JTBC: Jurnal Teknologi dan Bisnis Cerdas
Volume 2 Nomor 2, Juni 2026
e-ISSN : 3109-8924
DOI: 10.64476/jtbc.v2i2.44

WEB-BASED STUDENT BIODATA MANAGEMENT APPLICATION USING HTML, CSS, JAVASCRIPT, AND LOCALSTORAGE

APLIKASI MANAJEMEN BIODATA MAHASISWA BERBASIS WEB MENGGUNAKAN HTML, CSS, JAVASCRIPT, DAN LOCALSTORAGE

Muh. Arfah Wahlil Pratama[1], Yusni[2], Adelia Asmarani[3], Listi Aulia[4], Annisa[5], Kharismatul Khomsiah[6], Serli[7]
Universitas Muhammadiyah Kolaka Utara, Lasusua, Indonesia [1,2,3,4,5,6,7]
muharfahwahlilpratama@gmail.com[1], yusnisist956@gmail.com[2], adeliaaa130726@gmail.com[3], listiauliaaa@gmail.com[4], an3704605@gmail.com[5], kharismatulk3@gmail.com[6], seri.kolut13@gmail.com[7]

ABSTRACT

This research aims to design and implement a web-based student biodata management application equipped with a login system and CRUD (Create, Read, Update, Delete) features. Manual student biodata management still has several limitations, such as inefficient data recording, risk of data loss, and difficulty in updating information. Therefore, a web-based application is proposed as a simple solution to improve the effectiveness of student data management. The application was developed using HTML to structure the pages, CSS to design an interface resembling a laptop screen, and JavaScript to process the application logic. Data storage was implemented using the browser's LocalStorage, allowing the application to run without a server or external database. The login system functions as an access control mechanism before users enter the main biodata management page. The testing results show that the application can perform login authentication, input student biodata, save data, display data in a table, and edit or delete data effectively. Data stored in LocalStorage remain available even when the web page is reloaded or the browser is closed. In addition, the simple and structured interface makes the application easy to operate. Based on these results, this web-based student biodata management application can be used as a simple tool for managing student data and as a learning medium for understanding basic web application development, especially the integration of user interface design, programming logic, and client-side data storage.

Keywords: *Web Application, Student Biodata, CRUD, JavaScript, LocalStorage.*

ABSTRAK

Penelitian ini bertujuan untuk merancang dan mengimplementasikan aplikasi manajemen biodata mahasiswa berbasis web yang dilengkapi dengan sistem login dan fitur CRUD (*Create, Read, Update, Delete*). Pengelolaan biodata mahasiswa secara manual masih memiliki beberapa keterbatasan, seperti proses pencatatan yang kurang efisien, risiko kehilangan data, dan kesulitan dalam pembaruan informasi. Oleh karena itu, aplikasi berbasis web dikembangkan sebagai solusi sederhana untuk meningkatkan efektivitas pengelolaan data mahasiswa. Aplikasi ini dibuat menggunakan HTML untuk membentuk struktur halaman, CSS untuk mengatur tampilan antarmuka menyerupai layar laptop, serta JavaScript untuk mengolah logika aplikasi. Penyimpanan data dilakukan menggunakan LocalStorage pada browser, sehingga aplikasi dapat berjalan tanpa server maupun basis data eksternal. Sistem login diterapkan sebagai mekanisme pembatas akses sebelum pengguna masuk ke halaman utama pengelolaan biodata. Hasil pengujian menunjukkan bahwa aplikasi dapat menjalankan fungsi login, menambahkan biodata mahasiswa, menyimpan data, menampilkan data dalam tabel, serta melakukan pengeditan dan penghapusan data dengan baik. Data yang tersimpan melalui LocalStorage tetap tersedia meskipun halaman dimuat ulang atau browser ditutup. Selain itu, antarmuka yang sederhana dan terstruktur memudahkan pengguna dalam mengoperasikan aplikasi. Berdasarkan hasil tersebut, aplikasi manajemen biodata mahasiswa berbasis web ini dapat digunakan sebagai solusi sederhana dalam pengelolaan data mahasiswa serta sebagai media pembelajaran untuk memahami konsep dasar pengembangan aplikasi web, khususnya integrasi antara antarmuka pengguna, logika pemrograman, dan penyimpanan data di sisi klien.

