

AI-Based Smart Event Management System with Event Risk Prediction and Audio Reminders

Ms. Saundarya K Khichade¹ and Ms. Asha Mane²

Sadhu Vaswani Institute of Management Studies for Girls, Pune

Savitribai Phule Pune University

saundaryakhichade21@gmail.com and ashamane90@gmail.com

Abstract: *Anyone who has organised an event knows how quickly deadlines, budgets and supplier promises pile up, and how easily a warning sign gets lost when the records are spread across notebooks and chats. This paper describes a web prototype that gathers events, tasks and reminders in a single application, labels every event as Low, Medium or High risk, and reads due reminders out loud through the browser. It is written in Python with Django and stores its data in SQLite. The risk label comes from a Random Forest classifier fed with five values: the days left before the event, the number of unfinished tasks, how many of those are high priority, whether the vendor has confirmed, and the percentage of the budget already spent. Reminders follow plain date-and-time checks, and the Web Speech Synthesis API handles the spoken part. Development and testing used synthetic data only: 100 events, 605 tasks, 200 reminders and 100 labelled risk records. On a test split of 25 records the Random Forest scored 84% accuracy, compared with 40% for a baseline that always guesses the most common class. Since the data are artificial and the test split is tiny, these numbers demonstrate that the idea is workable; they do not predict how the system would perform on real events. Each module is described below, followed by suggestions for further work.*

Keywords: Event Management, Artificial Intelligence, Random Forest, Risk Prediction, Django, Task Monitoring, Audio Reminders, Machine Learning

I. INTRODUCTION

Weddings, birthday parties, conferences and office meetings differ in scale, yet each one comes down to the same thing: many small jobs that all have to be finished by a date that will not move. If urgent jobs remain open, or a supplier has not yet said yes, the organiser often realises it only when time is nearly gone. An ordinary event register keeps the details but never warns anyone that an event is falling behind.

In this work we built a web application where organisers handle their events and tasks and, alongside them, see a machine-learning estimate of how much risk each event currently carries. The estimate is intended to draw attention to an event, not to pass judgement on it. We chose Random Forest because it copes well with small tables of data and can pick up non-linear effects and interactions between the indicators. Convolutional networks were left out, as the system does not process images. Reminders are spoken through the browser's Web Speech Synthesis API, and they only play while the application is open and the device permits sound.

II. PROBLEM STATEMENT

Organizers need one place to store event details, follow their tasks and spot the events that are heading for trouble. A basic register cannot express risk, and moving risk assessment to a different page would make the tool clumsy to use. This paper therefore asks how event handling, task tracking, easy-to-read risk categories and time-triggered spoken reminders can sit together in one Django application, while model performance is reported truthfully and no claim goes beyond what the data can support.



III. LITERATURE REVIEW

Breiman (2001) introduced Random Forest, an ensemble method in which many decision trees vote on the outcome, which makes classification steadier and lets it handle non-linear relationships among features. The implementation used in this project for training and evaluation comes from scikit-learn, described by Pedregosa et al. (2011). For the web side, Holovaty and Kaplan-Moss (2009) explain Django, which provides the data models, views and authentication here, and the official Django documentation (Django Software Foundation, 2024) shows how those pieces connect to the database. The spoken output relies on the Web Speech API as documented by MDN Contributors (2024); it turns text into audio in the browser and has no role in the prediction itself.

IV. RESEARCH GAP

A good number of event-management tools are available, but most of them only record information and expect the organiser to work out the risk personally. The present prototype tries to fill that gap by combining three things in one application: indicators of event risk, three easy categories (Low, Medium, High) and audio reminders that fire at chosen dates and times. Should the trained model turn out to be more useful than a hand-written rule, that would be an extra contribution of this work, but checking this on real events is left for the future.

V. PROPOSED METHODOLOGY

The first step was to define Django models for events, tasks and reminders. The CSV files were then cleaned and loaded into the database, and the features needed by the model were calculated for each event. Several classifiers were trained on the risk data and compared with one another. The model that was selected sits behind a prediction function, which the views call and whose result is shown on the dashboard. For reminders, the application looks at the date and time of each one and, once a reminder falls due, hands its text to the Web Speech Synthesis API.

