ISSN (e): 2250 – 3005 || Volume, 16 || Issue, 6 || June – 2026 ||
International Journal of Computational Engineering Research (IJCER)

Smart Chatbot for College Enquiry - Askace

Dr.K.Prem Kumar[1]
Professor & HOD Department of AI & ML
ACE Engineering College (Autonomous), Hyderabad, Telangana, India.

A.Nikitha[2]
Dept.of AI & ML
ACE Engineering College (Autonomous)
Hyderabad, Telangana, India

G. Chandrika[3]
Dept.of AI & ML
ACE Engineering College (Autonomous)
Hyderabad, Telangana, India

Y. Rahul[4]
Dept.of AI & ML
ACE Engineering College (Autonomous)
Hyderabad,Telengana,India

V. Lalith Kumar[5]
Dept.of AI & ML
ACE Engineering College (Autonomous)
Hyderabad, Telengana, India

V.Sanjay Kumar[6]
Dept.of AI & ML
ACE Engineering College (Autonomous)
Hyderabad, Telengana, India

ABSTRACT
Modern engineering institutions face growing pressure to centralise academic resources, simplify administrative workflows, and improve communication between students, faculty, and staff. This paper presents the design, architecture, and implementation of a feature-rich mobile application developed for ACE Engineering College, Hyderabad. The application is built with the Flutter cross-platform framework and powered by the Firebase backend-as-a-service ecosystem. The system incorporates a five-tab navigation model that offers a public-facing information portal, a dedicated departments hub, an AI-driven conversational assistant (AskAce), a placements information centre, and a role-based authentication gateway. Three distinct dashboards cater to students, faculty members, and administrators with tailored functionality. The application additionally integrates a speech-to-text interface and a markdown-rendering AI chatbot connected to a cloud-hosted API endpoint. Real-world deployment outcomes demonstrate significant improvements in information accessibility and administrative efficiency. The findings suggest that Flutter combined with Firebase provides a rapid, scalable, and cost-effective pathway for developing smart campus solutions.
KEYWORDS: *Flutter, Firebase, Mobile Application, Smart Campus, Role-based Access Control, AskAce AI Chatbot, Cloud Firestore, Cross-platform Development.*

Date of Submission: 08-06-2026 Date of acceptance: 18-06-2026

I. INTRODUCTION

The proliferation of smartphone usage among college students and faculty has fundamentally altered how academic institutions communicate and deliver services. Traditional notice boards, printed timetables, and face-to-face administrative interactions are progressively being replaced by digital alternatives that offer immediacy, convenience, and scalability. Engineering colleges in particular must manage a heterogeneous set of stakeholders, including undergraduate students spread across multiple departments, teaching and non-teaching staff, placement coordinators, and institutional administrators, all of whom have distinct information needs.

ACE Engineering College, located in Hyderabad, Telangana, offers programmes across ten engineering disciplines. Managing communications, academic records, placement activities, and departmental updates

across this breadth of stakeholders using conventional means proved increasingly inefficient. This situation motivated the development of a unified, role-aware mobile application that consolidates the institution's digital presence into a single platform.

Flutter, introduced by Google, is a UI toolkit that compiles natively for Android, iOS, and web from a single codebase written in Dart. Firebase, also a Google product, provides a suite of backend services including authentication, a real-time NoSQL database (Cloud Firestore), serverless cloud functions, and hosting. Together, they enable rapid application development with production-grade reliability, making them well-suited for resource-constrained institutional projects.

This paper documents the full software design and development journey of the ACE Engineering College application, covering requirements analysis, system architecture, feature implementation across all five navigation tabs, the AI chatbot integration, and role-specific dashboard design. The remainder of the paper is organised as follows: Section II reviews related literature; Section III outlines the system architecture; Section IV describes each functional module; Section V presents implementation specifics; Section VI reports results and discussion; and Section VII concludes with future directions.

II. LITERATURE REVIEW

Several studies have examined the role of mobile applications in educational settings. Ally [1] established that mobile devices enable anywhere, anytime learning and that well-designed interfaces reduce cognitive load for students accessing academic content. Traxler [2] argued that institution-specific mobile tools yield higher engagement than generic learning management systems because they are tailored to local workflow expectations.

Cross-platform frameworks have attracted growing attention in the software engineering community. Biorn-Hansen et al.

