Mzanzi Lingo

Comprehensive Document

Table of Contents

Links

- GitHUb Link: https://github.com/Rhea0524/MzansiLingo
- Demo Video: https://youtu.be/85hXIaZRtzs

Introduction

The increasing demand for mobile-first educational tools has created opportunities to design applications that blend technology with cultural relevance. Mzansi Lingo is positioned to address this need by offering an inclusive, interactive platform for learning South Africa's diverse languages. The system integrates gamification, structured lessons, and speech-centred learning to make language acquisition engaging and effective.

A key component of the design is the conversational feature, which enables learners to practise their speaking and comprehension skills in real time. This ensures that users are not only exposed to vocabulary and grammar but are also encouraged to apply their knowledge in a practical, feedback-driven environment (CivicScience, 2023).

Overview of the Application

- **App Name:** Mzansi Lingo
- **App Logo (visual description):** A friendly cartoon green rhinoceros head on a bright yellow background. The rhino has large round eyes with sparkles, a cheerful smile, white horns, and is drawn in a modern flat illustration style.
- **Brief Description**: Mobile-first interactive language learning for seven South African languages: isiZulu, isiXhosa, Sesotho, Setswana, Sepedi, Afrikaans, and English. Combines gamification, cultural content, and speech-first practice.
- **Target Audience:** South African citizens, tourists, expatriates, students, and professionals seeking cultural and language fluency.

Purpose of the Application

The purpose of Mzansi Lingo is to provide an accessible, culturally rich, and interactive platform for learning South Africa's diverse languages. The application aims to close the gap between traditional learning methods and modern, mobile-first education by offering users an engaging way to build real-world conversational skills.

Mzansi Lingo empowers learners by combining structured lessons, speech-based interactions, gamified progress tracking, and cultural context into a single, easy-to-use application. It is designed to promote multilingualism, improve communication across South Africa's diverse communities, and support both locals and visitors in understanding and appreciating the linguistic heritage of the country. Through AI-driven pronunciation guidance, real-time conversational practice, and meaningful progress analytics, the app helps users develop confidence, fluency, and cultural insight.

Features:

1. **Progress-Based Learning and Points:**

Mzansi Lingo uses a structured progress system where users earn points for completing lessons, quizzes, and pronunciation exercises. Instead of relying on “game-like” mechanics such as lives or energy systems, the app emphasizes meaningful learning progress. Each completed lesson contributes to the learner’s overall skill growth, and users can visually track which topics they have mastered and which require more practice. The points system provides boosts in motivation by showing clear evidence of improvement over time. As users accumulate points, they unlock more advanced content, allowing for a personalised, self-paced learning journey.

2. **Leader boards & Real-Time Rankings:**

The leader board system introduces a sense of friendly competition by displaying users’ rankings based on their accumulated points, quiz results, and learning streaks. Rankings update in real time whenever a user completes lessons or challenges. This feature encourages consistent engagement because learners naturally want to improve their standing in comparison to peers. It also creates a community-driven environment where progress becomes fun, socially interactive, and goal-oriented.

3. **Daily Targets and Reminders:**

Users can set personal daily goals for the number of words, phrases, or lessons they want to complete. The system sends reminders so learners do not lose momentum, reinforcing good habits and encouraging long-term consistency. If users fall behind, the app gently notifies them to resume learning. Completing daily targets triggers achievement pop-ups, making the experience rewarding and helping learners create a sustainable learning routine.

4. **Audio-Based Learning (Playback + Text-to-Speech):**

To help learners develop proper pronunciation and listening comprehension, every word, phrase, and sentence includes clear audio playback. Users can listen to native-like pronunciation through professionally produced or AI-generated audio. The app also integrates Text-to-Speech (TTS) so learners can hear spoken examples for any phrase-including dynamic chat responses and custom requests. This ensures high accessibility and supports learners who depend heavily on auditory learning styles. These tools help improve listening, mimicry, fluency, and overall speaking confidence.

5. **AI Chatbot for Pronunciation Help (REST API Integration):**

The AI chatbot enables users to ask how to say specific words or phrases in their selected language. When a user types or speaks a request, the chatbot returns:

- The translated word or phrase
- A phonetic pronunciation guide (e.g., *"Sawubona (sah-woo-BOH-nah)"*)
- Optional audio playback for correct pronunciation

The chatbot is powered by a REST API hosted on a cloud platform (Railway/Firebase Functions). This ensures quick responses, scalability, and accurate language outputs. This feature transforms the app from a static learning tool into an interactive, on-demand tutor available 24/7.