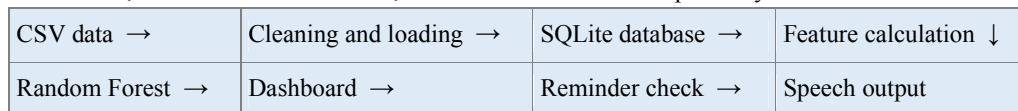


Figure 1: Modules and information flow

VI. SYSTEM MODULES AND TECHNOLOGIES

Module	Main function	Technology / implementation
Login and access	Signs in organisers and administrators and manages who can access what	Django authentication
Event management	Keeps the name, type, date, venue, guest count, budget, vendor, confirmation and status	Django models, forms and admin
Task management	Connects each task to an event and records its deadline, priority and completion status	Task model with event foreign key
Dashboard	Displays event dates, days remaining, pending tasks, risk category and reminder text	Django views and HTML/CSS templates
Risk prediction	Predicts Low, Medium or High from the event and task indicators	Scikit-learn RandomForestClassifier
Reminder management	Holds each reminder's date, time, title, message and active/triggered flags	Django reminder model and view logic



Module	Main function	Technology / implementation
Audio reminders	Reads the reminder message aloud in the browser	JavaScript Web Speech Synthesis
Database	Stores events, tasks, reminders and users	SQLite (prototype)

VII. DATASET

Every record used in this study is synthetic. Four CSV files were generated: Events.csv with 100 events, Tasks.csv with 605 tasks, Reminders.csv with 200 reminders and Event_risk_training.csv with 100 labelled records. The fields include event type, date, location, guest count, budget, budget used, vendor confirmation, pending tasks, task priority, reminder schedule and the risk label (Low, Medium or High). The labels were assigned by predefined rules for demonstration, and the files were then loaded into the Django application.

SQLite held the events, tasks and reminders, whereas the risk file served only to train the model. All records were checked for missing values and correct data types. Because the labels were produced by rules and not taken from real events, the results show merely how closely the model can reproduce those labels. A high accuracy on such data says little about real events, and a fair evaluation will need larger sets of genuine events.

VIII. MODEL AND TRAINING

A. Input features

Feature	Definition	Expected relevance
Days_remaining	Number of days between the observation date and the event date	A shorter lead time leaves less room to recover
Pending_tasks	Count of tasks that are still not completed	More unfinished work points to more pressure
High_priority_tasks	Pending tasks that carry a high priority	Reflects urgent work that is still outstanding
Vendor_confirmed	Binary flag showing whether the vendor has confirmed	An unconfirmed supplier adds uncertainty to operations
Budget_used	$100 \times \text{budget used} / \text{total budget}$	Heavy spending can point to budget pressure

B. Random Forest model

The classifier was built with scikit-learn and stored with joblib once training was finished. It takes a vector made of the five features listed above and returns one of three classes. The data were split into training and test portions using a fixed random seed, so the experiment can be repeated with the same outcome. If the classes are unbalanced or the same event appears more than once, stratified splitting and grouping by event would be sensible, and the saved model should only ever be used with the same preprocessing that was applied during training.

IX. MODEL EVALUATION

Four approaches were compared: a majority-class baseline, a Decision Tree, Logistic Regression and Random Forest. The baseline reached 40% accuracy. The Decision Tree achieved 84% with a macro F1 of 0.8440, Logistic Regression achieved 76% with a macro F1 of 0.7546, and Random Forest achieved 84% with a macro F1 of 0.8397 and a balanced accuracy of 85.24%.



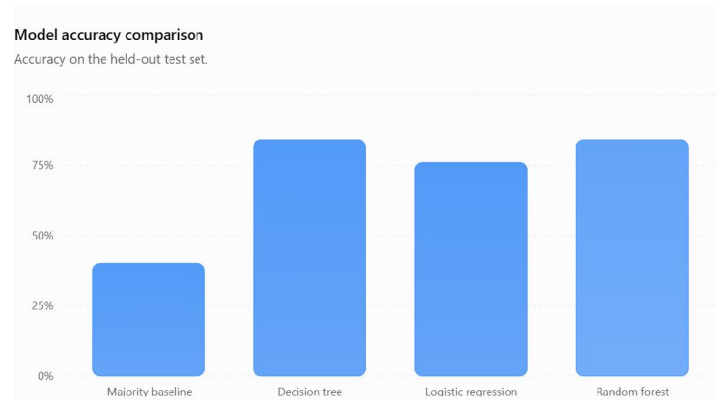


Figure 2: Accuracy of the four models on the held-out test set

Decision Tree and Random Forest finish level on accuracy, and both are far ahead of the baseline. The Decision Tree has a marginally higher macro F1, while the Random Forest has a balanced accuracy of 85.24%.

