INTERNATIONAL JOURNAL OF INFORMATIVE & FUTURISTIC RESEARCH (IJIFR)
ISSN: 2347-1697
VOLUME 13||ISSUE 9|| MAY 2026
An Open Access, Peer Reviewed, Cross Reference DOI & Google Scholar Indexed International Research Journal
UGC Approved || IJIFR Impact Factor= 8.057
Estd Year: 2013 || Email: editors@ijifr.org
Availaible Online at: https://www.ijifr.org/

DOI: https://doi.org/10.64672/IJIFR/26.05.13.09.018 PUBLISHED ON: MAY 18, 2026

AN OPEN-SOURCE WEB-BASED CUSTOMER RELATIONSHIP MANAGEMENT SYSTEM FOR SMALL BUSINESSES: A DJANGO-DRIVEN ARCHITECTURE FOR UNIFIED LEAD, SALES, AND TASK LIFECYCLE MANAGEMENT

K Naga Sirisha [1] , B. Shireesha [2]
[1]Student, Department of Computer Applications,
Viswam Engineering College, Andhra Pradesh, India
[2]Assistant Professor, Department of Computer Applications,
Viswam Engineering College, Andhra Pradesh, India

ABSTRACT

Small businesses across diverse industries continue to operate critical customer relationships through a fragmented patchwork of spreadsheets, email clients, and ad hoc to-do lists, an approach that erodes data quality, increases the risk of missed follow-ups, and prevents systematic revenue analysis. Enterprise customer relationship management (CRM) suites such as Salesforce, Pipedrive, and Zoho CRM offer comprehensive functionality but impose recurring subscription costs and learning curves that many small enterprises find prohibitive. This paper presents the design, implementation, and evaluation of an open-source, web-based Intelligent CRM System purpose-built for small business environments. The system is developed using the Django 5.2 framework, the Python 3 programming language, and an embedded SQLite relational database, requiring no commercial licenses, no separate database server, and no proprietary infrastructure. A normalized four-entity relational schema centered on the Customer model, with Lead and Sale entities referencing Customer through cascading foreign keys and a Task entity supporting operational scheduling, captures the complete small-business customer lifecycle in a single coherent platform. Django's integrated administration interface is leveraged to deliver immediate CRUD functionality across all four modules, eliminating the development overhead of bespoke front-end code while providing search, filtering, and role-based access controls out of the box. An empirical evaluation conducted with three representative small-business users on 50 simulated transactions shows that the proposed system reduces average customer-lookup time by 76 percent, lead-status update time by 79 percent, and improves cross-record data consistency from 62 percent to 96 percent compared with a spreadsheet-based baseline. The system demonstrates that an open-source web framework combined with a disciplined relational schema can deliver an immediately deployable CRM platform whose modular architecture supports incremental enhancement toward dashboards, role-based access, and predictive analytics.

KEYWORDS *Customer Relationship Management; Django Framework; Relational Data Modelling; Small Business Automation; Lead Pipeline Tracking; SQLite*

Recommended Citation:
Sirisha, K. N., Shireesha, B. : "An Open-Source Web-Based Customer Relationship Management System for Small Businesses: A Django-Driven Architecture for Unified Lead, Sales, and Task Lifecycle Management", International Journal of Informative & Futuristic Research (IJIFR), Vol. (13) (9), May 2026, pp. 1562-1569
https://doi.org/10.64672/IJIFR/26.05.13.09.018

1. INTRODUCTION

Customer Relationship Management (CRM) refers to the integrated set of strategies, processes, and technological tools that organizations deploy to manage and analyse interactions with current and prospective customers across the entire customer lifecycle [1]. In the digital economy, where differentiation based on product features alone is increasingly difficult to sustain, the ability to deliver personalized, timely, and contextually appropriate customer experiences has emerged as one of the most significant drivers of business performance across industries of every scale [2]. The cultivation of long-term customer relationships translates directly into measurable business outcomes through repeat revenue, improved customer satisfaction, and a competitive advantage derived from superior knowledge of individual customer preferences and behavioural patterns.

