WOXSEN UNIVERSITY

Software Engineering Case Study 2

University Management System

Sumit upadhyay
23WU0101167

Abstract:

This case study examines the limitations of existing university management systems, where professors often engage in time-consuming manual academic tasks that reduce their capacity to focus on research. The study proposes an automated framework to streamline these processes, with a particular focus on the manual updating of marks after examination copies are evaluated. By integrating machine learning techniques, the system aims to automatically process and record marks, thereby reducing repetitive workload, improving accuracy, and enabling professors to dedicate more time to academic innovation and research activities. This approach highlights the potential of automation and ML in enhancing efficiency within higher education administration.

Problem with existing solutions:

1. Slow performance and latency

- Universities with legacy ERP/management systems often rely on poorly optimized databases, old servers, or shared hosting. This causes pages like grade entry, attendance, or exam-form submissions to load slowly.
- Example: At many public state universities, staff report delays when accessing exam-related modules during peak times (end of semester), causing backups in marking & result entry. (Implicit in “Admission, Examination … huge problems” in Jai Narain Vyas University case. ([universityerp.com][1]))
- Effect: Professors waste time waiting; manual backups (spreadsheets) are used to avoid the lag, increasing chance of error.

2. Poor usability and user experience

- Interfaces are form-heavy, not intuitive. Navigation between modules (e.g. exams → marks → reports) is non-seamless. Mobile/universal access is often neglected.
- Example: Many ERP portals expect faculty to download CSV/Excel, fill marks offline, then re-upload. This adds friction. In DUIMS (Mangalore University), manual retrieval of information was “time taking” before portal deployment. ([ekspertech.com][2])
- Consequence: Low adoption; faculty prefer manual or local tools; inconsistent use of the system; more mistakes.

3. Mismatch with academic processes

- Academic workflows are complex: eligibility checks, exam revaluation, eligibility of student forms etc. Off-the-shelf ERP often doesn’t support all these.
- Example: In Jai Narain Vyas University and Mohanlal Sukhadia University, erroneous exam form submissions by ineligible students occurred because manual checks were weak. ([universityerp.com][1])
- Also, in examination module of Rajasthan University of Veterinary Sciences (RAJUVAS), institution had manual evaluation, delayed compilation etc. The ERP introduced had to cover the entire exam cycle to fix that. ([universityerp.com][1])

4. Data quality and migration failures

- Legacy systems/data stored in spreadsheets, paper logs, localized copies. This leads to missing, duplicated, or inconsistent records when porting to ERP.
- Example: In Mangalore University (DUIMS), data flow was asynchronous and info "not readily available" because of scattered, manual data. ([ekspertech.com][2])
- Also vague/incomplete student data affected merit lists, enrolment numbers.

5. Integration and interoperability gaps

- ERP modules aren't always well-integrated or don't provide APIs. Different departments use separate software for attendance, exam, finance. No central integration causes double entry and errors.
- Example: In many Indian universities, separate software is used for finance vs exam vs admissions vs student management. In GGU, prior to IUMS, some modules used FoxPro, others Excel. ([universityerp.com][1])
- Consequence: Data silos, delays in report generation, manual consolidation of data.

6. Low change management and training

- Faculty and staff often untrained or resistant. Old habits of manual sheets, paper forms persist. Systems deployed without adequate hand-holding.
- Example: In Sohar University (outside India but comparable), significant improvements came when top management support + training + interdepartmental cooperation were strong. Without that, performance stayed poor. ([ijmar.org][3])
- In Indian context, several ERP implementations stall because users don't understand modules well, find interface confusing.

7. Security, compliance, and privacy issues

- Student records, exam marks are sensitive. Poor access control, weak encryption, unsecure hosting can cause data leaks.
- Example: In DR-NTR University (case by Eksper Technologies), software had to pass rigorous security audits before being hosted, because data was sensitive. ([ekspertech.com][4])
- Also regulatory compliance (with UGC, state / national data protection norms) often neglected.

8. Vendor lock-in and high customization cost

- To support academic specific workflows, institutions demand heavy customization. Vendors charge high fees for bespoke features. Custom code is harder to maintain.
- Example: Sohar University needed high level of customization because its teaching / exam processes differed from "standard best practice" embedded in ERPs. ([ijmar.org][3])
- Universities then face recurring cost for upgrades, maintenance; sometimes certain modules break in updates because custom parts unsupported.

