian pada peran mahasiswa dan dosen, berjalan sesuai kriteria keberhasilan yang ditetapkan (100% sesuai). Hasil ini menunjukkan bahwa Docker menyediakan lingkungan praktikum yang terisolasi dan stabil, mendukung proses praktikum pemrograman yang lebih konsisten.

Kata Kunci: Containerized; Runtime; Docker; Platform Praktikum; Algoritma Pemrograman Interaktif.

This work is licensed under a Creative Commons Attribution 4.0 International License.

Copyright © 2027 by the authors. Published by Lembaga Komunitas Informasi Teknologi Aceh (KITA). All rights reserved.

E-ISSN: 2580-1643

1. Pendahuluan

Pembelajaran praktikum pada bidang pemrograman memerlukan lingkungan kerja yang seragam agar mahasiswa dapat menjalankan kode, mengikuti instruksi praktikum, dan memperoleh hasil eksekusi yang konsisten. Pada praktiknya, perbedaan perangkat, sistem operasi, konfigurasi, serta versi perangkat lunak sering menyebabkan kendala instalasi, perbedaan hasil eksekusi, dan kesulitan dalam mereplikasi materi praktikum. Kondisi tersebut dapat menghambat kelancaran proses pembelajaran, terutama pada praktikum yang membutuhkan kesiapan lingkungan pemrograman dan basis data. Oleh karena itu, diperlukan platform pembelajaran praktikum berbasis *web* yang mampu menyediakan lingkungan eksekusi terstandarisasi, mudah diakses, dan tidak bergantung pada instalasi perangkat lunak di perangkat masing-masing mahasiswa. Beberapa penelitian terdahulu telah membahas pemanfaatan teknologi digital untuk mendukung pembelajaran. (Stevanus Ocktoberrikho & Noviyanti P, 2025) mengembangkan platform *e-learning* berbasis *Android* untuk siswa sekolah dasar yang berfokus pada penyampaian materi dan aktivitas belajar, sedangkan (Jinan dkk., 2025) merancang sistem *e-learning* berbasis *framework Laravel* untuk pengelolaan materi, kelas, dan pengguna secara umum. (Prasetya & Shofiya Syidada, 2025) mengintegrasikan *serious games* ke dalam platform pembelajaran pemrograman untuk meningkatkan keterlibatan mahasiswa, dan (Sukendar, 2023) merancang sistem penilaian otomatis (*auto-grading*) untuk program *Java* berbasis *web* yang memberikan umpan balik langsung terhadap kode mahasiswa.

Sementara itu, (Cahyanto, 2023)menganalisis pemanfaatan platform *e-learning* berbasis *cloud computing* dalam meningkatkan efektivitas pembelajaran jarak jauh. Kelima studi tersebut menunjukkan bahwa platform pembelajaran berbasis *web* dapat meningkatkan keterlibatan dan kemudahan akses mahasiswa, namun tidak satu pun yang secara eksplisit mengatasi heterogenitas lingkungan eksekusi kode di perangkat mahasiswa. Mahasiswa tetap memerlukan instalasi bahasa pemrograman atau *dependency* secara lokal sesuai spesifikasi perangkat masing-masing. Pada sisi lain, teknologi Docker telah banyak digunakan dalam konteks infrastruktur dan *deployment*. (Isron & Putra, 2021) menerapkan Docker sebagai *virtual server* untuk sistem informasi geografis, (Megantara dkk., 2022) memanfaatkan Docker untuk memaksimalkan utilitas *server* universitas pada masa pandemi Covid-19, (Asmara dkk., 2024) menggunakan Docker untuk pengelolaan aplikasi *web* pada instansi pemerintahan, dan (Butarbutar dkk., 2024) menunjukkan efisiensi *container* Python pada sistem operasi Fedora Linux. (Afrizal & Prihanto, 2022) membandingkan kebutuhan sumber daya antara *single server*, virtualisasi, dan *container*, dan menemukan bahwa *container* lebih ringan karena berbagi *kernel* dengan *host* tanpa memerlukan sistem operasi tersendiri seperti pada *virtual machine*. Selain itu, (Sutanto dkk., 2021)merancang arsitektur *container* Docker untuk aplikasi berbasis Node.js dan basis data NoSQL guna meningkatkan skalabilitas dan portabilitas sistem, (Zulfahrizan dkk., 2024) mengimplementasikan *MinIO* sebagai sistem penyimpanan objek yang diintegrasikan dengan *container* Docker pada *learning management system*, dan (Alqaisi dkk., 2024) menganalisis performa berbagai teknologi *container* untuk aplikasi *computer vision* pada perangkat *edge*. Studi-studi ini membuktikan bahwa Docker mampu menyediakan lingkungan yang ringan, efisien, dan terisolasi, tetapi penerapannya masih berfokus pada kebutuhan infrastruktur dan *deployment* aplikasi, bukan pada alur pengerjaan praktikum interaktif yang langsung diakses mahasiswa. Penelitian yang secara khusus mengintegrasikan *containerized runtime environment* ke dalam platform praktikum interaktif untuk mendukung pembelajaran seperti Python dan basis data secara langsung, sekaligus menyediakan sisi pemantauan bagi dosen, masih terbatas. Keterbatasan tersebut menjadi dasar pengembangan platform yang tidak hanya menyediakan materi praktikum, tetapi juga lingkungan eksekusi yang seragam, terisolasi, dan dapat digunakan mahasiswa melalui peramban *web*. Penelitian ini menawarkan penerapan *containerized runtime environment* menggunakan Docker pada platform pembelajaran praktikum interaktif berbasis *web*.

