Volume: 13 Issue: XI Month of publication: November 2025
DOI: https://doi.org/10.22214/ijraset.2025.75553

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

Ontop - An Intelligent Workspace Management Application with Machine Learning Integration

Madhumati Pol[1], Alesha Mulla[2], Ajinkya Ghule[3], Aliza Kundal[4]
Department of Engineering, Sciences and Humanities, Vishwakarma Institute of Technology, Pune, India

Abstract: In today's fast moving digital environment, managing the large number of clients, contacts and projects is a problem faced by professionals throughout the world. Existing solutions focus on a specific technique for boosting productivity, such as automated note-taking or To-Do list generation. This paper presents Ontop, a personal workspace management application that combines multiple known productivity tools into one. In a single interface, it allows for client detail storage, To-Do list, Project management and event logging. It supports real-time file access, contextual client interactions, and seamless organization of work data. By providing major productivity tools in a single interface, it increasing efficiency and improving UX. Ontop also uses its own ML layer (built with Random Forest, BERT-based transformer models and Neural Ranking Model) which allow smart contact-based notifications which show relevant project information when receiving a call from a client. This layer also helps with task and note recommendations. This paper describes the motivation, system architecture, feature implementation, and future scope of Ontop, and how it can be used to enhance productivity.

Keywords: Mobile application, cross-platform development, machine learning, natural language processing, project management, client tracking, real-time data handling, productivity tools, Flutter, MongoDB, NodeJS.

I. INTRODUCTION

In an increasingly digital and high workload professional environment, effective workspace management has never been more important. Freelancers, remote employees, and small business professionals often manage several separate workspaces with their own clients, ongoing projects, scheduled meetings, and time-sensitive tasks simultaneously. Most existing productivity tools either focus on a single function, such as task tracking, note-taking, or calendar integration, or are designed for teams, making them unnecessarily complex or unsuitable for individual users. As such, we're presenting Ontop, a cross-platform mobile application developed to address this gap by providing a centralized app for personal workspace management. It allows users to maintain structured records of clients, link them to specific projects, manage associated tasks and events, and store important files, all within a single interface. The application has been built using Flutter for the front end with Node.js, which serves as an API layer to handle communication between the app and a MongoDB database hosted on MongoDB Atlas. This tech stack enables scalable data management, real-time data handling and cross-device data syncing. With Ontop, we expect our users to concurrently have dozens of business contacts. As such, keeping track of all projects, tasks and deadlines would be difficult. To address this, Ontop implements a machine learning layer to look at all projects, tasks and contacts to generate a summary of what all the user is collaborating on with a specific contact. On receiving a phone call from this contact, this summary is sent to the user in the form of a push notification. This summary makes it easier to multi-task and the context awareness helps users respond more effectively in real time. The contents of this paper include the development details of Ontop, like its system architecture, data models, database implementation, connecting API layers. It also discusses the future scope for the app, such as analytics dashboards, and a web-based workspace interface.

II. LITERATURE REVIEW

According to A. M. Smirnov and Y. A. Shelomentseva [1], minimalism facilitates optimal user engagement due to avoidance of superfluous complexities and clear, transparent, evenly displayed offerings without compromising intent for differing user needs. M. Bano and D. Zowghi [2] evaluated that user engagement is an essential factor of system development success but needs to be moderated meaning that the more appropriate engagement levels helped requirements gathering and systems during development, the better.

S. R. Chavare et al. [3] reveal that deep learning structures and models are necessary for advancement of recommendation accuracy in personal productivity applications which assist in the formulation of suggestive action systems with a smart suggestion outcome. A. Khamaj and A. M. Ali [11] evaluated how reinforcement learning can successfully tailor user experiences through real-time evaluations of users and thus, customized interface components and suggested tasks.

K. Akhil et al. [4] demonstrate significant NLP processing of unstructured communicative data from elsewhere. Their results substantiate how contemporary implementations can extract real task-based information from emails, reminders, and notes from a call. Chandrakala et al. [6] implemented an intent detection pipeline for conversational AI that could assist in systems that evaluate phone calls and use automatic intelligence to gather the most noted applicable information and reminders from talking over the phone.

