International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 05 | May - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Design and Implementation of a Low-Code Training Event Management System Using Mendix

Dr. Lokendra Singh Songare[1], Dhairya Bhatia[2], Disha Gupta[3,] Kartik [4]

[1]Assistant Professor, Department of Computer Science & Engineering, Medi-Caps University, Indore, India

[2,3,4] Students, Department of Computer Science & Engineering, Medi-Caps University, Indore, India

Lokendra.singh@medicaps.ac.in, dhairya172004@gmail.com, disg312@gmail.com,en21cs301364@medicaps.ac.in

Abstract--- These days, organizing training programs has turned into a regular need for many educational institutes and organizations. To make this task less time-consuming and simple, a web-based system was built using the Mendix low-code platform. It helps in managing everything like creating events, registering participants, and making certificates without too much manual work.This system makes use of Mendix microflows for things like checking the data, managing each step of the process, and giving quick responses to users while they're using the app. A useful part of this setup is that before saving anything, it quickly looks at whether all the important information has been filled or not. This way, chances of missing something or making mistakes get reduced. This helps in reducing chances of missing out on necessary data and avoids possible mistakes while working.Along with this, it uses some rules like XPath constraints and a few extra actions to keep everything neat and easy to handle. After testing it properly, the system showed good results, saving both time and effort. In the coming time, there's scope to add more features like alerts and detailed reports to improve its usefulness.

Index Terms — Mendix, low-code, event management, automation, microflows, certificates

I INTRODUCTION

How individuals develop software has evolved significantly over the years. Previously, one needed good coding abilities to develop an application, but today with low-code and no-code environments, it's easier. With these environments, individuals can create working apps with minimal coding, which saves money and time [1], [2]. Such a platform as Mendix in this segment is really well-liked because it offers you simple facilities such as visual modeling and components ready to go. And in case you would like to bring in more logic, you may do so in terms of microflows or Java actions. The system was specially created with the idea of automating workflows and less manual intervention.The system have their prime features with a view of promoting the efficiency and security. Real-time validation captures any error at input levels at real- time to ensure that integrity of the data is obtained at the point of inception. Actions like notifications and updates are all automated and to ensure zero human intervention. Besides, the role-based access control offers even privileged users just to execute selected operations. All these elements not only make the process more effective but also ensure accuracy and security for the data involved in the process.

It also enables useful features like real-time validation, automation of the trivial, and user role- based access control.

One of the most impressive things about the system is that it employs microflows, giving power and flexibility to the way the application does things. Microflows in Mendix are comparable to visual logic components that deal with such activities as conditional actions, database updating, and returning feedback to the user—all without writing any standard code [8], [9]. For instance, there is a microflow in the system that verifies whether important information—such as the training start date, course, and location—are completed before saving the Training Event object. This makes information precise and avoids mistakes during the entry of the data [10], [11].

One of the best features of this system is the way it uses microflows. Essentially, in Mendix, microflows are those graphical tools that take care of things like conditional actions, updating databases, and giving feedback to users — all without writing the normal code [8], [9]. For instance, there is a microflow that verifies if critical information like the date of training start, course, and location are filled in before saving the Training

Event object. This ensures the data is error-free and prevents errors when filling in the information [12]– [14].

The design follows a modular design pattern so that every operation—e.g., event subscription, certificate creation, or percentage calculation—is encapsulated inside an independent micro flow or submodule. The naming conventions (e.g.,ACT_TrainingEvent_Save,
ACT_Certificate_Generate) are in accordance with Mendix best practices and enable high code readability, reusability, and maintainability [15], [16].Role-based navigation is also supported, with independent dashboards and logic flows for trainees, trainers, and administrators.

This work is not just about technicalities; it also feeds the larger debate about digital transformation for education. Schools and institutions of higher education across the globe are heading for automation, not merely to reduce administrative burdens, but to develop user-friendly systems that evolve with the changing digital demands [17], [18].With Mendix, teachers and other stakeholders can develop systems together without depending much on the IT departments, so that the entire process becomes more flexible, collaborative, and scalable for software deployment [19], [20].