6. **Offline Mode with Automatic Sync:**

Mzansi Lingo allows users to continue learning even without internet access. In offline mode, users can:

- Access downloaded lessons
- Play the quiz game
- Practise vocabulary and phrases

Progress made offline such as earned points, completed lessons, quiz scores, or streaks-is stored locally on the device. Once an internet connection becomes available, the app automatically syncs the offline progress with the user's online profile. This makes learning flexible and reliable, even in areas with limited connectivity.

7. **Biometric Login and Google SSO:**

To improve convenience and security, the app supports multiple authentication methods:

- Google Single Sign-On: Allows users to log in instantly using their Google account, eliminating the need to manually enter passwords.
- Biometric Authentication: Fingerprint or facial recognition is available for fast and secure access. This ensures that returning learners can resume where they left off without repeating login steps.

Both methods enhance accessibility while maintaining strong security standards.

8. **Conversation Logging and Analytics:**

Every conversation session between the user and the AI chatbot is logged and stored in a secure database. Each session captures:

- Session ID
- User ID
- Selected language
- Timestamps
- Message direction (user → system / system → user)
- Feedback on pronunciation or grammar
- Progress indicators

These logs allow users to revisit previous sessions to review their growth.

From a system perspective, analytics help identify:

- Commonly requested phrases
- User learning difficulties
- Language proficiency trends
- Session frequency and engagement patterns

This data makes the learning experience more personalised and improves the quality of future app updates.

9.Sharing Learning Progress:

Mzansi Lingo features integrated sharing capabilities that allow users to proudly display their learning achievements with friends or classmates. Learners can share highlights such as:

- Overall points earned
- Ongoing or completed streaks
- Milestones achieved
- Badges unlocked

This feature boosts motivation by fostering social encouragement, accountability, and a sense of community involvement.

10. **Exporting Test and Quiz Data:**

The app also provides an option to export personalised learning data, including quiz scores, test results, and lesson completion records, in formats such as CSV or PDF. This functionality is valuable for:

- Students who need to monitor academic progress
- Educators tracking learner performance
- Individuals wanting a documented record of their improvements

By enabling data export, the app supports transparency and gives users greater ownership of their learning journey.

11. Enhanced Gamification Features:

To further enrich the user experience, Mzansi Lingo introduces additional gamification elements such as:

- Unlockable badges for key achievements (e.g., “100 Words Learned”, “7-Day Streak”)
- Visual animations when levelling up
- Timed challenge modes to test knowledge under pressure
- Limited-time events and themed challenges to keep engagement high

These enhancements help maintain user interest and motivation while ensuring the app remains focused on meaningful learning rather than traditional gaming mechanics.

Design Considerations

Architecture

1. Mobile-First Native Android App (Kotlin)

Mzansi Lingo is built as a native Android application using Kotlin, ensuring high performance, smooth animations, and full access to modern Android APIs. The entire architecture is designed with a mobile-first mindset, meaning the app is optimized for mobile screens, touch input, and on-the-go learning.

- Clean, Layered Architecture:
    - The system is structured using standard Android architectural best practices:
- UI Layer:
    - Activities & Fragments handle navigation and screen rendering.
    - Jetpack Compose is used for modern, declarative UI elements, improving performance and reducing boilerplate.
    - The UI focuses on simplicity, accessibility, and engaging visual feedback (e.g., progress indicators and leader board highlights).
- ViewModels:
    - All business logic, state, and UI-related data are stored inside ViewModels, following MVVM.
    - ViewModels survive configuration changes (rotation, backgrounding), ensuring data stability.
    - They expose data through StateFlow or LiveData, enabling reactive updates.
- Repository Pattern:
    - The Repository abstracts data sources, allowing the app to seamlessly combine: Remote API data, Firebase real-time data, Local Room database cache.
    - This creates a clean separation between UI and data logic.

- RoomDB for Local Persistence:
    - Local storage is implemented using Room, the official Android ORM. RoomDB allows the app to:
        - Cache lessons, vocabulary, and leader board data
        - Store user progress and quiz results offline
        - Sync back to the cloud when online
    - This guarantees that learners can continue using the app even with unstable or limited connectivity.

2. Backend: Lightweight REST API (Railway):

Mzansi Lingo uses a minimalistic, optimized backend hosted on Railway, chosen for its fast deployment, low cost, and easy API hosting.