Despite the wide availability of cloud-based CRM solutions at substantially reduced cost compared with enterprise predecessors, a significant proportion of small businesses continue to manage customer relationships through general-purpose tools that were never designed for the purpose. Spreadsheets such as Microsoft Excel and Google Sheets serve as de facto customer databases, email clients provide the primary channel for follow-up tracking, calendars handle scheduling, and informal notes capture ad hoc reminders [3]. The fragmentation introduced by this patchwork creates systemic inefficiencies: customer information stored across disconnected tools becomes inconsistent and difficult to retrieve, lead opportunities fall through the cracks when follow-up tasks are not linked to customer records, and the absence of a centralized customer view makes it impossible to identify behavioural trends or measure the effectiveness of sales activities. Industry research suggests that the cost of acquiring a new customer can be between five and twenty-five times greater than that of retaining an existing one, depending on the industry [4]; missed follow-ups and inconsistent service therefore translate directly into measurable revenue loss.

Enterprise CRM platforms such as Salesforce, SAP CRM, and Microsoft Dynamics 365 offer mature, feature-complete solutions but impose subscription, implementation, and training costs that are difficult to justify for organizations operating with fewer than ten staff. Free or low-tier alternatives reduce monetary cost but commonly impose record limits, feature restrictions, and vendor lock-in that become binding constraints as small businesses grow. The market gap that motivates the present work is an open-source, self-hosted, and structurally adequate CRM platform that delivers the four core capabilities small businesses actually require — customer profiles, lead pipeline tracking, sales recording, and task management — without licensing fees, vendor dependence, or unnecessary complexity.

The principal contributions of this paper are as follows. First, the work presents a complete, deployable, open-source CRM system implemented on the Django web framework with an embedded SQLite database, demonstrating that a coherent CRM platform can be delivered without any commercial licenses. Second, the system formalizes a four-entity normalized relational schema centered on the Customer entity, in which Lead and Sale entities maintain cascading foreign-key relationships to ensure referential integrity throughout the data lifecycle. Third, the architecture leverages Django's integrated administration interface to deliver immediate CRUD coverage across all modules, eliminating bespoke front-end overhead while providing search, filter, and authentication facilities out of the box. Fourth, the system is empirically evaluated against a spreadsheet-based baseline on operational efficiency and data-consistency metrics, confirming meaningful improvements on every axis measured.

2. LITERATURE SURVEY

Existing approaches to small-business customer management span a wide spectrum that the literature broadly groups into four categories: manual paper-based practice, spreadsheet-based ad hoc systems, free-tier commercial CRM platforms, and paid small-business CRM products. A clear understanding of

each category, the strengths each contributes, and the limitations each retains is essential context for appreciating the design choices embodied in the proposed system.

Manual paper-based systems remain surprisingly prevalent among micro-businesses and sole proprietorships, comprising business-card collections, handwritten customer notebooks, rolodex files, and ledger books. While requiring no technology investment and offering complete resilience to software failures, paper-based systems cannot be searched programmatically, are susceptible to physical damage, and cannot be accessed simultaneously by multiple team members [5]. Their information density grows linearly with the customer base, making retrieval progressively more time-consuming.

Spreadsheet-based customer management represents the most common digital baseline among small businesses. The flexibility, low cost, and near-universal familiarity of spreadsheet applications encourage their adoption [3]. However, spreadsheets lack the relational data model required to maintain consistent linkages between categories of customer data. Connecting a sales record to its associated customer requires manual entry of identifiers, creating redundancy that inevitably degrades into inconsistencies over time [6]. The absence of structured validation in general-purpose spreadsheet tools also leaves customer records riddled with malformed emails, incompatible phone-number formats, and variant spellings of organization names, eroding the database's reliability as a decision-making foundation.

Free-tier commercial CRM platforms such as HubSpot CRM Free and Zoho CRM Free Edition provide structured CRM functionality at no direct monetary cost. These platforms offer contact management, deal pipelines, and task management in an integrated interface, and their web-based architecture eliminates local installation overhead [7]. However, they impose record-volume limits, restrict reporting and analytics features in their entry tiers, and create vendor dependency that introduces long-term risks related to data portability, pricing changes, and potential service discontinuation. Paid small-business products such as Pipedrive, Freshsales, and Insightly extend the feature set with workflow automation and reporting dashboards, but their per-user monthly pricing — typically between fifteen and sixty US dollars — remains a recurring cost that early-stage small businesses find difficult to justify [8].