[3] conducted a systematic review of cross-platform mobile development approaches and concluded that Flutter's ahead-of-time compilation and widget-level rendering give it a performance edge over hybrid frameworks such as Ionic or Cordova. Stoll et al. [4] compared Flutter with React Native across 12 dimensions and found Flutter superior in animation smoothness and cold-start time.

Firebase as a backend platform for educational applications has been studied by Mohammadi [5], who demonstrated that Cloud Firestore's real-time synchronisation significantly reduces the development effort for collaborative academic tools. Role-based access control (RBAC) in mobile applications is discussed by Sandhu et al. [6], who provide a formal model that maps user roles to permission sets — a model directly applicable to student, faculty, and administrator differentiation.

AI chatbots in campus environments have been explored extensively. Winkler and Söllner [7] reviewed 35 chatbot deployments in higher education and found that conversational agents reduce repetitive enquiry load on administrative staff by up to 40 percent. The integration of speech-to-text interfaces further lowers barriers for users with limited typing comfort, as shown by Munteanu et al. [8]. The present work builds on these findings by embedding an AI assistant directly within the campus application rather than deploying it as a standalone tool.

III. SYSTEM ARCHITECTURE

3.1 Architectural Overview

The application follows a layered client-server architecture. The client tier is a Flutter application compiled into native Android and iOS binaries. The service tier consists of Firebase Auth for identity management, Cloud Firestore as the primary data store, and Firebase Cloud Functions for server-side logic.

An additional external service, the AskAce backend deployed on Render (https://askace-backend.onrender.com/ask), handles natural language query processing. Figure 1 illustrates the overall system topology.

3.2 Entry and Navigation

Application startup is handled by main.dart, which resolves the splash screen and proceeds to the MyHomePage widget after a two-second animated logo display. Navigation is implemented as a PageView controlled by a floating bottom navigation bar styled with the institution's brand colour (#0189C8) and rounded corners of radius 35 pixels. The bar employs a bubble animation on the active tab. Five tabs are defined: Home (index 0), Departments (index 1), AskAce AI (index 2), Placements (index 3), and Login (index 4). The AppBar persistently displays the college logo and title, with a supplementary Feed button that navigates to a gated announcements page.

3.3 Data Layer

All dynamic content is stored in Cloud Firestore and organised into top-level collections: Home, Departments (one sub-collection per department), Placements, Users, Feeds, and PYQ. This schema allows administrators to update any content through the in-app admin dashboard without requiring a new application release. Firestore security rules enforce role-based read and write permissions, ensuring that students cannot modify records outside their own profile.

3.4 Authentication and Role Resolution

Firebase Authentication manages user credentials through email/password sign-in. Upon successful authentication, the application queries the Users Firestore collection, retrieves the authenticated user's document, and inspects the role field. Depending on whether the value is student, faculty, or admin, the corresponding dashboard is rendered. A forgot password flow is implemented through Firebase's password reset email mechanism.

IV. FUNCTIONAL MODULE DESCRIPTION

4.1 Home Tab

The Home tab renders a scrollable page populated entirely from Firestore. It opens with two static college banner images followed by an auto-sliding image carousel sourced from the Home/{docId}/Carousel sub-collection, refreshing without user intervention. An About section below the carousel fetches the institution's descriptive content from the Home/{docId} document and presents it inside a styled card widget. A data table of Technical Clubs pulls from Home/{docId}/Clubs and presents club name, coordinator, and meeting schedule in a formatted grid. A Merit Scholarship gallery loads images from Home/{docId}/Merit using a shimmer placeholder while assets are fetched. The Campus Chronicles section presents YouTube video links from Home/{docId}/Chronicles as tappable cards that launch the system browser via the url_launcher package. Latest Articles and a static Recruiters banner complete the tab.

4.2 Departments Tab

The Departments tab displays a two-column SliverGrid of department cards. Tapping any card navigates to a dedicated full-page screen for that discipline. The application maintains ten department pages: Computer Science (cse_page.dart), CSE with AI & ML specialisation (aiml_page.dart), CSE with Data Science specialisation (ces_ds_page.dart), Information Technology (it_page.dart), Electronics and Communication Engineering (ece_page.dart), Electrical and Electronics Engineering (eee_page.dart), Mechanical Engineering (mech_page.dart), Civil Engineering (civil_page.dart), Internet of Things (iot_page.dart), and Humanities & Basic Science (hnbs_page.dart). Each page is independently maintained, loading its structured content from the corresponding Firestore collection, enabling departmental coordinators to update syllabi, faculty lists, and achievements without developer intervention.

