International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Tailored Module Development in Odoo: Challenges and Solution Developers' Guide

Dhruviba G. Zala[1], Piyush Kashiyani[2]

[1]Odoo Developer Rajkot, India
[2]Assistant Professor Rajkot, India

Abstract
Odoo is an open-source, comprehensive and modular EPR platform that provides businesses with a wide range of built-in modules to address a wide range of business functions. But for unique and specialized business needs Odoo provides the functionality of making custom modules for enhancing and providing best quality results for EPR. Odoo offers two editions i.e. Odoo Enterprise Edition And Odoo Community Edition. Odoo Enterprise edition and Odoo Community Edition each have around 70-80 and 150-200 built-in modules respectively including core and extended features. This paper examines the design, development, deployment as well as the challenges and solution of creating tailored modules in Odoo.This research demonstrates the adaptability and scalability of Odoo's architecture by going into detail about the procedure, including model generation, user interface customisation, and security measures.We demonstrate through this research that the open-source nature and modular architecture Odoo offers a solid foundation of creating customized ERP capabilities.

Keywords: Customized ERP Development, Enterprise edition, Community Edition, Odoo;

1. INTRODUCTION

Enterprise Resource Planning [ERP] is a widely used term for corporate companies. Odoo is one of the most used ERP in small and medium sized companies. While odoo itself provides users with a wide range of built-in modules and functionalities it also provides a way to create customized modules for the extra functionalities required by the clients. Every organization has a variety of different needs and that is where the actual need of creating custom modules arises. Odoo also enables the integration of the third-party apps like Paypal, Razorpay, Stripe, Worldpay to provide the functionality of secure payments in eCommerce, Point Of Sale (POS).

Tailored Modules are created to meet the distinctive needs for the clients which provides additional features with enhanced integrations that are not part of the Odoo's unconventional capabilities. These custom modules can be either created by inheriting one of the already existing modules and changing as well as adding new features to it or by creating a custom module from scratch to create a whole new component respective to the requirement of the client. These modules are built with Odoo's powerful framework, which ensures versatility and by sustaining XML for view design and Python for backend development. The development of custom modules is extensive and involves steps like defining data modules, creating the corresponding views and user interfaces, establishing the business logics and ensuring the precise access control for the secure and shielded use of the same in the organizations.

This paper is designed to provide a guide based on the making and functioning of the custom modules in

odoo. It emphasizes the significance of custom-tailored modules, steps involved as well as the challenges one encounters when doing so.

2. Standard vs. Tailored Modules in Odoo

Odoo has a modular ERP architecture with a wide range of standard inbuilt modules that serves as a foundation to the organizations. These modules provide the client with exceptional functionalities but the businesses often face many other operational issues which can be addressed using the custom-developed modules.

2.1 Standard Modules

Standard Modules are the modules that are pre-built in Odoo and accessible through the Odoo App Store. These are integrated to resolve and address the basic needs of the business with minimal coding and configuration making it a go-to for a wide range of industries.

Core Features of the Standard Modules:

- Preconfigured Functionalities: HR, CRM, Sales, and Inventory modules are prepared for instant deployment.
- Ease of Implementation: Standard modules are easy to use and require slight modification.
- Economical: They are either free or offered for a small fee as part of Odoo's core system.
- Frequent Updates: Odoo keeps these modules up to date and aligned with changing technology and business standards.
- Standard modules have limits when it comes to handling special or sector-specific procedures, even though they work very well for companies with broad needs.

2.2 Tailored Modules

Tailored Modules are custom-designed modules that are developed or created for specific needs of clients or businesses. These modules are designed to modify the existing functionalities and introduce entirely new workflows.

Core Features of the Tailored Modules:

- Personalisation: Custom fields, processes, and organisation-specific features can all be included in custom modules.
- Industry-specific solutions: They address specialized industries where conventional modules are insufficient, such as manufacturing, healthcare, or construction.
- Third-Party Integration: Odoo may be integrated with external applications, specialist tools, or legacy systems using customized modules.
- Scalability: Personalized modules may be modified to accommodate changes as the company expands or develops.