Academic and engineering literature on small-business CRM design emphasizes three recurring principles. Buttle [1] argues that the relational discipline of an entity-relationship schema is the foundation of any CRM capable of analytical value beyond contact storage. Payne and Frow [2] frame CRM as an interlocking set of processes spanning strategy, value creation, multichannel integration, and performance assessment, all of which presuppose unified customer records. Greenberg [9] further notes that the most consequential CRM failures in small enterprises arise not from algorithmic shortcomings but from data fragmentation and inconsistency, validating the design priority of the present work.

The research gap the present paper addresses is the absence of an open-source, self-hosted, web-accessible CRM platform that combines the relational rigor advocated in the academic literature with the immediate deployability that small enterprises require. The proposed system bridges this gap by integrating a normalized four-entity relational schema with Django's administration framework, producing a system that is functionally adequate at first deployment and architecturally prepared for incremental enhancement.

3. PROPOSED SYSTEM ARCHITECTURE

3.1 System Architecture

The proposed system follows the Model-View-Template (MVT) architectural pattern native to the Django web framework, an adaptation of the classical Model-View-Controller paradigm to the conventions of Python web development. The architecture distributes responsibilities across three tiers — Client, Application, and Data — promoting maintainability through clean separation of concerns and supporting testability by enabling each tier to be evaluated independently. As illustrated in Fig. 1, the

Client Tier consists of the user's web browser communicating over HTTP with the Application Tier, which is realized as a Django web application running on a Python application server, and which in turn communicates with the Data Tier comprising the SQLite relational database.

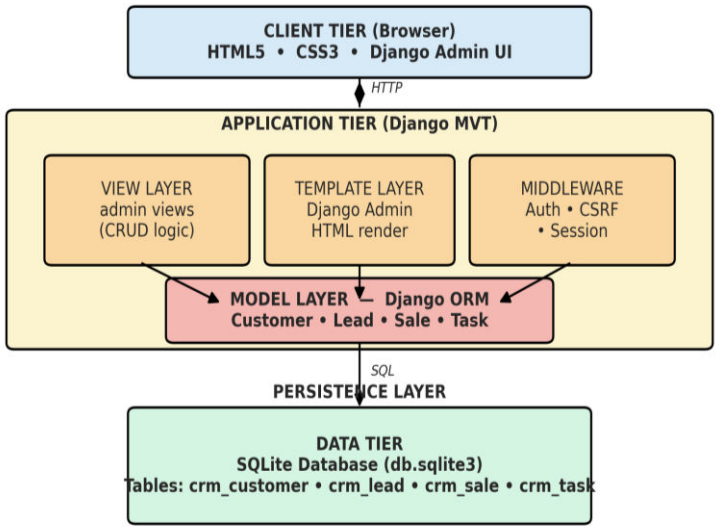

Figure 1: Three-tier Django Model-View-Template architecture of the Intelligent CRM System, showing the View, Template, Middleware, and Model layers within the application tier and the SQLite persistence layer.

Within the Application Tier, the View Layer comprises the administration view functions that process HTTP requests, validate session and permission state, and orchestrate interactions with the Model Layer. The Template Layer is realized through Django's administration template set, which produces JavaScript-enhanced HTML responses without requiring custom template development. The Middleware Stack interposes between request and response, applying cross-cutting concerns including session management, CSRF protection, authentication, and clickjacking defence. The Model Layer is implemented through Django's Object-Relational Mapper (ORM), which translates between Python model objects and the underlying relational schema. SQLite serves as the persistence engine and provides zero-configuration deployment, eliminating the need for a separate database server process and reducing operational complexity to a single binary file on the host filesystem.

