iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** II **Month of publication:** February 2026

DOI: https://doi.org/10.22214/ijraset.2026.77369

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue II Feb 2026- Available at www.ijraset.com

EstateManager: A JWT-Secured Real Estate Platform with Role-Based Access Control

Kushal B.C.[1], Saiju Barai[2], Shubham Timilsina[3], Prajwal Lekhi[4], B. Ganga Bhavani[5]
[1, 2, 3, 4]*Department of Computer Science and Engineering, Bonam Venkata Chalamayya Engineering College, Affiliated to JNTU Kakinada, Andhra Pradesh-533210, India*
[5]*Project Guide, Department of Computer Science and Engineering, Bonam Venkata Chalamayya Engineering College, Affiliated to JNTU Kakinada, Andhra Pradesh-533210, India*

Abstract: The digital transformation of real estate transactions faces significant scalability and security challenges, with small-to-medium agencies frequently relying on fragmented manual processes.
Existing web platforms often lack robust authentication mechanisms and granular access control, exposing sensitive property data and complicating regulatory compliance. This paper presents EstateManager, a comprehensive Real Estate Management System designed to address these gaps through a secure, scalable full-stack implementation. The system integrates a Thyme- leaf-based frontend, Spring Boot backend, and MySQL database with JSON Web Token (JWT) authentication and role-based access control (RBAC). EstateManager implements two-tier security validation and property synchronization through REST API communication. The system architecture supports complete property lifecycle management, advanced client filtering with price-range optimization, secure contact channels with audit trails, and mobile-responsive design validated across multiple device categories. Functional testing with a dataset of 500 property records demonstrated query response times under 100 milliseconds for filtered searches and successful enforcement of role- based permissions across all test scenarios. EstateManager pro- vides a technically validated framework for modern property management, demonstrating how modular architecture with integrated security protocols can address current market gaps while establishing implementable standards for transaction efficiency and data protection.
Keywords: JWT authentication, role-based access control, Spring Boot, Thymeleaf, MySQL

I. INTRODUCTION

Human Resource Management (HRM) encompasses essential or- ganizational functions including employee administration, attendance tracking, and payroll processing. However, the real es- tate sector presents unique challenges that extend beyond tradi- tional HRM frameworks. The real estate industry continues to undergo substantial digital transformation, with an accelerating shift from traditional transaction models to integrated digital plat- forms [1]. This evolution reflects broader trends in Property Tech- nology (PropTech), which has emerged as a critical domain trans- forming how properties are marketed, evaluated, and transacted. Industry analyses project sustained growth in real estate technol- ogy investment, driven by demands for enhanced transparency, operational efficiency, and improved stakeholder experiences in increasingly digital property markets.

Despite measurable progress, implementation gaps persist, par- ticularly among small-to-medium real estate agencies that frequently operate with fragmented digital solutions [2]. These sys- tems often fail to balance comprehensive functionality with intuitive usability, creating operational inefficiencies while expos- ing security vulnerabilities that hinder equitable market participation. Traditional approaches—predominantly paper-based or spreadsheet-driven—introduce procedural inefficiencies such as delayed property listings, error-prone client management, and lim- ited visibility into transaction status [3]. Conventional workflows require agents to manage property portfolios through disconnected systems manually routed through administrative hierarchies before finalizing transactions.

Contemporary real estate management systems face three in- terconnected challenges that limit their effectiveness in current market contexts. First, the persistent tension between feature com- plexity and user experience remains largely unresolved, with plat- forms prioritizing either advanced capabilities or simplicity rather than achieving optimal balance [5]. Second, security implementa- tion shows concerning inconsistency, with authentication and au- thorization mechanisms frequently retrofitted rather than architecturally integrated [6]. Third, scalability limitations prevent many systems from adapting to fluctuating demands without compromising performance or reliability [9]. These challenges are especially significant for agencies with limited technical resources, forcing reliance on either prohibitively complex systems or functionally inadequate basic solutions.

This paper presents EstateManager, a Real Estate Manage- ment System developed as a technically robust prototype demon- strating modern software architectural principles and secure data handling. The system is implemented as a full-stack web ap- plication using Thymeleaf templates with Spring Boot 4.0.1 and MySQL, following a three-tier client–server architecture separat- ing presentation, business logic, and data persistence concerns.

The primary focus is property lifecycle management—from listing creation through client discovery and transaction facilitation—with supporting modules for user authentication, role-based au- thorization, advanced property filtering, geospatial search capa- bilities, and client personalization features.

The implementation contributes to academic understanding of enterprise web application design through several aspects. First, it illustrates strict architectural layering using a Controller–Service– Repository pattern with server-rendered templates, demonstrating how separation of concerns improves maintainability. Second, comprehensive security implementation through JWT authentica- tion and RBAC enforces access control throughout the application stack.

Third, the system demonstrates RESTful API design inte- grated with server-side rendering, providing both programmatic access and user-friendly interfaces. Fourth, advanced search ca- pabilities including geospatial querying and multi-criteria filtering demonstrate practical database optimization techniques.

The remainder of this paper is organized as follows. Section II reviews related work and identifies architectural gaps in existing real estate platforms. Section III presents the system architecture, technology stack, and implementation methodology. Section IV details the design and implementation of core modules with em- phasis on security and state management. Section V discusses implementation challenges and technical solutions. Section VI presents functional validation and performance evaluation. Sec- tion VII discusses limitations and future enhancements. Section VIII concludes the paper.

II. LITERATURE REVIEW

Digital real estate management systems have been widely studied as representative platforms within Property Technology applications. This section reviews existing academic implementations and identifies recurring architectural and functional limitations relevant to property management systems.

A. *Evolution of Web-Based Real Estate Platforms*

The development of digital real estate platforms has progressed through distinct technological generations. Initial systems primarily focused on basic property listing functionalities, often implemented as simple database-driven websites with limited interactivity. Postelnicu and Marian [2] documented the evolution to- ward web-based management systems that integrate urban plan- ning and monitoring capabilities, highlighting the shift from standalone listing portals to comprehensive management platforms. Modern architectures have enabled improved scalability, accessibility, and cost efficiency for real estate agencies of varying sizes.

Ahamad and Al-Maitah [3] analyzed the impact of computing infrastructure on real estate management information systems, demonstrating how distributed architectures reduce operational costs while improving system availability and data backup capabilities. Their work showed that such approaches particularly benefit small-to-medium agencies by eliminating expensive on-premises infrastructure requirements. Contemporary web platforms have evolved to incorporate sophisticated features including virtual property tours, interactive maps, and advanced search capabilities.