V. Radu et al. [5] studied multimodal deep learning for activity detection and contextual awareness. The more information that can be assessed from multiple modalities of information, the deeper the relevant context found when it comes to intelligently assessing a phone call for situationally aware task creation.

A. Aldayel and K. Alnafjan [7] studied mobile application development approaches which reveal best practices that promote performance and usability from iterative testing and UX design emphasis. I. Qasim et al. [9] studied practices of mobile user experience development which reveals best practices to keep usability effectiveness. A. Jain [15] studied cross-platform applications with scalable solutions that service AI/ML implementation with performance integrity.

N. Freitas, A. D. Rocha, and J. Barata [8] explored data management solutions from an industrial perspective, offering the best principles of effective systems as necessary applications to maintain a diversified means of information as necessary for smart productivity. J. Ferreira and J. Noble [10] studied how UI design within an agile cycle fosters user testing, which is beneficial for defining complex AI characteristics that need feedback incorporation to improve.

S. Sulkifli and A. M. Putri [12] studied freelancers as hybrid workers and found that flexibility and work/life balance trump social isolation ramifications which infer a smart communicative solution is necessary. K. Renard et al. [13] studied such findings in an employment-based context. Y. A. Rizky et al. [14] note how clutter limits productivity due to increased stress which supports the idea of a intelligently run organizational task manager to ensure cluttered scenarios like these do not happen in the first place.

III. METHODOLOGY

A. Components

1) Flutter (Frontend)

Flutter serves as the frontend framework. It allows developers to build high performance apps that compile natively from one codebase. This works across various devices without issues. When compared to options like React Native or pure Android and iOS coding, Flutter stands out for a few reasons.

- It uses the Skia engine for quick rendering. The user interface responds well with smooth animations and is portable.
- Performance stays consistent on both Android and iOS platforms.
- It has a large set of widgets for custom designs that take little time.
- The community provides thousands of libraries to support UI work.

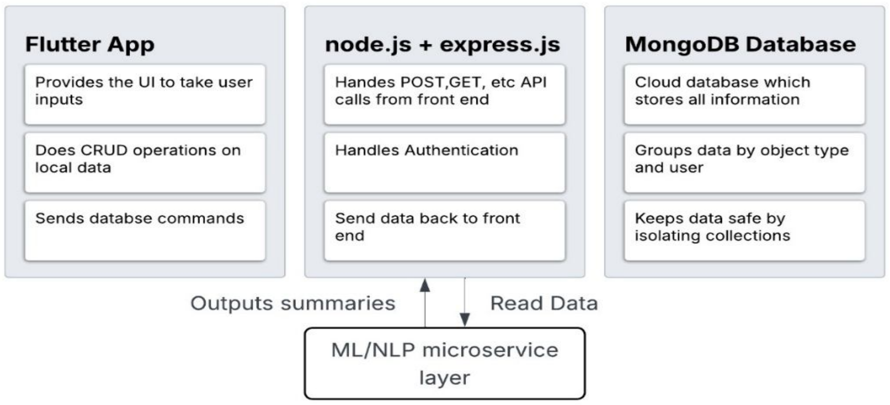
Flutter App
Provides the UI to take user inputs
Does CRUD operations on local data
Sends databse commands
node.js + express.js
Handes POST,GET, etc API calls from front end
Handles Authentication
Send data back to front end
MongoDB Database
Cloud database which stores all information
Groups data by object type and user
Keeps data safe by isolating collections
Outputs summaries
Read Data
ML/NLP microservice layer

Fig 1: Block Diagram of Ontop

2) Node.js with Express.js (Backend API)

For the backend API, Node.js is paired with the Express.js framework. We chose Node.js for its event driven setup and handling of asynchronous input output tasks. Alternatives such as Django or Spring Boot exist, but Node.js keeps things leaner. It also works better with JavaScript in the frontend when required. In Ontop,

- This setup handles concurrent API requests quickly.
- Routing for REST stays simple, middleware integrates easily.
- The non-blocking server fits real time needs well.