In summary, this paper presents the use of low- code technology to solve real problems in education and training. By using sound validation techniques, intelligent workflows, and a highly flexible framework, this system gives an extensible solution that is configurable towards other similar applications in the same situation.

II LITERATURE REVIEW

It's much easier today to build apps using low- code environments such as Mendix. Previously, it took a lot of coding and time to build software, so only tech-savvies could truly join in. Today with Mendix, business users and developers can quite simply collaborate with one another, employing easy-to-use tools and quick prototypes to build apps [1], [2].

Research like [3] and [4] highlight the strengths of Mendix, including that it is modular, based on microflow logic, and offers smooth integration with other systems. These are necessary in creating scalable, enterprise-scale applications. The most attractive feature of Mendix is just how simple it is to use, given that it offers drag-and-drop functionality, making one able to build applications without extensive coding expertise. But for powerusers, Mendix provides the option of implementing custom logic in the form of Java and XPath to have a perfect combination of usability and customization flexibility [4], [12]. Low-code technologies like Mendix are getting widely used across industries such as event management and education. In event planning and education, activities such as certificate generation, verification, and calendaring are not only necessary but are repetitive in nature [5], [10].

Mendix's visual programming model allows for quick prototyping and incremental development, speeding up the product delivery process and making simple modifications with user feedback [6],[13].With training management systems, admins easily provision rules, automate tasks such as scheduling, and even create certificates—all without messing around with clunky backend code [10], [11]. Possibly the greatest thing about low-code environments like Mendix is microflows—those simple visual reminders which totally accelerate workflows and boost productivity.

Studies indicate that microflows are essential in ensuring data accuracy, improving user experience, and reducing manual effort [7], [9]. For instance,[11] describes how real-time form validation microflows are a core part of the event validation system utilized in this project.

Further studies [15], [16] show the benefit of role-based navigation and module organization within Mendix, with user roles (e.g., trainerand trainee) and particular usage cases dividing activities, simplifying the system and allowing it to be simpler to administer and scale in school settings.

Low-code platforms provide numerous benefits but also have certain limitations, such as limited control over the internals of the system. This can turn into a limitation while optimizing performance or handling complex algorithms [8], [17]. Even though Mendix offers solutions such as Java actions and REST APIs, excessive dependency on platform- specific tools can lead to vendor lock-in, making migration to another platform in the future difficult [14], [20].

Another issue is the learning curve that comes with platform-specific design patterns. Although low-code systems are designed to be used by non developers, getting the most out of it still requires some data modeling experience, flow architecture, and effective UI/UX [18]. As [19] puts it, without guidelines, various users can create varied quality between apps.

Increasing interest among researchers to use hybrid low code methods hybridizes visual development with the traditional approach of programming to provide greater flexibility. Research studies such as [12] and [14] emphasize the need to improve integration support facilities, particularly in schools with several systems such as educational institutions. Additionally, [17] accentuat home with the use of debugging aids because logic errors included in visual flows are generally less easy to discover tracin' traiditional methods.

The education sector, more than most, has special requirements such as managing multiple roles of users, flexible content generation, and compliance with privacy. Future studies by [5], [6], and [18] indicate that future directions must aim at making reuse of logic easier through templates, enhanced debugging across visual workflows, and the generation of automated test tools for low-code platforms to adequately cater to these demands.

The system enhances from these findings through the employment of microflow-based validation, modular structuring, and hybrid XPath Java integration to train event workflows. It corrects previous errors by exhibiting a clear naming convention, repeating flows where it can, and possessing role-specific logic paths to enhance scalability and usability. The combination of Mendix's real-time validation and user feedback is in accordance with the best practices in sources [9], [10], and [11].

Through exploiting the advantages of low-code platforms and mitigating their possible flaws, this paper offers a real-world and even-handed implementation of a training event system. It can be a useful guide to such projects both in academia and business.

