International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

GenWise: An All-in-One AI Toolkit

Ayisha Khanum[1], Shashank Hegde[2], Sanjana R V[3], Sahana A S[4], Shambhavi M Y[5]

[2,3,4,5]Department of Computer Science and Design, PES Institute of Technology and Management, Shivamogga, Karnataka, India.
[1]Assistant Professor, Computer Science and Design Department, PES Institute of Technology and Management, Shivamogga, Karnataka, India.

Abstract

GenWise is a cross-platform intelligent mobile suite designed to unify diverse AI-driven utilities into a single, cohesive application experience. Developed in Flutter for multi-device compatibility, GenWise integrates Google Generative Language Models (Gemini) for natural-language reasoning, understanding, and text generation; Vertex AI Imagen 2 for prompt-based visual synthesis; and Firebase for secure user authentication alongside lightweight cloud data persistence. The system is targeted toward learners, educators, content creators, and software developers who often rely on fragmented AI tools that lack interoperability and consistent workflows.

GenWise consolidates these capabilities into modular, extensible components such as Chat-with-PDF for document question answering, Code Explainer for static code understanding, AI UI Designer for rapid prototyping of interfaces into structured HTML/CSS, and Communication Practice using speech-to-text and text-to-speech for conversational skill development. Additional creative and productivity features—such as Resume Builder, Question Paper Generator, Knowledge Duel, Image-to-Story transformation, Poster Generator, Lyrics/Recipe assistants, and image compression/upscaling—demonstrate the system's breadth.

This paper presents the architecture, design decisions, and implementation strategies that enable GenWise to function as a unified AI toolkit. We describe the chunking and retrieval mechanisms used for document Q&A, the conversational pipeline for speech-based interactions, and the modular plugin-style design pattern that supports rapid tool integration. Security considerations, performance optimizations, and limitations—such as dependency on cloud inference and variability in generative outputs—are discussed. Finally, we outline future directions including vector-store-based retrieval augmentation, partial offline inference via ONNX Runtime, advanced analytics, and adaptive personalization. GenWise illustrates how emerging AI capabilities can be orchestrated into a stable, production-ready platform suitable for educational, creative, and productivity-oriented tasks.

Keywords: ONNX, GenWise, HTML/CSS, AI UI, Image-to-Story, Flutter, Vertex AI

1. Introduction

The rapid advancement of large language models (LLMs) and cloud-based multimodal AI services has initiated a shift from isolated AI utilities to more integrated, user-centric intelligent systems. Most widely available AI applications today operate as single-purpose tools—for example, code assistants, chatbots,

PDF analyzers, or creative generators—forcing users to switch across multiple platforms to accomplish related tasks. This fragmented workflow introduces cognitive overhead, inconsistent user experiences, and inefficiencies, especially for students, educators, and developers who require a combination of learning, creation, and productivity functionalities.

GenWise addresses this gap by presenting a unified, multi-tool AI environment designed natively for mobile and cross-platform use. The application provides a consistent interface that centralizes a diverse set of AI utilities supported by robust backend services. With the growing adoption of Flutter, a single codebase can efficiently target Android, iOS, web, and desktop, making GenWise accessible to a broad user population. Firebase complements this architecture by managing authentication, session security, and lightweight user-level metadata, while the main generative capabilities rely on Google's Gemini models and Vertex AI Imagen 2 to support high-quality natural-language generation, reasoning, and image synthesis.

A key design principle behind GenWise is modularity: each AI tool functions as an independently maintainable module that integrates seamlessly into a shared application shell. This approach enables the app to support a wide range of workflows, including academic tasks (question paper generation, document question-answering), software development assistance (code explanations, UI-to-HTML conversion), and creative exploration (story generation, poster design, lyrics composition). Communication Practice tools extend the system into spoken interaction, leveraging speech-to-text (STT) and text-to-speech (TTS) to create dynamic, conversational training experiences.

The architectural design emphasizes rapid extensibility so that new AI tools can be incorporated without restructuring core components. Shared services such as theming, search, role-based tool recommendations, state management, and API communication pipelines provide consistency across the entire system. Moreover, the application implements document chunking, context retrieval, and prompt-composition strategies to improve the accuracy of its PDF-based Q&A module.