References of above casestudies

[1]: https://universityerp.com/clients/Case_studies.php?utm_source=chatgpt.com "University ERP :: Case Study"

[2]: https://www.ekspertech.com/case_studies_uom.php?utm_source=chatgpt.com "Eksper Technologies Limited | India's Leading Online ERP solutions For School, College and Universities"

[3]: https://ijmar.org/v3n1/16-003.html?utm_source=chatgpt.com "Critical Success Factors across the Stages of ERP System Implementation in Sohar University: A Case Study| Shatat | International journal of management and applied research"

[4]: https://ekspertech.com/case_studies_uom.php?utm_source=chatgpt.com "Eksper Technologies Limited | India's Leading Online ERP solutions For School, College and Universities"

Our Solution:

Woxite is an innovative solution designed to streamline administrative tasks and improve user experience for both professors and students within an academic ERP (Enterprise Resource Planning) system. By integrating several key features, Woxite aims to boost productivity, enhance fairness, and simplify daily academic processes.

AI-Powered Chatbot for Enhanced Navigation

A core feature of Woxite is its integrated AI chatbot. This tool is designed to significantly improve user navigation and productivity for professors. Instead of manually searching through the entire ERP system to find a specific function, professors can simply interact with the chatbot. The chatbot will understand their request and redirect them to the exact function they need, saving valuable time and reducing frustration.

Accurate Handwriting-to-Text Conversion

Woxite includes a groundbreaking handwriting-to-text conversion feature. This system is designed to convert handwritten notes and exam papers into legible, digital text. To achieve a high degree of accuracy, each student will have their own personal database of their unique handwriting. Over time, as the system learns and adapts, it can achieve up to 99% accuracy in converting a student's handwriting.

This feature can also be used to digitize exam papers. Once converted, the answers can be broadcast on a portal without the student's name to ensure impartiality. This allows teachers to

evaluate the work without bias, providing marks and remarks that students can view to understand their mistakes.

Automated Facial Recognition for Attendance

To further increase efficiency, Woxite incorporates a facial recognition system for attendance tracking. This feature automates the process of taking attendance, eliminating the need for manual roll calls and speeding up the daily routine for both students and professors. This not only saves time but also provides a reliable and fast method for monitoring student presence in class.

Methodology

The software is being built as a foundational ERP system designed to handle essential business processes. Its core purpose is to provide a robust platform for organizational management while prioritizing high security and usability from the ground up.