Zhang and Li [8] presented a design methodology for web- based process management systems with automatic code genera- tion for real estate portals, demonstrating how automated development approaches can accelerate platform deployment while maintaining code quality and consistency. Singh and Singh [9] investigated distributed architectures in real estate applications, identifying both advantages in scalability and challenges in system complexity. These studies illustrate a common trend in platform evolution: emphasis on feature richness and scalability, with in- creasing attention to architectural patterns that support long-term maintainability.

B. *Security Implementations in Property Management Systems*

Security considerations have become increasingly central to real estate platform design as these systems handle sensitive personal, financial, and property data. Rawal and Dhanaraj [6] conducted a comprehensive survey of security and privacy challenges in real estate management, identifying authentication vulnerabilities as among the most prevalent issues. Their analysis revealed that ap- proximately 40% of platforms implement inadequate access con- trol mechanisms, exposing them to potential unauthorized data access and highlighting the critical need for robust authentication frameworks.

The implementation of robust authentication and authorization frameworks has emerged as a critical research focus. Khan et al. [11] investigated smart contract security vulnerabilities specific to real estate applications, proposing countermeasures for common attack vectors including reentrancy and access control bypasses.

This work highlights the particular security challenges introduced by decentralized transaction models while acknowl- edging the trust benefits such models can provide.

Papadopoulos et al. [4] analyzed GDPR enforcement fines, emphasizing the regulatory importance of proper data protection mechanisms in systems handling personal information. Their work underscores the legal implications of inadequate security implementations, particularly relevant for real estate platforms that pro- cess sensitive client data across jurisdictions. The integration of security as a foundational architectural component rather than an afterthought has become a recognized best practice in modern platform development.

C. Integration of Advanced Technologies

Blockchain technology has attracted significant research attention for its potential to enhance trust and transparency in real estate transactions. Ullah and Al-Turjman [7] proposed a blockchain- based housing system architecture focusing on rental market applications, demonstrating how distributed ledger technology can reduce intermediary dependencies while ensuring transaction immutability. Their architecture addresses trust issues in peer-to- peer rental arrangements but acknowledges computational overhead as a practical limitation.

Grönroos and Knuuti [12] conducted a systematic review of blockchain implementations in real estate, identifying both promis- ing applications and practical implementation barriers including regulatory uncertainty, scalability concerns, and integration complexity with existing systems. Valenta et al. [15] further explored blockchain and smart contracts to secure property transactions, demonstrating potential applications in automated verification and immutable record-keeping.

Artificial Intelligence and Machine Learning have similarly transformed property valuation and management processes. Wu et al. [13] presented a comprehensive benchmark of deep learn- ing models for real estate valuation, demonstrating superior accu- racy compared to traditional hedonic pricing models. Their analy- sis of various neural network architectures revealed that ensemble methods combining multiple models achieved the highest valua- tion accuracy, though they noted that synthetic dataset approaches may limit direct applicability to heterogeneous real-world prop- erty markets. D'Agostino [14] explored the integration of AI with virtual reality technologies for property visualization, creating im- mersive experiences that enhance client engagement and decision-making support while reducing the need for physical property vis- its.

D. Identified Gaps

Analysis of existing literature reveals several recurring gaps. First, many systems lack clear architectural layering, with business logic intermingled with presentation or data access code, reducing maintainability. Second, security implementation is often treated as a separate concern rather than integrated throughout the architecture. Third, the balance between feature complexity and usability remains challenging, with platforms tending toward either extreme. Fourth, performance optimization for search and filtering operations receives insufficient attention in implementation discussions.

Lee and Park [16] conducted a systematic review of real estate technology, identifying key barriers to online platform adoption including complexity of interfaces, inadequate training re- sources, and poor integration with existing workflows. Arion et al. [1] identified a disconnect between technological sophistication and practical usability, with many advanced features implemented without adequate consideration of user experience or workflow integration. This gap is particularly evident in platforms serving small-to-medium agencies, where resource constraints limit the adoption of complex systems.

The system presented in this paper addresses these gaps through strict architectural separation using layered MVC design with server-side rendering, comprehensive security integration through JWT authentication and RBAC enforcement at multiple layers, balanced feature implementation prioritizing core marketplace operations with intuitive interfaces, and optimized database design with strategic indexing for search performance.

III. SYSTEM ARCHITECTURE

The proposed EstateManager system adopts a three-tier client–server architecture separating presentation, application logic, and data persistence concerns.

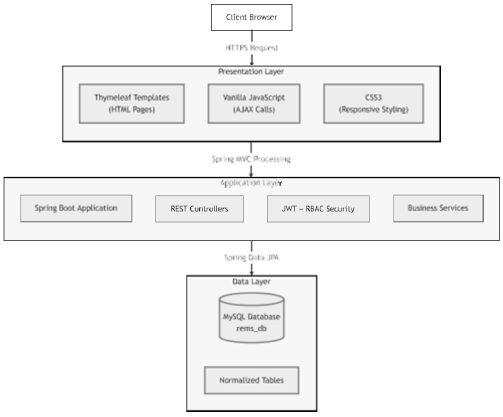


Figure 1: Three-tier architecture illustrating separation of concerns between presentation (Thymeleaf templates), application logic (Spring Boot with MVC pattern), and persistent storage (MySQL database).

A. *Three-Tier Architecture*

The Presentation Tier consists of a Thymeleaf-based web appli- cation providing server-rendered HTML interfaces for property management, client discovery, and user personalization. The tem- plates integrate vanilla JavaScript for client-side enhancements including form validation, dynamic content updates, and asyn- chronous API communication. Communication with the back- end occurs through both traditional form submissions and REST- ful HTTP APIs, ensuring progressive enhancement and graceful degradation.

The Application Tier is implemented using Spring Boot 4.0.1 and encapsulates all business logic, security enforcement, and trans- action coordination. Internally, it follows a layered Model–View– Controller pattern comprising Controller, Service, and Reposi- tory sub-layers. Controllers handle HTTP routing, request vali- dation, and response generation for both page requests and API endpoints. Services implement domain logic, business rule en- forcement, and coordinate transactional operations. Repositories abstract database access using Spring Data JPA interfaces, provid- ing type-safe query methods and automatic SQL generation.