3. Challenges in Module Development

Odoo has a modular ERP architecture with a wide range of standard inbuilt modules and those modules can be customized according to the clients' needs. But there are also a list of challenges developers face during that process.

3.1 Customization Complexity:

A key component of Odoo development is customisation, which enables companies to adapt standard modules to their particular workflows. However, inappropriate customisation can result in a number of

dangers, such as compatibility problems, functionality faults, and user interface malfunctions.

3.1.1 Dangers of Changing Essential Odoo Features-

- Violating Standard Functionality: Modifying core models and views directly may cause unexpected behaviour in modules that are already in place.
- Upgrade Challenges: When Odoo is updated, custom modifications could be lost, necessitating a thorough reimplementation.
- Conflicts with Third-Party Modules: Direct changes have the potential to disrupt other installed modules, leading to unstable systems.

3.1.2 Top Customisation Techniques-

Developers should use appropriate extension protocols, like these, to steer clear of these pitfalls:

- To increase current functionality without changing core files, use Model Inheritance (_inherit).
- To change UI elements in an organised and upgrade-safe way, use View Inheritance (xpath).
- When business logic doesn't need altering Odoo's underlying models, external scripts are used.

Developers may make sure that customisations are compatible with next releases and maintainable by adhering to certain best practices.

3.2 Version Compatibility:

Odoo is constantly evolving, with new iterations bringing structural modifications to database models, APIs, and code. Odoo 18 is now the most recent version, and maintaining compatibility across versions is essential for the long-term viability of modules.

3.2.1 Version Migration Challenges

It is frequently necessary to make significant code changes when migrating an Odoo module from one version to another because:

- Modifications to ORM Methods: Newer versions may include modified versions of functions like search(), write(), and unlink().
- Changes to Model Fields and Naming Conventions: Some models and fields can be deprecated or renamed.
- Variations in API Structures: Integrations may be impacted by variations in API endpoints and parameters across versions.

3.3 Performance Considerations:

Poorly optimized custom modules can negatively impact the overall performance of an Odoo instance. It is essential to develop efficient code that ensures system responsiveness and scalability.

3.3.1 Common bottlenecks in performance

- Ineffective Database Queries: Slow response times and higher server load might result from poorly constructed queries.
- Overuse of Synchronous Processing: When business logic activities are blocked, the user interface becomes unusable.
- Excessive Background Jobs: Automated tasks or poorly optimised cron jobs might use up too much resources, which slows down the system.

3.3.2 Methods for Performance Optimisation

- Optimise ORM Queries: To increase performance, use batch processing (search_read() rather than several search() calls) and indexed fields.

- Use Asynchronous Processing: Set up Odoo queue jobs to carry out lengthy processes without blocking.
- Track System Performance: To find and fix bottlenecks, make use of Odoo's integrated logging and profiling features.

Even as the complexity of custom modules increases, maintaining optimal performance aids in keeping the Odoo system responsive and scalable.

3.4 Integration Challenges:

Integration with external APIs, SMS providers, and payment gateways is necessary for many Odoo systems. Although these linkages improve functionality, they also present a number of difficulties.

3.4.1 Dependency on External Services

- The reliance on outside services Downtime: Odoo modules that depend on an external API may malfunction, interfering with business operations.
- Rate Limits: Restrictions placed on API requests by numerous third-party services may result in service disruptions.
- Unexpected API Changes: Modules may need to be modified to retain compatibility when providers update their APIs without warning.

3.4.2 Handling Webhooks & Event-Driven Integrations

Webhooks are used by many external services for real-time data changes, which need to be managed effectively:

- Webhook Reliability refers Data consistency depends on preventing lost or duplicate webhook events.
- Asynchronous Processing: By handling webhooks with Odoo queue tasks, UI delays are avoided and system performance is increased.
- Error Handling & Logging: Failed webhook events are reprocessed thanks to appropriate logging and retry procedures.

