Daily Report

Jurgen Halili 11/09/2025

Today, I started working on a new Epic called Implement Comprehensive QA Functionality for Agent Testing. This Epic consists of multiple subtasks, and I was able to complete several of them.

I began with the database schema and backend foundation. For this, I designed and implemented two new tables: qa_contacts and qa_call_logs. The qa_contacts table stores information such as user, name, phone, email, and role, while the qa_call_logs table records details about calls, including call status, duration, agent configuration snapshots, and notes. I also created the necessary database migrations, set up proper relationships between the tables, added indexing for performance, and ensured that the migrations are reversible. This establishes the base needed for supporting QA contacts and logging functionality.

Next, I worked on refactoring the configuration interface. The goal here was to reorganize it so that voice configuration is separated from QA contacts management. I split the interface into two distinct tabs: Voice Config and QA Contacts. The Voice Config tab maintains all its existing functionality, while the new QA Contacts tab provides a dedicated space for managing contacts. I also made sure that the navigation between tabs is responsive, intuitive, and that the state is preserved when switching back and forth.

After this, I implemented QA contacts management itself, focusing on CRUD operations. Users can now create, view, update, and delete QA contacts directly within the QA Contacts tab. I added a form with validation for creating contacts, a searchable and filterable contact list view, and edit functionality with pre-populated forms. Deleting a contact now requires confirmation to prevent mistakes, and I also included bulk operations such as import and export to make it easier to manage larger sets of contacts.

Once the UI side was complete, I developed the supporting API endpoints for QA contacts. These include endpoints for creating, listing with pagination and filtering, updating, deleting, and retrieving individual contact details. I implemented validation, sanitization, and proper error handling to ensure stability. The API now returns meaningful messages with the correct status codes, and I updated the documentation to reflect these new endpoints.

Finally, I integrated the QA section into the Testing tab. This new section displays all the saved QA contacts and allows users to select one or more of them using checkboxes. I added “Select All” and “Deselect All” options, along with clear visual

indicators for selected contacts. The section handles both normal and empty states cleanly, making the QA testing process more intuitive for users.

Overall, this work sets up a solid foundation for comprehensive QA functionality, covering both backend infrastructure and front-end usability.