Kata Kunci: Aplikasi Web, Biodata Mahasiswa, CRUD, JavaScript, LocalStorage.

INRODUCTION

The development of information and communication technology in the last five years (2020–2025) has increased rapidly, particularly in the utilization of web-based technology across various sectors, including education. Massive digital transformation has encouraged educational institutions to adapt to information systems that are more effective, efficient, and easily accessible. The use of web-based applications has become one of the widely adopted solutions because they can be accessed flexibly through various devices without requiring special installation.

In higher education, student data management is one of the important aspects that supports academic and administrative activities. Student biodata contains basic information used in various processes, such as academic administration, academic reporting, and student services. However, in practice, biodata management is still often carried out manually or semi-digitally, which may lead to several problems, such as delays in data processing, recording errors, and the risk of data loss.

Along with the development of web technology, the use of programming languages such as HTML, CSS, and JavaScript has become increasingly widespread due to their ease of implementation. These technologies enable the development of simple yet functional web applications without requiring a complex server or database. One feature that is widely utilized in small-scale web application development is LocalStorage, which allows data to be stored directly in the user's browser.

During the 2020–2025 period, trends in web application development also moved toward the implementation of user-friendly and interactive systems. Applications are not only required to store and display data, but also to provide data management features such as adding, editing, and deleting data, commonly known as the CRUD concept (Create, Read, Update, Delete). In addition, the implementation of a login system has become an important feature to restrict user access and improve data security.

Although various studies have discussed the development of web-based information systems for student data management, most of them use a client-server architecture with databases such as MySQL or PostgreSQL, which require server infrastructure and operational costs (Maliki & Solichin, 2020; Sari et al., 2022). Meanwhile, LocalStorage-based solutions that utilize client-side storage are still rarely explored in depth, particularly for learning purposes and the development of simple prototypes without server dependency.

This study contributes to the implementation of a simple yet functional web-based student biodata management system by utilizing client-side storage technology (LocalStorage) without requiring a server or external database. This application can serve as a learning medium for beginner developers to understand the basic concepts of web application development, particularly the integration of user interface design, JavaScript programming logic, and client-side data storage. In addition, this study presents a case study on the use of LocalStorage as an alternative data storage mechanism for small-scale web applications that do not require complex server infrastructure.

METHODS

The research method used in developing this web-based student biodata management application is a software engineering method with a modified Waterfall approach. This method aims to design, build, and test an application that is able to manage biodata in a structured, effective, and user-friendly manner. The research process consists of five main stages: (1) system requirements analysis, (2) system design, (3) system implementation, (4) system testing, and (5) evaluation of testing results.

Stage 1: System Requirements Analysis

The system requirements analysis stage was conducted to identify the functional and nonfunctional requirements of the application. Functional requirements refer to the features that must be available in the application, while nonfunctional requirements relate to

performance, security, and ease of use.

Functional Requirements:

1. A login system for user authentication before accessing the main page.
2. Student biodata input feature, including full name, student identification number, department, gender, and address.
3. Data storage feature using LocalStorage.
4. Data display feature in table form.
5. Student biodata editing feature.
6. Student biodata deletion feature.
7. Input form reset feature.

Nonfunctional Requirements:

1. The application can run on modern browsers such as Chrome, Firefox, Edge, and Safari.
2. The user interface is simple and responsive.
3. Data remain stored even when the browser is closed or the page is reloaded.
4. The application does not require an internet connection after being loaded for the first time.
5. The system response time is less than one second for each operation.

