Team G

OopsFix
Design Report

Version 1.0

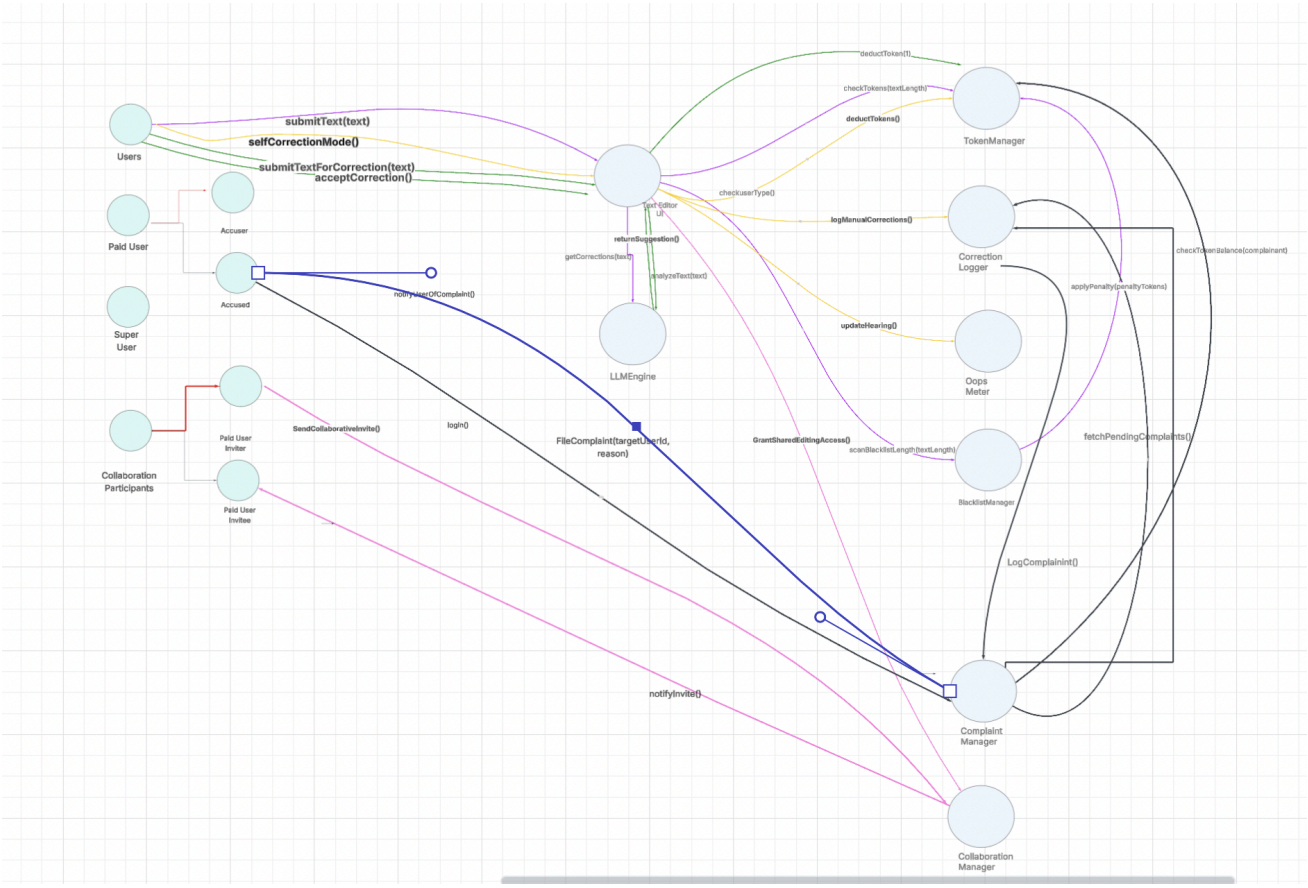
Revision History

Date	Version	Description	Author
4/13/25	1.0	Created Collaboration Class Diagram Worked on OopsFix UI	Nur Dogrusoz
4/13/25	1.0	Updated teamwork Updated use-case description	Klea Meta
4/13/25	1.0	Detailed design	Sujana Yeasmin
4/13/25	1.0	Created use-case diagrams	Gaurav Gupta
4/13/25	1.0	Updated ER Diagrams	Tony Christopher

Table of Contents

1. Introduction

collaboration class diagram



2. Use-Cases

Section 2.1 will describe each of the use-cases, their normal and exceptional scenarios, and the classes involved in each case; a sequence class diagram will be provided for each use case. Section 2.2 will display petri nets for select use cases.

2.1 Cases and Sequence Class Diagrams

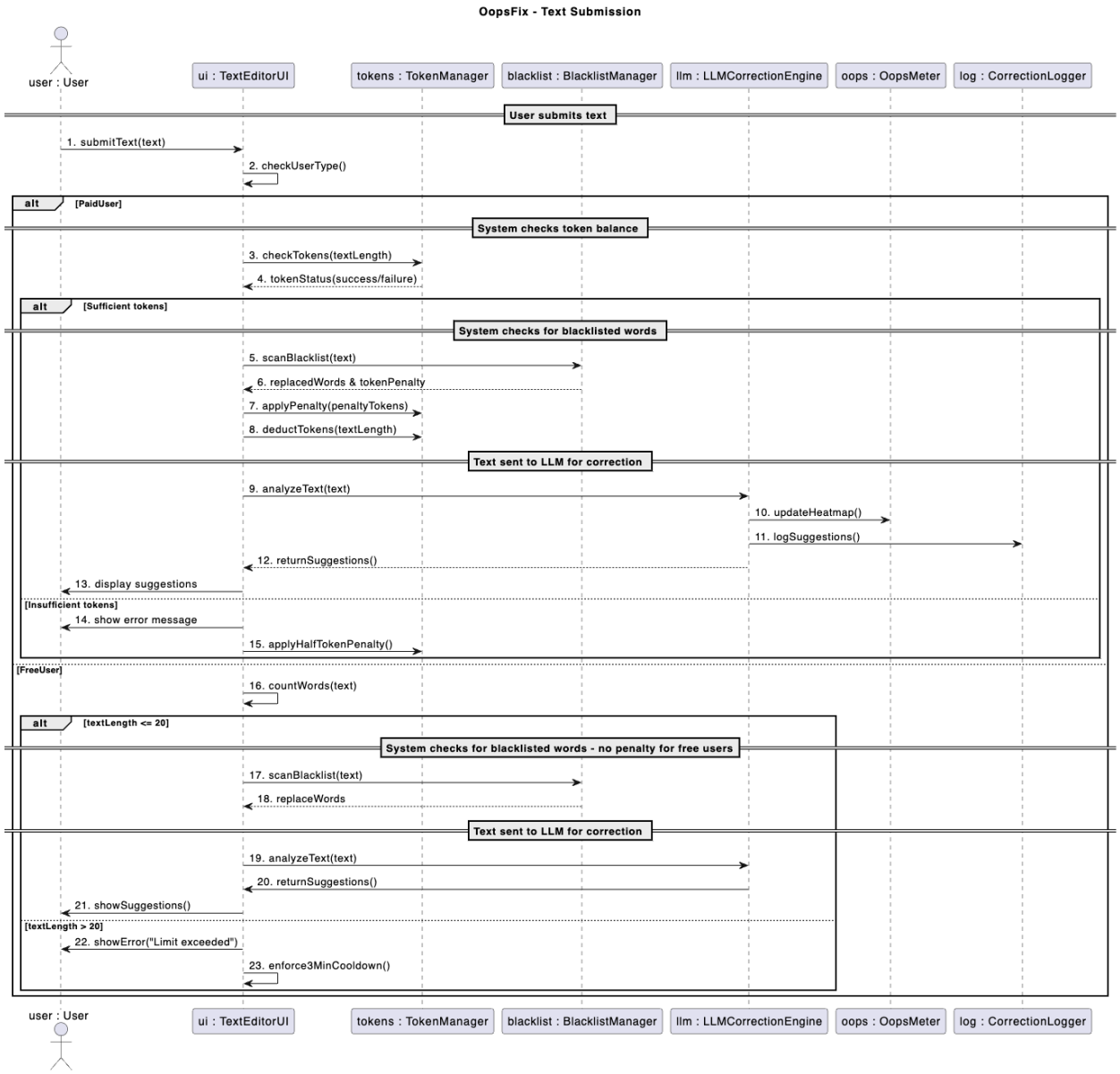
2.1.1 Text Submission

Users submit text for editing via the OopsFix UI.

