Technical Documentation
Coding Journal Application

Naveen Prabakar

August 3, 2025

1 Overview

The Coding Journal project is an interactive journaling application designed specifically for software developers to track their programming activities, reflect on progress, and analyze challenges or learning curves. It supports both command-line and web interfaces and features integrated natural language processing tools for grammar and spell checking.

The core functionalities include journal creation, stepwise journal entry with guided questions, retrieval and modification of previous entries, and correction of language input.

2 Architecture and Components

2.1 Modules and Responsibilities

- `codingJournal.py`:
  - Command-line interface for creating, modifying, and retrieving journal entries.
  - Provides a menu-driven user experience allowing users to:
    - Create new journals and entries,
    - Edit or find previous journal entries by date,
    - Quit the program.
  - Integrates calls to other modules for database operations and natural language processing.
- `PreviousJournal.py`:
  - Handles retrieval and modification of existing journal entries stored in a MySQL database.
  - Performs database queries to:
    - Check the existence of a journal,
    - Extract journal entries based on date,
    - Update entries with grammar and spelling corrections.
  - Uses parameterized queries to prevent SQL injection and allows field-specific updates.
- `Newjournal.py`:
  - Manages database creation and writing new journal data into MySQL.
  - Supports:
    - Creation of new journal databases,
    - Creation of journal table schemas,

    - Insertion of new entries with cleaned text,
    - User login creation (partially implemented).
  - Integrates natural language processing for grammar and spell correction before storing entries.

- `questions.py`:
  - Contains a fixed list of reflection questions targeted for developers.
  - Questions serve as prompts during the journal entry creation process.

- `Grammer.py`:
  - Provides natural language processing utilities for spell checking and grammar correction.
  - Uses `TextBlob` for spelling correction.
  - Uses `language_tool_python` for advanced grammar checking.
  - Returns cleaned and corrected strings for insertion into journal entries.

2.2 Technology Stack

- **Backend:** Python 3.x
- **Database:** MySQL accessed through `mysql.connector` Python library
- **NLP Libraries:** `TextBlob` and `language_tool_python`
- **Interface:** CLI in `codingJournal.py`; potential for web integration via Flask extension (not included)

3 Detailed Module Descriptions

3.1 `codingJournal.py` (CLI Interface)

- Provides a menu-driven interface with options to:
  - Create new journal or entries,
  - Find previous journal entries by date,
  - Modify existing entries,
  - Quit the application.
- User inputs and selections are validated.
- Calls `Newjournal` or `PreviousJournal` for persistence operations.

3.2 `PreviousJournal.py`

- Supports:
  - `checkname(journal)`: Checks if the specified journal database exists.
  - `extract(date, name)`: Retrieves all journal fields for a given date.
  - `modify(date, name)`: Allows selective field updates with grammar/spell correction.
- Uses parameterized SQL queries and ensures safe, accurate modifications.
- Handles database connection opening/closing.

3.3 `Newjournal.py`

- Functions:
  - `createJournal(name)`: Creates a new MySQL database named after the journal.
  - `createTables()`: Creates the `entries` table within the journal database.
  - `InsertData(...)`: Inserts a new journal entry with grammar/spell cleaned data.
  - `createLogin(name, password)`: Intended to create user login entries (currently minimal).
- Calls `Grammer.py` for text correction before database insertion.

3.4 `questions.py`

- Contains a list of domain-specific structured journal prompts for programming reflection, e.g.:
  - "What did you work on today?"
  - "What problems or bugs did you face?"
  - "What could you have done better?"
  - "What will you tackle next?"
- Used by the CLI and web routes to guide user journaling systematically.

3.5 `Grammer.py`

- Corrects user input text by:
  - Spell checking and correcting with `TextBlob`.
  - Grammar checking and fixing with `language_tool_python`.
- Returns the corrected strings ready for database storage.

4 Database Schema

```
Database: <journal_name>

Table: entries

Columns:
- description     VARCHAR(500)
- errors          VARCHAR(200)
- buddy           VARCHAR(100)
- Todo            VARCHAR(300)
- additional      VARCHAR(100)
- date            DATETIME PRIMARY KEY
```

Note: The field "additional" may be misspelled as "addtional" in the code and should ideally be corrected.

5 Usage and User Flow

1. User launches the `codingJournal.py` script.
2. Menu options allow creating a new journal database, entering responses to questions prompt-by-prompt, or retrieving/modifying past entries.
3. As user inputs text, it is automatically corrected for grammar and spelling.
4. Journal entries are saved persistently in MySQL under the named databases.
5. Previous entries can be searched by date and edited seamlessly.

6 Security and Validation

- Uses parameterized SQL to prevent injection attacks.
- Currently, user login handling is minimal and could be enhanced with hashed passwords.
- Input validation and error handling exists but can be improved for robustness.

7 Potential Future Improvements

- Complete user authentication and secure login system with password hashing.
- Develop a web interface (e.g., Flask) for richer user experience.
- Add input validation, error feedback, and logging.
- Extend NLP processing for sentiment analysis or automated summary generation.
- Add export features (PDF, Markdown) for journal entries.
- Integrate reminders or journaling habit tracking.

8 Summary

This Coding Journal system offers programmers an integrated journaling platform with natural language corrections and a structured data backend. Its modular Python design separates concerns of persistence, input correction, and user interaction while providing a user-friendly way to reflect on coding progress. The codebase supports easy expansion toward web deployment and richer features.