Dengan adanya integrasi Docker, setiap proses eksekusi kode dan kebutuhan praktikum dapat dijalankan dalam lingkungan yang lebih stabil dan terkontrol. Pendekatan ini diharapkan dapat membantu mahasiswa mengerjakan praktikum tanpa instalasi manual, sekaligus memudahkan dosen dalam manajemen konten dan memantau progres pembelajaran. Tujuan penelitian ini adalah menerapkan *containerized runtime environment* berbasis Docker pada platform pembelajaran praktikum interaktif berbasis *web* untuk menyediakan lingkungan kerja praktikum yang seragam, terisolasi, dan mudah diakses. Penelitian ini juga bertujuan mendukung proses praktikum Algoritma dan Pemrograman serta basis data agar berjalan lebih konsisten, terarah, dan sesuai dengan kebutuhan pembelajaran. *Virtual machine* (VM) merupakan teknologi virtualisasi yang menjalankan sistem operasi secara penuh di atas *hypervisor*, sehingga setiap VM memiliki *kernel* dan sumber daya sendiri yang terpisah dari *host* maupun VM lain. Sebaliknya, *container* merupakan bentuk virtualisasi pada tingkat sistem operasi (*OS-level*

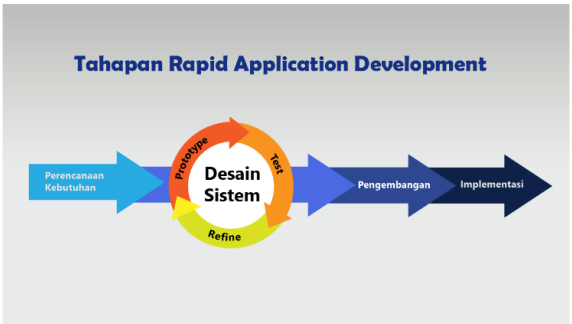
E-ISSN: 2580-1643

virtualization) di mana seluruh *container* berbagi *kernel* yang sama dengan sistem operasi *host*. Karena tidak memerlukan *kernel* atau sistem operasi terpisah untuk tiap lingkungan aplikasi, *container* jauh lebih ringan dan lebih cepat dijalankan dibandingkan VM (Afrizal & Prihanto, 2022). Temuan ini sejalan dengan hasil pengujian *benchmark* (Pratama & Raharja, 2023) yang menunjukkan performa Docker yang lebih unggul dibandingkan lingkungan virtualisasi berbasis VM pada pengujian aplikasi berbasis Node.js. Perbedaan arsitektur inilah yang menjadi dasar pemilihan *container* dalam penelitian ini, karena kebutuhan menjalankan banyak sesi eksekusi kode mahasiswa secara bersamaan pada satu *server* akan lebih efisien menggunakan *container* dibandingkan VM. Docker adalah platform yang menggunakan teknologi *container* untuk menjalankan aplikasi beserta *library*, *dependency*, dan konfigurasi yang dibutuhkan dalam satu *image*, sehingga aplikasi dapat berjalan secara konsisten pada berbagai lingkungan sistem (Isron & Putra, 2021). *Containerized runtime environment* adalah lingkungan tempat aplikasi dijalankan menggunakan teknologi *container*, di mana seluruh *container* berjalan di atas satu *kernel* sistem operasi yang sama namun tetap terisolasi satu sama lain (Butarbutar dkk., 2024). Setiap sesi eksekusi kode Python maupun *query* basis data milik mahasiswa dijalankan pada *container* terpisah yang dibuat dan dihentikan secara dinamis oleh *server*, sehingga kegagalan pada satu sesi tidak memengaruhi sesi mahasiswa lain.