3) MongoDB Atlas (Database)

The database is use for Ontop is MongoDB. A NoSQL document style database fits better than relational ones here as:

- No fixed schema means the app can grow, and data models can shift over time.
- Storage in JSON like format aligns with the data flowing between Flutter and Node.js.
- It scales horizontally with cloud features built in.
- Each user gets their own collection, so sensitive information stays separate.
- As Ontop deals with big data, storage and querying is easier in a NoSQL environment.
- User data synchronization becomes a lot easier as each new login on a separate device only needs to connect to MongoDB.

All changes were also stored to a local SQLite database, which would update first. This is slaved to the primary source of truth which is the MongoDB database.

4) REST API Architecture

It follows the RESTful API design pattern to normalize communication between the mobile app and the backend. Among such solutions, REST was chosen because of its stateless design, easiness of testing, and wide support across mobile platforms. Restful APIs are easy to scale and thus provides space for future extension, such as the inclusion of web or desktop interfaces.

5) Intelligent Processing Layer (ML Integration)

Ontop's intelligent processing layer uses Python-based microservices to host trained ML models (Random Forest for task priority prediction, BERT-based transformer models for NLP and a Neural Ranking Model for call notification context). This layer helps us with task priority prediction, NLP for client notes, and context generation for call notifications. It analyzes user behaviour, processes unstructured data and generates contextual insights. This is integrated using REST API interfaces and the Node.js backend.

B. Design

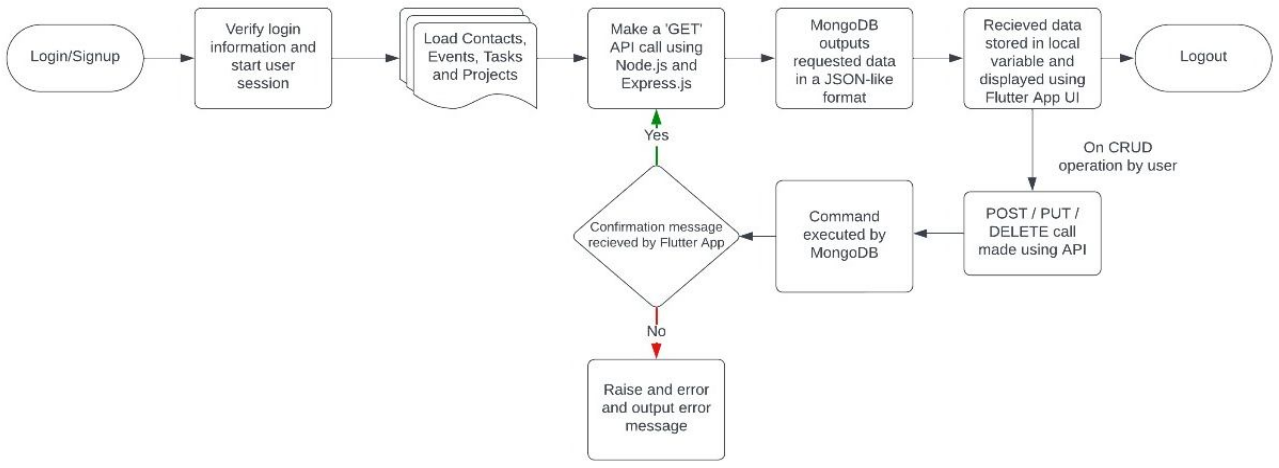
Login/Signup
Verify login information and start user session
Load Contacts, Events, Tasks and Projects
Make a 'GET' API call using Node.js and Express.js
MongoDB outputs requested data in a JSON-like format
Recieved data stored in local variable and displayed using Flutter App UI
Logout
On CRUD operation by user
POST / PUT / DELETE call made using API
Command executed by MongoDB
Confirmation message recieved by Flutter App
Yes
No
Raise and error and output error message

Fig 2: Flowchart of How User Data is Fetched and Displayed

1) MongoDB

Ontop uses the non-relational MongoDB database to store user data. The structure has two tiers – one for centralized user management and the other to hold individual user collections. The users collection is where all information necessary for authentication and identity formation will exist. It has usernames, passwords, names, emails and the like for profiling. The relevant documents that will be created for each user are user id, name, email, phone, password encrypted, timestamp created.