Advanced Security and Authentication

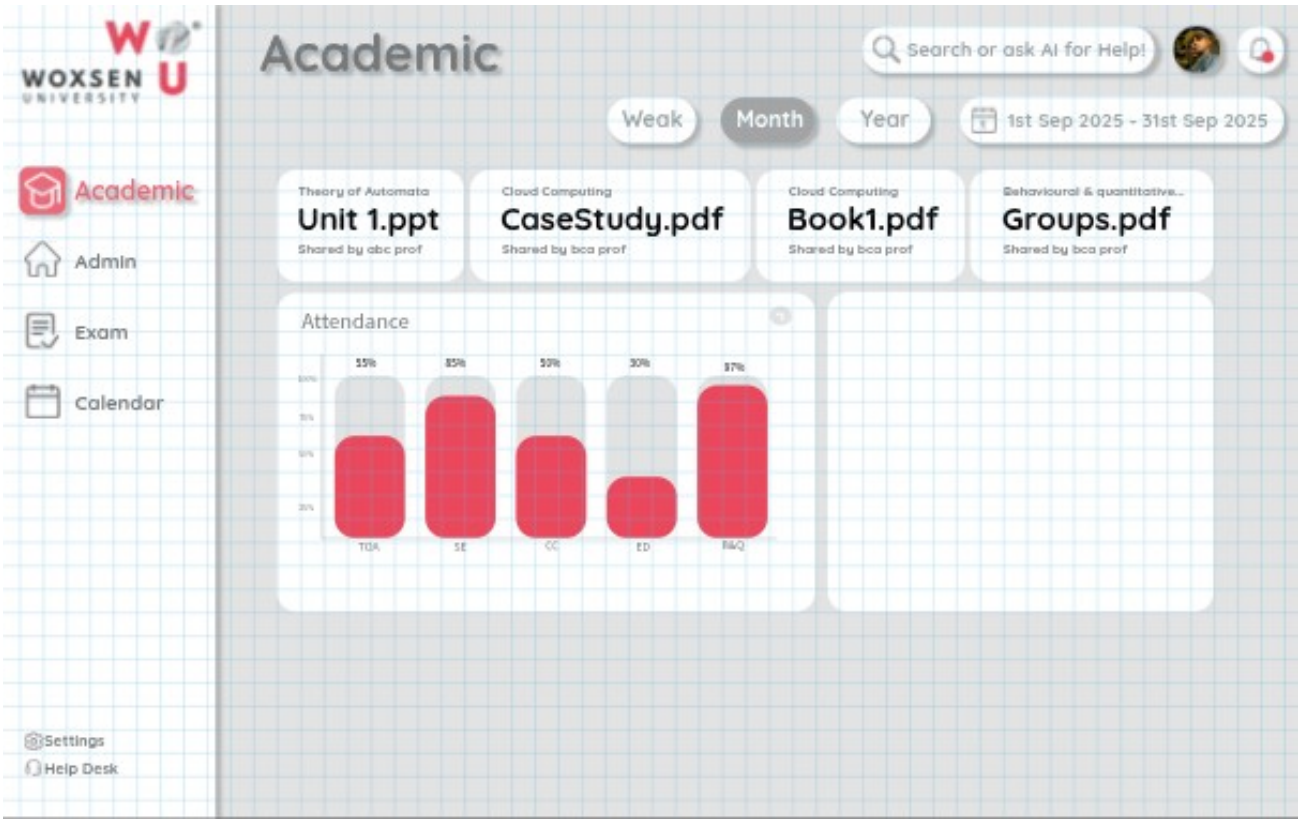
A primary focus of this ERP is its security architecture. Key features include:

- Robust Authentication System: The system will feature advanced authentication capabilities, including a multi-role recognition system. This allows for precise access control, ensuring that users can only access information and functions relevant to their specific role within the organization.
- Data Protection: To prevent unauthorized data dissemination, the software will have built-in security measures that prohibit users from downloading web pages or sensitive information, thereby protecting the integrity of the data.

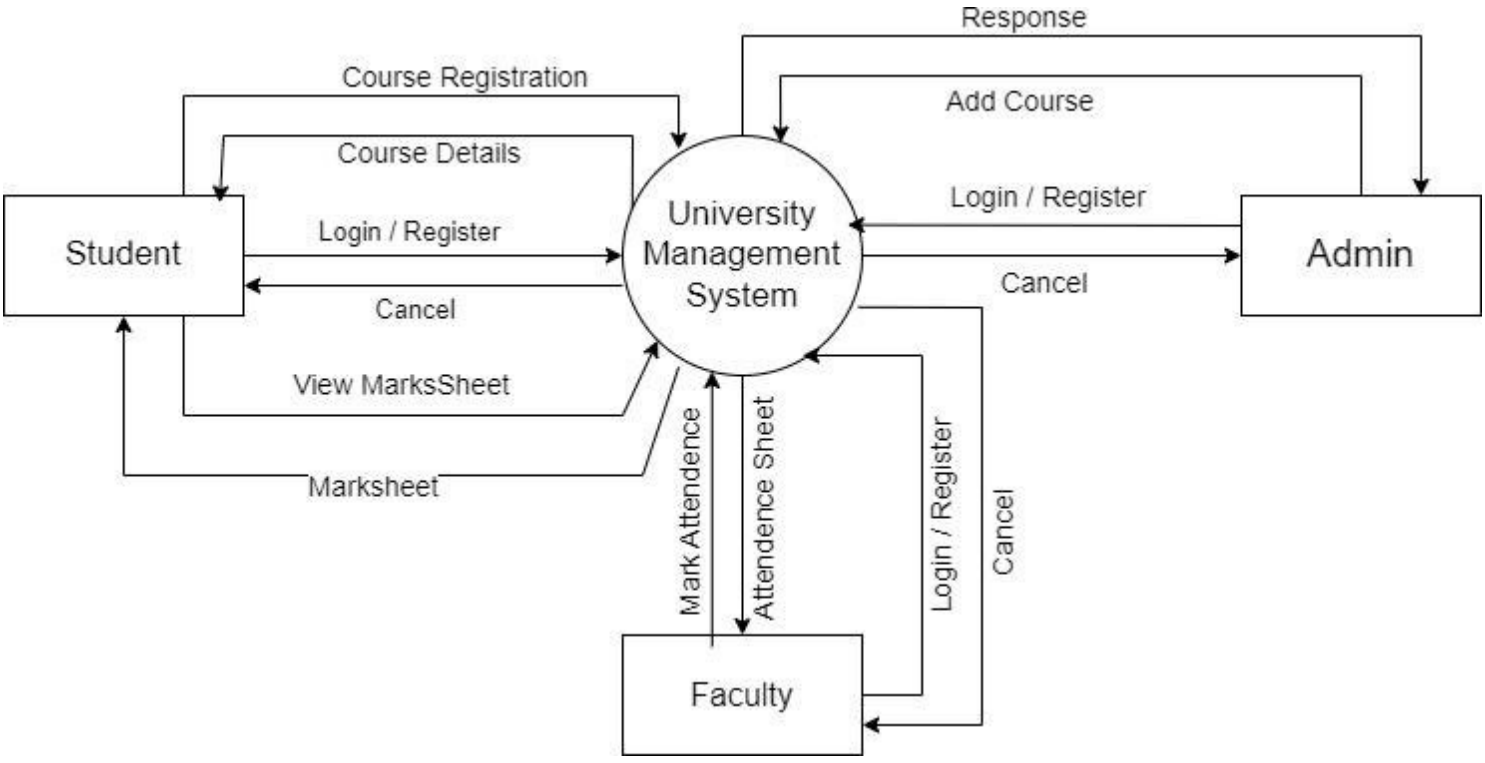
Superior User Experience (UI/UX)