Rapid Application Development (RAD) adalah metode pengembangan perangkat lunak yang menekankan kecepatan pembangunan sistem melalui perancangan bertahap dan keterlibatan pengguna secara berkelanjutan (Rinduh Iriane & Soter Japi, 2025). RAD dipilih dibandingkan *Waterfall* karena kebutuhan fitur platform perlu disesuaikan secara iteratif berdasarkan masukan dosen sebagai pengguna utama selama pengembangan berlangsung, sesuatu yang sulit diakomodasi oleh *Waterfall* yang bersifat linear. Dibandingkan *Agile* yang menekankan iterasi berkelanjutan dalam jangka panjang dengan banyak *sprint*, RAD lebih sesuai untuk pengembangan bertahap tunggal dengan target waktu terbatas seperti pada penelitian ini. *Black box testing* merupakan metode pengujian perangkat lunak yang berfokus pada pengujian fungsional sistem dengan memeriksa masukan dan keluaran tanpa memperhatikan struktur kode program di dalamnya (Adisty Yudianto Putri dkk., 2025). Pengujian ini bertujuan memastikan bahwa seluruh fungsi aplikasi berjalan sesuai kebutuhan pengguna dan spesifikasi yang telah ditetapkan, dengan menyusun skenario pengujian berdasarkan fitur sistem kemudian menguji respons sistem terhadap masukan yang valid maupun tidak valid.

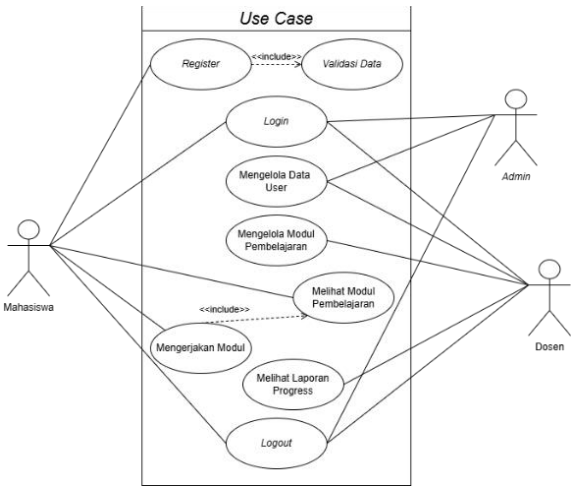
2. Metode Penelitian

Penelitian ini menggunakan metode *Rapid Application Development* (RAD). Metode ini dipilih karena sesuai untuk pengembangan aplikasi yang membutuhkan proses perancangan dan implementasi secara cepat dengan tetap memperhatikan kebutuhan pengguna. Tahapan dalam metode RAD meliputi perencanaan kebutuhan, perancangan sistem, pengembangan, implementasi, dan pengujian. Pada tahap perencanaan kebutuhan, dilakukan identifikasi kebutuhan sistem berdasarkan permasalahan praktikum, yaitu kebutuhan lingkungan kerja yang seragam, fitur pengerjaan praktikum interaktif, serta pemantauan hasil pengerjaan mahasiswa.



Gambar 1. Tahapan *Rapid Application Diagram* (RAD)

