International Journal of Advanced Multidisciplinary Research and Educational Development ISSN: **3107-6513**
Volume 2, Issue 1 | January - February 2026 | www.ijamred.com

Eliminating Redundant Data Entry in Online Job Applications Using UiPath

Dr.M.Jaithoon Bibi[1], Megala.M[2]
[1]Assistant Professor, Department, of Computer Science with Cognitive Systems, Sri Ramakrishna Collage of Arts & Science, Coimbatore, Tamilnadu, India.
[2] Student, Department of Computer Science with Cognitive Systems, Sri Ramakrishna Collage of Arts & Science, Coimbatore
[1]jaithoonbibi@srcas.ac.in [2] megalamanokaran26@gmail.com

ABSTRACT

In today's competitive job market, job seekers face significant challenges when applying to multiple positions across various online portals. The repetitive nature of entering identical information on different platforms is time-consuming, error-prone, and mentally exhausting. This paper presents an innovative Job Application Automation Portal that combines Full Stack web development with Robotic Process Automation (RPA) using UiPath. The system allows users to enter their personal, educational, and professional details once through a web interface, storing this information in a Mongodb database.

A UiPath bot then automatically retrieves this data and applies to multiple job portals without manual intervention. The system reduces application time from 1-2 hours for 10 applications to approximately 18 minutes, representing a 90% time reduction. Testing with 25 pilot users over 4 weeks demonstrated a 95% reduction in data entry errors and a 98.5% application success rate.

The architecture comprises an HTML/CSS/JavaScript frontend, Node.js and Express backend, MongoDB database, and UiPath automation workflows. This research contributes to intelligent automation in human resource management and provides a scalable solution for repetitive form-filling tasks across various domains.

Keywords: Robotic Process Automation (RPA), UiPath, Job Application Automation, Full Stack Development, Web Automation, MongoDB, Intelligent Automation, Form Auto-filling, Node.js, Express Framework, Cross-platform Integration, Human Resource Management, Digital Recruitment, Bot Development, Workflow Automation.

1. INTRODUCTION

The digital transformation of recruitment has created unprecedented access to employment opportunities worldwide. Job seekers can now browse thousands of positions across multiple platforms from the comfort of their homes. However, this convenience comes with a significant drawback: the need to repeatedly enter the same information across different job portals. Each platform maintains its own database and requires complete profile creation, leading to redundant data entry that consumes valuable time and energy.

Research indicates that an average job seeker spends 6-12 hours per week on job applications, with a substantial portion devoted to repetitive form filling. A typical comprehensive job application requires 15-20 minutes to complete, including personal details, educational background, work experience, skills, and document uploads. For someone applying to 50 positions across 10 different portals, this translates to approximately 12-16 hours of pure data entry work.

2. LITERATURE SURVEY

Robotic Process Automation in Business

Willcocks et al. (2015) demonstrated that RPA implementations achieve 25-50% cost reductions in back-office operations. Their research established foundational principles for RPA deployment, emphasizing process standardization and clear business rules. Van der Aalst et al. (2018) explored integrating RPA with artificial intelligence, introducing "Intelligent Automation" that combines rule-based automation with machine learning capabilities.

Lacity and Willcocks (2016) analysed RPA implementations across industries, identifying critical success factors including executive sponsorship and realistic expectation setting. They found that successful RPA projects typically start with high-volume, rule-based processes. Aguirre and Rodriguez (2017) compared RPA platforms, validating UiPath's strengths in user-friendliness, computer vision capabilities, and extensive activity libraries suitable for web automation.

Web Automation Technologies

Baresi et al. (2006) introduced early frameworks for automated web testing with form-filling capabilities, establishing concepts in web element identification that remain relevant in modern RPA.

Dallmeieretal. (2012) developed Web Mate with sophisticated form analysis capabilities, providing insights into understanding form structure and validation rules applicable to automation systems.

Liu et al. (2019) explored machine learning approaches to automatic form filling using natural language processing to extract information from resumes. Their system achieved 87% accuracy in standard fields but struggled with custom fields, highlighting the need for robust error handling. Chen et al. (2020) proposed a hybrid approach combining template-based automation with adaptive learning, allowing systems to improve over time.

