# Implementations, Refinements and Plans for the Italian Chat Bot Application: "La Vita Italiana"

DANIEL BERRY, Clemson University, USA

The original designs of the Italian Chatbot Application have been transformed and refined into a more coherent and attainable vision. Over the course of the last five weeks, the project has approached an appropriate stopping point to begin testing and deployment.

CCS Concepts: • **Human-centered computing** → **Human computer interaction (HCI)**; • **Applied computing** → **Language translation**.

Additional Key Words and Phrases: Language, Italian, LLMs, Game, Learning

**ACM Reference Format:**
Daniel Berry. 2018. Implementations, Refinements and Plans for the Italian Chat Bot Application: "La Vita Italiana". 1, 1 (March 2018), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 Introduction

The original vision of the application has morphed into a clean and efficient design targeted towards capturing one or two example Italian conversations. The focus from now until deployment will be to refine and reduce as much application scope as possible. The project has already reached an untenable number of moving parts and needs to be reigned in. Between the interface, chat bot, and back-end engineering, the application is fast approaching the inflection point to begin considering deployment.

## 2 Interface Design

The prototype has all necessary parts of the application currently in place. The landing page for the web application is just a login form with no other content except a link to a similar registration form. All users will be required to register before they can use the chat bot. Once registered, they are immediately redirected to the game interface.

### 2.1 Point and Click Interface

Users are able to navigate around the game world with a mouse and choose destinations to visit in the fictional Italian village. Each of the images was generated as placeholders through the use of an image generation service provided by DeepAI[1]. When hovering over a specific location, users will see that they can click to navigate to a different location. In each world location, there are different characters users can choose to interact with (Refer to Fig. 1).

Author's Contact Information: Daniel Berry, dpberry@clemson.edu, Clemson University, Clemson, South Carolina, USA.


ACM XXXX-XXXX/2018/3-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn


Fig. 1. Drogheria" Location Interface

## 2.2 Chat Interface

When they choose a character to interact with, they can strike up a conversation with them through the chat interface. User prompts are submitted to the back-end, go through several levels of validation and a generated response is returned to the user. The primary responses are from the character, but the more valuable information comes from the grammar check that displays a grammar analysis at the bottom of the chat interface (Refer to Fig. 2).

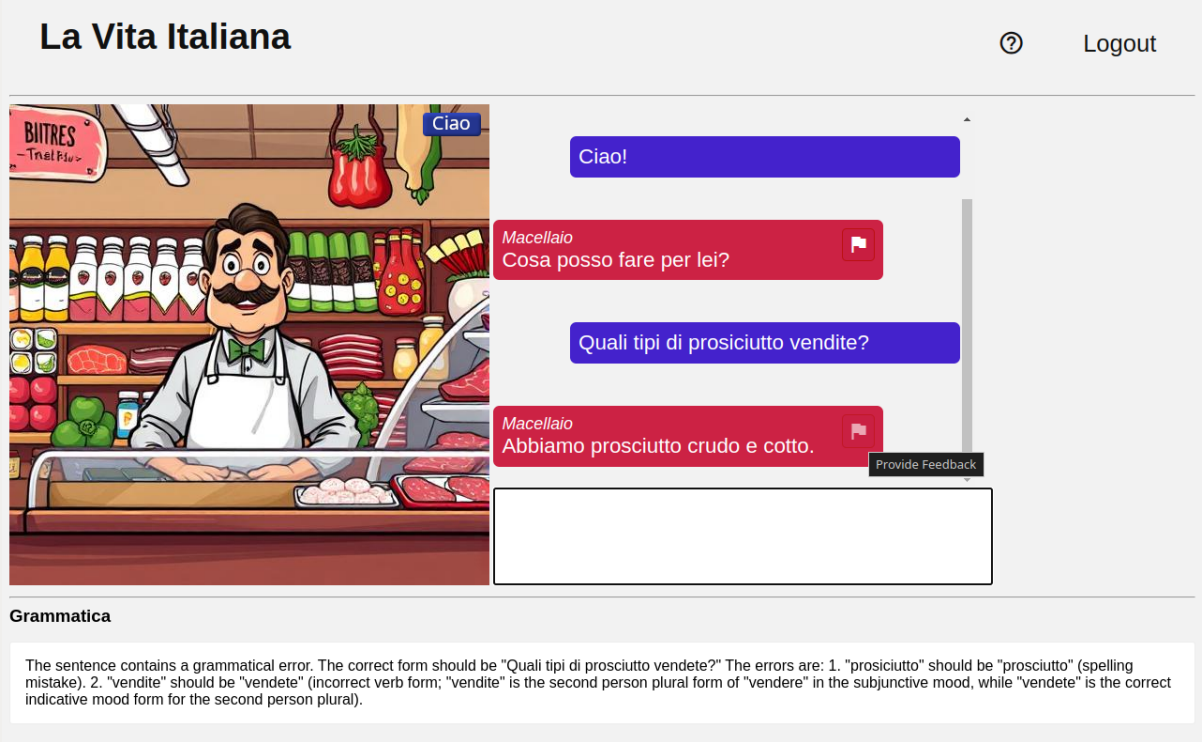


Fig. 2. "Chat and Grammar" Interface

## 3 Feedback

Currently, there are three avenues for collecting feedback from users that we will use to improve the application experience.