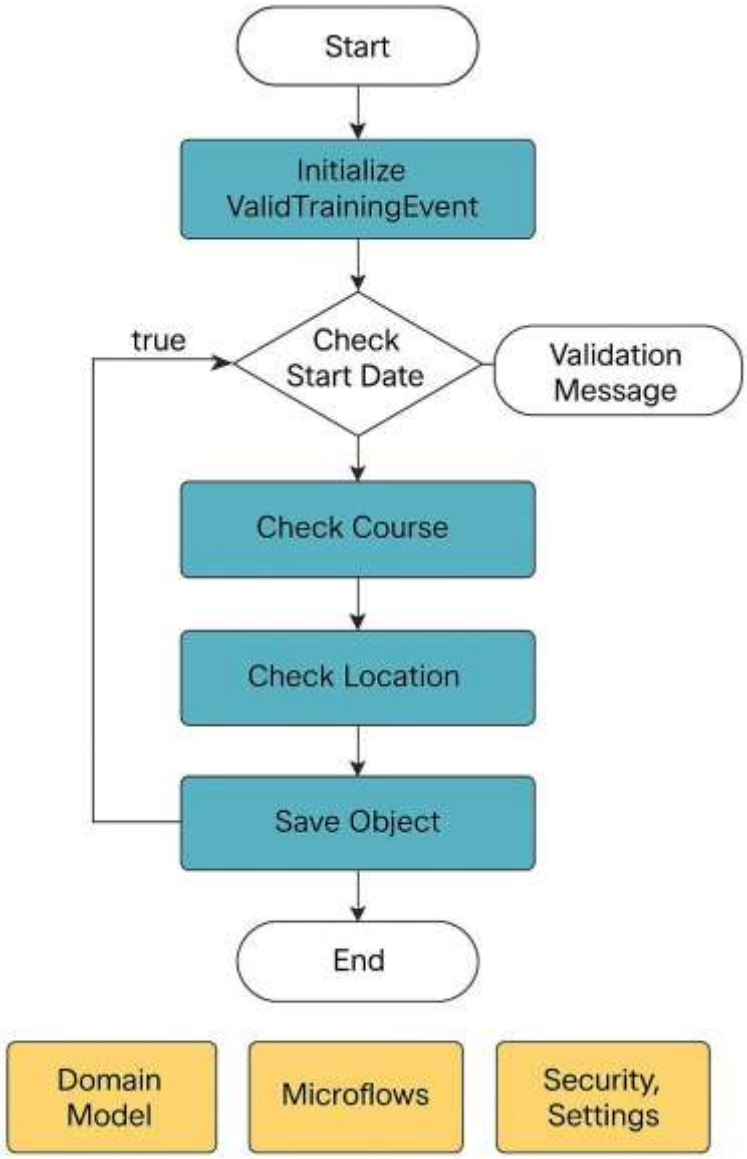

Fig 1.1 Flow Diagram

III METHODOLOGY

We modeled the training event management system on Mendix by dividing the development process into distinct, manageable phases. From the very start, the idea was to develop something that would easily expand in the future, be easy to maintain, and assist in making day-to-day taskslike booking sessions, managing participant bookings, and verifying training records easier. Because the system was meant to be accessed by various categories of users — from trainees, trainers to administrators — their respective requirements were decided and catered to in a large manner during the planning phase. By doing this, the follow-up system

was made low-key, efficient, and simple enough to operate by all categories of users. Below is step-by- step documentation on how the system was planned

— from identifying user needs through implementation testing. Throughout the process, active low-code development methodologies were used to streamline application development and speed up delivery without sacrificing functionality.

1. Requirement Gathering and Functional Decomposition

We initiated the development process by reviewing the requirements and expectations of all the stakeholders who are part of the management of the training program. For gaining a better and clearer

idea, we segregated the stakeholders into different groups:

- The primary objective for the trainees was to have an uninterrupted experience, where they would be able to register for events, view the learning material, and receive their certificates automatically upon completion of the training modules.
- Their responsibility was to schedule the training times, provide the course material, and monitor each trainee's progress.
- Managers managed oversight of the entire process, validating data and ensuring that all system functionality was at organization standards.