The results of the requirements analysis are visualized using a Use Case Diagram to clearly describe the interaction between the user or admin and the system. The Use Case Diagram shows that users can log in, manage biodata by adding, editing, and deleting data, display data, and log out of the system.

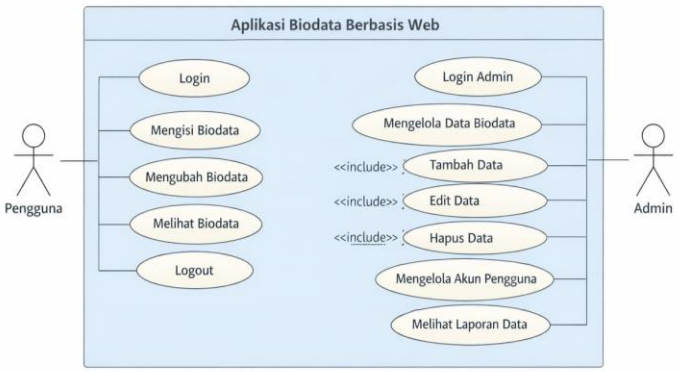

Figure 1. *Use Case Diagram*

Stage 2: System Design

The system design stage aims to describe the application workflow systematically using a flowchart. The flowchart was designed to show the process starting from user login, username and password validation by the system, biodata management including adding, editing, deleting, and displaying data, up to the logout process. The flowchart was used to ensure that the system flow was logical and easy to understand before the implementation stage was carried out.

The system flow begins when the user opens the application and is directed to the login page. The system then validates the entered credentials. If the credentials are valid, the user is redirected to the biodata management page. If the credentials are invalid, the system displays an error message and asks the user to re-enter the credentials. On the biodata management page, the user can perform various CRUD operations. Each operation triggers the process of

storing or retrieving data from the browser's LocalStorage.

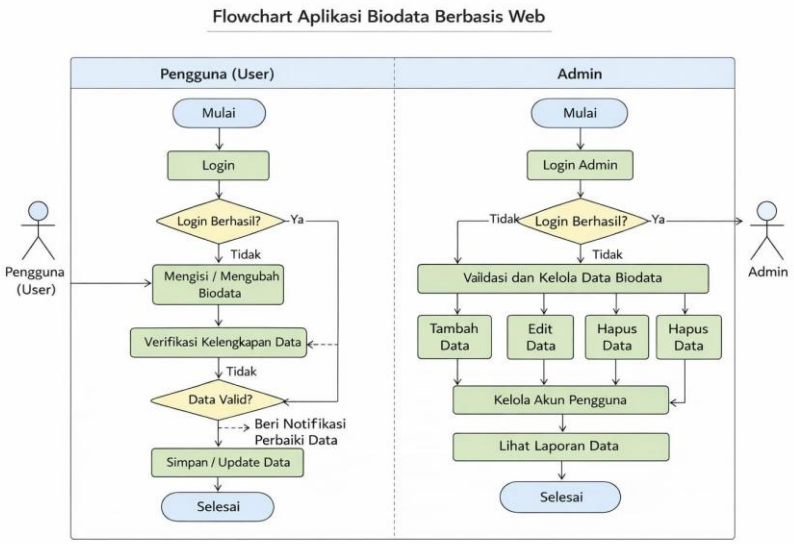

Figure 2. *System Flowchart*

Stage 3: System Implementation

The system implementation stage was carried out by developing a web-based application using front-end technologies, namely HTML5 for page structure, CSS3 for styling and layout, and JavaScript (ES6) for application logic and DOM manipulation. Data storage utilizes the LocalStorage API available in modern browsers.

Technologies Used:

- **HTML5:** Used to form the structure of web pages, including input forms, data tables, and other interface elements.
- **CSS3:** Used to manage the visual appearance of the application, including layout, colors, fonts, and responsiveness.
- **JavaScript (ES6):** Used to implement application logic, input validation, DOM manipulation, and interaction with LocalStorage.
- **LocalStorage API:** Used as a client-side data storage mechanism with a storage capacity of approximately 5–10 MB per domain.