- Core Backend Functions
    - AI Chatbot Language Assistance
    - Handles queries like "How do I say ___?"
    - Returns pronunciation guides, example sentences, and corrected phrasing.
    - Communicates with language models for natural, human-like responses.
    - Session Management:
        - Tracks conversation history
        - Stores progress across different learning sessions
        - Logs timestamps and message metadata for analytics
- Firebase Integration for Real-Time Features

The app uses multiple Firebase services:

- FireStore
    - Syncs leader board updates
    - Stores user progress and achievements in real time
    - Manages multi-language lesson content
- Firebase Storage
    - Stores audio files (recorded speech, listening exercises)
    - Stores pronunciation samples and generated TTS clips
- Firebase Cloud Messaging (FCM)
    - Sends notifications for: Daily targets, Missed streaks, New lesson content, Ranking changes

This combination of Railway and Firebase provides reliability, scalability, and low cost.

3. Processing Layer: Serverless Cloud Functions

Mzansi Lingo uses Serverless Cloud Functions to handle compute-heavy operations. This improves performance and reduces server load because the system only runs functions when needed.

- Speech-to-Text (STT) Integration

Cloud Functions connect to the speech engine to convert recorded audio into text.

- Used for:
    - Checking if pronunciation was correct
    - Analysing conversation practice
    - Processing spoken quiz answers

- Text-to-Speech (TTS) Generation

Functions generate audio clips for:

  a. Lesson pronunciations
  b. Word practices
  c. Example sentences
  d. Chatbot responses

Audio files are then stored in Firebase Storage.

- Pronunciation Scoring

Cloud Functions run a scoring algorithm that:

  - Detects clarity
  - Compares pronunciation to native samples
  - Identifies mispronounced syllables
  - Returns feedback such as:
    - *"You pronounced the second syllable incorrectly."*
    - *"Try to shorten the vowel here."*

- Dynamic Lesson Adjustments

The system can automatically adjust difficulty based on:

  - User accuracy
  - Time taken
  - Mistake patterns

This creates a personalized learning experience for every user.

Data Flow

1. User Interaction & Local Validation

- Users interact with the UI layer via Activities, Fragments, or Jetpack Compose components.
- Input actions include:
    - Answering quiz questions
    - Recording speech
    - Chatting with the AI pronunciation assistant
    - Navigating lessons and practice exercises
    - Local validation occurs immediately
    - Quiz answers are checked for correctness
    - Speech input is pre-processed for basic validation (e.g., recording length, audio format)
    - UI feedback (correct/incorrect indicators, points, streaks) is displayed instantly

This provides instant responsiveness and ensures the user experience remains smooth even if the network is slow or unavailable.

2. Offline Storage & Queued Sync (RoomDB)

- All user progress, scores, and session data are stored locally in RoomDB when offline.
- Any changes are queued for synchronization with the cloud once an internet connection is available.
- Offline caching covers:
    - Completed lessons
    - Daily goals and progress streaks
    - Chatbot conversation logs
    - Audio recordings for TTS/STT processing

This offline-first strategy guarantees that learning is uninterrupted, regardless of connectivity.

3. Cloud Sync & AI Processing

Once the device is online:

- FireStore Synchronization
    - Queued data is automatically synced to FireStore in real-time.
    - Ensures:
    - Accurate leader board rankings
    - Updated user progress across device

    - Centralized lesson completion tracking

- Backend Notification
    - After syncing, the REST API backend is notified for AI processing.
    - Tasks include:
        - Speech-to-text (STT) conversion
        - Pronunciation analysis
        - Dynamic lesson adjustment based on performance
- AI Chatbot & TTS Responses
    - The AI chatbot returns:
        - Text-based responses
        - TTS-generated audio for proper pronunciation
        - Structured feedback on fluency, tone, and grammar
    - All responses and audio are stored locally and in Firebase Storage for future reference and offline access.

4. End-to-End Flow Summary

- User interacts with the app → immediate local validation.
- Progress written to RoomDB if offline → queued for sync.
- Device online → syncs to Firestore + notifies backend for AI processing.
- AI processes data → returns chatbot responses, TTS audio, pronunciation scores. - Data stored locally and in the cloud → ready for review, playback, or analytics.

This ensures that every interaction is tracked, analysed, and returned as actionable feedback, making Mzansi Lingo a seamless and effective language-learning platform.