Upon text submission by a PU, the system computes the word count and deducts one token for each word prior to processing the content. If any banned terms are detected, they are substituted with asterisks, and further tokens are deducted in accordance with their length. When the user possesses an inadequate number of tokens, the submission will be unsuccessful, resulting in the loss of half of the remaining tokens as a penalty.

Normal: user possesses sufficient tokens, resulting in successful text processing.

Exceptional: user possesses insufficient tokens, in which case the submission is rejected and a penalty is applied.

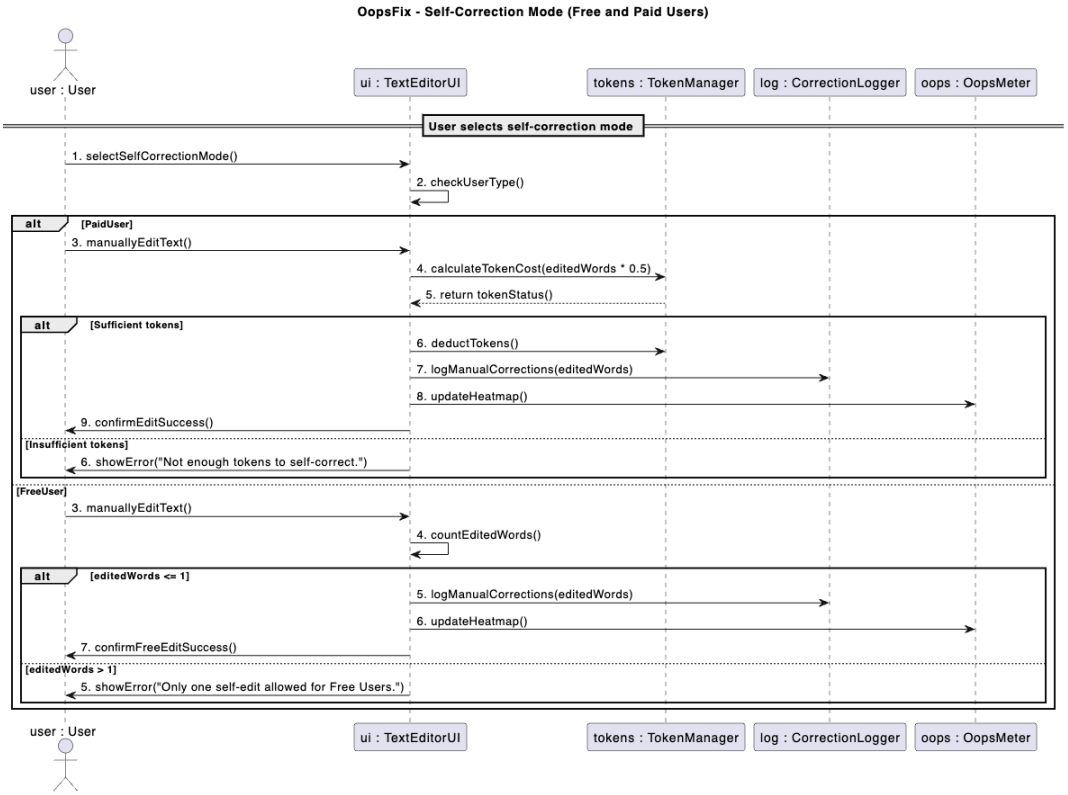


2.1.2 Self-Correction

Users can choose self-correction mode to manually edit their text. Paid users are charged 0.5 tokens per word they edit, while free users are allowed to make only one manual edit per session.

Normal Scenario: The user selects self-correction mode and begins editing their text. If the user has enough tokens, each word edited deducts 0.5 tokens from their balance. The editing session continues smoothly without interruption, and the changes are successfully applied.

Exceptional Scenario: The user starts editing but does not have enough tokens to complete the changes. In this case, the system halts the editing process, displays an error message indicating insufficient tokens, and prevents the user from making further edits until more tokens are acquired.