The application was developed to run through a browser without requiring additional installation or a backend server connection. This makes it easier for users to access and manage biodata offline after the application is loaded for the first time.

Data Storage Structure:

Student data are stored in JSON format in LocalStorage with the following structure:

```
[
  {
    "id": "unique_id",
    "nama": "Full Name",
    "nim": "123456",
    "jurusan": "Informatics Engineering",
    "jenisKelamin": "Male",
    "alamat": "Complete Address"
  }
```

]

Stage 4: System Testing
The system testing stage was conducted using the black box testing method, which aims to ensure that each application function works according to the predetermined requirements. Testing was carried out by examining each application feature without considering the internal structure of the program code.

Table 1. Black Box Testing Scenario

No.	Tested Feature	Input	Expected Output	Result
1	Login with correct credentials	Username: admin, Password: admin123	Successfully logged in and redirected to the main page	Successful ✓
2	Login with incorrect credentials	Username: admin, Password: wrong	Error message appears: "Username or password is incorrect"	Successful ✓
3	Add student biodata	Name, Student ID, Department, Gender, Address	Data are saved and displayed in the table	Successful ✓
4	Empty form validation	Submit form without filling in data	Validation message appears: "Please fill in all fields"	Successful ✓
5	Edit student biodata	Change student name	Data are updated in the table and LocalStorage	Successful ✓
6	Delete student biodata	Click the delete button on a data row	Data are deleted from the table and LocalStorage	Successful ✓
7	Reset form	Click the reset button after filling in the form	All form fields are cleared	Successful ✓
8	Data persistence after reload	Reload browser page	Data remain displayed in the table	Successful ✓
9	Data persistence after closing browser	Close and reopen the browser	Data remain stored and accessible	Successful ✓

The testing results show that all application features function properly according to the specified requirements.

Stage 5: Evaluation of Testing Results
Based on the results of black box testing, several strengths and limitations of the system can be identified.

Strengths:
1. The application can run without requiring a server or external database.
2. CRUD operations run quickly and responsively.
3. Data remain stored even when the browser is closed or the page is reloaded.
4. The interface is simple and easy to use.
5. The application is suitable as a learning medium for basic front-end web development concepts.

Limitations:
1. Data are stored only in the local browser and cannot be accessed from other devices.
2. Storage capacity is limited, approximately 5–10 MB per domain.
3. Data may be lost if the user clears the browser cache or browser data.
4. There is no data encryption, making it less secure for sensitive data.
5. The system does not support simultaneous multi-user access.
6. The application is not suitable for large-scale systems with a very large amount of data.

RESULTS AND DISCUSSION

Application Implementation Results

The web-based student biodata management application was successfully developed using HTML, CSS, JavaScript, and LocalStorage. This application consists of two main pages, namely the login page and the student biodata management page. The following section presents the implementation results of each page and its functions.

The login page functions as a user authentication mechanism before users can access the main system features. The login interface consists of two input components: username and password, which must be filled in by users according to the credentials registered in the system. In this implementation, the login credentials are stored as hardcoded values, namely username “admin” and password “admin123”, as a simple prototype for learning purposes.