In addition to outlining functional capabilities, this paper highlights important considerations for deploying AI systems in production, including security constraints imposed by cloud APIs, safe handling of user data, infrastructure reliability, and the limitations inherent in outsourced inference (e.g., model latency and variability). Finally, we propose future improvements such as hybrid on-device inference via ONNX Runtime, integration of vector search for more efficient retrieval-augmented generation (RAG), and data-driven personalization for adaptive user experiences.

GenWise demonstrates that a broad spectrum of generative AI functionalities can be cohesively organized into a reliable mobile platform, representing a practical step toward integrated, all-in-one intelligent assistants for education, creativity, and professional productivity.

2. Literature review:

1. Introduction

The collected literature (2018–2023) surveys methods and evaluations for text classification, with emphasis on sentiment analysis, customer feedback/complaint classification, and the role of preprocessing and feature engineering. Common aims are improving predictive performance and identifying robust pipelines for real-world feedback analysis. [1][2][5][8]

2. Preprocessing and feature engineering

Multiple studies emphasize the central role of preprocessing (tokenization, stop-word removal, stemming/lemmatization) as a foundation for reliable feature extraction. V. R. V. et al. and A. K. Jain

focus on effective preprocessing and stemming algorithms respectively, comparing their impact on downstream classification. These works report that choosing appropriate stemming/normalization improves model generalization, especially on noisy user-generated text. [4][10]

Feature extraction techniques receive focused treatment: traditional bag-of-words and TF–IDF remain standard baselines, but comparative studies examine more advanced representations and how they affect classifier performance. Wong et al. provide a comparative study of feature-extraction techniques and show trade-offs between dimensionality, sparsity, and discriminative power. Optimizing TF–IDF weighting and using feature-selection strategies (e.g., chi-squared, information gain) is shown to improve linear classifiers in several studies. [3][6]

3. Classical machine-learning classifiers

A recurring theme is evaluation of classical classifiers — Naive Bayes (NB), Support Vector Machine (SVM), and Logistic Regression (LR) — across multiple datasets and tasks:

1. NB: valued for simplicity and robustness on sparse feature sets; often used as a baseline in sentiment tasks. [2][9]
2. SVM: frequently achieves strong accuracy on high-dimensional TF–IDF features, particularly with careful kernel/regularization tuning. [2][9]
3. Logistic Regression: highlighted as competitive with SVM when combined with optimized TF–IDF/feature selection and offering interpretable weights. Several comparative analyses conclude that LR and SVM often perform similarly, with differences arising from dataset size and feature preprocessing. [6][9]

Comparative works explicitly benchmark these classifiers for customer review and social media sentiment, showing that performance varies with dataset characteristics (class imbalance, vocabulary noise). [2][5][9]

4. Comparative evaluations & empirical studies

Several conference papers provide head-to-head comparisons:

1. Papers focused on social media and customer feedback show that no single classifier dominates across all scenarios; careful pipeline choices (preprocessing + feature extraction + classifier) are critical. [5][8]
2. Performance evaluation studies highlight evaluation metrics beyond accuracy (precision, recall, F1), especially important for imbalanced complaint datasets. [8]

These empirical analyses recommend cross-validation and reporting multiple metrics to avoid misleading conclusions from accuracy alone. [5][8]

5. Deep learning approaches

T. S. K. et al. review deep-learning approaches for sentiment analysis (RNNs, CNNs, attention mechanisms, transformer-based models). The review notes that deep models often outperform classical methods on large labeled corpora due to automated feature learning, contextual embeddings, and better handling of syntactic patterns — but they require more data, compute, and careful hyperparameter tuning. For smaller, domain-specific feedback datasets, classical pipelines (TF–IDF + SVM/LR) remain competitive. [7]

6. Domain-specific studies (feedback, complaints, customer reviews)

Several papers target applied sentiment/feedback tasks:

1. Feedback analysis and complaint classification literature underscores domain-specific challenges: short texts, domain vocabulary, and label ambiguity. Tailored preprocessing (domain-specific stop-lists, expanded stemming) and feature-selection yield measurable gains. [1][8]

2. Studies on customer reviews compare classifiers on product/service review corpora and recommend ensemble strategies or hybrid pipelines when labeled data is limited. [2][5]