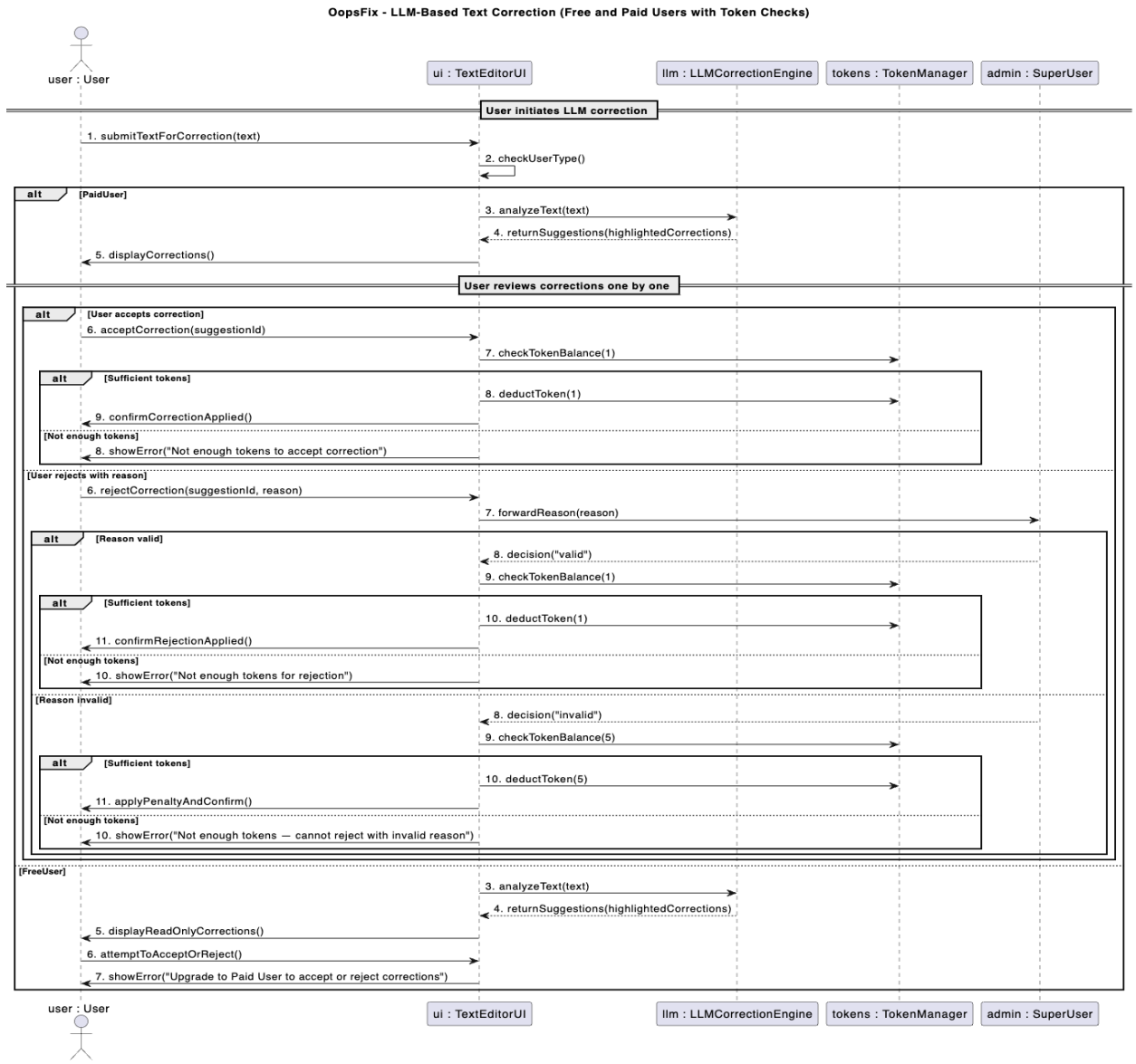


2.1.3 LLM-Based Text Correction

The system scans the submitted text for grammar, spelling, and syntax errors and highlights suggested corrections. For every accepted correction, 1 token is deducted. if the user rejects an LLM-generated correction, they must provide a reason, which is then reviewed by the super user (SU). If the reason is deemed valid, only 1 token is deducted; if the reason is invalid, the user loses 5 tokens as a penalty.

Normal: All corrections either accepted or rejected with valid reason

Exceptional: Corrections are rejected with invalid reason

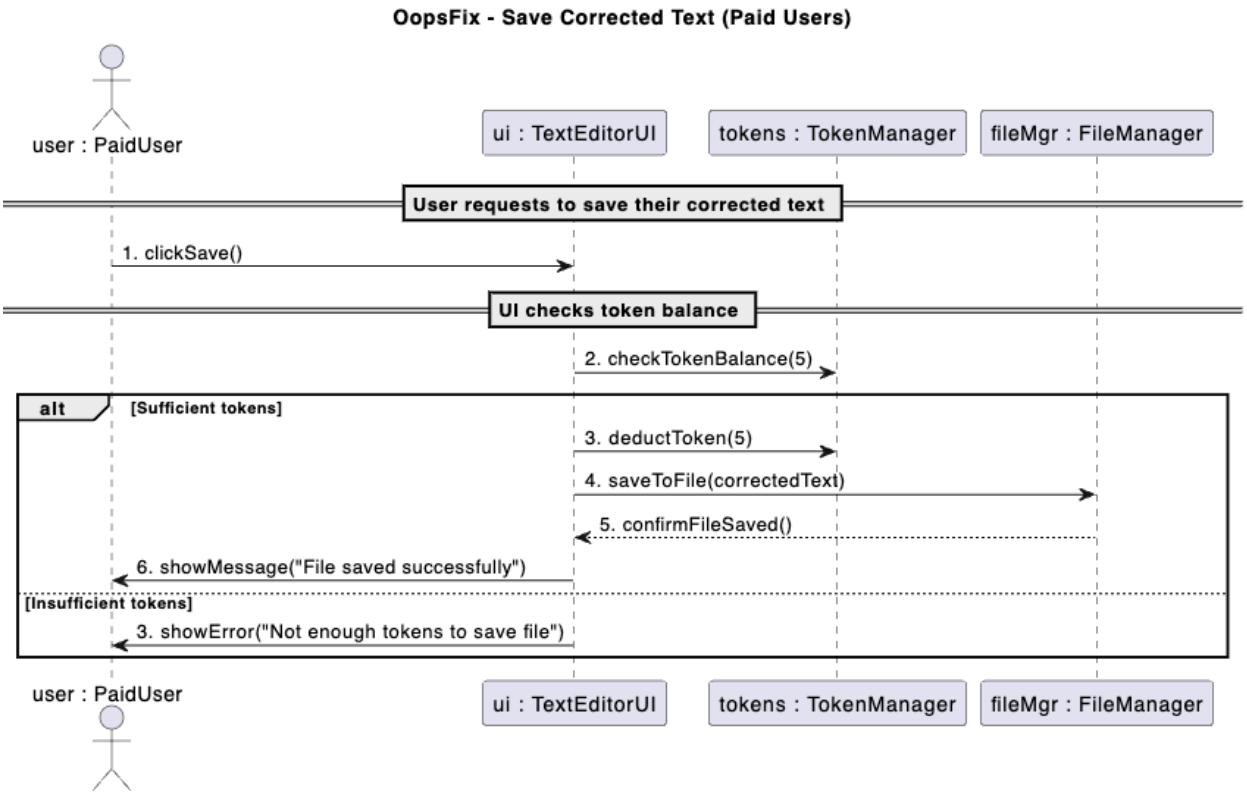


2.1.4 Save Edited Text (Paid Users Only)

Paid users can save their edited text after completing corrections. This action costs 5 tokens, which are deducted from the user's token balance at the time of saving.

Normal Scenario: The user has at least 5 tokens available. Upon clicking “Save,” the system successfully stores the edited document and confirms the save with a success message. The token balance is updated accordingly, and the document is made accessible for future viewing or download.

Exceptional Scenario: The user has fewer than 5 tokens. When the “Save” button is clicked, the system blocks the save operation and displays an error message indicating insufficient tokens. The user is prompted to purchase more tokens before trying to save again.

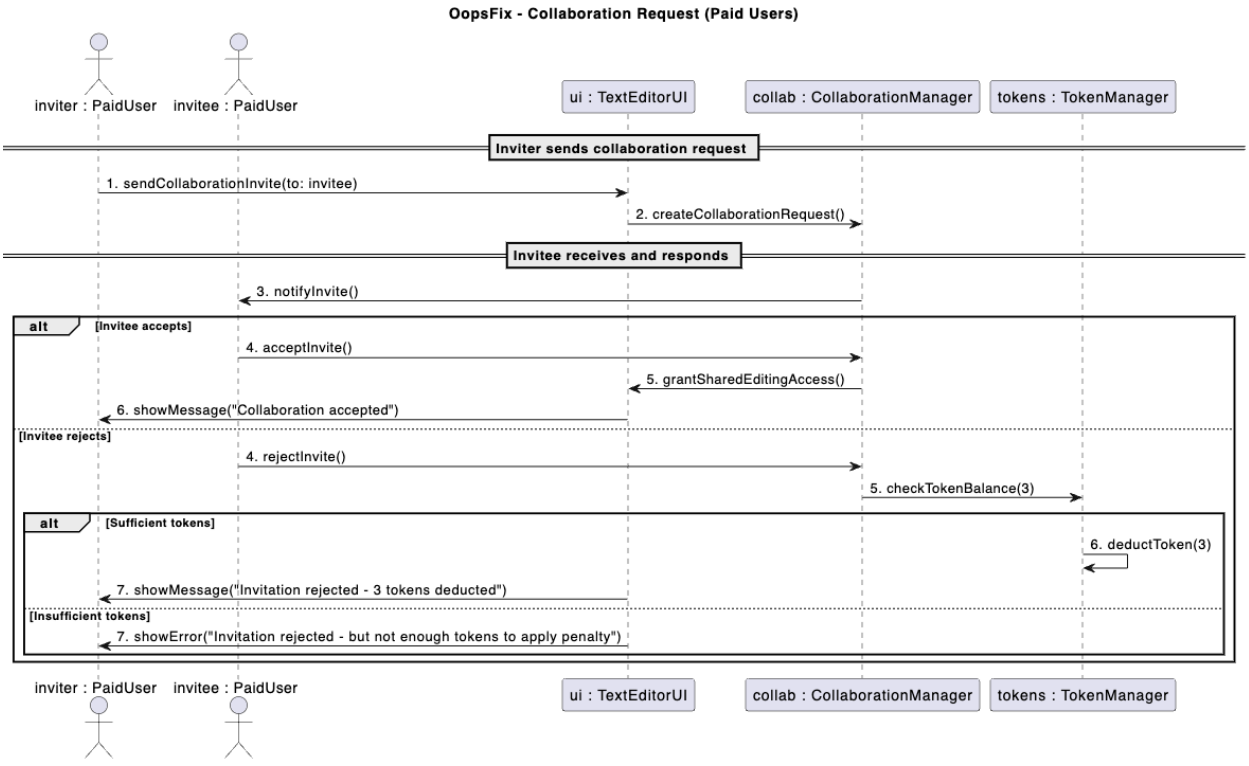


2.1.5 Collaboration (Paid Users Only)

Paid users can invite other paid users to collaborate on a shared text file. This feature allows both users to co-edit the same document using either self-correction or LLM-assisted correction tools.