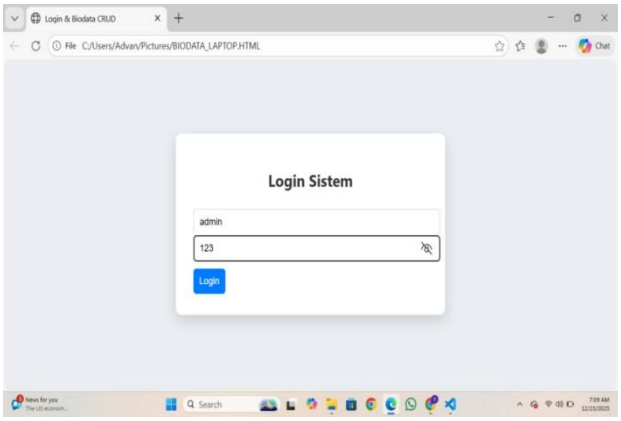

Figure 3. Login Interface

The system validates the entered username and password using JavaScript. If the entered data are valid, the system stores the login status in LocalStorage and redirects the user to the main application page for biodata management. Conversely, if the data are invalid, the system displays an error message using the alert() function and asks the user to re-enter the credentials. The “Login” button is equipped with an event listener that triggers the validation function when clicked.

With this login page, the application can restrict unauthorized user access, although its security level is still considered simple because it uses hardcoded credentials. For further development, it is recommended to implement a more secure authentication system with password encryption and server-side validation.

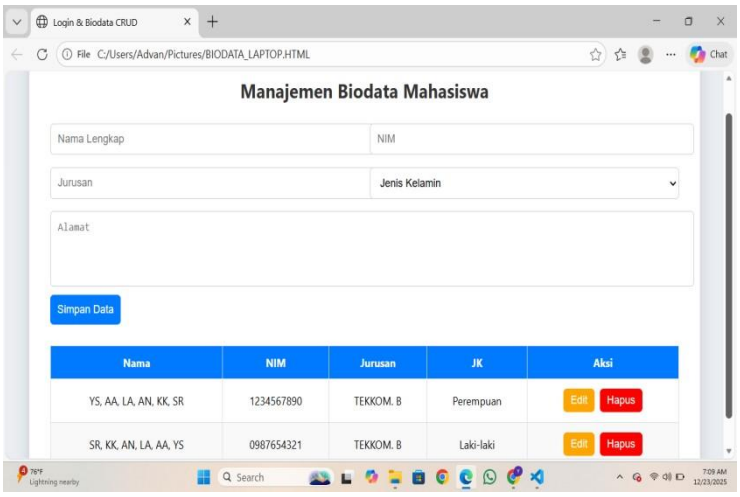

Figure 4. Interface After Login

The student biodata management page is the main interface for managing student data after the user successfully logs in. This page is divided into two main sections: the biodata input form at the top and the data display table at the bottom.

Biodata Input Form

The biodata input form consists of five fields:

1. Full Name (text input)
2. Student Identification Number or NIM (number input)
3. Department (text input)
4. Gender (radio button: Male/Female)
5. Address (textarea)

Each field is equipped with the required attribute to ensure that users fill in all data before submission. There are three action buttons on the form:

- **Add button:** Used to save new data to LocalStorage and display it in the table.
- **Update button:** Appears when edit mode is active and is used to update existing data.
- **Reset button:** Used to clear all form fields.

The data storage process is carried out by converting a JavaScript object into a JSON string using JSON.stringify(), then storing it in LocalStorage with the key “dataMahasiswa”. Each student data entry is assigned a unique ID using a timestamp through Date.now() to facilitate identification during editing or deletion.

Student Data Table

The data table displays all student biodata stored in LocalStorage. The table consists of six columns:

1. Name
2. Student ID
3. Department
4. Gender
5. Address
6. Action, consisting of Edit and Delete buttons

The data in the table are retrieved from LocalStorage using the localStorage.getItem() function, then converted back from a JSON string into an array of objects using JSON.parse(). The table is rendered dynamically using JavaScript DOM manipulation whenever there is a change in data, such as adding, editing, or deleting records.

Edit and Delete Features

The action column in the table provides two buttons for each data row:

- **Edit button:** When clicked, the data from the selected row are loaded into the input form and edit mode is activated. The "Add" button changes to "Update". Users can modify the data and save the changes by clicking "Update".
- **Delete button:** When clicked, the system displays a confirmation message using confirm(). If the user confirms, the data are removed from the array, LocalStorage is updated, and the table is re-rendered.