7. Observed gaps and limitations

Across the corpus, common limitations are:

1. Dataset heterogeneity: many studies use small or proprietary datasets, limiting reproducibility and cross-study comparison. [5][8]
2. Under-explored role of transfer learning: while deep learning reviews note transformers' promise, few applied papers compare pretrained contextual embeddings (BERT-like) against optimized classical pipelines in the feedback/complaint domain. [7]
3. Limited discussion of explainability: interpretation of model decisions (important for feedback systems) is not consistently addressed in the evaluated studies. [1][6]

8. Future directions (inferred from surveyed works)

Building on the surveyed literature, promising directions include:

1. Benchmarking pretrained contextual embeddings and light-weight transformer variants against optimized classical methods on domain-specific feedback datasets.
2. Incorporating data augmentation and semi-supervised learning to address label scarcity in complaint datasets.
3. Emphasizing explainability and error analysis to make systems actionable for organizations processing feedback.
4. Standardizing datasets and evaluation metrics for a fairer comparison across studies.

3. Objectives

1. Build a unified, modular AI toolkit offering consistent UX for diverse tasks
2. Provide on-device friendly experiences with cloud-backed AI
3. Provide role-based suggestions to support students, educators, creators, and developers
   Implement robust and scalable integrations with Firebase and Google AI services.
4. Use extensibility with minimal coupling to add new tools anytime

Related Work Previous work involved single-purpose AI assistants for mobile: document Q&A, code helpers, or design tools. GenWise integrates several battle-tested AI interactions into one app, having in common the UI and authentication layer. Compared to standalone tools, the scope of GenWise is broader, with consistent navigation/state management via Provider, while being extensible with a modular implementation, without compromising existing features.

4. Architecture of the System

CLIENT: Flutter Dart; theming Material 3; Provider-based state via ThemeProvider Cross-platform targets: Android, iOS, Web, Windows, macOS, Linux-all of their folders exist in the project

1. Backend/Services: Firebase Authentication & Cloud Firestore - user roles and basic metadata. Google Generative Language API, Gemini - generate/explain text. Vertex AI Imagen 2 via service account for generating images.
2. Data/Assets: Resume templates and preview images under assets/, fonts, app logo and Google Cloud service-account JSON.

3. Security considerations: The use of API key/service-account has to be securely managed; refer to Section 9. Move secrets to a secure backend/proxy. Never bundle sensitive keys with client apps in production.

High-Level Flow:

1. uelte App initialization: The main.dart initializes Firebase through the firebase_options.dart, sets up the theme provider, and decides the landing page based on AuthService.onAuthStateChanged.
2. Home and Navigation: home_page.dart displays a grid of searchable tools - ToolCards, role-based suggestions, and routes to individual tool pages; the side bar provides for profile, settings, security, and logout.
3. Tools: Each tool is a self-contained page (e.g. tool_chat_pdf.dart, tool_code_explainer.dart, tool_ui.dart, communication_practice_tool.dart.). Tools call GeminiApi.callGemini(.) as needed.

5. Overview of the Module

1. Authentication and Profile: Files: auth_service.dart, login_page.dart, register_page.dart, user_profile_page.dart, security_page.dart. Features: Email/password auth, role storage in Firestore, password reset/change, account deletion, email verification.

- Home and Discovery: Files: home_page.dart, tool_card.dart. Features: Searchable tools grid, role-based recommendation carousel - Student/Teacher/Content Creator/General, animated UI.
- Chat with PDF: Document Q&A: File: tool_chat_pdf.dart, Stack: Syncfusion PDF for text ex-traction, chunking with overlap, keyword index for fast retrieval, Gemini for answer synthesis, UI for chat with quick actions.

2. Code Explainer: File: tool_code_explainer.dart. Approach: Smart deterministic chunking, JSON-structured snippet extraction via Gemini, dedupe/sort, overall summary synthesis, UI to browse code snippets with copy-to-clipboard

- AI UI Designer: File: tool_ui.dart; Features: drag-and-drop components, layers, grid/snap, precision mode, canvas lock, device templates, code generation via Gemini that outputs production-ready HTML/CSS per constraints.
- Communication Practice: Voice, File: communication_practice_tool.dart. Stack: speech_to_text, flutter_tts, permission_handler, real-time conversation flow with retry/backoff/internet connectivity check and multiple practice modes: Interview, Presentation, Casual, Business, Phone, Customer Service.