Normal Scenario: The inviter sends a collaboration request, and the invited user accepts. Both users are granted shared editing access to the document. They can now make edits collaboratively in real time or asynchronously. No penalty is applied, and the collaboration session begins successfully.

Exceptional Scenario: The invited user rejects the invitation. As a result, the inviter is charged a penalty of 3 tokens. If the inviter has fewer than 3 tokens at the time of rejection, the system blocks the invitation or applies an alternative penalty (such as a warning or partial deduction), and the user is notified about the insufficient token balance.

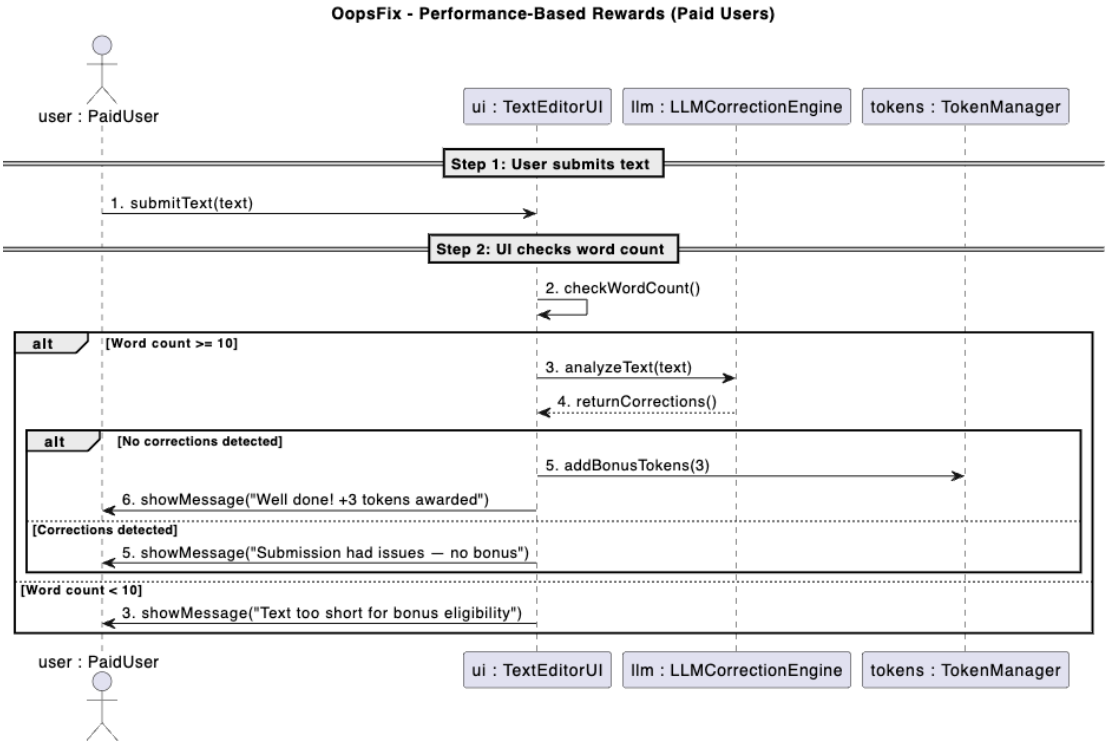


2.1.6 Performance-Based Rewards (Paid Users)

To promote high-quality writing, OopsFix rewards paid users who submit well-structured text. If a submission contains 10 or more words and the AI detects no grammar, spelling, or syntax errors, the user is granted a bonus of 3 tokens.

Normal Scenario: The user submits a well-written text of at least 10 words. The AI scans the content and finds no errors. As a result, the user is awarded 3 bonus tokens, which are added to their token balance. A confirmation message is displayed to notify them of the reward.

Exceptional Scenario: The user submits a high-quality text, but the AI incorrectly flags one or more parts as errors. Due to the false detection, the system does not issue the bonus. The user may feel the submission was mistakenly evaluated but receives no tokens unless a manual review is requested through the super user.

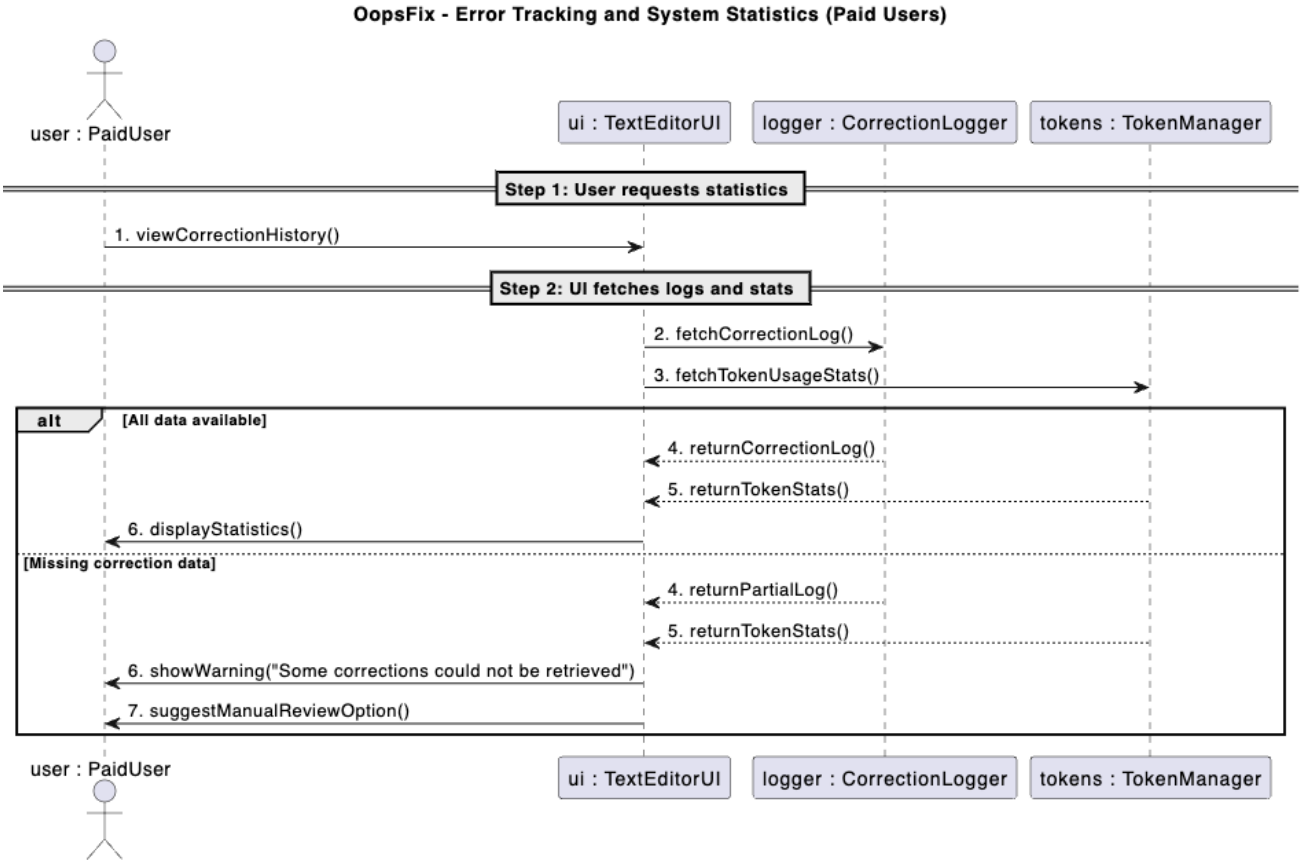


2.1.7 Error Tracking and System Statistics (Paid Users)

OopsFix provides detailed logs that allow paid users to track their error history, correction patterns, and token usage statistics. This feature enables users to identify frequent mistakes and monitor their progress over time.