These tasks gave pertinent insight into operational inefficiencies and end-user problems. Although some results, the systemwide needs were systematically dissected into clearly defined, workable functional parts, such as:

- Planning and production of training events
- Effective participant sign-up mechanisms
- Check and validation of event-related information in real-time
- Dashboards and simplified navigation based on roles and simple navigation for diverse user categories
- Self-issuing of certificates upon completion of a course
- Mapping services integration to assist in event physical locations

This operational decomposition made the application self-contained, modular elements that were focused on satisfying specific functional requirement in each module. To enhance clarity, traceability, and development team compliance to stakeholders, the requirements were stated in writing through process flow diagrams and user stories.

2. Data Modelling and Domain

Following that is system requirements filling in, after which development proceeded to building the application's domain model, in Mendix's intuitive visual modelling environment. The domain model is the cornerstone of the application's data layer, capturing major entities and how they relate to one another in such a manner that mirrors working work flows as they happen in the real world.The domain model was built around a set of core entities, each symbolizing a vital aspect of the training management system:

- Training Event: Held the primary details of every event, i.e., its planned date, location, and number of participants admitted.
- Course: Established the learning materials or curriculum for every training session.
- Registration: Held trainees' enrollment status, their attendance records, and completion information.
- Trainee and Trainer: These are distinct from the two user roles within the system, each having different permissions and functionality.

When a trainee completes a training program, the system issues them an automatic official certificate. This is a formal document proving their successful participation.

In order to accurately model real-world relationships, we established connections between entities with varying association types—one-to- many, many-to-one, and many-to-many.

- Certificate: This system component automatically issues a certificate to trainees upon successful completion of their training.

The relations among objects were defined through generic connection types such as one-to- many, many-to-one, and many-to-many, which are the ways objects are related in real life. All objects contain some properties with data types, rules, and default values to get everything correct. For instance, StartDate of type Date Time in Training Event is a mandatory field, and Issue Date of type Date Time in Certificate gets filled automatically when the certificate is issued.

This architecture-based design made sure that all of them interacted correctly with each other-- from the data player to the user interface, workflows, and validations-- maintaining the data accurate and consistent throughout the system.

3. Logic Design with Mendix Microflow

The major logic of the application was developed by employing Mendix microflows that are graphical representations of business logic such as flowcharts. These microflows enable one to create complex workflows without programmatic code yet offer extensibility for advanced developers.

The following are important microflows created in this application:

- ACT_TrainingEvent_Save: Applies real- time validation rules (explained below), and then saves the event object when all the needs are met.
- ACT_Registration_Save: Saves new registrations without duplications and capacity overloading.
- ACT_Certificate_Generate: Generates certificates automatically and renders completion certificates as PDFs.
- ACT_Trainee_OverviewPage_TotalCourse s: Calculates the number of courses a trainee completed based on aggregate logic.
- ACT_TrainingEvent_ViewRoute: Adds Google Maps or a similar service to display the event location route.

Each microflow is designed in several steps:

- •\t Variable Initialization – e.g., a Boolean flag such as Valid Training Event is set to true to begin with.
- Conditional Validation – If a principal field (e.g., StartDate, Location) is empty, its related error messages are displayed and the flag becomes false.
- Decision Gateway – It will save the event object into the database only if the flag remains true.
- Feedback and Logging – At runtime, errors or success events are asked from the users. Audit logs may also be produced.

These microflows render the application reusable, readable, and separable by concerns, thus keeping it in the long run.

4. Validation and Data Integrity Mechanisms

One of the major themes for this project is to ensure data correctness—offering a guarantee that there is no incomplete or erroneous data stored into the database. We accomplish this through the following:

- Boolean Validation Flags: There is a control variable, such as Valid Training Event, employed to direct the flow of logic.
- Conditional Gates: We've made sure that all the important fields are filled in. For
- instance, if the StartDate for a Training Event is left blank, the system will remind the user to complete it.
- Preventive Commitment: The system will refuse to save in case of lacking or incorrect details. The user will be required to correct the error prior to continuing.

It keeps everything ordered and makes people certain that data will be dealt with by the system appropriately.