UI / UX

1. Camera-First & Speech-First Lessons

- Lessons are designed around camera and microphone input:
- Users can record themselves speaking for pronunciation practice.
- Certain exercises may use the camera for interactive gestures or visual cues.
- This approach supports multimodal learning, combining auditory, visual, and kinesthetic engagement to improve retention and fluency.

2. Clear Visual Hierarchy

- The UI follows a logical visual hierarchy, guiding users naturally through each screen:
- Primary actions (e.g., “Start Lesson,” “Record Answer”) are highlighted.
- Secondary actions (e.g., settings, help, profile) are accessible but less prominent.
- Typography, color coding, and spacing are used to emphasize progression, completion, and feedback.

3. Accessible Controls

- All interactive elements are designed for ease of use:
- Large touch targets for buttons and controls − Voice-over compatibility for visually impaired users
- Simple gestures for common actions like swiping between exercises or replaying audio
- Notifications and prompts are designed to be non-intrusive but informative.

4. Progressive Disclosure for Advanced Settings

- Advanced or less frequently used options are hidden by default and revealed progressively:
- Users can access notifications, privacy settings, and profile management only when needed.
- Keeps the interface clean and uncluttered for first-time or casual learners.
- This ensures a streamlined experience while still offering full configurability for power users.

Accessibility

1. Semantic Components for Screen Readers

- All UI elements are built with semantic labelling, allowing screen readers to interpret content accurately.
- Examples include:
    - Buttons, sliders, and interactive cards with descriptive labels
    - Lesson content tagged with proper headings and roles
    - This enables visually impaired users to navigate the app and access all functionality without missing context.

2. High-Contrast Theme

- The app provides a high-contrast colour theme option:
- Text and interactive elements stand out clearly against backgrounds
- Improves readability for users with visual impairments or low vision
- The theme can be toggled in settings, supporting personalized accessibility preferences.

3. Adjustable Font Sizes & Captions

- Users can adjust font sizes across the app to suit their reading comfort.
- Audio and video content include captions and transcripts:
    - Learners who are deaf or hard of hearing can follow along with lessons
    - Supports better comprehension for non-native speakers.
- These features ensure that the app remains inclusive and usable for a wide range of learners

Security & Privacy

1. TLS for All Network Requests

- All data transmitted between the app and backend servers is encrypted using TLS (Transport Layer Security).
- This ensures:
  - Protection against eavesdropping
  - Secure communication for chat messages, lesson data, and audio files
- Users can safely interact with the AI chatbot and access cloud-based features without risk of data interception.

2. Authentication via Firebase Auth + Optional Biometrics

- Firebase Authentication is used for secure login and registration:
  - Users can sign in quickly with Google Single Sign-On (SSO)
- Optional biometric authentication (fingerprint or facial recognition) is available on supported devices:
  - Provides faster login
  - Adds an additional layer of security for user accounts - This combination ensures both convenience and security.

3. Minimal Data Collection & User-Controlled Chat Logs

- Mzansi Lingo collects only essential data required for learning and analytics.
- Users have full control over their conversation history:
  - Chat logs can be deleted at any time
  - Ensures compliance with privacy best practices and builds user trust
- Audio recordings, lesson progress, and other personal data are stored securely and encrypted in Firebase Firestore and Storage.

Performance & Scalability

1. Offline-First UX with RoomDB and WorkManager

- RoomDB is used for local persistence of all user data:
    - Lesson progress, daily goals, chat logs, and audio recordings are stored locally
    - Ensures that users can continue learning even without internet access - WorkManager handles background synchronization:
    - Queued data is automatically synced to the cloud when connectivity is restored
    - Guarantees data consistency across devices without interrupting the user experience
- This offline-first approach ensures uninterrupted learning in areas with poor connectivity.

2. CDN for Static Assets & Firebase Storage for Media

- Static assets (UI images, icons, app illustrations) are served via a Content Delivery Network (CDN) for fast, reliable loading worldwide.
- User-generated media (audio recordings, TTS outputs) is stored securely in Firebase Storage:
    - Low-latency access
    - Secure storage and retrieval
    - Optimized for mobile bandwidth and performance

3. Horizontal Scaling of Stateless Services on Railway

- Stateless backend services, including AI chatbot processing and session management, are hosted on Railway.
- Horizontal scaling ensures that additional service instances are created automatically under high load:
    - Supports large numbers of simultaneous users
    - Maintains responsiveness and low latency
- Stateless design reduces bottlenecks and allows robust, scalable infrastructure without downtime.