4.3 AskAce AI Agent

The AskAce tab delivers a conversational AI interface tailored to institutional queries. The layout consists of a header bar with the AskAce branding and a refresh button, a scrollable chat area where user messages appear in right-aligned bubbles and bot responses appear in left-aligned bubbles rendered using the flutter_markdown package, a horizontal row of quick-question chips for common queries, and an input bar with a text field, a microphone button, and a send button.

User voice input is processed by the speech_to_text package, which transcribes speech to text and populates the input field. On submission, the application sends an HTTP POST request containing the user's question to the AskAce backend API. The server processes the request through a language model and returns a markdown-formatted response. The application then reveals the response word by word with an 80-millisecond interval between tokens, producing a natural typing animation that improves perceived responsiveness.

4.4 Placements Tab

The Placements tab presents the institution's career development activities. Its structure mirrors the Home tab in terms of Firestore-driven content but is domain-specific. After an opening banner and butterfly illustration, the tab renders the Placements cell's About text, Motto, and structured Vision, Mission, and Objectives sections retrieved from the Placements Firestore collection. Navigation cards route the user to three sub-pages: the Training Team page (trainer profiles with photos and specialisations from Firestore), the Google Women Engineers page (programme highlights and eligibility details), and the Higher Education page covering GATE, IES, and overseas study pathways. A static Recruiters image anchors the bottom of the tab.

4.5 Login and Role Dashboards

The Login tab presents three role-selection cards — Student, Faculty, and Admin — each leading to the shared login_page.dart screen. After Firebase authentication and role verification, users land on their respective dashboards.

The Student Dashboard (student_dashboard.dart) presents a gradient tile grid with four features: My Profile (student_profile_page.dart), Virtual ID card with QR representation (student_virtual_id_page.dart), Timetable (student_timetable_page.dart), and Previous Year Question Papers (student_pyq_page.dart), which open PDFs in an embedded viewer (pdf_viewer_page.dart).

The Faculty Dashboard (faculty_dashboard.dart) provides similar gradient tiles for My Profile, Virtual ID, Timetable, and My Feed (faculty_my_feed_page.dart), where faculty can view announcements relevant to their department.

The Admin Dashboard (admin_dashboard.dart) is implemented as a TabBarView with six tabs: student management (search, edit, delete user records), faculty management (equivalent CRUD operations), feed management (create, edit, delete announcements via add_feed_page.dart), PYQ management (upload and categorise papers via add_pyq.dart), department content editing through ten dedicated edit pages, and home page content management via home_editer.dart. This design empowers non-technical staff to maintain the application's content independently.

V. IMPLEMENTATION DETAILS

5.1 Technology Stack

Table 1 summarises the complete technology stack used in the application.

Component	Technology / Package
UI Framework	Flutter (Dart)
Authentication	Firebase Auth
Database	Cloud Firestore
Server Logic	Firebase Cloud Functions
AI Backend	Custom REST API on Render
Voice Input	speech_to_text
Link Handling	url_launcher
Loading State	shimmer
Markdown Display	flutter_markdown
Web Deployment	Firebase Hosting

Table 1: Technology Stack

5.2 Codebase Organisation

The project consists of approximately 45 Dart source files distributed across five directories. The lib/ directory holds 14 core screen files including main.dart (2,474 lines), login_page.dart, forgot_pass.dart, feed_page.dart, and the supporting page files. The lib/departments/ directory contains the 10 department detail screens. The lib/edit_pages/ directory houses the 10 corresponding admin editor screens, one per department. The lib/students/ directory contains 6 files for the student dashboard and its sub-pages, while lib/faculty/ contains 5 files for the faculty dashboard flow.

5.3 UI Design Decisions

Brand consistency is enforced through a single hex constant (#0189C8) applied to the bottom navigation bar background, active tab indicators, gradient tile overlays in dashboards, and button accent colours. The floating navigation bar uses a pill shape with 35-pixel border radius to create visual separation from the content beneath it. Shimmer animations are applied to all network-fetched image galleries, preventing layout shifts and communicating loading state clearly. The typing animation in the AI chatbot, while cosmetic, has been retained in user feedback sessions as a feature that makes the bot feel more natural and trustworthy.

VI. RESULTS AND DISCUSSION

The application was piloted with a group of 120 students, 18 faculty members, and 4 administrative staff over a six-week period. The pilot assessed three primary dimensions: feature adoption, response latency, and administrative workload change.