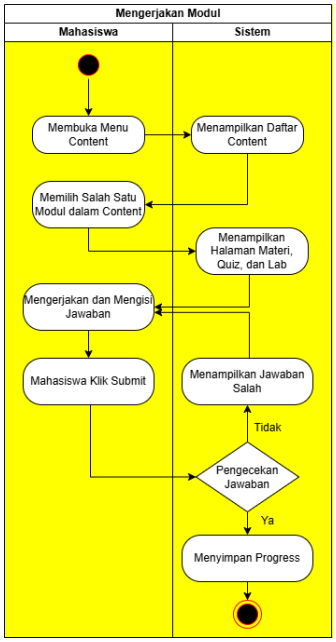
Tahap perancangan sistem dilakukan dengan membuat rancangan alur penggunaan aplikasi, struktur sistem, dan desain antarmuka. Perancangan ini meliputi pembuatan *use case diagram*, *activity diagram*, *sequence diagram, class diagram*, serta rancangan tampilan halaman aplikasi. Rancangan tersebut digunakan sebagai acuan dalam proses pengembangan agar sistem yang dibuat sesuai dengan kebutuhan mahasiswa dan dosen sebagai pengguna utama. Interaksi antara pengguna (mahasiswa, dosen, dan admin) dengan sistem digambarkan melalui *use case diagram* pada Gambar 2.



Gambar 2. Use Case Diagram

E-ISSN: 2580-1643

Gambar 3 menggambarkan alur aktivitas mahasiswa dalam mengerjakan modul pembelajaran. Proses dimulai ketika mahasiswa membuka menu *content*, kemudian sistem menampilkan daftar *content* yang tersedia. Mahasiswa kemudian memilih salah satu modul, dan sistem menampilkan halaman yang berisi materi, *quiz*, dan *lab*. Setelah mahasiswa mengisi jawaban dan menekan tombol *submit*, sistem melakukan pengecekan jawaban secara otomatis. apabila jawaban salah, sistem menampilkan notifikasi kesalahan sehingga mahasiswa dapat mengerjakan ulang, sedangkan apabila jawaban benar, sistem menyimpan progres pengerjaan mahasiswa. Alur ini menegaskan bahwa pengecekan jawaban secara langsung (*real-time answer checking*) berjalan pada setiap iterasi pengerjaan, bukan hanya pada akhir sesi praktikum.



Gambar 3. *Activity Diagram*

Interaksi antara mahasiswa dan sistem dalam proses pengerjaan praktikum digambarkan melalui sequence diagram pada Gambar 4. Proses dimulai ketika mahasiswa membuka menu pembelajaran, kemudian *controller* mengambil data modul beserta materi, *quiz*, dan lab, serta data progres mahasiswa yang tersimpan. Setelah mahasiswa menyelesaikan sesi *quiz* dengan benar, sistem menyiapkan lingkungan eksekusi melalui Docker *container* dan menampilkan soal lab beserta editor kode. Ketika mahasiswa menuliskan kode dan menekan tombol *run*, kode tersebut dieksekusi langsung di dalam Docker *container* untuk menghasilkan *output*, yang kemudian divalidasi oleh sistem terhadap kunci jawaban sebelum hasilnya disimpan sebagai progres mahasiswa. Alur ini menunjukkan bahwa isolasi eksekusi kode melalui *container* terjadi secara per-sesi dan terintegrasi langsung ke dalam alur kerja praktikum, bukan sekadar lapisan infrastruktur terpisah.

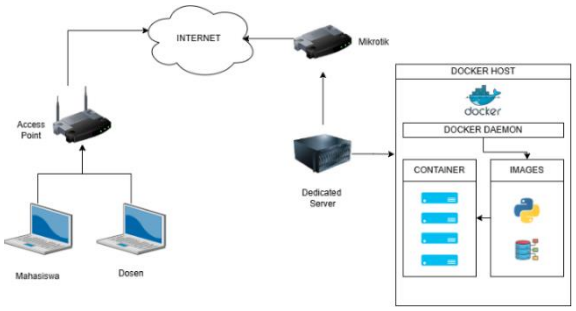


Gambar 4. *Sequence Diagram*

Tahap pengembangan dilakukan dengan membangun platform pembelajaran praktikum berbasis *web* menggunakan *framework* Laravel (PHP). Sistem menyediakan fitur registrasi dan login, pengelolaan materi dan sub materi praktikum, editor kode berbasis *web*, pengecekan jawaban secara langsung, penyimpanan progres, serta laporan progres pembelajaran. Setiap proses eksekusi kode Python atau *query* basis data dijalankan melalui *container* Docker yang dikelola langsung oleh aplikasi Laravel. Tahap implementasi dilakukan dengan menyiapkan lingkungan pengembangan dan pengujian sebagai berikut agar penelitian dapat direproduksi: *dedicated server* dengan prosesor Intel Xeon E5-2695 v4 (18 *core*/36 *thread*), RAM 16 GB ECC, dan penyimpanan SSD 256 GB, dengan sistem operasi Ubuntu Server 24.04.3 LTS. Arsitektur infrastruktur sistem, mencakup jalur akses pengguna melalui jaringan hingga pengelolaan *container* pada *Docker Host*, ditampilkan pada Gambar 3. Perangkat lunak pendukung yang digunakan meliputi *Docker* versi 4.56.0, Python 3.12, dan MariaDB 10.11. Platform kemudian dijalankan pada *server* tersebut agar dapat diakses melalui peramban. Setelah sistem berjalan, dilakukan pengujian menggunakan metode *black box testing*. Skenario pengujian disusun berdasarkan fungsi-fungsi utama sistem pada dua peran pengguna, yaitu mahasiswa (registrasi, *login*, *logout*, profil, akses materi pembelajaran, pengerjaan