Normal: users can access their correction logs and view system-generated insights with no errors.

Exceptional: system fails to record a correction, user may need to request a manual review from a super user.

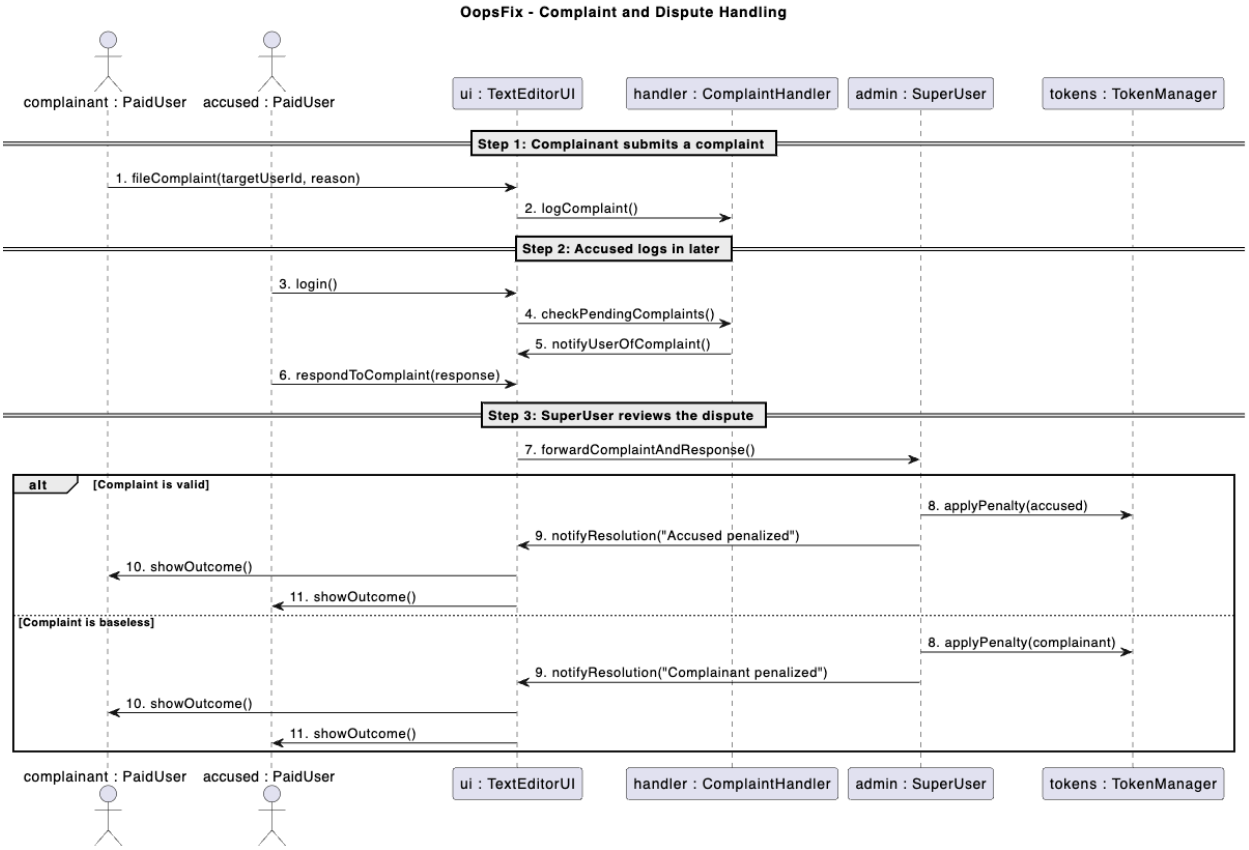


2.1.8 Complaint and Dispute Handling

If a user feels that a collaborator has violated system policies, they can file a complaint with the super user. Upon login, the accused user is notified and must respond to the complaint. The super user reviews the complaint and determines a resolution, which may involve penalizing either the complainant or the accused user.

Normal: disputes are fairly resolved, with appropriate token deductions applied.

Exceptional: complaint is found to be baseless, in which case the complainant may face penalties instead.