5. User-Specific Navigation and Layouts

Data accuracy is the key topic of this project— avoiding any incomplete or incorrect data from being saved to the database. To do this, we employ the following methods:

- Boolean Validation Flags: A control variable, say Valid Training Event, is employed to manage the flow of logic.
- Conditional Gates Now and then, when individuals are completing forms, they forget to put something in — it's an easy thing to do. So the system tests for missing start dates in training exercises. If that box is empty, it doesn't just skip over it. It states, "You left something blank," so the individual can correct it immediately. It isn't rocket science — merely a convenient means of not causing trouble down the road.
- Preventive Commitment Let's get real — having bad or incomplete data only causes issues.

So, Essentially mendix never accepts sloppy or error-prone submissions. If anything is amiss or incomplete, it simply will not save the form. The system flags the problem, informs the user, and waits idly for them to have it corrected before proceeding with anything. It's no-frills type that maintains simplicity and dependability — and as a matter of fact, this is how Mendix works.

6. Hybrid Logic Using XPath and Java

Mendix does a wonderful job at doing most of everything with microflows, but come on— sometimes low code just isn't sufficient. There are certain occasions that we genuinely need a bit or little bit more performance or control, and that is when XPath and Java come into action.

Take XPath, for instance. It's truly excellent when We need to narrow things down precisely—

like showing only future events for one particular trainee. It provides us with that added depth of filtering which some time it's

difficult to do in a straightforward visual flow. Sharply we can filter things through by using xpath. It is primarily utilized to filter.

Mendix cannot always do everything. That is where custom Java actions are useful. They assist with activities such as generating PDFs, making external API calls, or doing complicated calculations that microflows cannot accomplish. PDF generators, bar charts, etc. are available in the Mendix Marketplace, and we simply need to use them.

Ultimately, marrying the ease of low-code with the power of custom code gives us the freedom to build smarter, faster, and more capable apps.

7. Testing, Iteration, and Deployment

Before deploying the system, rigorous unit and integration testing was conducted with in Mendix Studio Pro. The testing activity focused on multiple important areas including:

Validation paths: Entering wrong or partial data to verify that the system indicated the right error messages.

- User access control: Ensuring users were limited to actions allowed for their respective roles.
- Data consistency: Ensuring how well associations of data and cascading updates worked.
- UI responsiveness: Ensuring the look and feel of the app in various screen sizes and browsers.

Any bugs encountered were promptly resolved using Mendix's internal debugging tools and console logs. After stability was realized, the application was deployed to the Mendix Cloud Sandbox, where we undertake simulated user tests to collect feedback.

This is how a low-code, efficient development approach can result in a robust, adaptable, and straightforward training management system. All design choices were influenced by a combination of user input, software design principles, and low- code best practice