E-ISSN: 2580-1643

materi, dan laporan pembelajaran) serta dosen (profil, manajemen pengguna, pengelolaan materi, pengelolaan sub materi, laporan pembelajaran, dan *monitoring container*). Kriteria keberhasilan pengujian ditetapkan sebelum pengujian dilakukan: suatu skenario dinyatakan "Sesuai" apabila keluaran aktual sistem identik dengan keluaran yang diharapkan pada rancangan skenario, dan dinyatakan "Tidak Sesuai" apabila terjadi penyimpangan. Pengujian pada tahap ini dilakukan dengan melibatkan 7 mahasiswa dan 1 dosen Program Studi Teknologi Informasi di lingkungan Politeknik Negeri Madiun sebagai penguji pada kedua peran pengguna. Pengujian dengan melibatkan jumlah mahasiswa yang lebih besar belum dilakukan dan menjadi salah satu keterbatasan penelitian yang dibahas pada bagian Pembahasan.

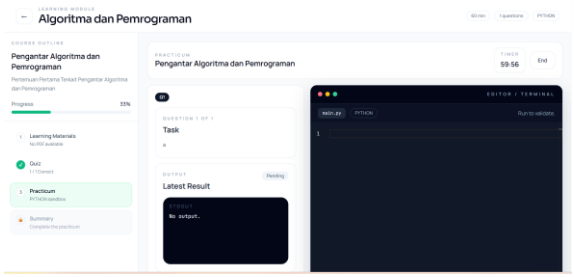


Gambar 5. Arsitektur Sistem

3. Hasil dan Pembahasan

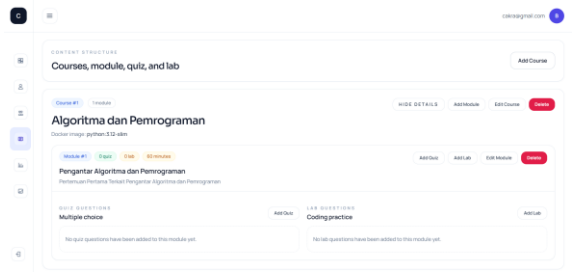
3.1 Hasil

Hasil dari penelitian ini adalah sebuah platform pembelajaran praktikum interaktif berbasis *web* yang terintegrasi dengan Docker sebagai penyedia *runtime environment*. Platform ini digunakan untuk mendukung kegiatan praktikum pemrograman Python dan basis data. Dengan adanya sistem ini, mahasiswa tidak perlu melakukan instalasi bahasa pemrograman, database, atau *dependency* secara manual pada perangkat masing-masing karena proses eksekusi dijalankan melalui *container* yang disediakan oleh server. Pada sisi mahasiswa, platform menyediakan fitur registrasi, login, akses materi praktikum, pengerjaan soal praktikum, editor kode berbasis *web*, serta pengecekan jawaban secara langsung. Mahasiswa dapat memilih materi yang tersedia, membaca instruksi praktikum, menuliskan jawaban, menjalankan kode atau *query*, kemudian melihat hasil eksekusi yang diberikan oleh sistem. Proses ini membantu mahasiswa melakukan praktikum secara lebih mandiri karena hasil pengerjaan dapat langsung diketahui tanpa menunggu pengecekan manual.