Random Forest class-wise results

Risk category	Recall (correctly identified)	Test samples
Low	100%	8
Medium	70%	10
High	85.71%	7

Table: Recall by class

Actual / Predicted	Low	Medium	High
Low	8	0	0
Medium	1	7	2
High	0	1	6

Table: Confusion matrix (rows = actual class, columns = predicted class)

The Random Forest found all eight Low-risk events and identified most of the High-risk ones correctly. Medium risk was the hardest category, with three of the ten events pushed into a neighbouring class. The results look promising, but the test set holds just 25 records, so a change in only a few predictions would shift the percentages visibly.



X. VISUAL RESULTS AND APPLICATION SCREENSHOTS

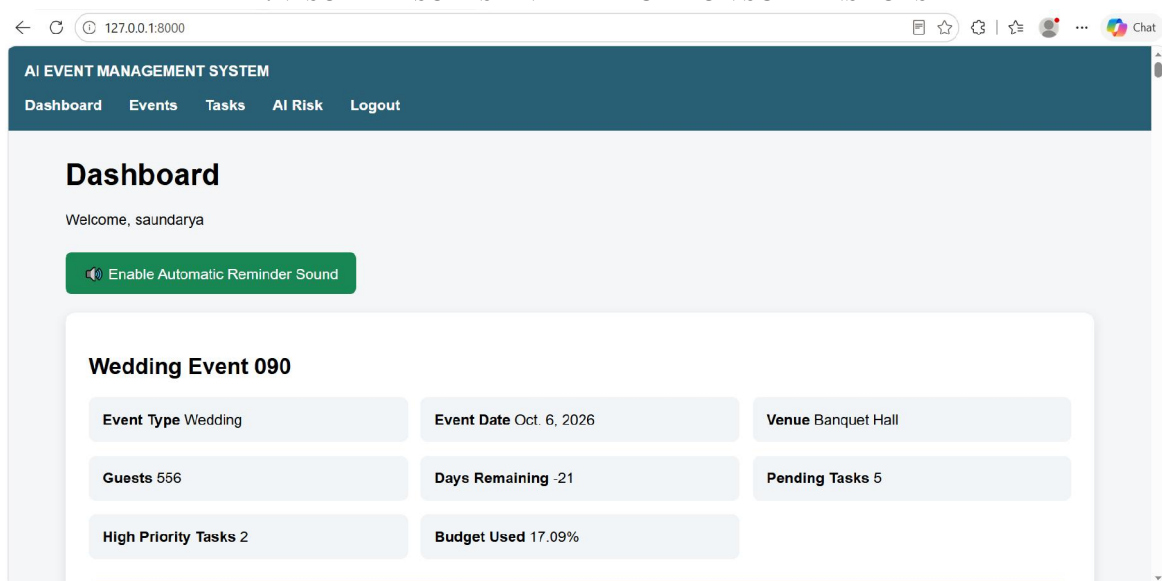


Fig 10.1

The above screenshot is of main Dashboard which displays the event details, event type, date, venue, guest count, days remaining, pending tasks, high-priority tasks, and budget utilization, along with an option to enable automatic reminder sounds.