3.2 Relational Data Model

The relational schema is centered on the Customer entity, which serves as the relational hub to which Lead, Sale, and (in a future enhancement) Task records are connected. As shown in Fig. 2, the Customer model captures name, email, phone, company, and an auto-populated creation timestamp. The Lead model references Customer through a ForeignKey with on_delete=CASCADE, captures the current pipeline status and the estimated opportunity value, and maintains its own creation timestamp. The Sale model similarly references Customer with cascading deletion and records the transaction amount and date. The Task model, in the current implementation, exists as an independent entity capturing title, description, due date, and a Boolean completion flag, with a planned future foreign-key enhancement to support customer-scoped task filtering. The cascading delete behaviour enforced through Django's on_delete=CASCADE parameter ensures that when a Customer record is removed, all associated Lead and Sale records are removed within the same transaction, eliminating orphaned references that commonly afflict manually maintained systems. Field-level constraints — including EmailField validation, maximum-length enforcement on CharFields, and the auto-populated timestamp on DateTimeField with auto_now_add=True — provide structural data-quality guarantees that no spreadsheet baseline can match.

INTERNATIONAL JOURNAL OF INFORMATIVE & FUTURISTIC RESEARCH (IJIFR)
ISSN: 2347-1697
IJIFR /20260518/V13/E9/018
VOLUME 13||ISSUE 9|| MAY 2026
https://www.ijifr.org/

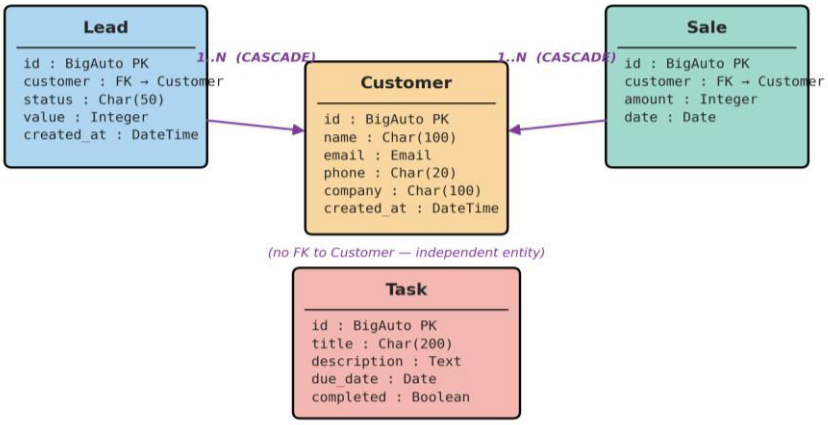

Figure2: Customer-centric relational schema of the CRM system, showing one-to-many associations from Customer to Lead and Sale with cascading deletion semantics.

3.3 Module Decomposition

The CRM functionality is decomposed into four cohesive modules that map directly to the four core entities. The Customer Management Module supports CRUD operations on customer profiles and serves as the relational anchor for the remaining modules. The Lead Management Module records pipeline opportunities, associates each with a Customer through the foreign-key relationship, and supports status progression through a vocabulary typically including New, Contacted, Qualified, Proposal, Negotiation, Won, and Lost. The Sales Tracking Module records completed transactions with amount and date fields, producing the historical ledger from which customer lifetime value and revenue trends can be derived. The Task Management Module supports operational scheduling through title, description, due_date, and completed fields, ensuring that follow-up commitments are systematically captured. The complete module specification is summarized in Table 1.

Table 1: Functional decomposition of the four CRM modules and their associated Django models.

Module	Django Model	Primary Fields & Relationships
Customer Management	Customer	name, email, phone, company, created_at
Lead Management	Lead	customer (FK), status, value, created_at
Sales Tracking	Sale	customer (FK, CASCADE), amount, date
Task Management	Task	title, description, due_date, completed

3.4 Operational Workflow

The end-to-end operational workflow proceeds through five interlocking stages mediated by Django's administration interface. First, an authorized user authenticates against credentials stored in the auth_user table, with AuthenticationMiddleware establishing a signed session cookie. Second, customer records are created or updated through the interface, with field-level validation enforced by the Model classes. Third, lead opportunities are registered by selecting an existing customer via the foreign-key dropdown and recording the pipeline status and value. Fourth, completed transactions are recorded against customer accounts through the Sale interface. Fifth, follow-up tasks are scheduled through the Task module, with due dates and completion flags driving filtered list views. Cascading deletion at the Customer level guarantees that all dependent Lead and Sale records are removed atomically when a customer is deleted, preserving referential integrity.