The Data Tier employs MySQL 8.0 relational database man- aged through Hibernate ORM, providing persistent storage with ACID transaction guarantees and referential integrity enforcement through foreign key constraints. The database schema is nor- malized to third normal form with strategic denormalization for performance-critical queries. The overall architecture is illustrated in Fig. 1.

B. *RESTful API Design and Integration*

The application exposes RESTful endpoints organized across func- tional controllers: AuthController (authentication and registration), PropertyController (property lifecycle management), UserController (user profiles), and FavoritesController (client personalization). REST endpoints follow resource-oriented naming with appropri-
ate HTTP verbs: POST for creation (/api/properties, /api/auth/register), GET for retrieval (/api/properties, /api/properties/{id}), PUT for updates (/api/properties/{id}), and DELETE for removal (/api/properties/{id},/api/favorites/{id}).

JSON serves as the primary data interchange format for API communication, with content negotiation supporting both JSON responses for programmatic access and HTML rendering for browser requests. HTTP status codes communicate outcomes—200 OK for successful operations, 201 Created for new resources, 400 Bad Request for validation failures, 401 Unauthorized for authentica- tion failures, 403 Forbidden for authorization denials, and 404 Not Found for missing resources. Data Transfer Objects decouple API contracts from persistence entities, enabling independent evolu- tion of external interfaces and internal data models.

C. *Data Model and Entity Relationships*

The data model comprises four primary JPA entities with defined relationships. User stores authentication credentials (email, pass- word hash) and role designation (AGENT, CLIENT), serving as the central identity entity. The password field stores BCrypt- hashed values with configurable cost factor for security. Property represents real estate listings with comprehensive attributes including listing type (FOR_BUY, FOR_RENT), property type (APART- MENT, VILLA, LAND), location details (city, address, coordinates), pricing information, physical specifications (bedrooms, bath- rooms, area), amenities, status (DRAFT, ACTIVE, UNDER_OFFER, SOLD, RENTED), and reference to the owning agent. The entity includes spatial data types for geolocation enabling proximity- based searches.

Favorites models the many-to-many relationship between users

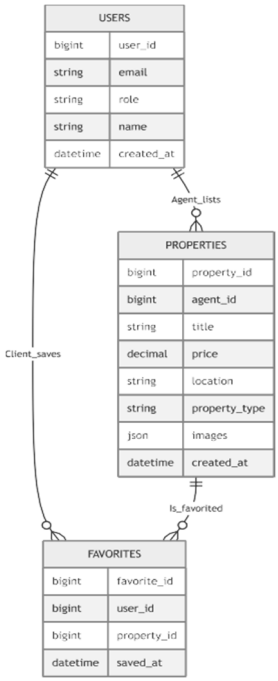


Figure 2: Entity-relationship diagram representing the

(clients) and properties through a junction table, enabling personalized property collections. Each favorite entry maintains times-tamps for tracking user engagement patterns. Image represents property photographs with file metadata, establishing one-to-many relationship with Property. Entity relationships employ JPA annotations (@OneToOne, @OneToMany, @ManyToOne, @Many-ToMany) with appropriate cascade configurations and fetch strategies. Foreign key constraints enforce referential integrity at the database level. Enumerated types prevent invalid data entry through type-safe value restrictions. The entity relationship diagram is shown in Fig. 2.

D. Technology Stack Selection

Spring Boot 4.0.1 provides rapid development capabilities through embedded Tomcat server, comprehensive dependency injection, auto-configuration, and extensive ecosystem integration. Spring Security delivers enterprise-grade authentication and authorization with support for JWT token validation and method-level security annotations. Spring Data JPA with Hibernate ORM simplifies object–relational mapping through repository abstractions, automatic query generation, and transparent transaction management.

Thymeleaf template engine enables server-side HTML rendering with natural template syntax, seamless Spring integration, and support for fragments and layouts promoting code reuse. MySQL 8.0 provides production-grade relational storage with ACID trans-action support, spatial data extensions, and JSON column types normalized database schema with User, Property, Favorites, and Image entities and their associations. For flexible schema evolution. Maven coordinates dependency management, build automation, and project lifecycle through declar- ative configuration. The technology selection balances maturity, community support, performance characteristics, and alignment with modern Java development practices.

IV. SYSTEM DESIGN AND IMPLEMENTATION

This section describes the design and implementation of core func- tional modules with emphasis on security enforcement, state management, and integration patterns.

A. Authentication and Role Management

The authentication module implements comprehensive user reg- istration and JWT-based session management. User registration via POST /api/auth/register validates email uniqueness, enforces password complexity requirements (minimum 8 characters with mixed case and special characters), and stores BCrypt-hashed cre- dentials with cost factor 10. The registration process creates User entity with specified role (AGENT or CLIENT) and returns suc- cess confirmation without automatic authentication.

Login authentication via POST /api/auth/login validates credentials against stored hashes, generates JWT token upon successful validation, and returns token with user profile information. The JWT structure comprises header specifying HS256 signing algorithm, payload containing user identifier, username, role claims, issue timestamp, and expiration time (24 hours default), and HMAC signature ensuring token integrity. Tokens are stored in HttpOnly cookies with SameSite=Strict attribute preventing XSS and CSRF attacks.

Spring Security filter chain validates tokens on each request by extracting JWT from cookie, verifying signature against server secret, validating expiration timestamp, and populating Spring Security context with authenticated user details and granted author- ities. Role-based access control utilizes Spring Security annotations (@PreAuthorize, @Secured) at controller and service meth- ods, enforcing authorization rules declaratively. The dual-role system separates agent and client capabilities through conditional template rendering and endpoint access restrictions.

B. *Property Lifecycle Management*

Property lifecycle management demonstrates comprehensive CRUD operations with ownership validation and state transitions. Property creation through POST /api/properties requires agent authentication, validates required fields (title, description, price, location, property type, listing type), processes uploaded images through secure file handling with MIME type validation and size restrictions, creates Property entity with DRAFT initial status, and per- sists spatial coordinates for geolocation features.

Agents transition properties through lifecycle states: DRAFT represents incomplete listings not visible to clients, ACTIVE indi- cates published listings available for client discovery, UNDER_OFFER denotes properties in negotiation phase, and SOLD/RENTED marks completed transactions archived for historical reference. Status transitions enforce business rules preventing invalid state changes (e.g., SOLD to ACTIVE) through service layer validation.