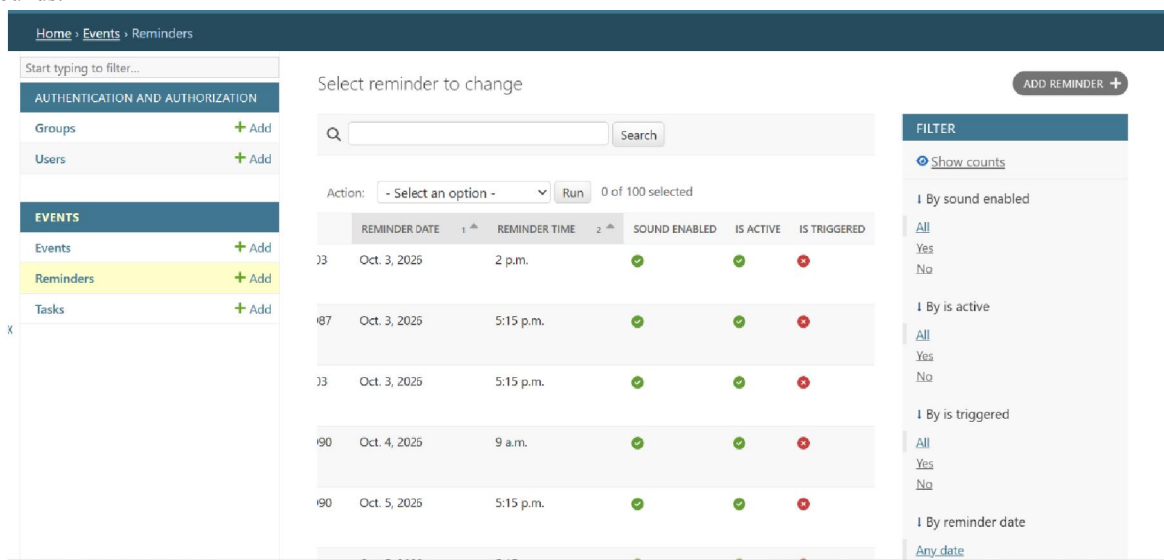


Fig 10.2

The above screenshot shows reminder page which allows the administrator to add, search, filter, and manage reminders, including reminder dates, times, sound settings, and activation status.



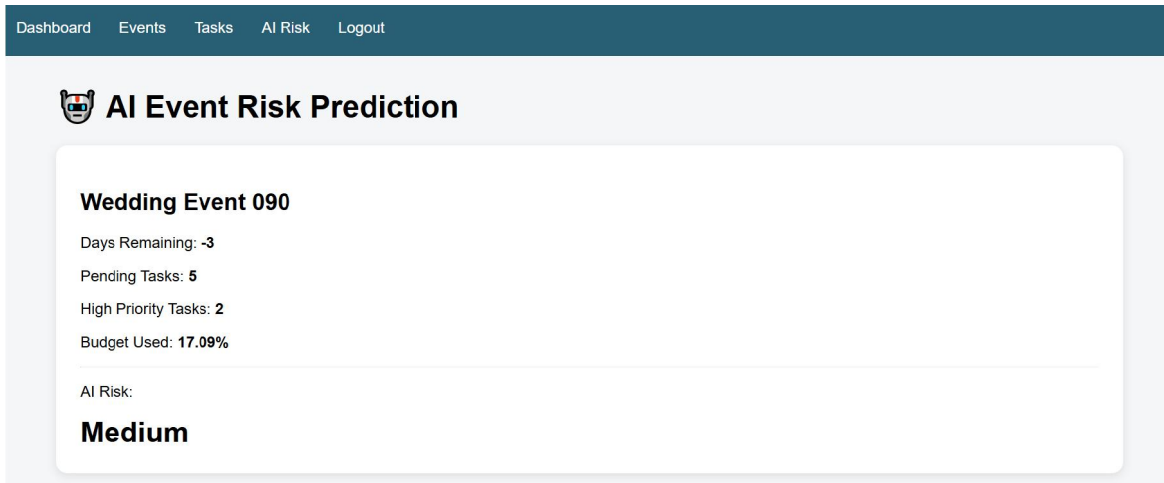


Fig 10.3

The above screenshot checks whether events display Low, Medium, or High risk and verifies that the predictions correspond to your saved event and task data.