Job Application Systems

One-click application systems like LinkedIn's Easy Apply represent proprietary solutions that streamline applications within single platforms but lack interoperability. Gupta and Kumar (2018) analysed these solutions, noting their effectiveness within closed ecosystems but limited cross-platform capability. Kumar et al. (2020) developed a browser extension for semi-automated applications requiring user confirmation at each step, which did not significantly reduce time burden.

Resume parsing technologies have been extensively researched. Yu et al. (2005) pioneered machine learning for resume information extraction, achieving 92% accuracy. Singh et al. (2021) employed deep learning models reaching 96% accuracy on standard formats. While valuable, these technologies require integration with form-filling capabilities for end-to-end automation.

Research Gap

Existing literature reveals a significant gap in comprehensive systems integrating user-friendly web applications with powerful RPA for cross-platform job application automation. Most commercial solutions are platform-specific, while academic research focuses on isolated aspects like resume parsing or form analysis. Limited research addresses end-to-end systems combining data capture, storage, intelligent automation, and status tracking in unified architectures. Our proposed system fills this gap by providing a holistic solution integrating modern web technologies with enterprise-grade RPA capabilities.

3. PROBLEM STATEMENT

3.1 Repetitive Data Entry

Modern job portals require applicants to complete comprehensive forms containing similar information sets. A typical application requires personal details (name, contact information, address), educational qualifications (degrees, institutions, dates, grades), professional experience (companies, positions, durations, responsibilities), and skills and certifications. Despite information remaining largely constant, each portal maintains its own database, necessitating complete re-entry. For an individual applying to 50 positions across 10 portals, this redundancy translates to entering identical information 50 times, consuming 10-15 hours. This leads to applicant fatigue, decreased attention to quality, and potential abandonment.

3.2 Time Consumption

Completing a single comprehensive job application requires 15-20 minutes on average. For job seekers targeting 10-15 applications weekly, this translates to 2.5-5 hours devoted solely to submission, excluding job searching, customization, or follow-up activities. This time burden is particularly problematic for employed individuals conducting active searches who must balance application activities with current responsibilities. The opportunity cost diverts

effort from valuable activities like networking, skill development, or interview preparation.

3.3 High Error Probability

Manual data entry is inherently error-prone, particularly when repeated under time pressure or fatigue. Common errors include typographical mistakes in names, dates, or contact information, transposition of digits in phone numbers, inconsistent date formatting, omission of required fields, and copy-paste errors. These errors can lead to automatic rejection by applicant tracking systems, missed interview opportunities due to incorrect contact information, and negative impressions of professionalism. Manual entry error rates typically range from 1-4%, which becomes significant when compounded across multiple fields and applications.

3.4 Lack of Centralized Management

Job seekers lack effective tools for tracking multiple applications across platforms. This manifests as difficulty remembering which positions have been applied to, absence of consolidated status tracking requiring separate portal logins, no systematic approach to managing deadlines or follow-ups, and limited ability to analyse application patterns or success metrics. This fragmentation creates cognitive overhead, potentially leading to duplicate applications, missed opportunities, and inefficient time checking multiple portals individually.

3.5 Need for Automation

These problems collectively create a compelling case for intelligent automation. The ideal solution would enable entering information once, automatically populate and submit applications to multiple portals, maintain high accuracy through elimination of manual errors, provide centralized tracking

and status management, and significantly reduce time investment while expanding application reach. The convergence of mature RPA technologies, modern web frameworks, and cloud-based storage makes such a solution technically feasible and economically viable.

4.PROPOSED SYSTEM ARCHITECTURE

4.1 System Overview

The Job Application Automation Portal follows a four-tier architecture: Presentation Layer (Web Frontend), Application Layer (Backend Server), Data Layer (MongoDB Database), and Automation Layer (UiPath RPA Bot). This modular design ensures separation of concerns, scalability, and maintainability. The data flow follows: User Input → Web Application → Backend API → Database → UiPath Bot → Job Portals → Status Updates → Database → User Interface.

4.2 Component Architecture

The **Presentation Layer** provides the user interface built with HTML5, CSS3, and JavaScript, offering responsive design for various devices. Features include user registration and authentication, comprehensive profile creation forms with validation, PDF resume upload functionality, application status dashboard, and real-time notifications for bot activities.