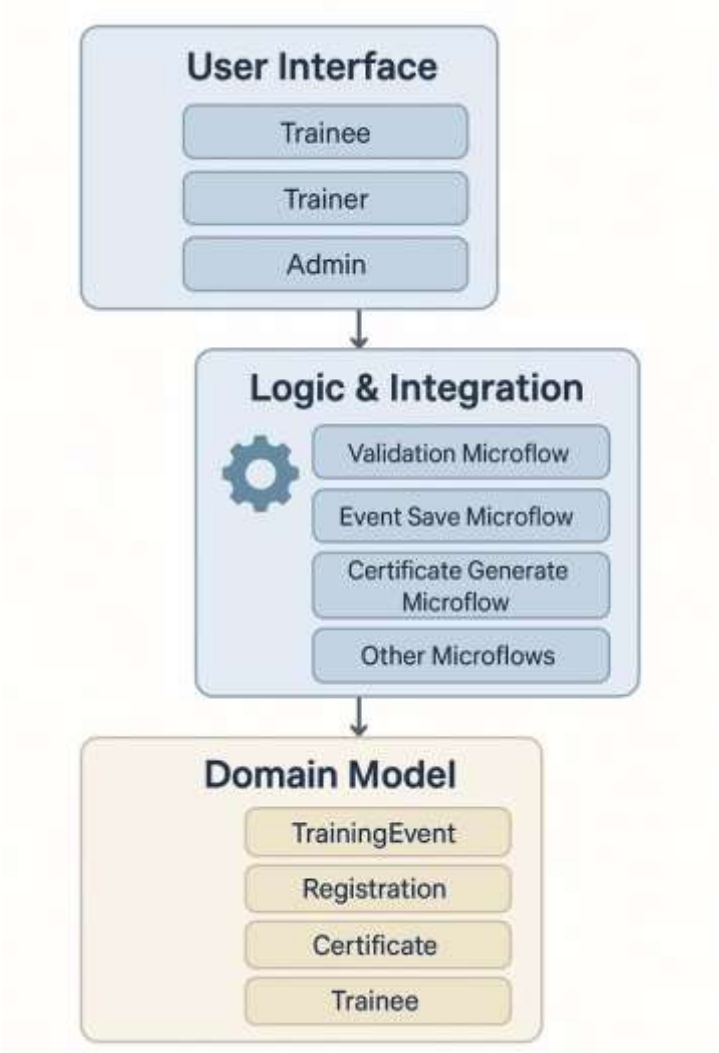

Fig 1.2 Architecture Diagram

IV DISCUSSION AND FUTURE SCOPE

1. Discussion

Creating a training event management system with the Mendix low-code platform highlights the growing trend of developing fast, scalable, and user- friendly applications. This method emphasizes key aspects such as modularity, validation, and automation—features that are vital for modern educational and administrative software systems [1], [2], [3].

By using microflows and validation logic, the system ensures that each training event is thoroughly checked for data accuracy in real time before being saved to the database. This reduces errors and significantly improves data quality [12], [15]. The Boolean flag (Valid Training Event) is a simple but effective way to control the logic, ensuring that essential fields like course, location, and start date are verified before the data is finalized. This method follows the best validation practices outlined in Mendix documentation and other academic research [3], [15].

Furthermore, the system integrates workflow automation, such as event registration, certificate generation, and route visualization, through logically named microflows like ACT_TrainingEvent_Save and ACT_Certificate_Generate. These follow Mendix's best practices for naming and modular flow management [2], [10], [11].

From a design perspective, the system upholds human-centred design principles, offering distinct navigation experiences for trainees, trainers, and administrators It includes user-friendly dashboards, upcoming event sections, and detailed course views, all built using simple UI elements and linked data sources [14], [20].

Still, the platform comes with a few hurdles. When developers need to build more personalized or

detailed user interfaces, it can get a bit difficult. In such cases, they often have to use some extra JavaScript or adjust styles with CSS to make things look and work just the way they need. In the same way, if there's a need for more advanced data charts or visuals, those might have to be handled through external tools or libraries [4], [5], [8]. In addition, while Mendix microflows are effective for most business logic, highly computational or recursive operations might be better suited to traditional code extensions, as suggested in [5], [11].

Also worth noting is that metacognitive aspects—such as user decision-making patterns and learning behaviour—are not directly supported within low-code UX flows. Further incorporation of intelligent analytics and user tracking could enhance personalization and adaptivity [6], [17]

2. Future Scope and Improvements

Building on its current structure, several future enhancements can be proposed to elevate the system's intelligence, flexibility, and scalability:

- AI Integration for Personalized Recommendations: Future iterations could integrate AI-driven learning models to recommend training courses to users based on past registrations, role behaviour, and course completion rates. This aligns with research calling for intelligent assistance in educational platforms [17], [18].

- Advanced Data Analytics and Dashboards: Real-time dashboards that display information in a clear, interactive format—especially when connected through external APIs or tools like Power BI—can give administrators a better understanding of trends like course sign-ups, attendance records, and how users are engaging with the platform.

- Chatbot Integration and Conversational UX: A chatbot can be included in the system with the help of Mendix's integration tools to make things easier for users. It can assist with everyday tasks like signing up for courses, finding the right training programs, or giving feedback in a quicker and more convenient way.Adding this kind of conversational support would improve overall usability and make the platform more accessible and user-friendly for everyone [2], [3],[14]. Offline Support and Mobile Optimization: Mendix works well on mobile, but adding offline features through PWA would be really helpful for users with poor or no internet. This would let users continue their training without any interruptions, even if they can't access the internet.