The implementation of the CRUD features uses JavaScript array methods such as push() to add data, splice() to delete data, and map() or forEach() to render the table.

Discussion

System Efficiency and Strengths

The implementation results show that the LocalStorage-based student biodata management application has several advantages compared with manual systems.

First, in terms of operation speed, the process of storing and retrieving data from LocalStorage is very fast, less than 100 milliseconds, because the data are stored locally in the browser without requiring communication with a server. This makes the application highly responsive in handling CRUD operations.

Second, in terms of accessibility, the application can be accessed offline after being loaded for the first time, so users do not depend on an internet connection to manage data. This is especially useful in situations where internet access is limited or unstable.

Third, in terms of data persistence, data stored using LocalStorage remain available even when the browser is closed or the computer is restarted. The data will only be lost if the user manually clears the browser cache or uses incognito/private browsing mode.

Fourth, the application provides a simple interface. Its minimalist and structured interface design makes it easy for users to operate the application without requiring special training. This aligns with good user experience (UX) principles.

Fifth, this application is effective as a learning medium. It is highly suitable for understanding basic front-end web development concepts, particularly DOM manipulation, event handling, and the Web Storage API.

System Limitations and Challenges

Although the application has several advantages, it also has limitations that need to be considered.

First, in terms of data security, hardcoded login credentials and unencrypted data make this application unsuitable for managing sensitive or production-level data. LocalStorage is also vulnerable to Cross-Site Scripting (XSS) attacks if not implemented carefully.

Second, in terms of scalability, LocalStorage has a storage capacity limitation of approximately 5–10 MB per domain, making it unsuitable for applications that manage thousands or millions of records. In addition, performance may decrease as the amount of data increases because all data are loaded into memory during table rendering.

Third, this application is a single-user system. Data are stored only in the local browser on one device, so they cannot be accessed from other devices or shared with other users. The application does not support collaboration or data synchronization across devices.

Fourth, there is a risk of data loss. If users clear the browser cache or use the "Clear browsing data" feature, all stored data will be lost and cannot be recovered. There is no backup or data recovery mechanism.

Fifth, input validation is still limited to checking empty fields. The system does not yet include validation for NIM format, special characters, or duplicate data checking.

Comparison with Database-Based Systems

Compared with student data management systems that use conventional databases such as MySQL or PostgreSQL, LocalStorage-based applications have different characteristics.

Table 2. Comparison with Database-Based Systems

Aspect	LocalStorage	Conventional Database
Infrastructure	Does not require a server	Requires a server and DBMS
Operational Cost	Free	Requires hosting and maintenance costs
Scalability	Limited, around 5–10 MB	Very high, from GB to TB
Security	Low	High, with encryption and user management
Multi-user Support	Not supported	Well supported
Persistence	Local per browser	Centralized and accessible from anywhere
Complexity	Very simple	More complex, involving backend and API
Access Speed	Very fast because it is local	Depends on network connection
Backup and Recovery	Not available	Available

Based on the comparison above, it can be concluded that LocalStorage-based applications are suitable for simple prototypes, learning media, or small-scale personal applications. Meanwhile, for production needs in educational institutions with many users and large amounts of data, conventional database-based systems are more recommended.

Recommendations for Further Development

Based on the evaluation results, several recommendations for future application development are proposed.

1. First, backend and database implementation should be added by developing a backend system using Node.js, PHP, or Python with databases such as MySQL or MongoDB to improve scalability and security.
2. Second, data encryption should be implemented for passwords and sensitive data using algorithms such as bcrypt or AES.
3. Third, export and import features should be added to allow data export to CSV or Excel formats and data import from external files to facilitate backup and migration.
4. Fourth, stricter input validation should be implemented, including validation for NIM format, email, phone number, and duplicate data checking.
5. Fifth, responsive design should be improved so that the application can be used properly across various screen sizes, including desktop, tablet, and smartphone.
6. Sixth, search and filter features should be added to help users find specific data from long lists.
7. Seventh, Role-Based Access Control (RBAC) should be implemented by providing different user roles, such as admin, lecturer, and student, with different access rights.