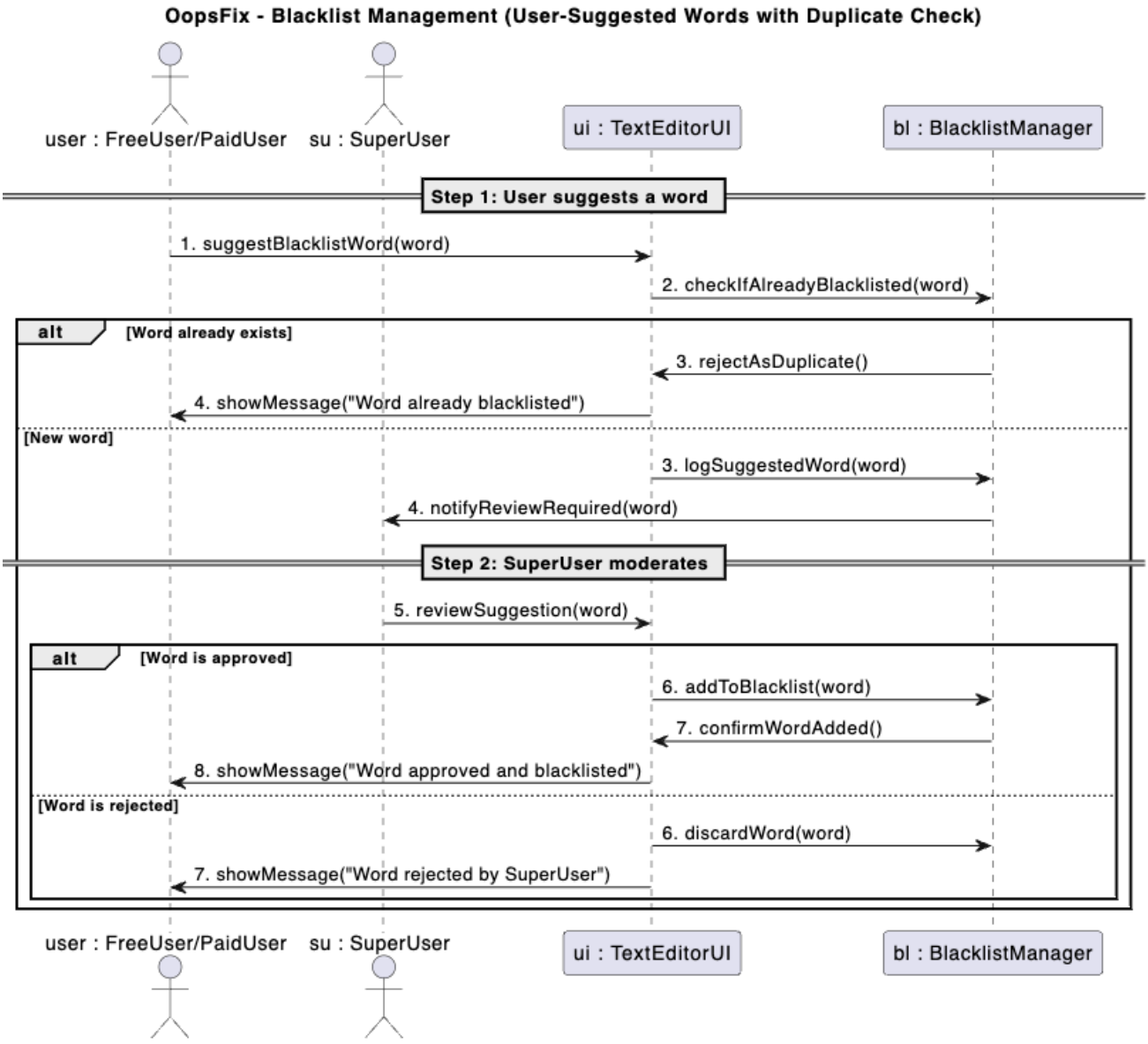


2.1.9 Blacklist Management

Users can suggest words for inclusion in the blacklist, but only super users (SU) have the authority to approve or reject these suggestions. If a word is approved, it is permanently added to the blacklist and applied to all future text submissions.

Normal: user suggests a word and SU approves it for the blacklist.

Exceptional: SU rejects the suggested word, suggested word is already in the blacklist

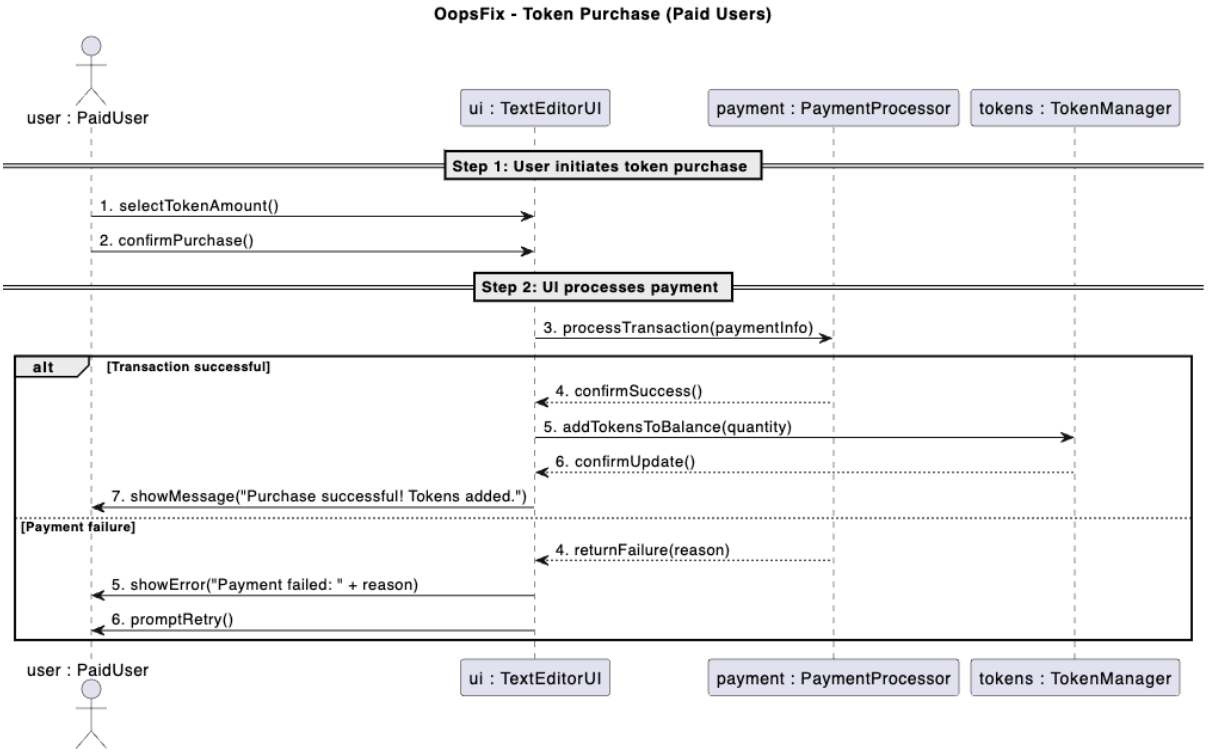


2.1.10 Token Purchase (Paid Users)

Paid users can purchase tokens to continue using premium features such as LLM-based correction, file saving, and collaboration. The user selects the number of tokens to buy, confirms the transaction, and the system processes the payment.

Normal Scenario: The transaction is successful. The system adds the selected number of tokens to the user's balance and displays a confirmation message. The user can immediately use the newly added tokens for editing or collaboration features.

Exceptional Scenario: The transaction fails due to reasons such as insufficient funds, an expired payment method, or a processing error. In this case, the system displays an error message and prompts the user to update their payment information or try the purchase again.

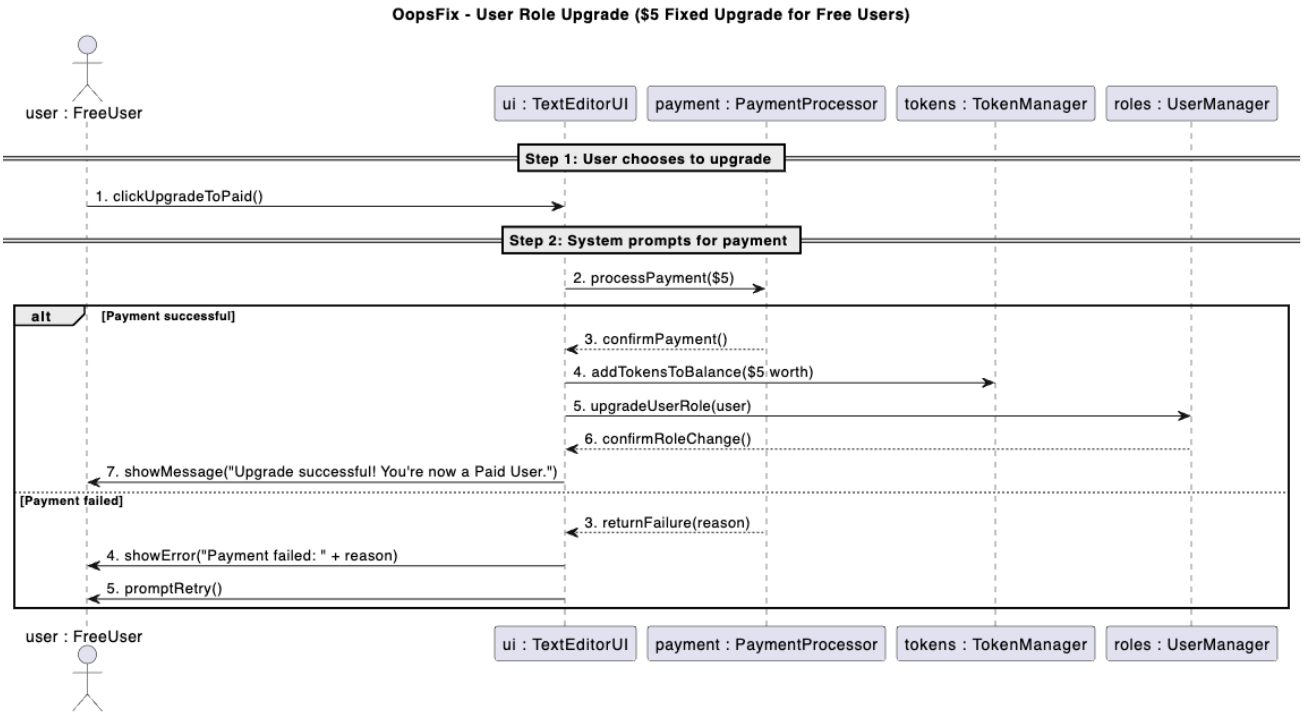


2.1.11 User Upgrade

Free users can choose to upgrade to paid users, granting them access to premium features. If they choose to upgrade, the system will prompt them to make a $5 payment. Once the payment is completed, $5 of tokens is deposited into their account and they are given access to paid user features.