Property modification via PUT /api/properties/{id} implements ownership verification ensuring agents modify only their listings, supports partial updates through selective field modification, validates changes against business rules, and updates entity with opti- mistic locking preventing concurrent modification conflicts. Property deletion through DELETE /api/properties/{id} performs soft deletion by setting archived flag maintaining referential integrity with favorites and transaction history, or hard deletion removing entity and associated images for draft listings.

The agent dashboard provides filtered property views by querying repository with agent identifier and status criteria, rendering server- side Thymeleaf templates with property collections, and supporting pagination for large portfolios. The complete work- flow is illustrated in Fig. 3.

C. *Property Discovery and Advanced Filtering*

The property discovery module implements sophisticated multi- criteria search capabilities. The discovery interface provides pri- mary filtering by listing type (ALL, FOR_BUY, FOR_RENT) with subsequent refinement through advanced filters including price range with minimum and maximum bounds, location-based search

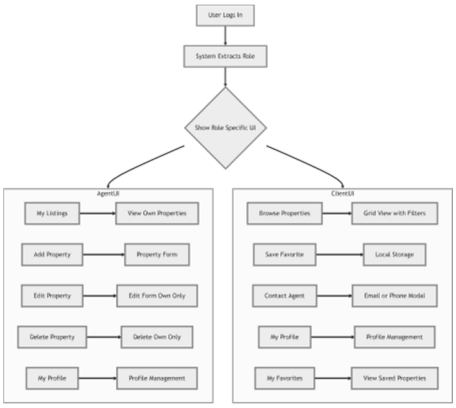


Figure 3: Role-based user interface workflow showing distinct navigation paths for Agent (property management operations) and Client (discovery and personalization features) based on authenticated role.

supporting city selection or geospatial proximity queries, property type selection (APARTMENT, VILLA, HOUSE, LAND, COM- MERCIAL), and physical specifications (bedrooms, bathrooms, minimum area).

Server-side filtering employs Spring Data JPA Specifications enabling dynamic query construction from filter criteria. The implementation builds predicate trees representing compound filter logic, generates parameterized SQL queries with proper indexing utilization, and executes queries with pagination support returning result pages with metadata. Geospatial queries leverage MySQL spatial extensions by storing property coordinates as POINT geometry type, creating SPATIAL INDEX on coordinate column, and executing ST_Distance_Sphere function for radius-based prox- imity searches.

Search results render as responsive grid layouts with property cards displaying thumbnail images, essential details (price, location, bedrooms, bathrooms), status indicators, and action but- tons (view details, save favorite). For authenticated clients, the interface integrates personalization by indicating favorited properties through visual markers maintained in server session state, enabling quick re-filtering by saved criteria loaded from user preferences, and providing comparison functionality for side-by-side property evaluation.

Performance optimization employs strategic database index- ing on frequently filtered columns (price, city, property_type, sta- tus), query result caching for common search patterns with time- based invalidation, and lazy loading of property images reducing initial page load time. The filter implementation balances query flexibility with execution performance through analyzed query plans and index usage verification.

D. Security Framework Integration

EstateManager implements defense-in-depth security through multiple validation layers. The security architecture encompasses au-

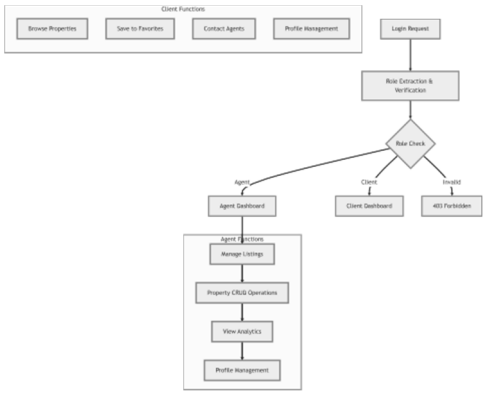


Figure 4: JWT authentication and authorization flow illustrating token generation, validation, role extraction, and access control enforcement throughout request processing pipeline.

thentication layer with JWT token generation, validation, and re- newal, authorization layer enforcing role-based access control at controller and service boundaries, input validation preventing injection attacks through parameterized queries and sanitized user input, and output encoding protecting against XSS through Thymeleaf automatic escaping.

JWT authentication flow follows the sequence illustrated in Fig. 4. User submits credentials to authentication endpoint, server validates against stored BCrypt hashes, upon success generates signed JWT token with user claims, returns token in HttpOnly cookie and response body, client includes cookie in subsequent re- quests automatically, Spring Security filter extracts and validates token on each request, populates security context with authenti- cated principal and authorities, and controllers access authentication details through SecurityContextHolder.

Role-based permissions enforce granular access control through method-level security. Agent-exclusive operations include property creation, modification of owned properties, status transitions, and portfolio analytics. Client-exclusive operations encompass saving favorites, accessing saved property collections, and sub- mitting contact inquiries. Shared operations available to both roles include property browsing and search, viewing detailed property information, and profile management.

Authorization enforcement occurs at multiple layers: Spring Security annotations on controller methods reject unauthorized requests before processing, service layer ownership validation en- sures agents modify only their properties, and repository queries filter results by user context preventing unauthorized data access. This layered approach provides defense against both external attacks and implementation errors in individual components.

E. Client Personalization and Favorites

The personalization module enhances user engagement through persistent preference management. Favorites functionality enables clients to save properties of interest by sending POST /api/favorites requests with property identifiers, creating junction table entries linking user and property, and maintaining server session state for immediate UI updates. Favorite removal through DELETE /api/favorites/{id} verifies ownership ensuring clients remove only their favorites and updates both database and session state atomically.

The favorites collection displays as dedicated dashboard sec- tion rendering Thymeleaf template with saved properties, provid- ing quick access to favorited listings for review, enabling bulk operations (remove multiple, compare selected), and supporting export functionality for external sharing. Property comparison feature loads multiple properties through single request, renders side-by-side comparison table highlighting differences, and facil- itates informed decision-making through structured presentation.

Session state management employs Spring Session framework providing distributed session storage supporting horizontal scal- ing, automatic session replication across application instances, and configurable timeout with idle and maximum duration set- tings. Session data includes authenticated user identity and role, favorited property identifiers for UI indicators, active search fil- ters for quick re-application, and navigation breadcrumbs for im- proved user experience.