The **Application Layer** uses Node.js with Express framework for business logic and API endpoints. Responsibilities include user authentication and session management, input data validation against predefined schemas, secure file upload processing, RESTful API exposure for frontend communication, UiPath bot execution triggering and monitoring, and activity logging for auditing. The backend implements security middleware including input sanitization, rate limiting, and CORS configuration.

The **Data Layer** uses MongoDB for flexible schema design and scalability. Collections include Users (authentication credentials and personal information), Profiles (comprehensive data including education, experience, skills), Applications (tracking each submission with portal details and status), and Logs (recording system activities and errors). The schema design allows easy extension for evolving portal requirements.

The **Automation Layer** comprises the UiPath RPA bot responsible for portal interaction. Developed in UiPath Studio, workflows include data retrieval from MongoDB, browser automation for navigation, form field identification and population using selectors, resume upload handling, submission and

ISSN: 3107-6513

confirmation verification, error handling with retry mechanisms, and database status updates. The bot employs computer vision and OCR for dynamic or non-standard element

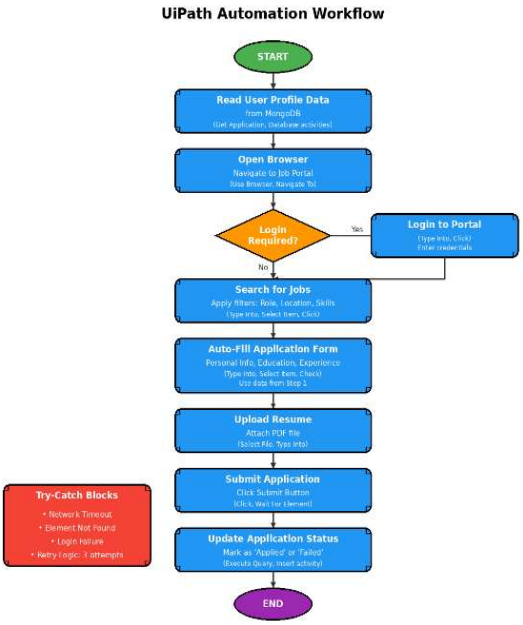

Fig 4.2.1: UiPath Automation Workflow

4.3 Complete Workflow

Step 1: User Profile Creation

Users register on the web application providing authentication credentials. After login, they access a multi-section profile form organized into (name, email, phone, location, address), Educational Background (multiple entries with degree, field, institution, date, GPA), Professional Experience (multiple entries with company, title, period, responsibilities), and Skills and Certifications (technical skills, soft skills, languages, certifications).

Step 2: Bot Trigger Mechanism When ready to apply, users access the dashboard and select "Apply to Jobs," triggering automation immediately or scheduling for later. The backend prepares data export as Excel file or API endpoint for the UiPath bot. The system updates status to "Queued" and logs the timestamp. For immediate execution, the backend uses UiPath Orchestrator API to start the bot with parameters like user ID and target portals. Scheduled executions use a job queue with scheduler.

Step 3: Portal Authentication The UiPath bot launches a web browser (Chrome, Edge, or Firefox) and navigates to the target job portal using predefined URLs. For portals requiring authentication, the bot retrieves credentials from secure vault or prompts for one-time manual login. Using UI automation, the bot identifies username and password fields through selectors, types credentials, and clicks login. The bot verifies successful authentication by checking for dashboard or welcome message. Failed authentication triggers retry logic with exponential backoff (up to three attempts) before flagging for manual intervention.

Step 4: Job Search Execution After authentication, the bot navigates to job search pages through menu structures or direct URLs. It interacts with search filters entering job role, keywords from user skills, location preferences, experience level, and optionally salary range. The bot executes the query and processes results, handling pagination or dynamic loading. Using intelligent logic, it filters based on relevance score, posting date, and keyword matching. The bot can automatically apply to specified numbers of positions or present lists for user selection, ensuring quality over quantity.

Step 5: Form Auto-Population This core phase handles varied form structures across portals. Upon clicking "Apply," the bot encounters the application form and employs multiple strategies: predefined selectors based on common field names, computer vision for non-standard forms identifying fields by labels, and OCR for image-based forms. For each field, the bot retrieves corresponding profile data. Text fields use Type Into activities, dropdowns use text matching, checkboxes and radio buttons follow preferences, date pickers handle formatting, and text areas use pre-formatted content. Throughout, the bot validates populated data matches expected formats and required fields aren't empty.