Recognizing the importance of a smooth workflow, the ERP software is being designed with a strong emphasis on a good UI (User Interface) and UX (User Experience) design. The goal is to create a system that is not only powerful but also intuitive and easy for all users to navigate, which will enhance productivity and adoption rates.

Interface:



Basic Architecture



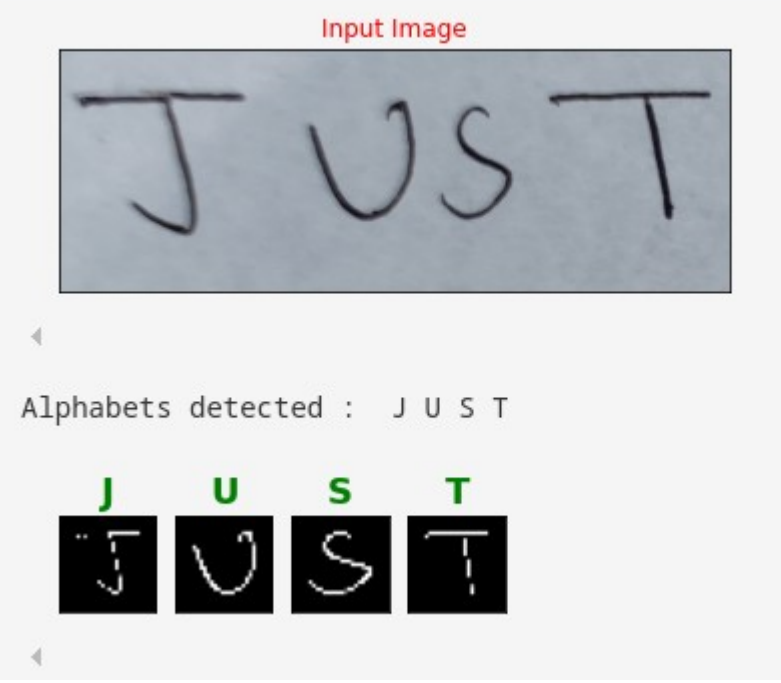
Accurate Handwriting-to-Text Conversion

The system uses TensorFlow as its foundation for accurate handwriting detection and matplotlib, numpy and pandas assisting the system for visulation and understanding. The process is integrated into the student registration flow and is designed to create a profile of each student's unique handwriting style.