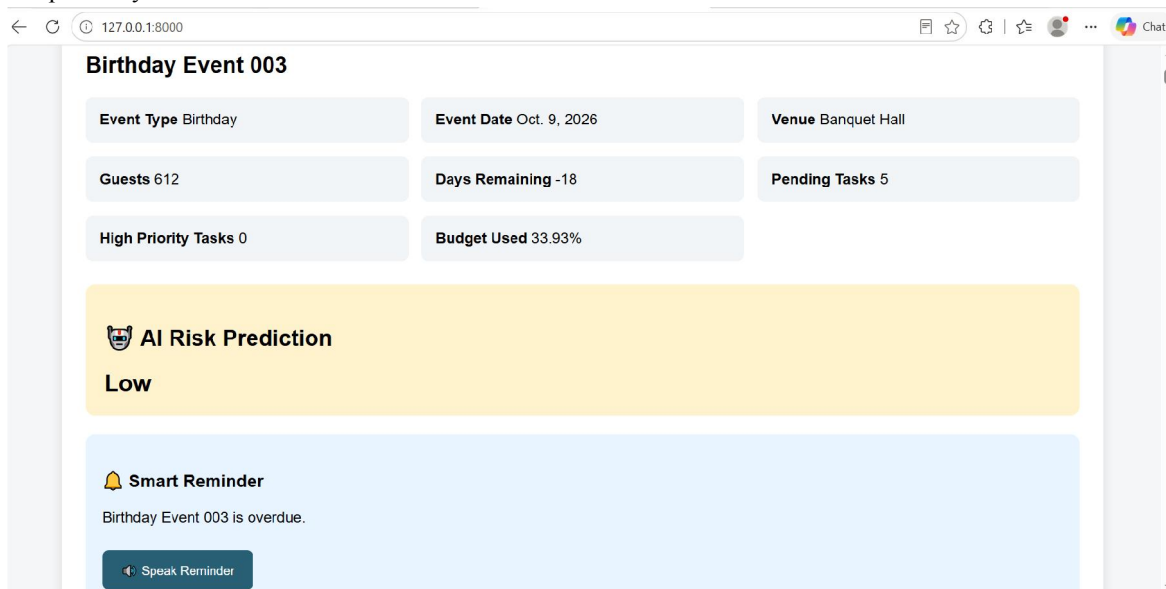


Fig 10.4

The above screenshot shows the risk and smart reminder for Birthday Event, which consists of event type, guests, event dates, pending tasks, etc.



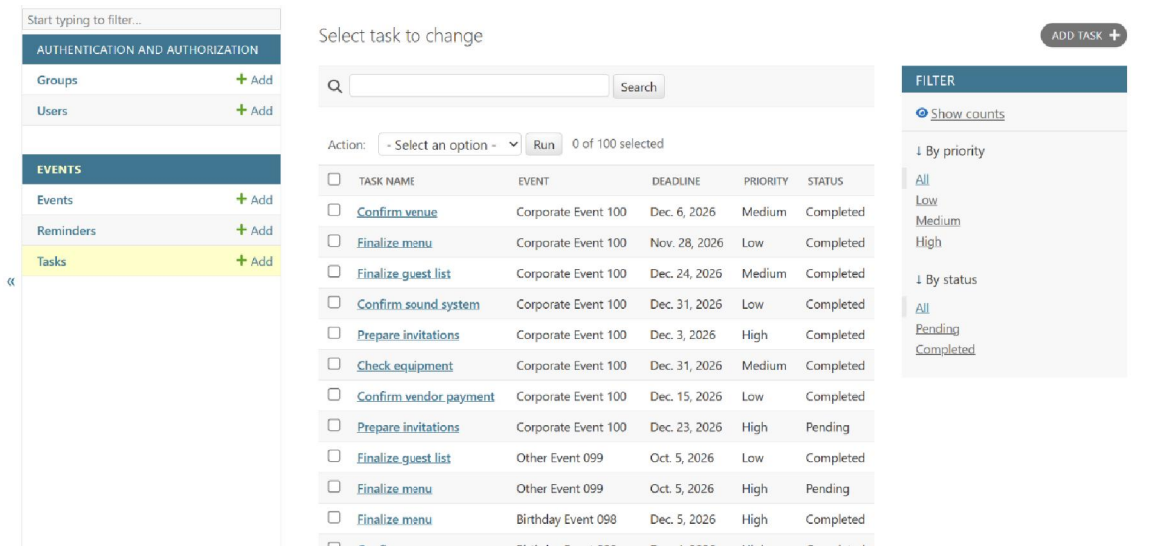


Fig 10.5

The above screenshot shows the number of tasks of an Event.

XI. FINDINGS

The prototype demonstrates that event management, task tracking, risk prediction and spoken reminders can live together in a single web application. On the synthetic test set the Random Forest reached 84% accuracy, using pending tasks, days remaining, vendor confirmation and budget use as its signals of risk. Audio reminders can also help users keep track of what they have committed to. Since every test relied on synthetic data, trials with real events are needed before anything can be said about practical usefulness.

XII. LIMITATIONS

The dataset is synthetic and labelled by rules, so it may not resemble real events, and the 84% accuracy might look different on other data. The test set is also small. The system depends on organisers entering correct information, and the audio reminders need an open browser, working speakers and an application that is actually in use. In addition, the model has no way of anticipating outside events such as bad weather or a supplier pulling out.

XIII. CONCLUSION

This paper presented a prototype for event management that brings together Django-based handling of events and tasks, an SQLite store, Random Forest risk prediction into Low, Medium and High, and spoken reminders in the browser. The chosen features give a rough picture of how well prepared an event is. The main contribution lies in putting these capabilities into one planning tool. A trained model cannot promise correct predictions, so future work should collect real event data, test on larger samples and compare the model against a simple rule-based approach.

REFERENCES

1. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.
2. Django Software Foundation. (2024). Django documentation. <https://docs.djangoproject.com>
3. Holovaty, A., & Kaplan-Moss, J. (2009). The definitive guide to Django: Web development done right (2nd ed.). Apress.
4. MDN Contributors. (2024). Web Speech API. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API



5. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
6. Scikit-learn Developers. (2024). RandomForestClassifier. Scikit-learn documentation. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