Strong error-handling and monitoring procedures are necessary when integrating Odoo with outside services in order to preserve system stability.

3.5 Custom Module Security Risks

Financial transactions, client information, and other private company data are all handled by Odoo modules. Unauthorised access and data breaches might result from improper security measures in bespoke modules.

3.5.1 Access Control & Permissions

- Role-Based Access Control (RBAC): Preventing unwanted access by ensuring users have the right permissions.
- Field-Level Security: Private data is safeguarded by limiting access to sensitive fields.
- Secure API Authentication: Unauthorised API access is avoided by using OAuth, API keys, or token-based authentication.

3.5.2 Compliance & Data Privacy

Custom Odoo modules have to abide by data protection laws like:

- The General Data Protection Regulation, or GDPR, safeguards the private information of citizens of the European Union.
- Healthcare data security is governed by the Health Insurance Portability and Accountability Act

(HIPAA).
- Payment card transactions are handled securely according to PCI DSS (Payment Card Industry Data Security Standard).

3.5.3 Use of Secure Data Transmission

Implement SSL/TLS encryption to provide secure data exchanges and API queries.
- Sanitise User Input: Validate and escape input data to avoid SQL injection and cross-site scripting (XSS) attacks.
- Frequent Security Audits: To guarantee adherence to security best practices, conduct regular code reviews and vulnerability assessments.
- Developers can safeguard confidential company information and guarantee regulatory compliance by putting in place robust security procedures.

3.6 Conclusion

Customised Odoo module development gives companies the freedom to modify ERP features to suit their requirements. Developers must, however, overcome a number of obstacles, such as the difficulty of customisation, compatibility with upgrades, performance snags, integration problems, and security threats. Developers may provide reliable and stable Odoo customisations that enable business expansion while guaranteeing long-term system stability by adhering to best practices including modular development, effective performance optimisation, safe API handling, and compliance with security requirements.

4. Solutions & Best Practices

4.1 Modular Development Approach

Odoo's architecture supports a modular development approach, allowing businesses to extend functionalities without modifying the core system.
- Avoid modifying core code: Directly altering Odoo's core files can lead to conflicts during upgrades. Instead, use inheritance (_inherit, _inherits) to extend models and views.
- Make use of unique modules: To maintain their independence from the normal Odoo modules, store customisations in distinct modules (custom_addons/).
- Utilise pre-existing Odoo frameworks: To effectively incorporate new features, make use of Odoo's built-in hooks, APIs, and extensions.
- Assure upgrade compatibility by adhering to Odoo's modular design principles, which facilitate updates and migrations.

4.2 Best Practices for Odoo ORM and APIs

Database operations are abstracted by Odoo's ORM (Object-Relational Mapping), which makes it safe and effective.
- Avoid direct SQL queries: Use ORM methods (create(), write(), search()) instead of raw SQL to maintain database consistency.
- Make sensible use of calculated fields: Performance can be slowed down by using @api.depends excessively; only include dependencies that are required.
- Optimise search queries: For greater efficiency, use search_read() or mapped() instead of several search() calls.
- REST API for integrations: To properly integrate third-party apps, use Odoo's JSON-RPC or XML-RPC APIs.

4.3 Version Control & CI/CD Integration

Version control and automated deployment techniques are necessary for efficient management of bespoke modules..

- Use Git for version control: To improve tracking and rollback capabilities, keep custom modules in repositories (such as GitHub, GitLab, and Bitbucket).
- Branching strategy: To manage changes without interfering with production, adhere to the Git Flow (develop, feature, release, and hotfix branches).
- CI/CD automation: To automate the testing and deployment of Odoo modules, use solutions such as Jenkins, GitHub Actions, or GitLab CI/CD.
- Database migration scripts: When updating Odoo versions, use odoo.addons.base.models.ir_model for structured data migrations.

4.4 Methods of Testing

Testing guards against unanticipated manufacturing faults and guarantees the dependability of customised components.