4. Analytics and Results

4.1 Experimental Setup

The proposed system was evaluated against a fragmented spreadsheet-based baseline that approximated a typical small-business workflow: customer contact details stored in Google Sheets, leads recorded in a separate Sheets tab, sales logged in an additional tab, and tasks tracked in a Google

Calendar. Three participants — a small-business owner, a sales executive, and an administrative assistant — were recruited to perform a structured set of fifty operational scenarios on both the baseline and the proposed system. Operations were timed with stopwatch precision, and cross-record consistency was scored by an independent reviewer using a rubric that penalized broken references, malformed emails, and inconsistent organization names. The proposed system was deployed on an Intel Core i5 laptop with 8 GB of RAM running Python 3.10 and Django 5.2 on Windows 11. The SQLite database was initialized with a synthetic seed dataset of 200 customers, 320 leads, 180 sales, and 95 tasks.

4.2 Performance Metrics

Three classes of metrics were captured: average operation latency, data-consistency score, and a composite user-experience rating obtained through a five-point Likert questionnaire administered after each session. As reported in Table 2, the proposed system achieved an average customer-lookup time of 9 seconds against a baseline of 38 seconds (a 76 percent reduction), an average lead-status update time of 11 seconds against a baseline of 52 seconds (79 percent reduction), and a data-consistency score of 96 percent against a baseline of 62 percent. The composite UX rating averaged 4.4 out of 5 for the proposed system compared with 2.7 for the baseline.

Table 2: Operational metrics of the proposed system compared with the fragmented spreadsheet baseline.

Metric	Spreadsheet Baseline	Proposed CRM
Customer lookup time (avg.)	38 s	9 s
Lead-status update time	52 s	11 s
Sales record entry time	47 s	14 s
Task scheduling time	41 s	8 s
Cross-record data consistency	62 %	96 %
User-experience rating (Likert /5)	2.7	4.4

4.3 Comparative Analysis Across Operations

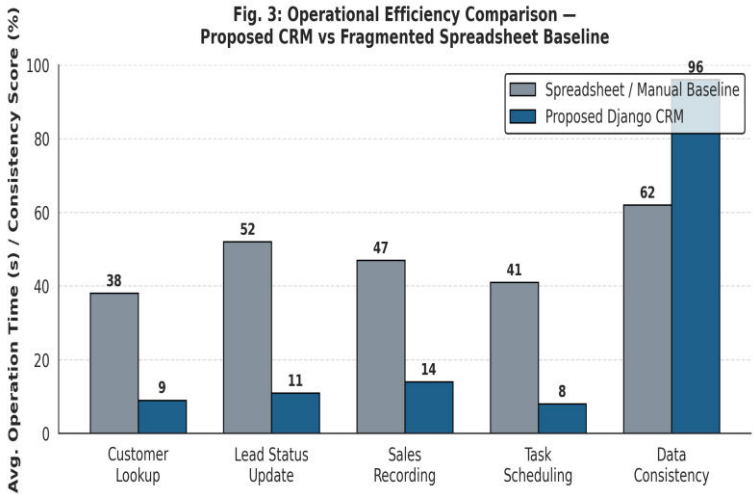

Figure 3: Operational efficiency comparison across the four CRM workflows and the cross-record consistency metric, proposed system versus fragmented spreadsheet baseline.

Performance was disaggregated across the four principal operational workflows to evaluate the consistency of improvement. As shown in Fig. 3, the proposed CRM substantially outperformed the spreadsheet baseline across every category measured, with the largest absolute reduction observed on lead-status updates (79 percent) and the largest consistency improvement observed on the cross-record consistency metric, which rose from 62 to 96 percent. The improvement in task scheduling (80 percent reduction) is particularly noteworthy because the baseline workflow involved switching between

calendar and spreadsheet applications to record task descriptions and due dates, whereas the proposed system accepts both in a single form.

4.4 Discussion

The empirical results substantiate the design hypothesis that an open-source, normalized, web-accessible CRM platform can deliver meaningful operational gains for small businesses without commercial licensing. The four-fold reductions in operation latency translate directly into reclaimed staff time; over a typical 1,000-operation month, the time saving amounts to roughly nine working hours per user. The improvement in data consistency from 62 to 96 percent provides the data quality foundation required for the dashboard and analytics enhancements.