Technical Stack

1. Android (Kotlin) & Jetpack Components

- Kotlin is used for native Android development, providing concise syntax and strong type safety.
- Jetpack libraries enhance app architecture and maintainability:
  - Room: Local database for offline-first storage of lessons, progress, chat logs, and audio recordings
  - WorkManager: Background synchronization of queued data with the cloud ○ ViewModel: Manages UI-related data in a lifecycle-aware way
  - LiveData / Flow: Provides reactive data streams to update the UI in real-time
- This combination ensures a responsive, maintainable, and scalable mobile application.

2. Firebase Services

- Firebase Authentication: Secure login via Google SSO and optional biometric authentication
- FireStore: Cloud-based, real-time database for storing progress, leader board rankings, and chat logs
- Firebase Storage: Stores media assets such as user recordings and TTS audio
- Firebase Cloud Messaging (FCM): Push notifications for lesson reminders, new content, and chatbot responses
- These services provide real-time sync, secure storage, and reliable notifications across devices.

3. Railway-Hosted REST API

- The AI chatbot and session management services are hosted on Railway, providing a scalable and lightweight backend.
- Responsibilities include:
  - Processing user input for AI conversational interactions
  - Managing session data and conversation logs
  - Handling requests from the mobile client in real-time
- This architecture allows horizontal scaling, ensuring smooth performance even under high load.

4. Speech Recognition & Text-to-Speech

- Google Speech-to-Text (STT) or equivalent is used to convert spoken language into text for analysis.
- Text-to-Speech (TTS) provides audio feedback with accurate pronunciation.
- Integration with serverless cloud functions enables:

    - Pronunciation scoring
    - Real-time feedback on fluency and grammar
- These components make learning interactive, immersive, and culturally relevant.

Utilization of GitHub & GitHub Actions

Mzansi Lingo uses GitHub as the central platform for version control, collaboration, and project management, while GitHub Actions automates workflows to streamline testing, building, and deployment.

1. Utilization of GitHub

- Version Control & Branching:
  - Source code is managed using Git, with `main` for stable releases and `develop` for ongoing development.
  - Feature branches are created for new functionalities like AI chatbot integration or audio improvements.
- Collaboration & Code Review:
  - Pull requests ensure code is reviewed before merging, maintaining quality and consistency.
  - Issues and project boards track bugs, enhancements, and tasks.
- Documentation Management:
  - All project documentation, including README, design diagrams, and API specifications, is maintained in the repository. ○ Images, flow diagrams, and screenshots are stored in the repo for easy reference.

- Benefit: Centralized source control and documentation make collaboration efficient and maintain a complete project history.