Thereafter, a subcollection will exist for each of them. These subcollections will be named for their unique id so records are entirely separated for more ease, and efficient, querying, since they can only query their own subcollection.
Subcollections can exist with any number of differing collections/documents, but each document contains a type field so it can differentiate between document types.
This is a scalable solution as data is not blended and is horizontally scaled as more users join over time. In addition, subcollections allow for horizontal scaling as more users join. Finally, subtype filtering allows for further types of information to be explored without needing to change anything about the hierarchy.

2) Node.js

We created a RESTful API with Node.js as the engine and Express.js framework for CRUD functionality. The whole app is modular as it's segmented into routing capabilities, information creation and access, and server information for set up.
The application uses the standard HTTP methods GET, PUT, POST and DELETE for API endpoints that enable the application to request and reply to the database in appropriate manners. The application includes comprehensive error catching capabilities to prevent HTTP level errors or account for them. All endpoints operate with try-catch statements to safeguard asynchronous operation and return standardized JSON replies for error reporting when necessary.

3) Machine Learning Model

Ontop features an intelligent layer of interpretation via microservices developed in Python. Such microservices contain trained ML models for prediction of task priority, NLP of notes taken by the user, and contextual adaptation on the fly during on the phone calls made to clients. The models send inference requests and reply over to the REST endpoint of the Node.js API which then finally saves their inferences to MongoDB.

4) Flutter

When you first open the app, it links up with the MongoDB database and checks to see if the user has logged in already. If the user is not logged in, they end up on the login or signup page. There they can type in their details to get back into their accounts. Those details get sent over securely to the Node.js server. From there, the server checks them against the database to either confirm the login or set up a new user. Once logged in, the app sends you to the home page. On that page, you pick from the different “tools” or tabs that Ontop provides, which are Contacts, To-Do, Events, and Projects. Each tab has its own custom data model designed to store all the key details. These data structures are stored as lists, which are loaded from the server into local memory whenever a tab is opened. All user interactions operate on this data, which causes the UI to update. Parallelly, these changes are sent to the database and a confirmation message is received. The app calls the begin() whenever it starts up or gets refreshed. A helper to cue the node.js portion to connect to the database and retrieve all the proper information. It comes in as a json file and is parsed and populated on the local end. All edits are made locally and are then pushed to the database. The changes are stored in a local SQLite DB which is queried and kept in sync with the MongoDB database. The results of the local database are used to build the UI. In case there is a disparity between the MongoDB results and SQLite results, the MongoDB data is used to rebuild the UI.

C. Intelligent Processing Layer (Machine Learning & NLP Modules)

1) ML-Based Task Recommendation Engine

Additionally, Ontop features a low-level machine learning aspect that watches its users to best recommend which tasks to accomplish based on user activity. For example, it pays attention to how many times a user does A, how many times they are recommended A over B, how many times it hears A in peer conversation, how many times A is submitted for grading on more assignment due dates than less than other options to understand what is important and what should automatically flagged/undermarked. Therefore, the program becomes accustomed to one person's unique way of operating and assists with the anticipated achievement of task facilitation.

2) NLP-Assisted Note Processing

The notes feature is enhanced with an NLP-based action-item extraction system. When users write or update notes, a text-processing pipeline identifies potential tasks, deadlines, and collaborator references. These action items can be added directly to the project’s to-do list, reducing manual effort and ensuring important details are captured consistently.

3) AI-Driven Call Context Awareness

To improve communication workflows, Ontop also includes an AI-driven context engine for incoming calls. When a saved contact calls, the system uses a ranking model to identify and display the most relevant project information, such as recent updates, files, or pending tasks. This turns simple call notifications into meaningful, context-aware summaries that help users stay prepared during client interactions.

D. System Workflow

1) User Authentication

New users can register by providing name, email, contact number, and a password, which is then hashed and stored securely. The app launches and checks if the user is already authenticated. If not, the user is asked to enter their email and password. The entered credentials are sent to database and verified against the hashed values stored. If the match is successful, the user is logged in and directed to the home screen. If authentication fails, an error message is returned.

2) Tab-Based Data Loading

Upon login, the user selects a tab: Contacts, To-Do, Projects, or Events. A query is constructed based on the selected tab type (e.g., "type": "contact"). The system fetches relevant documents from the user-specific MongoDB collection. The retrieved data is parsed into local lists for in-memory interaction. These lists are used to populate the UI. Since this operation takes a varying amount of time, usually in seconds, as discovered via testing, we store all user information in a local SQLite database. This information is used to construct the UI until get request is received and processed.