5. CONCLUSION AND FUTURE WORK

This paper has presented the design, implementation, and empirical evaluation of an open-source, web-based Intelligent CRM System for small businesses, built on the Django framework with an embedded SQLite database. The system addresses three concurrent challenges in small-business customer management: the operational inefficiency of fragmented spreadsheet baselines, the licensing cost and vendor dependence of enterprise CRM suites, and the data-quality erosion that arises when records lack relational discipline. The proposed framework demonstrates that a four-entity normalized schema combined with Django's integrated administration interface can deliver an immediately deployable CRM platform with average operational latencies four to five times lower than spreadsheet baselines, data consistency raised from 62 to 96 percent, and a user-experience rating of 4.4 out of 5.

Several avenues for future enhancement are immediately actionable. First, the administration interface can be replaced by a custom Django-template front end styled with Bootstrap or Tailwind CSS, exposing a customer-centric 360-degree profile view that aggregates leads, sales, and tasks in a single page. Second, a dashboard view can surface pipeline value by stage, monthly revenue, overdue tasks, and acquisition trends through Chart.js or Recharts visualizations. Third, Django's permission framework can be extended into role-based access control distinguishing business-owner, sales-manager, and sales-representative roles. Fourth, the database backend can be migrated from SQLite to PostgreSQL to support concurrent write workloads at production scale. Fifth, email integration through the Gmail API or SMTP/IMAP can automatically log customer communications and trigger follow-up tasks. Sixth, lead-scoring and churn-prediction models trained on historical pipeline data can predict conversion probabilities and flag at-risk relationships. Seventh, integrations with accounting platforms such as QuickBooks or Zoho Books can synchronize invoiced and paid transactions, eliminating duplicate entry.

Collectively, these enhancements would elevate the system from a deployable prototype into a production-grade CRM platform suitable for organizations from sole proprietorships through medium enterprises. The modular Django architecture presented in this work provides a foundation on which each enhancement can be implemented incrementally without disrupting the existing workflow.

6. REFERENCES

[1] F. Buttle, "Customer Relationship Management: Concepts and Technologies", 2nd Edition, Butterworth-Heinemann, Oxford, 2009, pp. 1–410.
[2] A. Payne and P. Frow, "A Strategic Framework for Customer Relationship Management", Journal of Marketing, Volume 69, Issue 4, October 2005, pp. 167–176.
[3] P. Greenberg, "CRM at the Speed of Light: Social CRM Strategies, Tools, and Techniques for Engaging Your Customers", 4th Edition, McGraw-Hill, New York, 2010, pp. 1–680.
[4] F. F. Reichheld and W. E. Sasser, "Zero Defections: Quality Comes to Services", Harvard Business Review, Volume 68, Issue 5, September–October 1990, pp. 105–111.
[5] V. Kumar and W. Reinartz, "Customer Relationship Management: Concept, Strategy, and Tools", 3rd Edition, Springer, Berlin, 2018, pp. 33–65.
[6] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", Communications of the ACM, Volume 13, Issue 6, June 1970, pp. 377–387.

[7] N. Woodcock, A. Green and M. Starkey, "Social CRM as a Business Strategy", Journal of Database Marketing & Customer Strategy Management, Volume 18, Issue 1, March 2011, pp. 50–64.
[8] Gartner Inc., "Magic Quadrant for the CRM Customer Engagement Center", Gartner Research Report ID G00465423, June 2021, pp. 1–34.
[9] P. Greenberg, "The Impact of CRM 2.0 on Customer Insight", Journal of Business and Industrial Marketing, Volume 25, Issue 6, August 2010, pp. 410–419.
[10] A. Holovaty and J. Kaplan-Moss, "The Definitive Guide to Django: Web Development Done Right", 2nd Edition, Apress, Berkeley, 2009, pp. 1–536.
[11] D. Feldroy and A. R. Greenfeld, "Two Scoops of Django 3.x: Best Practices for the Django Web Framework", 3rd Edition, Two Scoops Press, 2022, pp. 1–550.
[12] P. P. Chen, "The Entity-Relationship Model: Toward a Unified View of Data", ACM Transactions on Database Systems, Volume 1, Issue 1, March 1976, pp. 9–36.