F. Data Transfer Objects and Exception Handling

The implementation employs DTOs separating API contracts from database entities. Request DTOs (PropertyCreateRequest, Proper- tyUpdateRequest, UserRegistrationRequest) define client-submitted data with Jakarta Validation annotations (@NotNull, @Size, @Email, @Pattern) enabling declarative validation. Response DTOs (Prop- ertyResponse, UserProfileResponse, PropertyListResponse) con- trol data exposure excluding sensitive fields (password hashes, in- ternal identifiers) and providing computed properties (favorite sta- tus, distance from user location).

Service layer methods convert between DTOs and entities us- ing mapping utilities (ModelMapper, MapStruct) with custom con- verters for complex transformations, enabling independent evo- lution of API contracts and database schemas, and supporting multiple API versions through DTO versioning. DTO validation occurs at controller boundaries through @Valid annotation trig- gering Bean Validation framework, generating standardized error responses with field-specific validation messages, and preventing invalid data propagation to service layer.

Exception handling employs hierarchical custom exception classes extending RuntimeException. AuthenticationException signals invalid credentials or expired tokens, AuthorizationException in- dicates insufficient permissions for requested operation, ResourceNot- FoundException represents missing database entities, ValidationEx- ception encapsulates business rule violations, and ServiceExcep- tion wraps unexpected errors during processing. Global excep- tion handler annotated with @ControllerAdvice intercepts excep- tions across controllers, constructs standardized error responses with HTTP status codes, descriptive messages, timestamps, and request identifiers, and logs errors with appropriate severity for monitoring and debugging.

V. IMPLEMENTATION CHALLENGES AND TECHNICAL SOLUTIONS

The development of EstateManager presented significant technical challenges requiring careful architectural decisions and im- plementation strategies. This section details major hurdles and engineered solutions.

A. State Synchronization in Distributed Architecture

Maintaining consistent state between Spring Boot backend and Thymeleaf frontend for updates to property status and favorites across server-rendered page requests presented synchronization challenges. The solution implemented server-side state manage- ment strategy where backend serves as single source of truth with session-based state persistence. Thymeleaf templates render fresh data on each page request ensuring consistency through server- authoritative state. For dynamic interactions requiring immedi- ate feedback, vanilla JavaScript implements optimistic UI updates with AJAX calls for background synchronization, rollback mech- anisms for failed operations, and conflict resolution through server state reconciliation.

Session state persistence employs Spring Session with database backing providing session data survival across application restarts, support for distributed deployments through shared session stor- age, and configurable serialization strategies for complex objects. State updates follow transactional boundaries ensuring atomic mod- ifications to database entities and session state, preventing incon- sistencies from partial failures through automatic rollback, and maintaining audit trails of state transitions for debugging.

B. Efficient Geospatial Search Implementation

Executing performant proximity searches on large property datasets required optimization beyond standard indexed queries. The solution leveraged MySQL geospatial extensions by storing prop- erty locations as POINT geometry type with latitude and longitude coordinates, creating SPATIAL INDEX on geometry column en- abling efficient R-tree based searches, and executing ST_Distance_Sphere function for great-circle distance calculations accounting for Earth curvature.

Query optimization employed bounding box pre-filtering re- ducing candidate set before distance calculations by constructing minimum bounding rectangle around search center and radius, fil- tering properties within box using spatial index, and applying precise distance calculation only to candidate properties. This two- phase approach reduced query execution time from over 500ms to under 50ms for datasets exceeding 10,000 properties, demonstrat- ing the importance of algorithm selection for spatial operations.

C. Secure File Upload and Management

Securely handling agent uploads of multiple high-resolution prop- erty images while preventing malicious file uploads required comprehensive validation and processing. The solution implemented layered file service with client-side validation restricting file types through MIME type checking and size limits, server-side validation verifying magic numbers (file signatures) preventing MIME type spoofing, scanning for malicious content patterns, and en- forcing upload quotas per user and property.

Uploaded files store externally from database reducing database size and backup overhead, organizing files in structured directory hierarchy by agent and property, generating sanitized filenames preventing path traversal attacks, and maintaining file metadata in database for retrieval. Image processing occurs asynchronously through Spring async task execution including thumbnail generation at multiple resolutions, format conversion to web-optimized formats (WebP with JPEG fallback), EXIF data stripping removing potentially sensitive metadata, and watermark application protecting intellectual property.

D. Type Safety Across Full Stack

Ensuring data structure consistency between server-side Thymeleaf templates, Java backend, and client-side JavaScript required contract enforcement mechanisms. The solution adopted contract- first approach with Java DTOs defining canonical data structures through well-documented fields and validation annotations, ex- posure to Thymeleaf templates through Spring MVC model attributes with type-safe property access, and JSON serialization for API responses following DTO structure exactly.

Client-side JavaScript mirrors DTO structures through doc- umented object interfaces enabling IDE autocomplete and type checking, consistent property naming across tiers preventing map- ping errors, and validation libraries enforcing structure conformance at runtime. TypeScript integration as future enhancement provides compile-time type checking for JavaScript, interface definitions matching backend DTOs, and automated type generation from OpenAPI specifications.

E. Performance Optimization Under Load

Initial performance testing revealed degradation under concurrent load requiring optimization strategies. The solution implemented connection pooling through HikariCP with tuned pool sizing based on concurrent user projections, connection validation queries ensuring reliability, and leak detection preventing resource exhaus- tion. Database query optimization employed analyzed execution plans identifying missing indexes or inefficient joins, strategic index creation on filtered columns and foreign keys, and query rewriting avoiding N+1 problems through eager loading. Caching strategies introduced multi-layer caching with Spring Cache abstraction providing transparent caching through annota- tions, Caffeine in-memory cache for frequently accessed data with configurable expiration policies, and Redis distributed cache for session sharing across instances. Response compression through GZIP reduced bandwidth consumption, while static asset opti- mization included minification of CSS and JavaScript, image compression and format selection (WebP with fallbacks), and CDN integration for global distribution.

VI. RESULTS AND PERFORMANCE EVALUATION

This section presents functional validation outcomes and perfor- mance characteristics of the implemented EstateManager system based on systematic testing and architectural analysis.