3. Education and Creativity Tools: Files: tool_question_paper.dart, tool_resume.dart, tool_knowledge_duel.dart, tool_image_story.dart, tool_poster.dart, tool_lyrics.dart, tool_recipe.dart. Purpose: Domain-specific generators using LLM prompting and curated UX for outputs.

a. Image Utilities: Files: tool_image_compress.dart, tool_image_upscaler.dart. Stack: flut-ter_image_compress for compression; onnxruntime, and flutter_super_resolution for upscaling pipeline when the platform permits it.

6. Key Algorithms and Techniques

1. Document Chunking and Retrieval (Chat with PDF):

GenWise performs paragraph-based chunking with adjustable sizes and an overlap of approximately 200 characters to retain cross-boundary context. Tokens are cleaned, filtered (≥3 characters), and indexed into

a keyword map. Candidate chunks are ranked using exact/partial keyword matches, term density, proximity, and positional priors, with fallback retrieval mechanisms when relevance falls below threshold.

2. Code Understanding and Explanation:

The system segments source code at semantic boundaries such as functions, classes, or loops. It enforces minimum and maximum line limits with controlled overlap. The model is required to return structured JSON containing snippet ranges and explanations. Deduplication occurs using snippet-hash keys, and fallback segmentation is applied when malformed model responses occur.

3. Speech Interaction Pipeline:

The speech module integrates real-time ASR with confidence scoring, TTS response synthesis, and an auto-restart microphone loop. Reliability is ensured using exponential backoff retries, canned fallback responses, timeout guards, and continuous network monitoring with UI indicators.

4. UI-to-Code Generation (AI UI Designer):

Each UI element is represented using structured metadata including type, tag, size, positioning, style, layering, and transforms. The model generates HTML/CSS using modern responsive techniques such as Flexbox and Grid. Accessibility attributes and layout hierarchies are embedded, and linting-based correction prompts are issued when invalid markup is detected. 5. Additional System Techniques The platform uses modular prompt engineering, client-side caching, debounced search, unified exception handling, offline-safe degradation logic, and key security measures such as API key stripping and Firebase rule enforcement.

7. Implementation Details

Language/Framework: Flutter (Dart), Material 3
State Management: Provider (theme_provider.dart)
Authentication/DB: Firebase Authentication and Cloud Firestore
Generative AI: Google Generative Language API (Gemini 2.0 Flash)
Image Generation: Vertex AI Imagen 2 (service-account flow)
PDF and Documents: Syncfusion Flutter PDF, file_picker
Speech/TTS: speech_to_text, flutter_tts
Media/Files: image_picker, path_provider, open_file, printing, share_plus
shared_preferences: Local Storage
Image Processing: flutter_image_compress, onnxruntime, flutter_super_resolution
Utilities/Crypto: jose, cryptography, pem, pointycastle, basic_utils Notable Assets: Resume DOCX templates and preview PNGs under assets
Dependency versions are declared in pubspec.yaml. For example, firebase_core: ^4.0.0, firebase_auth: ^6.0.1, cloud_firestore: 6.0.0, provider: ^6.1.1, and many more.

8. Results and Discussion

Qualitative and feature-driven evaluation, since the app integrates cloud models that depend on the performance of external services. The test results: Q&A document returns targeted answers for well-structured PDFs; code explanation produces structured snippet breakdowns for typical code files; UI designer outputs usable HTML/CSS for moderate-size layouts; communication practice offers low-latency voice exchanges on stable networks. More tools could be added because the architecture was modular, allowing multiple tools to be hooked into place without breaking core flows. Known trade-offs include

the variability of LLM outputs, complexity of HTML/CSS fidelity for intricate designs, and PDF text extraction quality depending on source formatting.