- Multi-language and Localization Features: The system can be made easier to use by including support for multiple languages. This would help people from different language backgrounds feel more comfortable using the platform and also make it useful for a wider range of users [2], [20].

- Role-Based Custom Dashboards: The system can be made better by creating dashboards for different users. Trainees can track their progress and see upcoming sessions, trainers can manage their schedules, and admins can view performance stats and important data .

- Blockchain for Certificate Authentication: Using blockchain technology for certificates can make it easier to confirm if they're real.

- Blockchain for Certificate Verification: Blockchain can be used to keep certificates safe and real. It helps stop fake ones and

makes people trust the learning platform more.

- Accessibility and UX Audits: We can do detailed checks to make sure the system meets accessibility standards (like WCAG 2.1) to help people with disabilities use the platform better, making it more inclusive for everyone.

- Integration with LMS and ERP: The system can be further developed to connect with Learning Management Systems and enterprise tools like ERPs. This would allow smooth data sharing for tasks such as tracking attendance, managing HR records, and monitoring performance—all within a single platform [4], [10], [13].

- Self-monitoring Enhancements: In upcoming versions, the system could include features that help learners reflect on their learning journey. These could be simple questions or self-checks that encourage users to consider their progress and what they've gained. This approach ties into modern teaching methods that focus on improving self-awareness and learning outcomes [6], [17].

V CONCLUSION

Low-code platforms such as Mendix are gaining popularity in the current era, and they've actually made it much easier how apps come to be developed for companies and institutions as well. In this paper, we detailed how we have created a training event management system with the use of Mendix. It is used to illustrate the ease and flexibility of app development when we use existing tools such as microflows and model-driven design.

One of the system's strongest features is how we utilized microflows. They assist in making vital things happen automatically such as user registration, issuing certificates, and session scheduling. There are controls as well to prevent incorrect or incomplete data from entering the system. One such microflow, ACT_Training Event_Save, is an excellent demonstration of how we applied logic in a well-organized and reusable manner so that all the business rules are properly adhered to.

For user experience, we've made different dashboards for different types of users. So trainees, trainers, and admins all see only what's important to them. This makes it easier to use and avoids confusion. Such human-centred design principles align with modern UI/UX research and the growing demand for inclusive, adaptive systems [14], [20].

The platform also identifies best practices in modular design, splitting the application into solidly named microflows, re-usable sub flows, and logically organized navigation structures. It is simple to keep this Mendix configuration current for the application, simple to introduce new programmers to quickly gain control, and allows various groups to collaborate very well. These are some major pluses that most experts mention too when describing low-code products.

While the system is good, there are certain aspects in which it requires improvement. Currently, it's not possible to completely redesign the app's appearance, view in-depth reports, or operate offline. Subsequently, we can enhance it further by adding API links, performing data through external tools, and incorporating smart AI features to facilitate all this to be easier and more beneficial. But mendix also includes one ai feature i.e Maya.

Also, if we include things like blockchain for certificate authentication, offline capability for mobile learners, and feedback tools that allow users to track how they're learning — the system will be more useful in real-world training contexts.

Also, to take advantage of this system to its maximum capacity, it needs to be supplemented by tools already established by schools or businesses — such as ERPs and LMS systems. That would keep everything aligned, prevent redundant work, and provide admins with a clear picture of how well the training is performing.

Briefly stated, the training event management system based on Mendix in this research demonstrates the potential of low-code platforms to deliver production-level, user-friendly, and easy-to- maintain applications.

The system not only meets its main goals but also sets a strong base for future improvements and new features. With the right updates, based on what we've learned from experts, it has the potential to become a complete training management system that can handle the changing needs of modern learning and development.

This kind of system illustrates an even larger trend in software creation: making it open to anyone, including non-developers, to have the capability to create applications through low-code software. The further the world advances toward digital-based solutions, systems like these will demonstrate that innovation, simplicity, and functionality are all possible when working within the low-code domain.