Normal: payment is successful, user is upgraded

Exception: payment failure (insufficient funds, expired card, etc), system displays error message and prompts user to try again

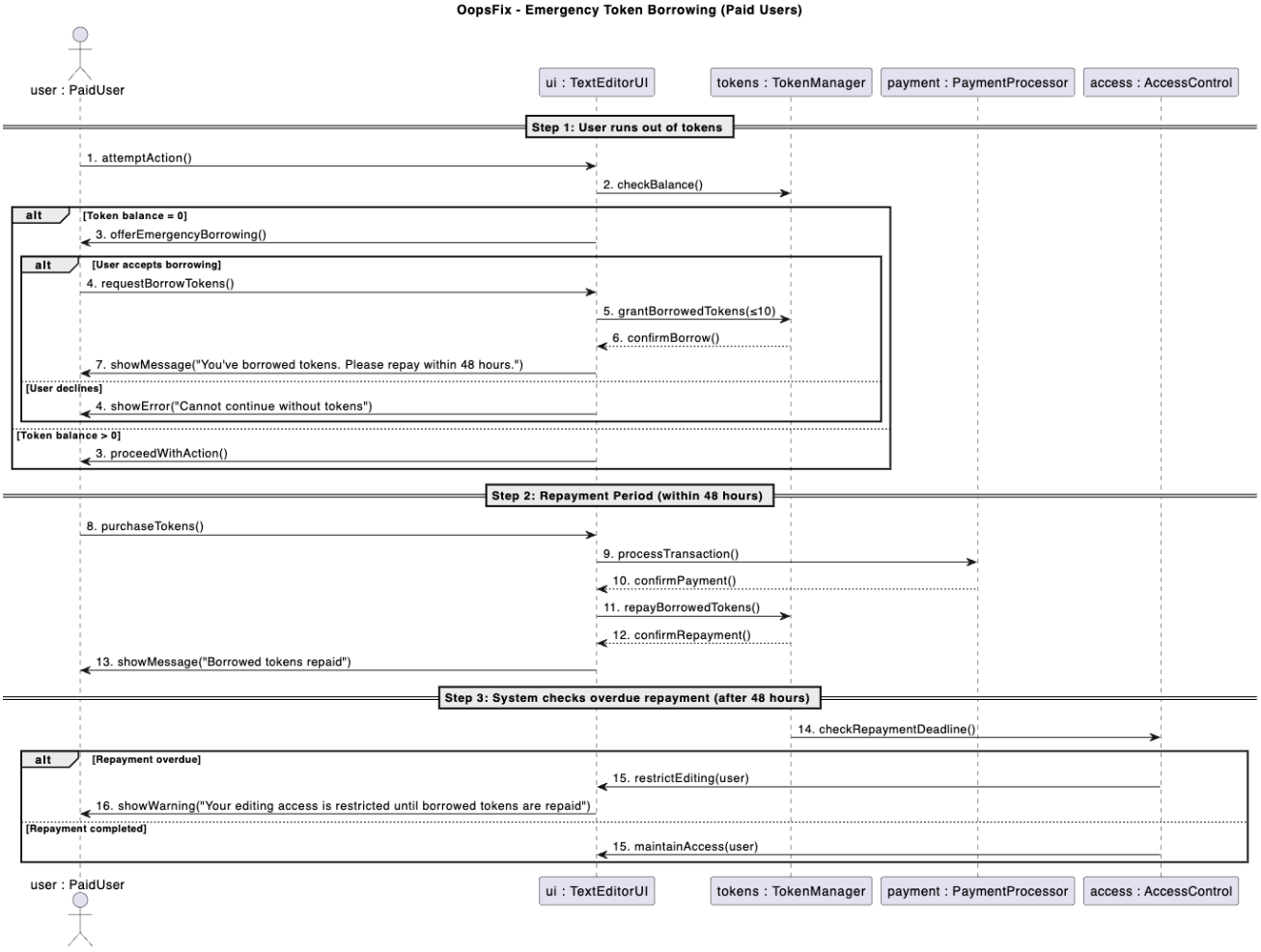


2.1.12 Emergency Token Borrowing System (Paid Users)

For users who run out of tokens mid-session, OopsFix provides an Emergency Token Borrowing option, allowing paid users to borrow up to 10 tokens at a time. Borrowed tokens must be repaid within 48 hours by purchasing new tokens.

Normal: users repay borrowed tokens on time.

Exceptional: user fails to repay on time, resulting in restricted editing access.

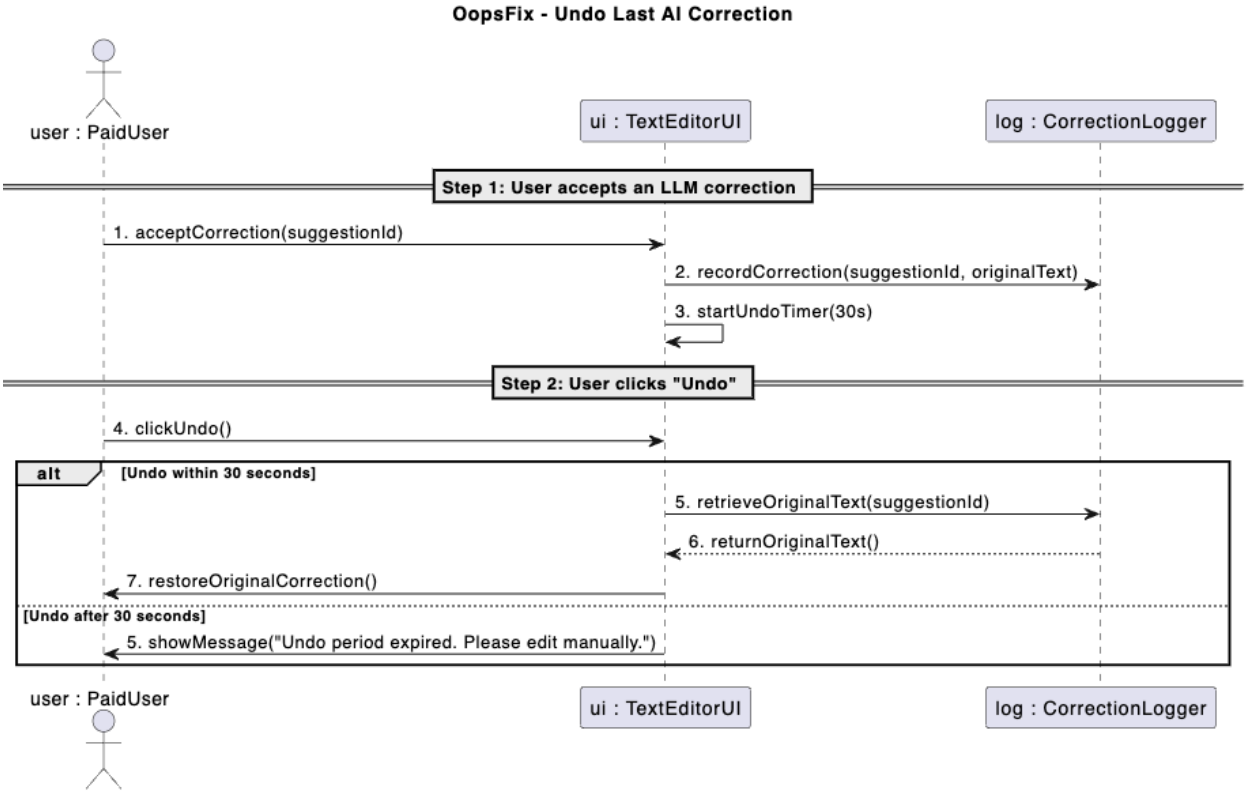


2.1.13 Undo Last AI Correction (Paid Users)

Users sometimes accidentally accept an LLM correction they didn't intend to. To fix this, OopsFix includes an undo button, allowing users to revert the last correction within 30 seconds of applying it.