### 3.1 Implicit User Feedback

During registration, new users are required to acknowledge that the website will be collecting and using their messages to improve the application experience. Saving the submitted user prompts will be the primary source of feedback in this project. We will use the messages to train later iterations of the chat bot and improve responses. The chat histories can provide insight into how users interact with the application and allow us to identify outlier responses or unintended behavior. Each message is keyed to the user that submitted it, allowing us to aggregate all the messages of a single user. This way, we can weight the value of messages for a single user. If, for instance, one user has much higher quality prompts than another user, we may want to adapt how the model learns from the lower quality messages. Each of these data collection strategies will give us the greatest leverage to improve the chat bot's responses.

### 3.2 Intentional User Feedback

On every response from the chat bot, the interface features a flag button that will open a popup modal that allows users to report an issue with the system (Refer to Fig. 2). There are four types of feedback they can submit: "Offensive Content", "Suggest an Improvement", "Incorrect Grammar", and "Other". Once submitted, the feedback


Fig. 3. Feedback Submission Interface

can be used to flag and review issues with the chat bot system (Refer to Fig. 3). The most useful of these feedback types will be the "Suggest an Improvement" submissions. This version of feedback gives users the chance to contribute a different version of the response that they consider more correct. With a careful curation of these submissions, they can be used to help train the model similarly to how we use the messages.

### 3.3 Survey

The final consideration for user feedback, comes in the form of a survey. Eight different and open-ended questions are integrated into the interface as a popup modal. Once a user finishes their very first character interaction, they are presented with the option to complete the survey. Users can submit any answer they want, allowing the opportunity to raise unforeseen concerns with the application.

## 4 Implementation

There are three distinct layers of the current implementation in place: the front-end, the back-end, and the Large Language Model. Each section passes data back and forth between layers handling its own distinct responsibility.

### 4.1 The Front-End

Currently the bulk of the "point and click" game and chat interface are designed using basic HTML and JavaScript elements. The interface is accessible to a wide variety of screen resolutions. Three distinct pages exist on the website: "Login", "Register", and "The Game".

### 4.2 The Back-End

The server side implementation is a Laravel[3] application setup with endpoints to handle submissions for logins, feedback, chats, and surveys. The back-end validates user inputs and will throttle requests as necessary. The application needs to be robust against automated programs and targeted attacks. Malicious users may attempt to expose user data, steal API keys, or exploit the LLM. Validated inputs are saved to a SQLite database and then connections are negotiated between the different endpoints within the LLM layer.

### 4.3 The Large Language Model

There are two separate large-language models being put to work to produce chat bot responses. The primary model is based around the Open Source Rasa Chat Bot[5] project. In the core of the Rasa[5] Project, we are able to plug in and get fluid and dynamic responses from a separate LLM such as the "ChatGPT 4o Mini[4]" model. The Rasa[5] chat bot model is primarily trained through their system of configuration files, story lines, and interactive shell interface. In its current state, it is not capable of very complex or unique conversations. This highlights the importance of collecting user prompts. With the input we collect from users, we will be able to train a much more capable chat bot.

### 4.4 Grammar Check

As far as utility to language learners, the grammar check system is one of the most valuable parts of the application. Users are able to receive immediate and direct feedback on their speech in real time. This system is parallel and separate to the Rasa[5] model. It communicates directly with the ChatGPT[4] API using the "4o Mini" model. Carefully constructed prompts are sent to the API with requests to return specific responses for different cases. In the event that the grammar is correct, the response will simply be "yes". If the user submits a prompt that's primarily in a language other than Italian, it will recognize this and force the user to interact in Italian. The main response we expect back is an in-depth analysis of the grammar of the user's message. This analysis is returned to the front-end and displayed in a separate area.

### 4.5 Paladin Check

The paladin check yet to be implemented, but should not be too difficult to add to the application. The intent for this check is to identify responses from the Rasa[5] chat bot that may be offensive, grammatically incorrect, or out of touch with cultural norms. Similar to the grammar check, we will send responses through the ChatGPT[4] API to check for unintended results before they are sent to the user. The primary obstacles to this final feature will be significantly slower chat response times and associated cost increases with OpenAI[4].

## 5 Plans for Deployment

The three layers of the application are packaged between two different Docker[2] containers. These containers will be hosted either on my personal server: "danielberry.tech" or split with a separate GPU accelerated server to handle the Rasa[5] model responses.

### 5.1 The Road to Production

There is very little left to complete between now and deployment. The primary drivers of development will be refinement of the existing interactions and expansion of content within the game world. The targeted date for deployment to production is an ambitious end of March deadline. In order to start collecting as much user data as possible, the application needs to be publicly available by March 31st. This release deadline will approximately allow for a two week period of user testing and evaluation. In order to bring in test users, a multifaceted marketing

campaign will be put in place across various online channels. With enough users and attention, there should be enough data to work with for the third and final checkpoint.

## References

[1] DeepAI. 2025. . Retrieved Mar 16, 2025 from https://deepai.org/machine-learning-model/text2img
[2] Docker. 2025. . Retrieved Mar 16, 2025 from https://docs.docker.com/engine/
[3] Laravel. 2025. . Retrieved Mar 16, 2025 from https://laravel.com
[4] OpenAI. 2024. . Retrieved Mar 16, 2025 from https://platform.openai.com/docs/quickstart
[5] Rasa. 2025. . Retrieved Mar 16, 2025 from https://rasa.com/docs/rasa/http-api

Received 16 March 2025