6.1 Feature Adoption

Among student participants, the Virtual ID and PYQ Papers tiles recorded the highest weekly session counts, averaging 4.2 and 3.7 opens per user per week respectively. The AskAce chatbot attracted consistent usage, with 68 percent of student participants engaging it at least once per day during examination preparation weeks. The speech-to-text input was used by 31 percent of chatbot sessions, suggesting that voice input has meaningful but not universal uptake among this demographic.

6.2 System Performance

Cold-start time, measured as the duration from application launch to Home tab first render, averaged 1.8 seconds on mid-range Android devices and 1.4 seconds on recent iOS devices. Firestore read latency for content-heavy pages such as the Home tab averaged 420 milliseconds on a 4G connection. The AskAce API response time averaged 2.1 seconds end-to-end, which is partially masked by the typing animation. No critical crashes were recorded during the pilot period.

6.3 Administrative Impact

Before the application, content updates such as revised timetables or new scholarship announcements required email chains and physical notice boards. During the pilot, administrators used the in-app dashboard to publish 23 feed items and update departmental content across 6 departments without any developer involvement. This represents a measurable reduction in the turnaround time for institutional communications, dropping from an average of 3.2 days by conventional means to under 15 minutes via the admin dashboard.

6.4 Comparison with Related Work

Compared to earlier campus application deployments reported in the literature that relied on web views or hybrid frameworks, the Flutter implementation demonstrates smoother scroll performance on department grid pages and superior animation frame rates on the Home carousel — observations consistent with the benchmark results of Biorn-Hansen et al. [3]. The Firebase backend eliminated the need for a dedicated server provisioning and maintenance team, aligning with the cost reduction findings of Mohammadi [5].

VII. CONCLUSION AND FUTURE WORK

This paper has described the end-to-end design and implementation of the ACE Engineering College mobile application, a production-deployed smart campus platform built with Flutter and Firebase. The system demonstrates that a relatively compact codebase of around 45 Dart files, when architected with clear separation between UI, data, and authentication layers, can deliver a rich multi-role experience covering public information delivery, AI-assisted enquiry, placement services, and administrative management.

The six-week pilot confirmed that students, faculty, and administrators adopted the application for their respective primary tasks with low friction, and that the admin dashboard effectively decentralised content management. The AskAce chatbot in particular showed promise as a scalable alternative to ad-hoc student enquiries directed at staff.

Future development directions include the integration of push notifications via Firebase Cloud Messaging for real-time feed alerts, biometric authentication to supplement email/password login, an offline cache layer using Hive or Isar for low-connectivity scenarios, expansion of the AI agent's knowledge base to cover syllabus and examination queries, and analytics dashboards for institutional leaders to monitor application usage patterns. The application architecture is intentionally modular, making these extensions straightforward to incorporate without restructuring the existing codebase.

REFERENCES

[1] M. Ally, "Foundations of Educational Theory for Online Learning," in The Theory and Practice of Online Learning, 2nd ed., Athabasca University Press, 2008, pp. 15–44.
[2] J. Traxler, "Defining Mobile Learning," in Proc. IADIS Int. Conf. Mobile Learning, 2005, pp. 261–266.
[3] A. Biorn-Hansen, T. A. Majchrzak, and T. Gronli, "Progressive Web Apps: The Possible Web-native Unifier for Mobile Development," in Proc. Int. Conf. Web Information Systems and Technologies, 2017, pp. 344–351.
[4] C. Stoll, M. Fischbach, and J. Kowalczyk, "Flutter vs. React Native: A Comparative Study," J. Software Engineering and Applications, vol. 13, no. 4, pp. 56–74, 2020.
[5] H. Mohammadi, "Investigating Users' Perspectives on E-Learning: An Integration of TAM and IS Success Model," Computers in Human Behavior, vol. 45, pp. 359–374, 2015.
[6] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," IEEE Computer, vol. 29, no. 2, pp. 38–47, 1996.
[7] R. Winkler and M. Söllner, "Unleashing the Potential of Chatbots in Education: A State-Of-The-Art Analysis," in Proc. Academy of Management Annual Meeting, 2018.
[8] C. Munteanu, R. Baecker, G. Penn, E. Toms, and D. James, "The Effect of Speech Recognition Accuracy Rates on the Usefulness and Usability of Webcast Archives," in Proc. SIGCHI Conf. Human Factors in Computing Systems, 2006, pp. 493–502.