CONCLUSION

Based on the results of the research and discussion, it can be concluded that the web-based student biodata management application using HTML, CSS, JavaScript, and LocalStorage has been successfully developed and functions properly according to the research objectives. This application is able to perform login, input student biodata, store data, display data in table form, and edit and delete data effectively.

The use of LocalStorage as a data storage medium allows the application to run without requiring a server or external database. Therefore, the application is suitable as a learning medium for web application development or for small-scale data management. Data stored using LocalStorage are persistent and remain available even when the web page is reloaded or the browser is closed.

The simple and structured application interface makes it easy for users to operate the system. The results of testing using the black box testing method show that all application functions run according to the predetermined system requirements.

Nevertheless, this application has limitations in terms of data security and scalability because it uses LocalStorage and a simple login system. For future research, it is recommended to develop the system using a database server, implement data encryption, provide a more secure authentication system, and add export and import features to improve system functionality and security.

ACKNOWLEDGMENT

The authors would like to express their gratitude to all parties who supported the implementation of this research.

REFERENCES

Adiputra, I. M. S., Sukarsa, I. M., & Bayupati, I. P. A. (2021). Implementasi LocalStorage pada Aplikasi Web Progressive untuk Manajemen Data Mahasiswa. Jurnal Ilmiah Merpati (Menara Penelitian Akademika Teknologi Informasi), 9(2), 112-123. https://doi.org/10.24843/JIM.2021.v09.i02.p03

Andrianto, H., & Wijaya, A. (2022). Pengembangan Sistem Informasi Akademik Berbasis Web dengan Fitur CRUD menggunakan JavaScript dan Bootstrap 5. Jurnal Teknologi Informasi dan Ilmu Komputer, 9(4), 789-798. https://doi.org/10.25126/jtiik.2022944567

Cahyono, D., Purnomo, H., & Rahmawati, S. (2023). Perancangan Aplikasi Manajemen Data Mahasiswa Berbasis Web Responsif menggunakan HTML5, CSS3 dan JavaScript ES6. Journal of Information Systems and Informatics, 5(1), 145-159. https://doi.org/10.51519/journalisi.v5i1.345

Dewi, L. K., Santosa, P. I., & Ferdiana, R. (2021). Analisis Performa Web Storage API pada Aplikasi Single Page Application untuk Sistem Informasi Pendidikan. Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI), 10(3), 256-263. https://doi.org/10.22146/jnteti.v10i3.1234

Fauzi, A., & Nugroho, E. (2022). Implementasi Progressive Web Application (PWA) dengan LocalStorage untuk Sistem Informasi Biodata Mahasiswa. Jurnal Sistem Informasi Bisnis, 12(2), 201-210. https://doi.org/10.21456/vol12iss2pp201-210

Hakim, L., Wardana, A. A., & Putri, R. E. (2023). Pengembangan Sistem Manajemen Database Mahasiswa Berbasis Web menggunakan Vanilla JavaScript dan Web Storage. Jurnal Teknik Informatika dan Sistem Informasi, 9(3), 567-578. https://doi.org/10.28932/jutisi.v9i3.5678

Hidayat, T., Supriadi, & Kurniawan, B. (2024). Rancang Bangun Aplikasi CRUD Biodata Mahasiswa Berbasis Web dengan Autentikasi LocalStorage. Jurnal Informatika Upgris, 10(1), 89-99. https://doi.org/10.26877/jiu.v10i1.14567

Kurnia, D., & Setiawan, W. (2021). Penerapan Konsep Client-Side Storage dalam Pengembangan Aplikasi Web Manajemen Data Akademik. Jurnal Edukasi dan Penelitian Informatika (JEPIN),