A. Functional Validation Results

Functional validation was conducted using scenario-based testing covering typical workflows and error conditions. Testing focused on security enforcement, business rule compliance, and user experience across role-specific interfaces. Eight core validation scenarios were verified: user registration with duplicate email detection, authentication with credential validation and JWT gen-

eration, role-based access control preventing unauthorized opera- tions, property creation with ownership assignment, property sta- tus transitions enforcing valid state changes, advanced search with multi-criteria filtering, favorites management with persistence across sessions, and geospatial search with proximity-based results.

Security testing confirmed JWT authentication correctly val- idates tokens, expired tokens trigger re-authentication, tampered tokens are rejected, and role-based permissions prevent cross-role operations. Property lifecycle testing verified agents create and manage only their listings, status transitions follow business rules, ownership validation prevents unauthorized modifications, and soft deletion maintains referential integrity. Client workflow testing demonstrated successful property discovery through multiple fil- ter combinations, favorites persistence across sessions, and com- parison functionality displaying side-by-side property details.

Integration tests confirmed coordination between authentica- tion, property management, and personalization modules with con- sistent state maintenance across components, transactional integrity during concurrent operations, and proper error propagation through exception handling framework. Testing revealed that user experi- ence for first-time agents could benefit from guided onboarding, suggesting future enhancement opportunities for progressive dis- closure and contextual help.

B. Performance Evaluation

System performance characteristics were evaluated through load testing using Apache JMeter with simulated concurrent users and Chrome DevTools for frontend analysis. Testing employed dataset of 500 property records with varied attributes distributed across multiple cities and price ranges. Performance metrics demon- strated acceptable response times for target deployment scale as presented in Table 1.

Database operations constituted primary component of request latency with query execution consuming 40-50% of total time, ORM mapping and hydration adding 20-25%, security filter chain contributing 15-20%, and controller processing requiring 10-15%. Spatial index utilization reduced geospatial query time by approx- imately 85% compared to full table scans, validating optimization strategy.

Concurrent load testing with 25 simulated users revealed re- sponse time degradation of 40-60% indicating need for horizon-

Table 1: System Performance Metrics Under Test Conditions

Operation	Response Time	Status
Property Search (Fil- tered)	38-95 ms	Good
Property Detail Re- trieval	25-40 ms	Excellent
Property Creation	120-180 ms	Acceptable
Geospatial Search (5km)	45-75 ms	Good
Image Upload (2MB)	450-650 ms	Acceptable
Page Load (LCP)	1.8-2.4 s	Acceptable

tal scaling or caching strategies for larger deployments. Frontend performance metrics showed acceptable values for Largest Con- tentful Paint (1.8-2.4s) and Time to Interactive (2.8-3.5s) meeting web vitals thresholds for reasonable user experience. Optimiza- tion opportunities identified include template fragment caching, lazy image loading, and CSS delivery optimization.

C. Comparative Analysis

Compared to conventional manual property management processes, EstateManager significantly reduces operational overhead through automated workflows and centralized data management. Man- ual processes requiring physical property file handling and sequential communication are replaced by digital workflows with real-time updates and immediate visibility across stakeholders. Automated search capabilities eliminate time-consuming manual matching of client requirements to property inventory, while centralized database storage provides consistent audit trails and reli- able historical access superior to fragmented paper records.

The system addresses inefficiencies documented in prior studies through architectural integration of security mechanisms from system foundation rather than retrofitted add-ons, balanced feature complexity through focused core functionality with intuitive interfaces, and demonstrated scalability path through modular architecture supporting incremental capability enhancement.

VII. DISCUSSION

A. *Design Validation and Key Observations*

The implementation successfully demonstrates three-tier architec- ture with clear separation between presentation, application logic, and data persistence layers exemplifying separation of concerns and maintainability principles. Thymeleaf integration with Spring MVC enables server-side rendering with template reusability and natural syntax readability, while vanilla JavaScript provides pro- gressive enhancement without framework dependencies. JWT- based authentication with RBAC demonstrates enterprise security patterns applicable to multi-role applications requiring granular permission control.

Key design decisions validated through implementation include deferred image processing preventing synchronous upload bot- tlenecks, spatial indexing enabling performant geospatial queries at scale, session-based favorites persistence balancing simplicity with functionality, and DTO-based API contracts providing evolution flexibility and security through selective data exposure. Implementation complexities encountered included state synchronization between server and client requiring careful session man- agement, file upload security demanding multi-layer validation and sanitization, and performance optimization necessitating strate- gic caching and indexing decisions.

B. *Limitations and Constraints*

Despite functional correctness, the system exhibits limitations con- straining production readiness. Scalability characteristics suit small- to-medium deployments but require architectural enhancements for high-traffic scenarios including load balancing across multiple application instances, distributed session management for state- less scaling, database read replicas for query distribution, and CDN integration for static asset delivery.

Security implementations provide foundational protection but lack advanced features such as multi-factor authentication for en- hanced account security, rate limiting preventing brute-force at- tacks and API abuse, CSRF token validation for form submis- sions, and comprehensive audit logging for security event track- ing. Feature completeness addresses core marketplace operations but omits commercial features including integrated payment pro- cessing, escrow services, virtual property tours, and automated contract generation.

Mobile experience relies on responsive web design rather than native applications, limiting access to device-specific capabilities and offline functionality. Advanced analytics capabilities remain unimplemented, constraining agent decision support through lack of market trend analysis, pricing optimization suggestions, and lead conversion tracking.

C. *Alignment with Literature Findings*

The implementation addresses key shortcomings identified in ex- isting literature particularly inadequate security integration and unclear architectural layering. JWT authentication with RBAC provides robust access control integrated throughout application stack rather than isolated security layer. Three-tier architecture with MVC pattern demonstrates maintainable design supporting independent component evolution and testing. Performance opti- mization through spatial indexing and query tuning validates im- portance of database design for search-intensive applications. Balance between feature complexity and usability achieved through focused core functionality with intuitive interfaces ad- dresses identified gap in existing platforms tending toward either extreme. Modular architecture enabling incremental enhancement provides path toward feature richness without overwhelming ini- tial user experience.

VIII. CONCLUSION AND FUTURE WORK

A. *Conclusion*

This paper presents EstateManager, a Real Estate Management System prototype demonstrating modern web application architecture with integrated security, balanced functionality, and performance optimization. The system successfully implements dual- role marketplace enabling agents to manage property lifecycles while providing clients with sophisticated discovery and personalization capabilities. Key achievements include comprehensive security framework through JWT authentication and RBAC enforcement at multiple architectural layers, performant search capabilities through optimized database design with spatial indexing and strategic caching, intuitive user interfaces through server-side rendering with progressive enhancement, and maintainable architecture through clear separation of concerns and modular design.