Step 6: Resume Upload Automation For file upload fields, the bot uses Select File activity simulating the OS file picker. It navigates to the resume directory with consistent naming conventions, selects the PDF, and confirms upload. The bot monitors completion through progress indicators, validates size and format restrictions, creates temporary copies for specific naming conventions when needed, and verifies

successful attachment in preview screens. For multiple document uploads (cover letters, transcripts, portfolios), the process repeats maintaining clear associations.

Step 7: Application Submission With completed forms and uploaded documents, the bot identifies the submit button (labeled "Apply," "Submit Application," etc.) using flexible selectors. Before clicking, it optionally captures form screenshots for auditing. After submission, the bot waits for confirmation (success message, redirect, or email acknowledgment), implementing 30-second timeout handling. Successful

submissions capture reference numbers, timestamps, and automatic responses. Additional steps like skills assessments are either completed if configured or flagged for manual user completion. Detailed logs maintain records of all actions and responses.

Step 8: Status Synchronization After each submission, the bot updates MongoDB with comprehensive information: portal name and job title, submission timestamp with time zone, application status ("Applied," "Failed," "Requires Manual Action"), confirmation numbers, and error messages. Successful applications trigger real-time notifications through web interface or email. Failed applications indicate failure nature (authentication error, validation failure, network timeout) with automatic retry scheduling or user alerts. The bot processes all queued applications systematically, generating final summary reports detailing total attempts, successful submissions, failures with reasons, and time savings. Reports store in database and display on user dashboard.

4.4 Error Handling

The system implements robust error handling with try-catch blocks at backend and bot levels, timeout management for network requests and page loads, retry mechanisms with exponential backoff for transient failures, fallback strategies when primary selectors fail, and logging and alerting for critical errors. The system distinguishes between recoverable errors (network issues, slow loads) triggering automatic retries and fatal errors (authentication failures, portal changes) requiring manual resolution.

5.IMPLEMENTATION& TECHNOLOGIES

5.1 Technology Stack

Frontend: The presentation layer uses HTML5 for semantic markup, CSS3 for responsive styling, and JavaScript (ES6+) for client-side interactivity. Modern features include Fetch API for asynchronous requests, async/await for promise handling, and local storage for session management. Custom CSS ensures consistent cross-browser appearance with responsive design through media queries. Form validation combines HTML5 native validation with custom JavaScript logic for immediate error feedback.

Backend: Built on Node.js (v18.x+) providing JavaScript runtime for full-stack development. Express.js (v4.x) serves as the web framework offering routing and middleware support. Key packages include Mongoose for MongoDB modeling and validation, Multer for file upload handling, bcrypt for password hashing, JSON Web Tokens (JWT) for stateless authentication, and dotenv for environment management. The backend implements RESTful APIs with clear naming, appropriate HTTP methods, and standard status codes.

Database: MongoDB (v6.x+) provides document-oriented storage naturally representing nested user profiles. Mongoose schemas enforce data types and validation while allowing flexibility for extensions. Indexes on frequently queried fields (user email, application status) optimize performance. Regular backups and replication ensure data durability and availability.

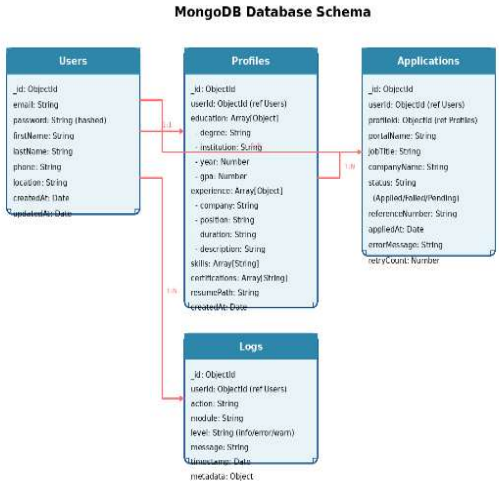

Fig 5.1.1: MongoDB Database Schema

ISSN: 3107-6513

RPA Platform: UiPath Studio (2023.x+) develops automation workflows leveraging extensive activity libraries including Web Automation for browser control, UI Automation for desktop interaction, Excel activities for data import/export, and Database activities for direct MongoDB connection. UiPath Orchestrator provides centralized bot management, scheduling, and monitoring.