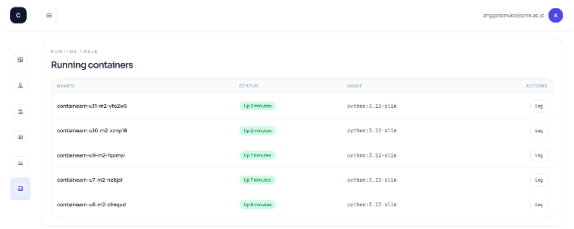
Gambar 6. Tampilan Halaman Mahasiswa Pengerjaan Konten Pembelajaran

Pada sisi dosen, platform menyediakan fitur untuk mengelola materi, mengelola sub materi atau soal praktikum, melihat data mahasiswa, memantau progres pembelajaran, serta melihat laporan hasil praktikum. Fitur laporan membantu dosen mengetahui mahasiswa yang sudah atau belum menyelesaikan praktikum. Selain itu, sistem juga menampilkan detail pengerjaan sehingga dosen dapat memantau hasil praktikum dengan lebih mudah dan terpusat.



Gambar 7. Tampilan Halaman Dosen Manajemen Konten

Selain memantau progres pembelajaran, dosen juga dapat memantau status *container* Docker yang sedang berjalan pada sistem, sebagaimana ditampilkan pada Gambar 6.



Gambar 8. Tampilan Halaman Dosen *Monitoring Container*

Verifikasi fungsional terhadap seluruh fitur di atas dilakukan menggunakan metode black box testing. Rekapitulasi hasil pengujian disajikan pada Tabel 1, sedangkan contoh rincian skenario, input, dan output pengujian disajikan pada Tabel 2.

E-ISSN: 2580-1643

Tabel 1. Rekapitulasi Hasil Black Box Testing per Fitur (63 skenario, 100% Sesuai)

No	Fitur	Peran	Status
1	Login	Mahasiswa/Dosen	Sesuai
2	Register	Mahasiswa	Sesuai
3	Logout	Mahasiswa/Dosen	Sesuai
4	Profil	Mahasiswa	Sesuai
5	Materi/Konten Praktikum	Mahasiswa	Sesuai
6	Pengerjaan Praktikum	Mahasiswa	Sesuai
7	Laporan Praktikum	Mahasiswa	Sesuai
8	Profil	Dosen	Sesuai
9	Manajemen Pengguna	Dosen	Sesuai
10	Pengelolaan Konten/Materi	Dosen	Sesuai
11	Pengelolaan Sub Materi	Dosen	Sesuai
12	Laporan (Report)	Dosen	Sesuai
13	Monitoring *Container*	Dosen	Sesuai

Tabel 2. Contoh Rincian Skenario Black Box Testing (Fitur Login dan Register)

Skenario	Input	Output Diharapkan	Output Aktual	Status
Login tanpa mengisi field wajib	Email dan password dikosongkan	Menolak akses, notifikasi field wajib diisi	Sistem menolak akses dan menampilkan notifikasi	Sesuai
Login dengan akun tidak terdaftar	Email/password tidak sesuai data	Menolak akses, pesan kesalahan login	Sistem menolak dan menampilkan pesan sesuai	Sesuai
Login dengan akun valid	Email dan password benar	Berhasil login, diarahkan ke dashboard	Sistem berhasil login sesuai role pengguna	Sesuai
Register dengan email terdaftar	Email yang sudah dipakai akun lain	Menolak, notifikasi email sudah terpakai	Sistem menolak dan menampilkan notifikasi	Sesuai
Register dengan data valid	Seluruh field diisi data valid	Berhasil daftar, diarahkan ke halaman Login	Sistem berhasil mendaftar dan mengarahkan	Sesuai