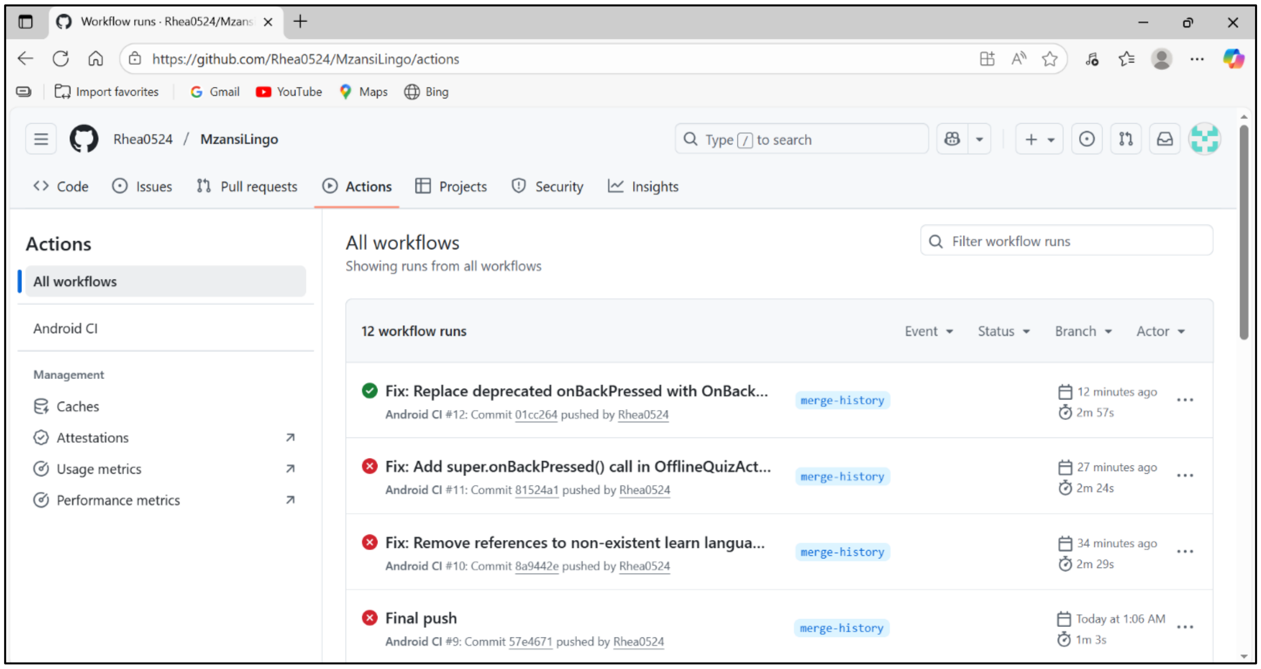
2. Utilization of GitHub Actions

- Continuous Integration (CI):

  - Automatically runs unit tests for key functionalities such as login, lesson completion, offline sync, and AI chatbot responses on every push or pull request.
  - Ensures that new changes do not break existing functionality. − Build Automation:
  - Android app builds (debug and release APKs) are generated automatically with Gradle, keeping artifacts up-to-date. − Deployment Automation:
  - Optionally deploys new builds to Firebase App Distribution for beta testing or internal QA.
  - Sends notifications for build status (success/failure), allowing rapid issue detection.
  - Scheduled tasks check code quality, run regression tests, and update documentation automatically.

- Benefit: GitHub Actions automates repetitive tasks, ensures consistent builds, and reduces manual errors, making development faster, safer, and more reliable.

GitHub Actions:

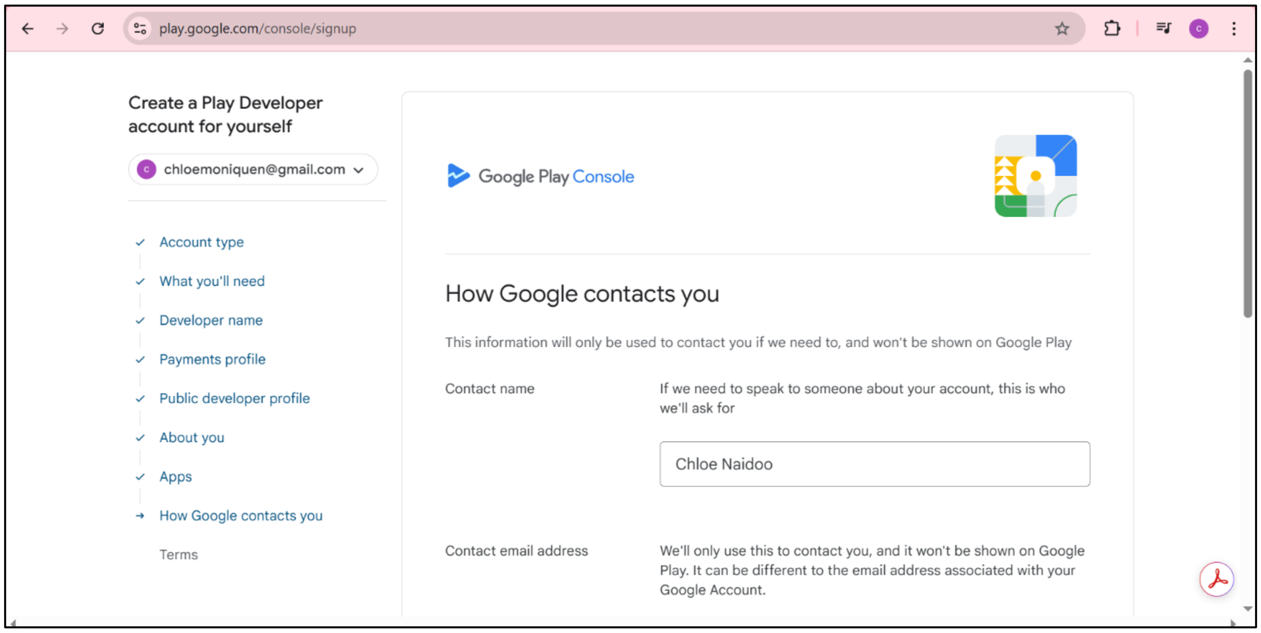


All workflows
Showing runs from all workflows

12 workflow runs

Fix: Replace deprecated onBackPressed with OnBack... Android CI #12: Commit 01cc264 pushed by Rhea0524	merge-history	12 minutes ago	2m 57s
Fix: Add super.onBackPressed() call in OfflineQuizAct... Android CI #11: Commit 81524a1 pushed by Rhea0524	merge-history	27 minutes ago	2m 24s
Fix: Remove references to non-existent learn langua... Android CI #10: Commit 8a9442e pushed by Rhea0524	merge-history	34 minutes ago	2m 29s
Final push Android CI #9: Commit 57e4671 pushed by Rhea0524	merge-history	Today at 1:06 AM	1m 3s

Play Store Publication:


play.google.com/console/signup

Create a Play Developer account for yourself

chloemoniquen@gmail.com

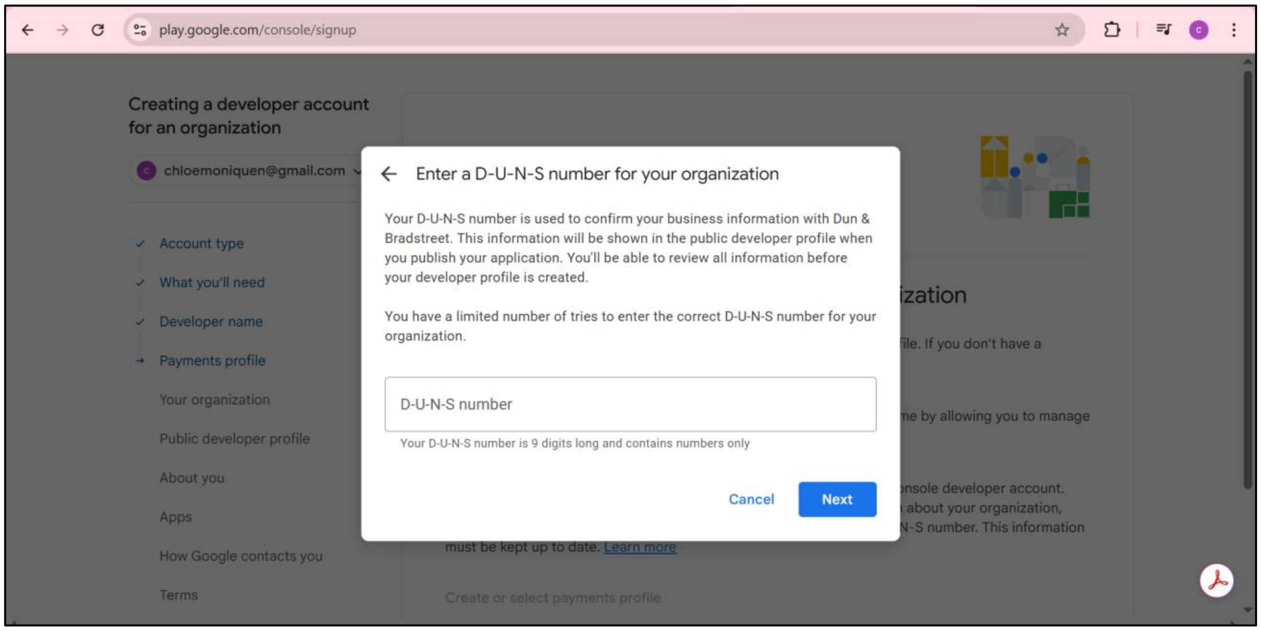
- ✓ Account type
- ✓ What you'll need
- ✓ Developer name
- ✓ Payments profile
- ✓ Public developer profile
- ✓ About you
- ✓ Apps
- → How Google contacts you
- Terms

Google Play Console

How Google contacts you

This information will only be used to contact you if we need to, and won't be shown on Google Play

Contact name	If we need to speak to someone about your account, this is who we'll ask for
	Chloe Naidoo
Contact email address	We'll only use this to contact you, and it won't be shown on Google Play. It can be different to the email address associated with your Google Account.


play.google.com/console/signup

Creating a developer account for an organization

chloemoniquen@gmail.com

- ✓ Account type
- ✓ What you'll need
- ✓ Developer name
- → Payments profile
- Your organization
- Public developer profile
- About you
- Apps
- How Google contacts you
- Terms

Enter a D-U-N-S number for your organization

Your D-U-N-S number is used to confirm your business information with Dun & Bradstreet. This information will be shown in the public developer profile when you publish your application. You'll be able to review all information before your developer profile is created.