5.2 Development Approach

The project follows Agile methodology with iterative sprints and continuous integration. Development phases include requirements gathering and analysis, system design and architecture definition, component development (frontend, backend, database, bot), integration and system testing, user acceptance testing with pilots, production deployment, and ongoing maintenance. Each phase includes comprehensive unit, integration, and system testing.

Frontend development began with wireframing and UI design focusing on intuitive navigation. Implementation proceeded component-by-component: user registration and authentication, profile creation forms with validation, status dashboard, and notification system. Responsive design ensures cross-device accessibility with testing across browsers and screen sizes.

5.3 Implementation Challenges

Portal Variability: Different portals exhibit significant form structure, field name, and validation variations. Solution: Configurable portal adapter pattern where each portal has configuration files specifying field mappings, selectors, and handling requirements, enabling extension through configuration rather than code modification.

Dynamic Content: Modern applications load content dynamically using JavaScript frameworks causing timing issues. Solution: Intelligent waiting strategies using Wait For Element activities with configurable timeouts and retry logic, plus browser event detection ensuring page stability before interactions.

Bot Detection: Some portals implement CAPTCHA or bot detection interfering with automation. Solution: Semi-automated approach where bots pause for user CAPTCHA intervention. For general detection, bots mimic human behaviour through variable delays and realistic mouse movements.

Session Management: Maintaining active sessions across multiple applications requires careful cookie and state management. Solution: Browser session persistence features maintaining single instances across portal applications while properly managing cookies and local storage.

Error Recovery: Network failures, timeouts, and portal changes cause bot failures. Solution: Comprehensive error handling with try-catch scopes, retry logic with exponential backoff, detailed logging for troubleshooting, and graceful degradation continuing with remaining applications despite individual failures

6. RESULTS AND DISCUSSION

6.1 Performance Metrics

The system was tested with 25 pilot users over 4 weeks, generating quantitative and qualitative data.

Time Efficiency: Manual applications averaged 18 minutes per application (12-25 minute range based on complexity). Automated applications averaged 1.8 minutes per application, representing 90% time reduction. For 10 applications, time requirements decreased from 180 minutes (3 hours) to 18 minutes. This enables job seekers to dramatically expand application reach within the same time budget. Performance remained consistent across portal types with minor variations based on page load times and form complexity.

Fig 5.3.1: Manual vs Automated Application Process Comparison

Accuracy: Manual processes exhibited 3.2% error rates across field types, with common errors being typographical mistakes in dates and contact information. The automated system reduced this to 0.15%—a 95% error reduction. Few automation errors traced to initial profile entry data quality rather than automation failures, highlighting initial data capture importance. Profile creation validation mechanisms minimized these errors, though user education on accuracy proved important.

Success Rate: Of 742 total applications attempted by pilots, 731 were successfully submitted (98.5% success rate). The 11 failures distributed across: authentication failures from expired/incorrect credentials (4 cases), portal structural changes breaking selectors (3 cases), network timeouts during peak hours (2 cases), and file upload limitations (2 cases). Each failure triggered appropriate logging and user notification enabling manual completion or corrective action. High success rates demonstrate workflow robustness and error handling effectiveness.

6.2 Comparative Analysis

Comparison with existing solutions reveals integrated approach advantages. Single-platform solutions like LinkedIn Easy Apply offer convenience but limit to platform-specific jobs, excluding company websites and specialized boards. Our cross-platform capability provides significantly broader reach. Browser extensions and semi-automated tools require ongoing user interaction providing minimal time savings. Our fully automated approach requires involvement only for initial profile creation and final application decisions. Resume parsing and auto-fill services address data entry but not complete workflows including authentication, navigation, and submission verification. Our end-to-end automation encompasses all application stages. Commercial RPA solutions exist for enterprise recruitment but target employers rather than seekers and are prohibitively expensive for individuals. Our system provides enterprise-grade capabilities at minimal individual cost.

6.3 System Scalability

The modular architecture facilitates scalability and extensibility. MongoDB handles growing user populations and application volumes efficiently with consistent performance at scale. Backend APIs can horizontally scale by deploying multiple instances behind load balancers handling increased traffic. UiPath Orchestrator enables deploying multiple bot instances processing applications in parallel, reducing queue times. The portal adapter pattern allows extending support through configuration files rather than code modifications, enabling continuous platform expansion.