Here's how the system works:

1. **Handwriting Sample Collection:** During registration, the system provides a series of sentences to the student.
2. **Image Capture:** The student writes these sentences by hand and presents the written sample to a camera. The camera captures an image of the handwritten text.
3. **Real-time Comparison:** The system immediately compares the captured handwritten image with the original digital text it provided.
4. **Character Recognition Training:** The process continues until the system has a complete understanding of the student's handwriting for all English alphabet characters and numbers. This ensures that the agent receives a comprehensive and accurate representation of the student's unique handwriting style.
5. **Agent Feedback:** The system's output gives the agent a clear idea of the student's handwriting, allowing for future use cases like document verification.

This process ensures that the system can accurately recognize and categorize each student's handwriting for later use within the ERP.



Prototype

Snipit of the function used to learn the handwriting of the student

```
def unseendata_test(filepath):
    image = cv2.imread(filepath)
    blur_image=cv2.medianBlur(image,7)

    grey = cv2.cvtColor(blur_image, cv2.COLOR_BGR2GRAY)

    thresh = cv2.adaptiveThreshold(grey,200,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV,41,25)

    contours,hierarchy= cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    preprocessed_digits = []


    boundingBoxes = [cv2.boundingRect(c) for c in contours]
    (contours, boundingBoxes) = zip(*sorted(zip(contours, boundingBoxes),key=lambda b:b[1][0], reverse=False))


    for c in contours:
        x,y,w,h = cv2.boundingRect(c)

        cv2.rectangle(blur_image, (x,y), (x+w, y+h), color=(255, 0, 0), thickness=2)

        digit = thresh[y:y+h, x:x+w]

        resized_digit = cv2.resize(digit, (18,18))

        padded_digit = np.pad(resized_digit, ((5,5),(5,5)), "constant", constant_values=0)

        preprocessed_digits.append(padded_digit)
    plt.xticks([])
    plt.yticks([])
    plt.title("Input Image",color='red')
    plt.imshow(image, cmap="gray")
    plt.show()

    inp = np.array(preprocessed_digits)
    figr=plt.figure(figsize=(len(inp),4))
    i=1
    alphabets_unseen=[]
    for digit in preprocessed_digits:
        [prediction] = nn_model.predict(digit.reshape(1, 28, 28, 1)/255.)
        pred=alpha[np.argmax(prediction)]
        alphabets_unseen.append(pred)
        figr.add_subplot(1,len(inp),i)
        i+=1
        plt.xticks([])
        plt.yticks([])
        plt.imshow(digit.reshape(28, 28), cmap="gray")
        plt.title(pred,color='green',fontsize=18,fontweight="bold")
    print("Alphabets detected : " ,*alphabets_unseen)
```

Automated Facial Recognition for Attendance

automated facial recognition system for attendance is a great project that combines computer vision, machine learning, and basic software development. Here's a high-level overview of the process and the key components you'll need.

Software & Libraries (Python is highly recommended):

- **OpenCV:** An open-source library for computer vision tasks. It's essential for accessing the camera, capturing video frames, and performing image processing.
- **`face_recognition`:** A Python library built on top of `dlib` that simplifies facial recognition tasks. It can detect faces, identify facial landmarks, and encode faces into numerical representations (embeddings).
- **`numpy`:** A library for numerical operations, used for handling the face encodings and other data.
- **pandas:** A data analysis library that's useful for managing and logging attendance data in a structured format like a CSV file.
- **Database:** A simple CSV file is fine for small projects, but for a more robust system, you would use a database like SQLite, MySQL, or a NoSQL database like Firestore.

The Step-by-Step Process

Step 1: Data Gathering and Enrollment

This is the "training" phase where you build the database of known faces.

- **Capture Images:** Take multiple pictures of each person you want to enroll.
- **Create Embeddings:** Use a library like `face_recognition` to process each image and convert the face into a unique numerical vector (an "embedding"). This embedding is a mathematical representation of the face's features.
- **Store Data:** Save these embeddings in a data store (e.g., a dictionary, a CSV file, or a database) along with the person's name or ID.

Step 2: Real-Time Recognition

This is the "attendance" phase where the system actively checks for faces.

- **Access Camera:** Use OpenCV to access the webcam and start a video stream.
- **Process Frames:** In a continuous loop, the system captures each frame from the video feed.
- **Detect Faces:** For each frame, it detects all faces present using a facial detection algorithm.
- **Create Embeddings (Real-time):** For each detected face, it generates a new embedding.