You have a limited number of tries to enter the correct D-U-N-S number for your organization.

D-U-N-S number

Your D-U-N-S number is 9 digits long and contains numbers only

Cancel Next

AI Automation

1. Debugging and Technical Assistance: AI tools were consulted to help troubleshoot various Android-related issues. This included identifying the causes of Gradle configuration errors, resolving Kotlin import and reference problems, and addressing compatibility issues involving Firebase services and Android SDK versions. The suggestions provided were treated as guidance, and every fix was verified manually within Android Studio to ensure it worked correctly.

2. Code Guidance and Implementation Support: AI-generated examples were used to assist with specific coding tasks such as configuring Firebase Authentication, storing and retrieving data from Firestore, improving Canvas-based drawing functionality, and refining XML UI layouts. These examples were not used verbatim; instead, they were adapted, rewritten, or optimised to match the project's structure, coding style, and functional requirements.

3. AI-Generated Visual Assets: AI image generation tools were used to create the application's logo and some visual elements featured in the app. The generated graphics served as a base for the design and were further adjusted and resized to meet Android's image standards (such as density variations for different screen types). Only general, non-sensitive visual assets were produced using AI.

4. Learning and Concept Clarification: AI support was used to better understand technical concepts relevant to Android development. Topics included Activity and Fragment lifecycles, Firebase workflows, and best practices for responsive layouts. The explanations helped reinforce understanding, but all final documentation, summaries, and implementations were written in my own words.

5. Documentation and Writing Support: AI tools assisted in editing and improving sections of the written documentation for this assessment. This included refining sentence structure, improving clarity, and helping organise the content. All writing was reviewed and rewritten where needed to ensure it aligned with academic integrity guidelines.

6. Ensuring Accuracy and Original Work: Any content or code suggested by AI was thoroughly reviewed, tested, and modified before being included in the final application. AI did not perform the development on my behalf; instead, it served as a supplementary tool. All final logic, design decisions, coding implementations, and written work are my own.

Code Attribution / References

Android Developers. (n.d.) *Android developer documentation.* Available at: https://developer.android.com (Accessed: 18 November 2025).

JetBrains. (n.d.) *Kotlin language documentation.* Available at: https://kotlinlang.org/docs/home.html (Accessed: 18 November 2025).

Google. (n.d.) *Material Design guidelines.* Available at: https://m3.material.io (Accessed: 18 November 2025).

Android Developers. (n.d.) *Guide to connecting Android apps to REST APIs.* Available at: https://developer.android.com/topic/libraries/architecture (Accessed: 18 November 2025).

Square. (n.d.) *Retrofit: A type-safe HTTP client for Android and Java.* Available at: https://square.github.io/retrofit (Accessed: 18 November 2025).

Firebase. (n.d.) *Firebase documentation.* Available at: https://firebase.google.com/docs (Accessed: 18 November 2025).

OpenAI. (n.d.) *API documentation and chatbot integration guides.* Available at: https://platform.openai.com/docs (Accessed: 18 November 2025).

GitHub Docs. (n.d.) *Git and GitHub documentation.* Available at: https://docs.github.com (Accessed: 18 November 2025).

Android Developers. (n.d.) *Layouts and activity structure: Empty Activity guide.* Available at: https://developer.android.com/guide/components/activities (Accessed: 18 November 2025).

Android Developers. (n.d.) *RecyclerView guide.* Available at: https://developer.android.com/guide/topics/ui/layout/recyclerview (Accessed: 18 November 2025).

Bumptech. (n.d.) *Glide: Image loading library documentation.* Available at: https://bumptech.github.io/glide (Accessed: 18 November 2025).

Stack Overflow. (n.d.) *Stack Overflow – Developer community discussions.* Available at: https://stackoverflow.com (Accessed: 18 November 2025).

YouTube. (n.d.) *Android development tutorials.* Available at: https://www.youtube.com (Accessed: 18 November 2025).

Kotlin Coroutines. (n.d.) *Kotlin coroutines guide.* Available at: https://kotlinlang.org/docs/coroutines-overview.html (Accessed: 18 November 2025).