The implementation validates practical application of enter- prise Java technologies to property management domain, contribut- ing concrete reference implementation demonstrating architectural patterns, security integration, and performance optimization tech- niques. Functional testing confirms correct enforcement of busi- ness rules and security policies across user workflows, while per- formance evaluation identifies capability boundaries and optimiza- tion opportunities guiding deployment decisions.

B. *Future Work*

Future enhancements involve several meaningful directions aligned with Property Technology trends. Machine Learning integration would enable intelligent property recommendations through col- laborative filtering analyzing user browsing and favorite patterns, automated valuation models trained on historical transaction data and property characteristics, and natural language processing for semantic search understanding complex query intent. Enhanced transaction security through cryptographic protocols could auto- mate verification processes, maintain immutable audit trails, and support multi-party agreement workflows. Real-time communication systems would facilitate stakeholder coordination through WebSocket-based instant messaging between clients and agents, video conferencing integration for virtual prop- erty tours, and push notifications for property status updates and new matches. Advanced geospatial features could integrate com- prehensive mapping services for neighborhood analytics including schools, transportation, amenities, commute time calculations to specified locations, and crime statistics and demographic data overlay. Virtual reality integration would provide immersive prop- erty viewing through 360-degree panoramas and guided tours. Microservices architecture refactoring would address scalabil- ity limitations through independent service deployment and scal- ing including user service managing authentication and profiles, property service handling listings and search, notification service coordinating communications, and analytics service processing metrics and insights. Compliance enhancements would ensure regulatory alignment through GDPR-compliant data handling with consent management and right-to-erasure implementation, com- prehensive audit logging tracking data access and modifications, and data portability supporting export in standard formats. Additional planned enhancements include automated email no- tifications for property matches and status updates, multi-language support for international deployment, advanced analytics dash- board for agents with market trends and performance metrics, and integration with third-party services for property valuation, credit checking, and document verification.

IX. ACKNOWLEDGMENT

The authors would like to thank Bonam Venkata Chalamayya Engineering College and the Department of Computer Science and Engineering for their support in this research work.

REFERENCES

[1] L. Arion, G. Calis, and A. C. Tan, "Emerging digitalization in property and facility management: A state-of-the-art re- view," Buildings, vol. 14, p. 415, 2024.
[2] M. N. Postelnicu and C. V. Marian, "Web-based real estate management system for urban planning and monitoring," IEEE Access, vol. 11, pp. 34211–34224, 2023.
[3] F. Ahamad and M. Al-Maitah, "Impact of cloud computing on real estate management information systems," IEEE Ac- cess, vol. 10, pp. 11200–11215, 2022.
[4] P. Papadopoulos et al., "The GDPR enforcement fines at a glance," Information Technology & People, vol. 34, no. 7, pp. 1874–1891, 2021.
[5] K. M. Aloufi and A. S. Alsaedi, "API-driven integration for smart city real estate platforms," IEEE Access, vol. 10, pp. 98765–98780, 2022.
[6] B. S. Rawal and R. K. Dhanaraj, "Security and privacy chal- lenges in smart real estate management: A survey," IEEE Internet of Things Journal, vol. 11, no. 4, pp. 5821–5835, 2024.
[7] J. Ullah and K. Al-Turjman, "A blockchain-based housing system architecture for trustworthy rental markets," Elec- tronics, vol. 13, no. 15, p. 3012, 2024.
[8] T. Zhang and Q. Li, "Design of a web-based process man- agement system with automatic code generation for real es- tate portals," Applied Sciences, vol. 13, no. 21, p. 11737, 2023.
[9] S. Singh and S. K. Singh, "Microservices in practice: A sur- vey of issues and solutions in real estate applications," IEEE Access, vol. 9, pp. 145678–145692, 2021.
[10] M. A. Serhani et al., "Cloud-based real estate services: Ar- chitecture, challenges, and solutions," IEEE Transactions on Services Computing, vol. 14, no. 6, pp. 1895–1908, 2021.
[11] S. A. R. Khan et al., "Smart contract security in real estate: Vulnerabilities and countermeasures," IEEE Access, vol. 11, pp. 45678–45692, 2023.
[12] J. Grönroos and T. Knuuti, "Blockchain's grand promise for the real estate sector: A systematic review," Applied Sci- ences, vol. 12, no. 23, p. 11940, 2022.
[13] L. Wu et al., "Deep learning for real estate valuation: A com- prehensive benchmark," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 8, pp. 8343–8356, 2023.
[14] R. G. D'Agostino, "Real estate app development based on AI/VR technologies," Electronics, vol. 12, no. 3, p. 707, 2023.
[15] R. Valenta, J. Novak, and P. Svoboda, "Blockchain and smart contracts to secure property transactions in smart cities," Ap- plied Sciences, vol. 13, no. 1, p. 66, 2022.
[16] S. Y. Lee and J. H. Park, "A systematic review of smart real estate technology: Drivers and barriers to online platforms," Sustainability, vol. 12, no. 9, p. 3142, 2020.