3) Local-First Data Update

When a user adds, updates or deletes an entry, first the local SQLite DB gets updated. The UI reflects this change instantaneously. In the background, an asynchronous request updates the server. The server, in turn, updates the MongoDB DB. If, however, in this moment, the server is down, it marks the entry as unsynced and alerts the user.

4) Smart Call Notification Trigger

When a user receives a call, the application employs machine learning to ascertain whether it's a customer calling who's on the application and utilizes smart number normalization should the customer save the number any differently. If so, it reveals the customer's name, associated project and relevant deadlines. Moreover, using NLP, it assumes what the customer wants to speak about based upon the last few sent/received texts to/from the customer. Thus, a contextual alert is generated with relevant customer data and AI-integrated assumptions. If it is not, the customer receives a more generic alert, predicting, via machine learning predictive analytics, to see if it's anyone that it recognizes.

5) Quick Communication Redirection

The client's interface has a WhatsApp message and email/phone call button. The app generates the appropriate URI (ie. tel +91XXXXXXXXXX) and system intent is opened for the respective app. This saves several steps between apps in one's workflow.

E. Testing

The majority of our testing focused on measuring the delay between user input (like taps or swipes) and UI updates. This delay was measured using Flutter DevTools. Two tests were run, one with only the remote MongoDB database connected and one with the both MongoDB and local SQLite database running. We found that waiting for confirmation from the MongoDB database took roughly 3x longer when compared to making the same changes to the local SQLite DB (80-110 ms vs 340-350 ms).

IV. RESULTS AND DISCUSSIONS

The final build of Ontop combines a variety of essential productivity features - calendar, notes, projects, contacts and to-do lists - all in one organic space so users do not have to inefficiently juggle between different apps that represent the same overarching theme. By creating native, interoperable applications for each, we also get dynamic references (my project may have milestones set in my calendar, a task may need reference to a colleague in my contacts, a note may have a live to-do check-box). This will increase user productivity by reducing time taken to position between these.

User testing supports that our local-first storage approach facilitates such ease. Given that all reads and writes occur on in-memory lists first, UI lags were never longer than expectation to write new information to a remote server and see if acknowledgement returned. Testing after benchmarking found 80-110 ms as the consistent UI response time throughout tasks aimed at readjusting task from due time until completion. Server responses were noted between 300-450 ms averaging similar information request/confirmation from submission dependent on whether writing acknowledgment was present.

Where even more leverage is gained from Ontop is with ML: Ontop houses a small machine learning model that analyzes effort in task to suggest recommended/priority tasks based on effort exertion (task marked as interesting), due dates and previous use across the platform. The notes tool is tied to an NLP pipeline which recognizes tasks, dates or collaborators mentioned in written notes and automatically tags them to the project to-do list. In addition, an AI-context engine analyzes recency of proximity to determine which information should be best displayed upon getting a saved contact's call based on when the number was last actively used in the project.

Thus, by accessing primary productivity features without the delay of cross-device enhancements over cloud reliance and thus, solely managing what's easy in local implementation, we create a quicker, seamless process without fault over device practicality. Thus, this trade-off is well-worth it to not rely on constant MongoDB round trips.

V. FUTURE SCOPE

Since an Ontop user's details are found in the MongoDB collection, whether the device is iOS or Android, there's no need for platform-specific access. Therefore, if the application was ever expanded, it would most probably start with a web-based interface/dashboard to facilitate easier access in a PC/Mac cross-platform world and facilitate true device-to-device syncing.

Therefore, any serious data analytics of such personal data would be beneficial - how many people have contacts per project, average project duration, etc. such data would be appreciated by many users in the professional realm for greater efficiency.

VI. CONCLUSION

The paper presented Ontop a personal workspace manager created to be a facilitator of task management with the potential for increased productivity. It employs a Node.js/MongoDB back-end and a Flutter front-end, incorporating real-time data capture/processing and modular design for backwards/forwards compatibility and ease of support and maintenance. The current iteration of Ontop meets all necessary CRUD requirements for personal use within the professional arena as well as task management integration. Furthermore, to make Ontop more than just a static management tool but an intelligent assistant, various ML and NLP modules work in conjunction with Ontop, for example. They also help ensure user patterns become predictive, adjustable and contextually appropriate.