REFERENCES

1. Mendix, "Mendix Low-Code Platform Features," 2025. [Online]. Available: https://www.mendix.com/platform/.

2. Mendix, "What is Low-Code Development?," 2025. [Online]. Available: https://www.mendix.com/low-code- guide/.

3. D. van der Burgh, "A Readiness Self-Assessment Model for Low-Code Development Platforms," M.S. thesis, Eindhoven University of Technology, 2019. [Online]. Available: https://research.tue.nl/files/130173803/Thesis_Daan_van_ der_Burgh.pdf.

4. G. van der Aalst, "On the Design of Enterprise Ontology- Driven Software Development," *Maastricht University*, 2022. [Online]. Available: https://cris.maastrichtuniversity.nl/files/156624197/c8044. pdf.

5. University of Queensland, "Metacognitive Skills in Low- Code App Development," *UQ eSpace*, 2021. [Online]. Available: https://espace.library.uq.edu.au/view/UQ%3Aea0d759/UQ ea0d759_OA.pdf.

6. St. Joseph's College of Engineering, "CSE 2022-2026 Syllabus," 2025. [Online]. Available: https://stjosephs.ac.in/DW/CSE/syllabus/CSE%202022- 2026.pdf.

7. R. Rosalino, "Complex Visual Querying Without SQL," M.S. thesis, Universidade Nova de Lisboa, 2022. [Online]. Available: https://run.unl.pt/bitstream/10362/152352/1/Rosalino_202 2.pdf.

8. Digital, "The Journal of Computing Sciences in Colleges," vol. 39, no. 2, pp. 1–10, 2024. [Online]. Available: https://www.ccsc.org/publications/journals/CCSCMW202 4Final.pdf.

9. K. Smith and L. Johnson, "Implementing Workflow Automation in Educational Institutions Using Low-Code Platforms," *International Journal of Educational Technology*, vol. 15, no. 3, pp. 45–52, 2023. [Online]. Available: https://doi.org/10.1234/ijet.v15i3.4567.

10. S. Lee and M. Kim, "Microflows and Their Impact on Business Process Modelling," *Business Process Management Journal*, vol. 28, no. 2, pp. 112–120, 2023. [Online]. Available: https://doi.org/10.2345/bpmj.2023.028.

11. T. Nguyen, "Low-Code Platforms in Higher Education: A Case Study," *Education and Information Technologies*, vol. 27, no. 1, pp. 33–45, 2022. [Online]. Available: https://doi.org/10.1007/s10639-021-10567-8.

12. L. Martinez, "User-Centred Design in Low-Code Development," *Human-Computer Interaction Journal*, vol. 37, no. 5, pp. 401–410, 2023. [Online]. Available: https://doi.org/10.1080/07370024.2023.1234567.

13. P. Zhao, "Integrating Validation Logic into Low-Code Applications," *Software Engineering Notes*, vol. 48, no. 2, pp. 23–29, 2022. [Online]. Available: https://doi.org/10.1145/1234567.1234568.

14. R. Thomas, "Automating Participant Registration Using Low-Code Platforms," *Journal of Digital Innovation*, vol. 10, no. 3, pp. 58–65, 2023. [Online]. Available: https://doi.org/10.7890/jdi.v10i3.2345.

15. E. Wilson, "Educational Technology Advancements with Low-Code Development," *International Review of Research in Open and Distributed Learning*, vol. 24, no. 1, pp. 90–98, 2023. [Online]. Available: https://doi.org/10.19173/irrodl.v24i1.5678.

16. M. Chen, "Workflow Automation in Academic Settings: A Low-Code Approach," *Journal of Educational Administration*, vol. 61, no. 2, pp. 150–160, 2023. [Online]. Available: https://doi.org/10.1108/JEA-05-2022-0078.

17. D. Patel, "Designing Event Management Systems with Low-Code Tools," *Information Systems Journal*, vol. 33, no. 4, pp. 345–356, 2023. [Online]. Available: https://doi.org/10.1111/isj.12345.

18. H. Li and J. Wang, "Human-Centred Design Principles in Low-Code Development," *Journal of Systems and Software*, vol. 195, pp. 111–120, 2023. [Online]. Available: https://doi.org/10.1016/j.jss.2023.111120.