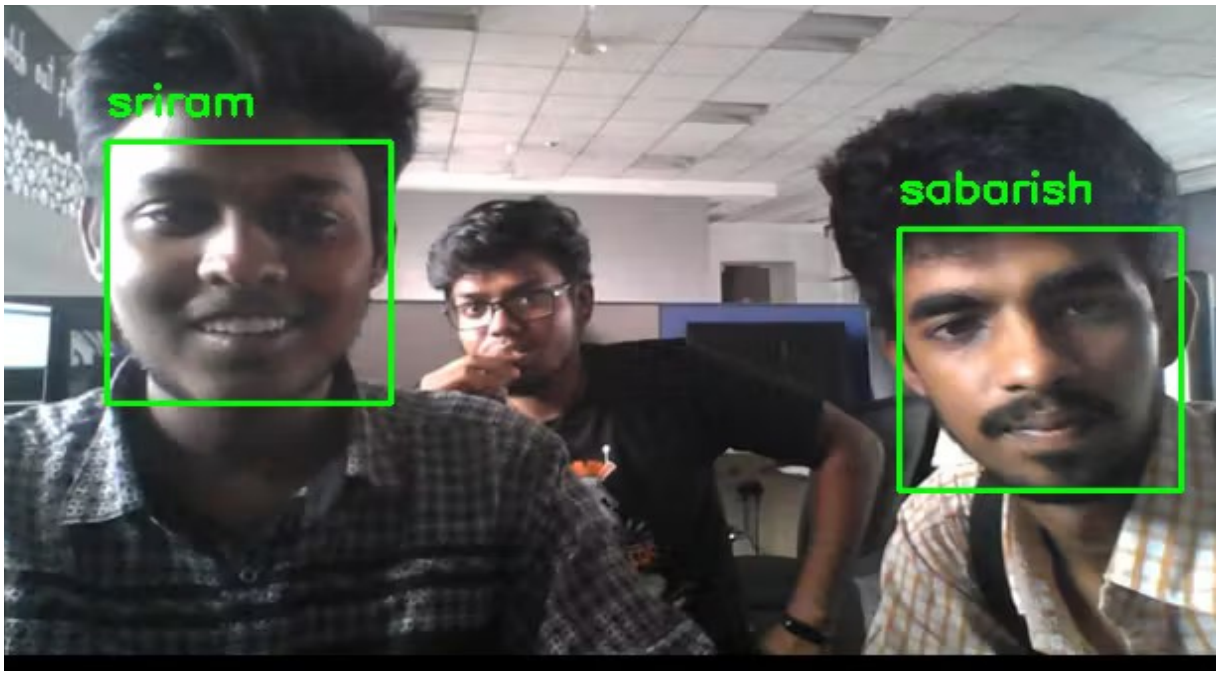
- **Compare and Match:** The system compares the new embedding to all the embeddings stored in your database from Step 1. It calculates the "distance" between the vectors to find the closest match. If the distance is below a certain threshold, a match is found.
- **Log Attendance:** If a match is found, the system logs the person's name and the current timestamp into your attendance record.

Step 3: Building a User Interface (Optional but Recommended)

For a more user-friendly system, you can build a front-end interface.

- **Flask or Django (Python):** These web frameworks can be used to create a simple web application. The front-end (HTML, CSS, JavaScript) can display the video feed, show attendance logs, and provide controls for enrolling new people.
- **GUI Libraries:** Libraries like Tkinter or PyQt can be used to build a desktop application with a graphical user interface.

This video from YouTube, "Face Recognition Based Smart Attendance System With Web Apps Using Machine Learning", provides a practical example of a project similar to what you're describing.



Conclusion

Based on our discussion, your proposed solution is a comprehensive system designed to significantly streamline academic management for both faculty and students. By automating key processes like handwriting detection for registration and facial recognition for attendance, the ERP aims to free up valuable time that can be redirected toward more critical academic activities such as research and projects.

It is clear that this is a forward-thinking initiative. The project is currently in active development, with an expected practical implementation date of April 2026. This timeline allows for the necessary development, testing, and refinement to ensure the system is robust, accurate, and ready for real-world use.