Codedness IDE is not a no-code/low-code platform; it builds **an actual codebase for your project** with all the flexibility and advantages of a completely custom-coded application. It also doesn't suffer the issues that come with LLM-based code generation, like unreadable code, hidden (generated) bugs, and bad maintainability. Instead, it delivers a stable and thoroughly tested project with **professional API documentation**.

Take a look at the table below and see how Codedness IDE differs from other solutions.

| | Codedness IDE | NoCode/LowCode Platforms | LLM-based Generation/ Vibe-coding | Traditional Software Engineering | COTS solutions |
|---|---|---|---|---|---|
| ***Security & Privacy*** | | | | | |
| **Granular ACL** | Professionally developed group-based user access with fully customizable granular levels. | Fixed profile levels with limited customizability. | Depending on the quality of the training data. Nothing is guaranteed. | Depending on the available budget, it needs to be coded from scratch. | Not customizable, functionality level depending on the product. |
| **2-factor authentication** | Implemented by default.<br>It can be switched off for, e.g., exclusive internal use. | Available with some platforms. | The quality of security-related code is questionable with LLM-based coding. | Native solutions need to be coded from scratch. | Available in some products. |
| **Deployment model** | Deploy anywhere: cloud, private cloud, local LAN, single machines, etc. Part of the standard contract. | No external deployment; applications run inside the vendor's platform. | Deployment options depend on the quality of the generated solutions. | Deploy anywhere. | Deploy anywhere or vendor-platform lock-in, depending on the product. |
| ***Development & Deployment Speed*** | | | | | |
| **Basic Application development** | Fast generation (seconds) from the data model. A functional data model can be defined in a few hours. | Fast setup of basic features. | Validating and debugging LLM-generated code can be very time-consuming. | Manual coding is extremely time-consuming. | No development is needed; the application can be deployed directly. |
| **Data model refactoring** | Screens are automatically updated based on the data model dictionaire. | Panels need to be reworked manually after data model changes. | Data model changes require manual code refactoring. | Data model changes require manual code refactoring. | Data model changes are not possible. |

| | | | | | |
|---|---|---|---|---|---|
| **Adding custom functionality** | Insert points let you add custom-coded functionality without hassle. | Only possible if the required functionality exists as an option for the platform. | Adding functionality on top of LLM-generated code is hard. | Adding functionality to existing code can be difficult and expensive. | In most cases it is not possible to add custom functionality. |
| **Deployment** | Automated (remote) deployment and data model migration. | Deployment outside the platform is not possible. | Deploying anywhere is possible depending on the chosen technology (LLM-dependent). | Deployment options are dependent on contract negotiations. | No automated deployment available. |
| ***Codebase Quality*** | | | | | |
| **Code correctness** | Stable code generation, based on professionally engineered templates. | Codebase is not available. | Codebase quality is debateable. | Depends on codebase availability for review. | Codebase is not available. |
| **Maintainability & Documentation** | A fully documented codebase, designed for maintainability and future development. | No application-specific documentation. | LLMs don't generate API documentation. | System/API documentation is often an afterthought. | No system documentation available. |
| ***Customizability*** | | | | | |
| **Custom features** | No limitations on custom feature development. | No custom features are possible inside the platform/application. | Custom features are only possible if they are part of the training set. | No limitations on custom feature development. | No custom features are possible inside the application. |
| **API availability** | Full API, covering the complete application codebase. | API availability is limited to connecting external tools and data. | Low-quality code prevents the development of a usable API. | API development is often overlooked because of the cost. | API availability is limited to connecting external tools and data. |
| **Codebase access** | Unlimited access to generated code for reuse in custom functionality. | No access to the application codebase. | The codebase is available, but the generated code is not fit for reuse. | Codebase access is dependent on contractual negotiations. | No access to the application codebase. |

| | | | | | |
|---|---|---|---|---|---|
| **CI & DevOps support** | | | | | |
| **Feature integration** | Insert points make integrating custom features fast and hassle-free. | Not available on most platforms. | Adding new features to existing code is dificult. | Adding new features to existing code is expensive. | Not possible. |
| **Remote deployment** | Direct (native) installs and containerized installs (either local or remote). | Deployments outside the platform are not available. | Remote deployment needs to be developed from scratch. | Remote deployment needs to be developed from scratch. | Remote deployments are not supported. |
| ***Vendor Dependency & Pricing*** | | | | | |
| **Licensing model** | You own the project, the codebase, and the developed application. | User-based licensing (user counts), no real ownership. | You own the developed application. Ownership of the generated code is unsure. | Ownership is dependent on contract negotiations. | User-based licensing (user counts), no real ownership. |
| **Codebase ownership** | The full codebase and related project files can be exported from the platform. | No access to the actual codebase. | The codebase is available but might be difficult to understand or even be unreadable. | The codebase is available if it is part of the project's deliveries. | No access to the actual codebase. |
| **Vendor lock-in** | Your application is completely self-contained and runs independently from the platform. | Developed applications run inside the platform. No independent installations. | No vendor lock-in. | Dependent on contract negotiations. | Version updates are dependent on the vendor. No individual maintenance options. |
| **Price point** | Fixed price per project, no additional costs for developers or end-users. No functional price tiers. | Price per user account (developers and end-users). Additional functionality at additional cost. | Unlimited users (if it works). | Traditional software engineering and coding are expensive. | Pricing per user license. Additional modules often at additional cost. |