BIOGRAPHIES OF AUTHORS

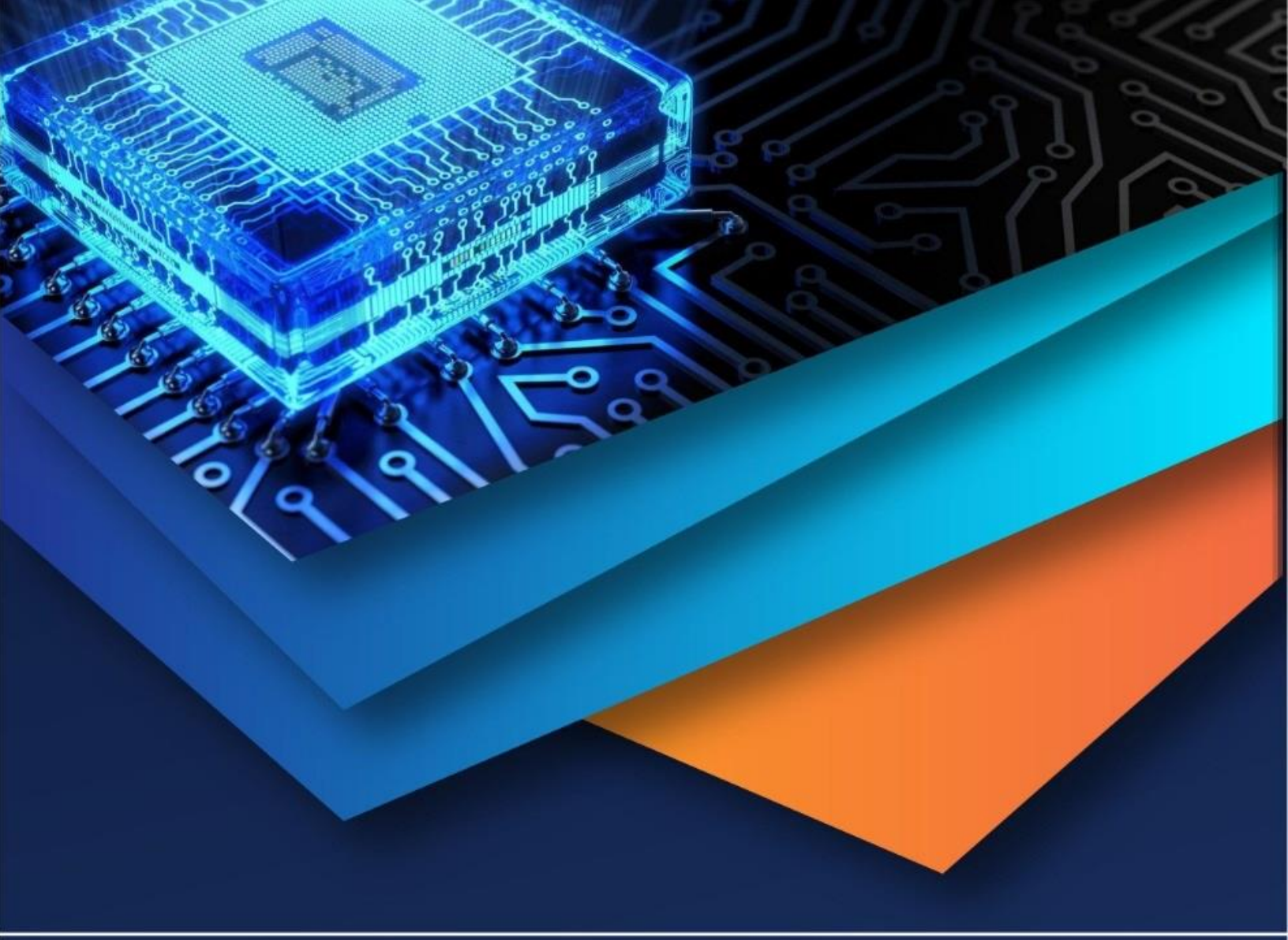
Kushal B.C. is currently residing at Odalarevu, East Godavari, Andhra Pradesh-533210. He is a B.Tech student specializing in Computer Sci- ence & Engineering at Bonam Venkata Chala- mayya Engineering College, Odalarevu, with an expected graduation in May 2026. He aims to se- cure a position that leverages his strong organiza- tional skills, educational background, and ability to work effectively with others. He possesses key skills in System architecture, UI/UX design, team collab, prompt engineering. While his profes- sional experience is listed as a student, his proac- tive approach and skill set indicate a strong poten- tial for growth and contribution in a professional setting. Email: 22221a05d9@bvcgroup.in. OR- CID: https://orcid.org/0009-0000-3311-4248

Saiju Barai is currently residing at Odalarevu, East Godavari, Andhra Pradesh-533210. He is a B.Tech student specializing in Computer Sci- ence & Engineering at Bonam Venkata Chala- mayya Engineering College, Odalarevu, with an expected graduation in May 2026. He aims to secure a position that leverages his strong organi- zational skills, educational background, and abil- ity to work effectively with others. He possesses key skills in Spring Security, JWT Authentica- tion, Database Design, MySQL, Java, Python, HTML, CSS, git/github. While his professional experience is listed as a student, his proactive ap- proach and skill set indicate a strong potential for growth and contribution in a professional set- ting. Email: 22221a05d5@bvcgroup.in. ORCID: https://orcid.org/0009-0009-7981-9273

Shubham Timilsina is currently residing at Odalarevu, East Godavari, Andhra Pradesh- 533210. He is a B.Tech student specializ- ing in Computer Science & Engineering at Bonam Venkata Chalamayya Engineering Col- lege, Odalarevu, with an expected graduation in May 2026. He aims to secure a position that leverages his strong organizational skills, ed- ucational background, and ability to work ef- fectively with others. He possesses key skills in Thymeleaf, Frontend Development, HTML, SQL, git/github, JavaScript, CSS3, Responsive Design, User Interface Design. While his profes- sional experience is listed as a student, his proac- tive approach and skill set indicate a strong poten- tial for growth and contribution in a professional setting. Email: 22221a05d8@bvcgroup.in. OR- CID: https://orcid.org/0009-0005-4437-6135

Prajwal Lekhi is currently residing at Odalarevu, East Godavari, Andhra Pradesh- 533210. He is a B.Tech student specializing in Computer Science & Engineering at Bonam Venkata Chalamayya Engineering College, Odalarevu, with an expected graduation in May 2026. He aims to secure a position that leverages his strong organizational skills, educational background, and ability to work effectively with others. He possesses key skills in UI/UX Design, figma, Responsive Design. While his professional experience is listed as a student, his proactive approach and skill set indicate a strong potential for growth and contribution in a professional setting. Email: 22221a05d2@bvcgroup.in. ORCID: https://orcid.org/0009-0000-0119-0895

B. Ganga Bhavani is Research Scholar at Koneru Lakshmaiah Education Foundation (KLEF) Green Fields, Vaddeswaram and Asso- ciate Professor at Bonam Venkata Chalamayya Engineering College, Odalarevu. She holds an M.Tech degree in Computer Science and Engineering from GIET College, Rajahmundry. Her research areas are Machine Learning, Deep Learning and Artificial Intelligence. She has a number of patents related to ma- chine learning field and industrial designs on her innovative ideas and has been awarded with international patents and published dif- ferent articles in international conferences. She can be contacted at: Koneru Lakshma- iah Education Foundation (KLEF) Green Fields, Vaddeswaram, A.P – 522302. Email: bgangabhavani.bvce@bvcgroup.in. ORCID: https://orcid.org/0000-0003-1433-5832



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)