- Unit Testing (unittest in Python): Test individual functions and ORM methods to catch issues early.
- Integration Testing: Validate module interactions with Odoo's built-in and third-party modules.
- Automated Testing (odoo.tests): Use @tagged decorators to categorize tests (e.g., performance, security).
- User Acceptance Testing (UAT): Ensure that the module aligns with business needs before deployment.

4.5 Performance Optimization Techniques

Optimising custom modules is necessary to guarantee a flawless user experience.

- Employ caching: To minimise database requests and store frequently used data, utilise @tools.ormcache.
- Indexing: To speed up queries, define indexes on fields that are frequently searched (index=True).
- Load balancing: In high-load situations, divide traffic among several Odoo instances.
- Asynchronous processing: Rather than preventing UI interactions, use odoo.queue_job for background operations.

5. Case Studies & Real-World Applications

5.1 Successful Custom Module Implementations

Case Study 1: Google Places API Address Autofill in Odoo.

Issue:

Inconsistent data and inefficiencies in customer and business records resulted from manual address entering.

Solution:

The solution was a specially designed module that automatically fills in the address fields (street, city, state, nation, and postal code) by integrating the Google Places API. It eliminates the need for human input and guarantees precise and organised address data.

Result:

- Increased precision of data in address fields.
- More rapid and effective data entry.
- Smooth user experience thanks to the integration of the Google Places API.

Case Study 2: Sales Target Module for Performance Management

Issue:

Companies needed a way to set, track, and evaluate different types of sales targets for teams and individual sales representatives.

Solution:

A custom Odoo module was developed to allow businesses to create and manage sales targets, track performance, and generate reports. The module supports various target types (monthly, quarterly, yearly) and integrates with Odoo's sales and reporting features..

Result:

- Clear goal-setting and tracking for sales teams.
- Better visibility into performance metrics.
- Enhanced motivation and accountability among sales representatives.

5.2 Comparison of Standard vs. Tailored Modules

Feature	Standard Module	Tailored Module
Customization	Limited to Odoo's built-in options	Fully customized workflows and fields
Integration	Supports Odoo's ecosystem	Can integrate with third-party applications
Scalability	Restricted by Odoo's default features	Scalable as per business needs
Cost	Lower initial cost	Higher initial cost but long-term benefits
Maintenance	Easy due to Odoo's updates	Requires active maintenance and updates

6. Conclusion & Future Scope

6.1 Summary of Key Insights

- With specially designed modules, Odoo's modular architecture allows for a great deal of customisation.
- Stability is improved through the use of best practices such ORM techniques, modular development, and performance optimisation.
- Version control and CI/CD provide seamless deployment and upkeep.

6.2 Development of Customised Modules

- Better upgrade pathways might be added in next Odoo releases to simplify migrations.
- By anticipating business needs, AI-driven automation could improve module customisation.
- Third-party integrations will be smooth thanks to improved API standardisation

6.3 Future Architecture Enhancements for Odoo

- Stricter access controls and integrated data encryption are examples of improved security measures.
- Improved backward compatibility to reduce downtime while updating Odoo versions.
- enhanced debugging capabilities to identify custom module performance snags.

References

- **Books & Official Documentation:**

1. Fabien Pinckaers, Greg Mader, Daniel Reis, Odoo Development Essentials, Packt Publishing, 2016.
2. Daniel Reis, Odoo 17 Development Essentials, Packt Publishing, 2024.
3. Odoo S.A., Odoo Official Documentation, https://www.odoo.com/documentation/latest/

- **Academic Papers & Research:**

4. T. Holz, F. Engelmann, and R. Lutz, Customization Strategies in Open-Source ERP Systems, ACM Computing Surveys, 2021.
5. A. Rashid, M. Nauman, and S. Khan, Enhancing Business Processes through ERP Customization: A Case Study of Odoo, International Journal of Information Management, 2020.

- **Industry & Case Studies:**

6. Deloitte, ERP Trends and Customization Challenges in 2024, Deloitte Insights, 2024.
7. Gartner, Odoo vs. Proprietary ERP Systems: A Comparative Study, Gartner Research, 2023.