3.2 Pembahasan

Hasil pengujian menunjukkan bahwa integrasi *Docker* sebagai *containerized runtime environment* berhasil menyediakan lingkungan eksekusi yang seragam bagi mahasiswa tanpa memerlukan instalasi perangkat lunak apa pun di perangkat masing-masing. Temuan ini sejalan dengan (Butarbutar dkk., 2024) yang menunjukkan efisiensi *container* Python pada lingkungan Fedora Linux, namun penelitian ini memperluas penerapannya ke konteks pembelajaran interaktif berbasis *web* yang belum dieksplorasi oleh studi tersebut. Perbedaan mendasarnya terletak pada kebutuhan manajemen *container* secara dinamis per sesi pengguna untuk mendukung pengecekan jawaban secara langsung (*real-time answer checking*), sebuah tantangan teknis yang tidak muncul pada skenario *deployment* statis seperti pada Butarbutar dkk., 2024), (Isron & Putra, 2021), maupun (Asmara dkk., 2024). Dibandingkan dengan (Jinan dkk., 2025) yang mengembangkan sistem *e-learning* berbasis Laravel untuk pengelolaan materi secara umum, dan (Prasetya & Shofiya Syidada, 2025) yang mengintegrasikan *serious games* untuk meningkatkan keterlibatan belajar, platform pada penelitian ini secara khusus menyediakan lingkungan eksekusi kode yang terisolasi sehingga mendukung praktikum pemrograman secara langsung tanpa konfigurasi tambahan di sisi pengguna. Kedua studi pembanding tersebut berfokus pada penyampaian materi dan keterlibatan belajar, sedangkan penelitian ini menambahkan lapisan eksekusi kode yang seragam sebagai kontribusi tambahan pada domain platform pembelajaran pemrograman. Dari sisi efisiensi sumber daya, hasil ini juga memperkuat temuan (Afrizal & Prihanto, 2022) bahwa *container* memerlukan sumber daya lebih ringan dibandingkan virtualisasi berbasis VM, karena seluruh sesi eksekusi kode mahasiswa pada penelitian ini berhasil dijalankan secara bersamaan pada satu *dedicated server* tanpa kendala performa yang teramati selama pengujian, meskipun pengukuran performa secara kuantitatif belum dilakukan secara sistematis. Penelitian ini memiliki beberapa keterbatasan. Pertama, pengujian *black box* dilakukan dengan melibatkan 7 mahasiswa dan 1 dosen Program Studi Teknologi Informasi, Politeknik Negeri Madiun, sehingga belum mencerminkan kondisi penggunaan pada skala jumlah mahasiswa yang lebih besar sebagai responden pengujian penerimaan pengguna (*user*

E-ISSN: 2580-1643

acceptance testing). Kedua, pengujian performa *container* di bawah beban pengguna simultan dalam jumlah besar belum dilakukan, sehingga ketahanan sistem pada kondisi praktikum serentak dengan jumlah mahasiswa yang banyak belum dapat dipastikan. Ketiga, dukungan bahasa pemrograman pada platform saat ini masih terbatas pada Python dan basis data, belum mencakup bahasa pemrograman lain yang umum diajarkan pada praktikum algoritma dan pemrograman.

4. Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, platform pembelajaran praktikum interaktif berbasis Docker berhasil menyediakan lingkungan kerja praktikum yang seragam, terisolasi, dan mudah diakses bagi mahasiswa, tanpa memerlukan instalasi perangkat lunak secara manual pada perangkat masing-masing. Hasil pengujian menggunakan *black box testing* menunjukkan bahwa seluruh 13 fitur utama sistem, yang diverifikasi melalui 63 skenario pengujian, berjalan sesuai dengan kriteria keberhasilan yang telah ditetapkan (100% "Sesuai"). Implementasi *containerized runtime* berbasis *Docker* pada platform pembelajaran praktikum interaktif dapat menjadi solusi yang sesuai untuk mendukung pembelajaran praktikum pemrograman dan basis data secara konsisten, sekaligus mempermudah dosen dalam mengelola materi dan memantau hasil pengerjaan mahasiswa. Penelitian selanjutnya disarankan untuk melakukan pengujian penerimaan pengguna dengan jumlah mahasiswa dan dosen yang lebih besar, mengevaluasi performa *container* di bawah beban pengguna simultan dalam skala besar, dan memperluas dukungan bahasa pemrograman selain Python untuk memperluas cakupan penggunaan platform pada berbagai mata kuliah praktikum.

5. Daftar Pustaka

Adisty Yudianto Putri, D., Cahyo Wibowo, N., & Karunia Farista Ananto, P. (2025). Implementasi pengujian blackbox menggunakan automation testing tools pada website monitoring balita. *JATI (Jurnal Mahasiswa Teknik Informatika), 9*(3), 4278–4285. https://doi.org/10.36040/jati.v9i3.13627.

Afrizal, H., & Prihanto, A. (2022). Analisis kebutuhan resource dan independensi antara teknologi single server, virtualisasi dan container. *Journal of Informatics and Computer Science, 04*. https://doi.org/10.26740/jinacs.v4n01.p26-33.

Alqaisi, O. I., Saman Tosun, A., & Korkmaz, T. (2024). Performance analysis of container technologies for computer vision applications on edge devices. *IEEE Access, 12*, 41852–41869. https://doi.org/10.1109/ACCESS.2024.3376570.