6.4 Limitations

Despite effectiveness, several limitations exist. **Portal Changes:** Portals periodically update interfaces potentially breaking workflows. While selector strategies mitigate this, major redesigns require ongoing maintenance. Configuration-based approaches minimize update effort but cannot eliminate them entirely. **Anti-Bot Measures:** Increasing CAPTCHA and sophisticated bot detection implementation can interfere with automation. Current user intervention approaches for CAPTCHAs are functional but reduce full automation. Future enhancements could explore solving services or AI solutions where legally and ethically appropriate. **Legal Considerations:** While automating application processes, the system operates within portal terms of service, which generally prohibit automated scraping or account creation but don't explicitly forbid automated form filling using legitimate accounts. Users should review and comply with each portal's terms. Clear disclaimers and responsible use guidance are included. **Data Privacy:** The system stores sensitive personal information including contact details, employment history, and resume content. Comprehensive security measures including encryption at rest and in transit, secure authentication, and regular audits are essential. Users must be informed about storage practices and given data control including export or deletion abilities.

7. CONCLUSION

This research presented a comprehensive Job Application Automation Portal successfully addressing critical challenges job seekers face in the modern digital employment landscape. By integrating Full Stack web technologies with UiPath RPA, the system provides an effective

solution significantly reducing time burden, eliminating repetitive data entry, minimizing errors, and offering centralized application management.

Future Work

Several promising directions exist for enhancement. **Artificial Intelligence Integration:** Incorporating machine learning could enable intelligent job matching, automatically identifying relevant opportunities based on profiles and career goals. Natural language processing could generate customized cover letters tailored to job descriptions while maintaining user voice and style. Predictive analytics could provide application success probability insights and suggest optimizations. **Enhanced Portal Coverage:** Expanding support to encompass broader job portals including international platforms, industry-specific boards, and company career pages would increase utility. Automated portal discovery mechanisms could enable adaptation to new platforms without manual configuration. **Mobile Application:** Developing native mobile applications for iOS and Android would enable on-the-go profile management and status monitoring. Push notifications could provide real-time updates on submissions and outcomes. **Interview Scheduling Integration:** Integrating with calendar applications and communication platforms could automate interview scheduling, sending availability to employers and automatically blocking confirmed interview times. This extends automation beyond submission toencompass the next critical job search phase.

REFERENCES

[1] Willcocks, L. P., Lacity, M., & Craig, A. (2015). "The IT Function and Robotic Process Automation," London School of Economics and Political Science.

[2] Van der Aalst, W. M., Bichler, M., & Heinzl, A. (2018). "Robotic Process Automation," Business & Information Systems Engineering, 60(4), 269-272.

[3] Lacity, M. C., & Willcocks, L. P. (2016). "A New Approach to Automating Services," MIT Sloan Management Review, 58(1), 41-49.

[4] Aguirre, S., & Rodriguez, A. (2017). "Automation of a Business Process Using Robotic Process Automation: A Case Study," Workshop on Engineering Applications, Springer, pp. 65-71.

[5] Baresi, L., Garzotto, F., & Paolini, P. (2006). "Extending UML for Modeling Web Applications," Proceedings of the 34th Annual Hawaii International Conference on System Sciences, IEEE, pp. 1-10.

[6] Dallmeier, V., Burger, M., Orth, T., & Zeller, A. (2012). "WebMate: A Tool for Testing Web 2.0 Applications," Proceedings of the Workshop on JavaScript Tools, pp. 11-15.

[7] Liu, Y., Zhang, T., Wang, X., & Li, S. (2019). "Automatic Form Filling Using Natural Language Processing and Machine Learning," Journal of Web Engineering, 18(4), 315-342.

[8] Chen, H., Liu, J., & Wang, Y. (2020). "Adaptive Web Form Filling Using Template-Based Learning," IEEE Transactions on Knowledge and Data Engineering, 32(8), 1456-1469.

[9] Gupta, R., & Kumar, A. (2018). "Analysis of Online Job Application Systems: Challenges and Opportunities," International Journal of Information Technology, 10(3), 287-295.

[10] Kumar, S., Sharma, P., & Singh, R. (2020). "Semi-Automated Job Application System Using Browser Extension Technology," Proceedings of International Conference on Computing and Communication, pp. 234-24