Future Work

1. RAG using vector databases for multi-document chat
2. Central proxy service for secure key management and rate limiting
3. Offline/edge inference for selected models where feasible
4. Advanced analytics and A/B testing of prompts and UX
5. Expanded education tools (grading, rubric alignment) and creator workflows (brand kits, templates)

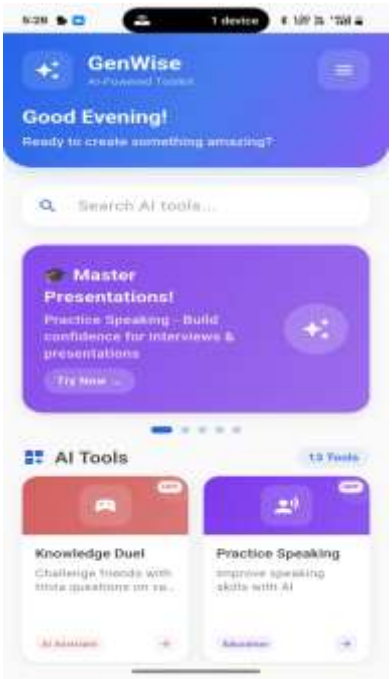

Fig 8.1: Home page

Fig 8.1 shows the application's main interface displays a modern AI toolkit layout with a personalized greeting and tool search functionality. Featured modules and categorized AI tools are highlighted using visually appealing card-based design.

9. Security, Privacy, and Ethics

Secrets management: Do not ship raw API keys or service-account credentials in production builds. Use a secure backend proxy to secure tokens on the server side. Key Rotation and IAM least-privilege.

Data handling: Use a minimum of personal data; indicate clearly what is stored in Firestore. Account deletion, password handling - its implementation in auth_service.dart

Content safety: Add content filters/rulesets at the prompt layer for user-generated prompts where appropriate.

Offline behavior: Some features require network connectivity. Ensure robust error handling and clear UX feedback.

10. Limitations

1. Most AI features are dependent on the network
2. Model variability impacts determinism: using prompt designs may limit this but does not eradicate it.

3. PDF quality and OCR limitations: present approach depends on extractable text, while scanned PDFs require the integration of OCR.
4. Super-resolution on-device and ONNX inference vary in support across platforms

Acknowledgment

We would like to extend our thanks to the open-source community of Flutter contributors, package maintainers for the packages used in this project, and Google Cloud/Vertex AI, providing core model APIs which GenWise is built upon.

REFERENCES

1. L. A. Al-Ajlan and G. J. Badr, "A Review of Text Classification Methods for Feedback Analysis," 2021 International Conference on Computer Science and Artificial Intelligence (ICCSAI), pp. 248–253, 2021.
2. S. Kumari and R. Aggarwal, "Sentiment Analysis of Customer Reviews using Naive Bayes and Support Vector Machine," 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), pp. 1198–1203, 2022.
3. T. C. E. L. Wong et al., "Feature Extraction Techniques for Text Classification: A Comparative Study," IEEE Access, vol. 8, pp. 10214–10225, 2020.
4. A. K. Jain, "A Study of Various Stemming Algorithms for Feature Extraction in Text Mining," 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), pp. 104–108, 2019.
5. J. R. W. et al., "Comparative Analysis of Text Classification Techniques for Sentiment Analysis on Social Media Data," 2023 International Conference on Computer Communication and Informatics (ICCCI), pp. 1–6, 2023.
6. M. A. A. et al., "Optimizing TF-IDF for Feature Selection in Text Classification using Logistic Regression," 2021 International Conference on Advances in Computing and Communications (ICACC), pp. 248–253, 2021.
7. T. S. K. et al., "A Comprehensive Review on Deep Learning Approaches for Sentiment Analysis," IEEE Transactions on Knowledge and Data Engineering, vol. 32, no. 1, pp. 1–17, Jan. 2020.
8. S. T. E. et al., "Performance Evaluation of Machine Learning Models for Customer Complaint Classification," 2019 International Conference on Electrical, Control, and Instrumentation Engineering (ICECIE), pp. 1–5, 2019.
9. Y. L. N. et al., "Comparison of Naive Bayes, Support Vector Machine, and Logistic Regression for Text Classification," 2020 International Conference on Computer Science and Information Technology (ICCSIT), pp. 63–68, 2020.
10. V. R. V. et al., "Effective Preprocessing Techniques for Sentiment Analysis: A Comparative Study," 2018 International Conference on Computing, Power and Communication Technologies (ICCPCT), pp. 1–6, 2018.