Future iterations of Ontop will include web-based dashboard access, client-sided analytics and encrypted cloud saves to further facilitate usability and data protection/access. Ultimately, Ontop represents a very practical addition to modern workspace management for increasingly demanding intelligent, accessible and secure productivity tools.

VII. ACKNOWLEDGEMENT

We would like to take this opportunity to express our heartfelt gratitude to our esteemed faculty for their guidance and encouragement. Their insightful feedback has been instrumental in shaping our understanding of the topic and refining our approach. We also express our deep sense of gratitude to this esteemed institution, Vishwakarma Institute of Technology, Pune for providing us with the opportunity and resources that have helped improve our research, analytical, and presentation skills. Our institutional environment has immensely contributed to shaping our academic and personal lives.

REFERENCES

[1] M. Smirnov and Y. A. Shelomentseva, "Minimalism in Interface Design: An Analysis of Tendencies and Effectiveness," Cultural Interface Framework for Responsive Applications (CIFRA), January 12, 2024.
[2] M. Bano and D. Zowghi, "A Systematic Review on the Relationship between User Involvement and System Success," Information and Software Technology (INFSOF), Vol. 5490, June 18, 2014.
[3] S. R. Chavare et al., "Smart Recommender System Using Deep Learning," Sixth International Conference on Inventive Computation Technologies [ICICT 2021], January 2021.
[4] K. Akhil et al., "Analyzing Unstructured Data: Natural Language Processing Frameworks, Challenges, Approaches, and Applications," 2024 IEEE 4th International Conference on ICT in Business Industry & Government (ICTBIG), Indore, India, 2024.
[5] V. Radu et al., “Multimodal Deep Learning for Activity and Context Recognition,” Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 1, 4, Article 157, December 2017.

[6] C. Chandrakala et al., “An intent recognition pipeline for conversational AI,” International Journal of Information Technology. 16. 10.1007/s41870-023-01642-8, December 2023.
[7] A. Aldayel and K. Alnafjan, "Challenges and Best Practices for Mobile Application Development: Review Paper," in Proceedings of the International Conference on Compute and Data Analysis (ICCDA), May 2017.
[8] N. Freitas, A. D. Rocha, and J. Barata, "Data Management in Industry: Concepts, Systematic Review and Future Directions," Journal of Intelligent Manufacturing, Springer, February 15, 2025.
[9] I. Qasim, F. Azam, M. Anwar, and H. Tufail, "Mobile User Interface Development Techniques: A Systematic Literature Review," in IEEE Information Technology, Electronics and Mobile Communication Conference (IEMCON), November 2018.
[10] J. Ferreira and J. Noble, "Agile Development Iterations and UI Design," in IEEE Xplore – Agile Conference Proceedings, September 2007.
[11] A. Khamaj and A. M. Ali, “Adapting user experience with reinforcement learning: Personalizing interfaces based on user behavior analysis in real-time,” Industrial Engineering Department, College of Engineering and Computer Science, Jazan University, Jazan, Saudi Arabia, April 2024.
[12] S. Sulkifli and A. M. Putri, "Hybrid Work Models and Freelancer Productivity: Challenges and Opportunities in the Modern Workplace," in International Conference on Government Education Management and Tourism (ICoGEMT), January 25, 2025.
[13] K. Renard, F. Cornu, Y. Emery, and D. Giauque, "The Impact of New Ways of Working on Organizations and Employees: A Systematic Review of Literature," Multidisciplinary Digital Publishing Institute (MDPI) – Administrative Sciences, April 7, 2021.
[14] Y. A. Rizky, K. Khuzaini, and S. Shaddiq, "Decluttering for Enhanced Workplace Performance: The 5S Solution," Jurnal Inovasi Ekonomi, October 2, 2023.
[15] A. Jain, “Scalable Frameworks for Cross-Platform Mobile App Development,” JETIR June 2025, Volume 12, Issue 6, June 2025.