Asmara, R., Fajri, A. F., Novinaldi, N., Iswandy, E., & Nanda, A. P. (2024). Membangun pengelolaan aplikasi web berbasis Docker pada Kominfo Padang Panjang. *Jurnal Pustaka Data (Pusat Akses Kajian Database, Analisa Teknologi, dan Arsitektur Komputer), 4*(1), 17–21. https://doi.org/10.55382/jurnalpustakadata.v4i1.666.

Butarbutar, L. E., Davina, S., & Tambunan, V. F. (2024). Implementasi Docker untuk cloud computing pada Fedora Linux: Studi kasus Python dalam kontainer. Dalam *Jurnal Studi Multidisipliner* (Vol. 8, Nomor 11).

Cahyanto, I. (2023). Pemanfaatan platform e-learning berbasis cloud computing untuk meningkatkan efektivitas pembelajaran jarak jauh. *Edum Journal, 6*(2), 18–31. https://doi.org/10.31943/edumjournal.v6i2.144.

Isron, M. H., & Putra, R. E. (2021). Implementasi virtual server berbasis container pada sistem informasi geografis Cagar Budaya Mojokerto. *Journal of Informatics and Computer Science, 03*. https://doi.org/10.26740/jinacs.v3n02.p143-154.

Jinan, A., Siregar, M. P., Suryani, D. F., Rolanda, V., & Muis, A. (2025). Desain dan rancang bangun sistem e-learning menggunakan framework Laravel berbasis web. *ROUTERS: Jurnal Sistem dan Teknologi Informasi*, 85–93. https://doi.org/10.25181/rt.v3i2.4182.

Megantara, R. A., Alzami, F., Pramunendar, R. A., & Prabowo, D. P. (2022). Pengembangan dan implementasi Docker untuk memaksimalkan utilitas server universitas pada masa Covid-19. *Transmisi, 24*(2), 48–54. https://doi.org/10.14710/transmisi.24.2.48-54.

E-ISSN: 2580-1643

Prasetya, N. I., & Shofiya Syidada. (2025). Integrasi serious games dalam pengembangan platform pembelajaran pemrograman. *Decode: Jurnal Pendidikan Teknologi Informasi, 5*(1), 12–26. https://doi.org/10.51454/decode.v5i1.864

Pratama, I. P. A. E., & Raharja, I. M. S. (2023). Node.js performance benchmarking and analysis at Virtualbox, Docker, and Podman environment using Node-Bench method. *JOIV: International Journal on Informatics Visualization, 7*(4), 2240. https://doi.org/10.62527/joiv.7.4.1762.

Rinduh Iriane, G., & Soter Japi, K. (2025). Penerapan metode RAD pada perancangan aplikasi pegawai berbasis mobile web pada Dinas Kelautan dan Perikanan Provinsi NTT. *JATI (Jurnal Mahasiswa Teknik Informatika), 9*(4), 6131–6137. https://doi.org/10.36040/jati.v9i4.13999.

Stevanus Ocktoberrikho, & Noviyanti P. (2025). Pengembangan platform e-learning berbasis Android untuk siswa sekolah dasar. *Instink: Inovasi Pendidikan, Teknologi Informasi dan Komputer, 4*(1), 1–11. https://doi.org/10.30599/8dkacm46.

Sukendar, A. (2023). Perancangan sistem penilaian otomatis program Java berbasis web pada perkuliahan pemrograman. *Jurnal Device, 13*, 216–222. https://doi.org/10.32699/device.v13i2.5841.

Sutanto, S., Gunawan, W., & Faeshal, F. (2021). Arsitektur container Docker pada aplikasi Expert Assist dengan teknologi Node.js, Express Framework, dan cloud database NoSQL MongoDB Atlas. *Jurnal Sistem Informasi dan Informatika (Simika), 4*(1), 73–89. https://doi.org/10.47080/simika.v4i1.1189.

Zulfahrizan, A., Raffi Akbar Tanjung, M., Zidane Al-Kautsar, M., & Kiswanto, D. (2024). Analisis perbandingan performa MySQL pada lingkungan virtualisasi pada Linux Debian dan Ubuntu berbasis web server Nginx. *JATI (Jurnal Mahasiswa Teknik Informatika), 9*(1), 1107–1112. https://doi.org/10.36040/jati.v9i1.12545.