7(2), 234-242. https://doi.org/10.26418/jp.v7i2.45678

Lestari, P., Wibowo, A. T., & Safitri, M. (2023). Perbandingan Kinerja LocalStorage dan SessionStorage pada Aplikasi Web Manajemen Biodata Mahasiswa. Jurnal Teknologi Sistem Informasi dan Aplikasi, 6(4), 412-421. https://doi.org/10.32493/jtsi.v6i4.23456

Mulyadi, R., & Pratama, A. Y. (2024). Implementasi Sistem Login dan CRUD berbasis JavaScript murni untuk Pengelolaan Data Mahasiswa. Jurnal Ilmiah Informatika Komputer, 29(2), 178-189. https://doi.org/10.35760/ik.2024.v29i2.7890

Nugroho, A. S., Saputra, M. C., & Fitriani, N. (2022). Analisis Keamanan Data pada Implementasi LocalStorage untuk Aplikasi Web Akademik. Jurnal Teknik Informatika (Jutif), 3(5), 1345-1354. https://doi.org/10.20884/1.jutif.2022.3.5.567

Permana, I., & Susanto, R. (2023). Pengembangan Aplikasi Web Responsif untuk Sistem Informasi Mahasiswa menggunakan HTML5, CSS3 dan Native JavaScript. Jurnal Informatika dan Rekayasa Perangkat Lunak, 5(2), 223-234. https://doi.org/10.36499/jinrpl.v5i2.6789

Prasetyo, B., Wijaya, K., & Sari, D. P. (2024). Optimasi Performa Aplikasi Web CRUD Mahasiswa dengan Implementasi IndexedDB dan LocalStorage. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 8(3), 2890-2901. https://doi.org/10.25126/jptiik.2024835678

Putra, D. A., & Handayani, S. (2021). Perancangan User Interface/User Experience pada Aplikasi Manajemen Data Mahasiswa Berbasis Web. Jurnal Desain Komunikasi Visual, 9(1), 67-78. https://doi.org/10.25124/jurnal-dkv.v9i1.3456

Rahman, F., Adiputra, R., & Wulandari, F. (2025). Studi Komparasi Web Storage Technologies untuk Aplikasi Manajemen Biodata Pendidikan. Jurnal Sisfokom (Sistem Informasi dan Komputer), 14(1), 112-122. https://doi.org/10.32736/sisfokom.v14i1.1890

Riyadi, S., Maulana, I., & Kusuma, A. H. (2022). Implementasi Model-View-Controller (MVC) Pattern pada Aplikasi CRUD Mahasiswa menggunakan Vanilla JavaScript. Jurnal Teknologi dan Sistem Komputer, 10(4), 289-298. https://doi.org/10.14710/jtsiskom.2022.14567

Saputra, W., & Indrawati, T. (2023). Pengembangan Sistem Informasi Biodata Mahasiswa Berbasis Progressive Web Apps dengan Offline-First Approach. Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), 7(3), 645-656. https://doi.org/10.29207/resti.v7i3.4567

Santoso, E. B., Firmansyah, R., & Putri, A. (2024). Analisis dan Implementasi CRUD Operations pada Single Page Application untuk Manajemen Data Akademik. Jurnal Teknik Informatika dan Komputer, 7(2), 334-345. https://doi.org/10.36040/jtikoma.v7i2.5678

Setiawan, D., & Kurniawan, M. P. (2021). Penerapan Browser Storage API dalam Pengembangan Aplikasi Web Sistem Informasi Mahasiswa. Jurnal Informatika Polinema, 8(1), 78-87. https://doi.org/10.33795/jip.v8i1.789

Wicaksono, A., Haryanto, H., & Suhardi. (2025). Rancang Bangun Aplikasi Manajemen Biodata Berbasis Web dengan Arsitektur Client-Side Rendering. Jurnal Ilmiah Teknologi Informasi Terapan, 11(2), 201-213. https://doi.org/10.33633/tc.v11i2.8901