Normal: Users undo mistaken changes quickly.

Exceptional: The exceptional case occurs when the undo period expires, requiring users to manually re-edit the correction.

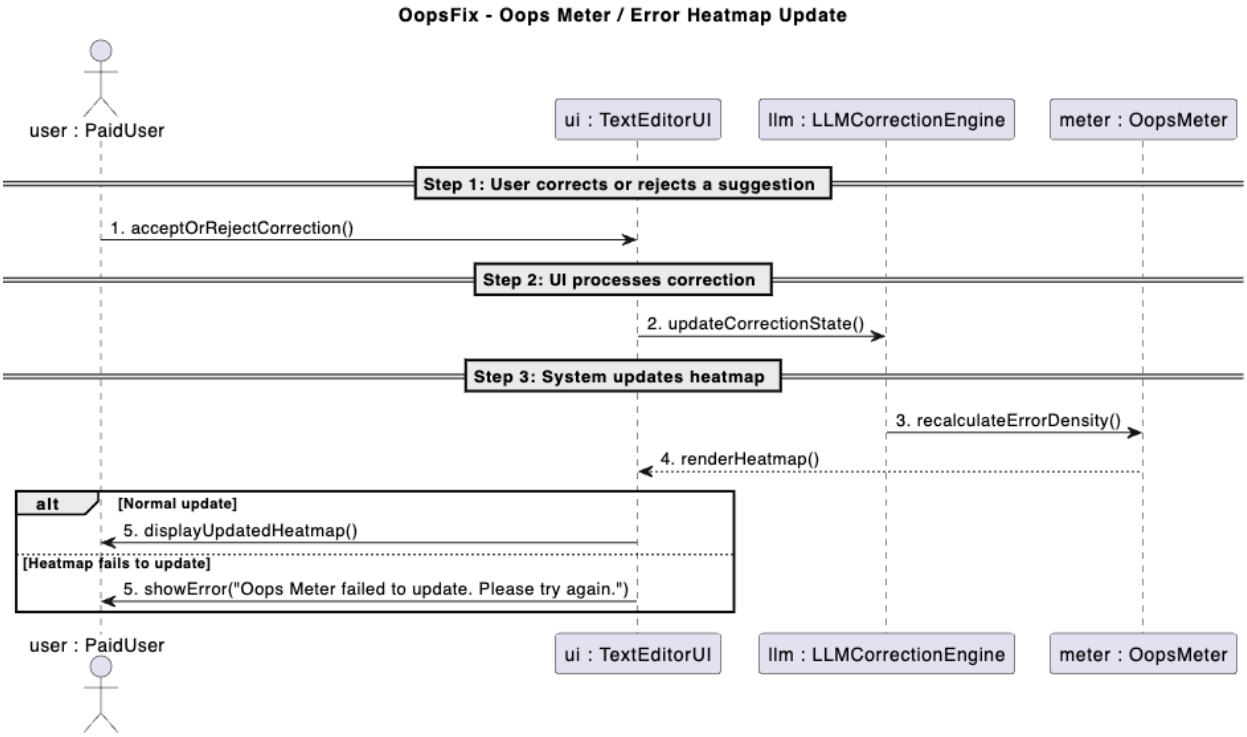


2.1.14 Update Error Heatmap

The Oops Meter provides a live visual representation of the error density in the user's text in the form of a heatmap. As corrections are made (or suggestions denied), the heatmap is automatically adjusted to reflect the new error density. This allows the user to analyze the strength of their writing in a fun and easy-to-see way.

Normal: user makes a correction , heatmap updated accordingly

Exceptional: Text is corrected but heatmap doesn't update

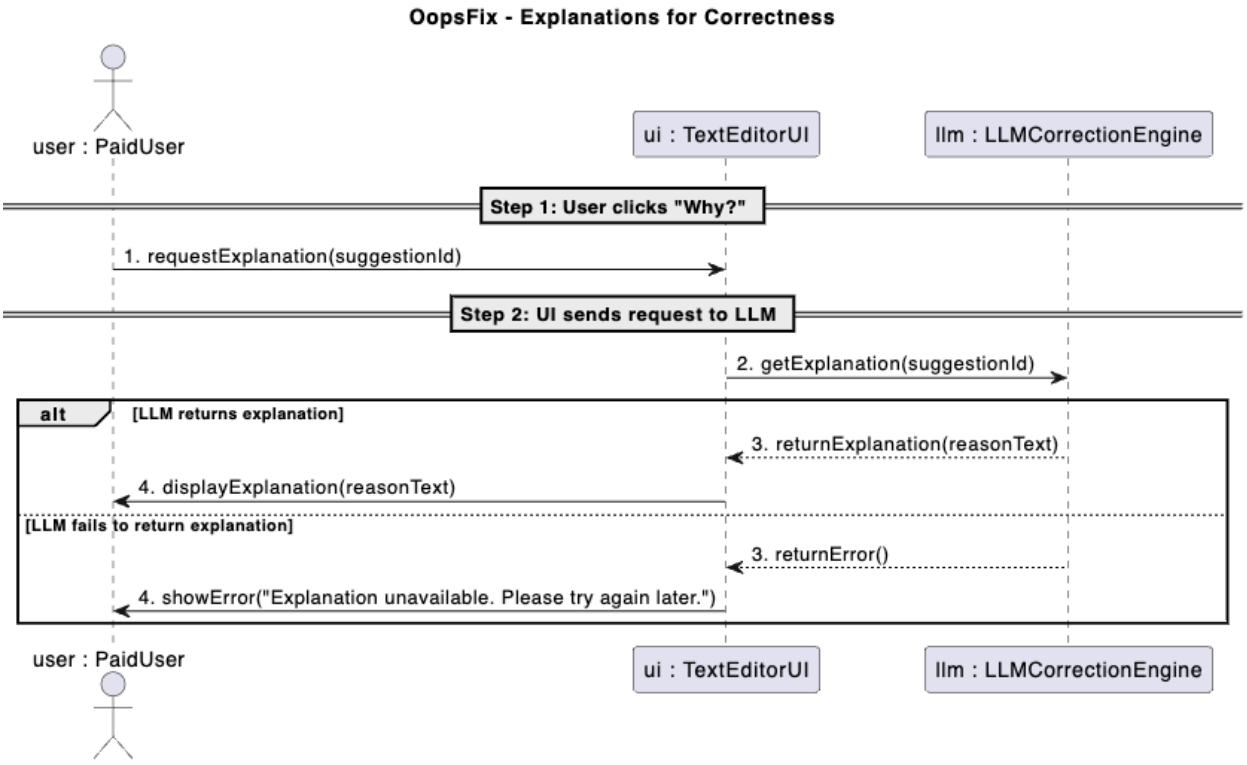


2.1.15 Explanations for Correctness (Paid Users)

For each suggestion, the user can request to see the reason for the suggestion. This will prompt the system to reveal the LLM model's logic for the recommended correction. This feature is closely linked with the UI in that each highlighted suggestion will have a corresponding reason which can be displayed on request.

Normal: user clicks on button, LLM returns explanation

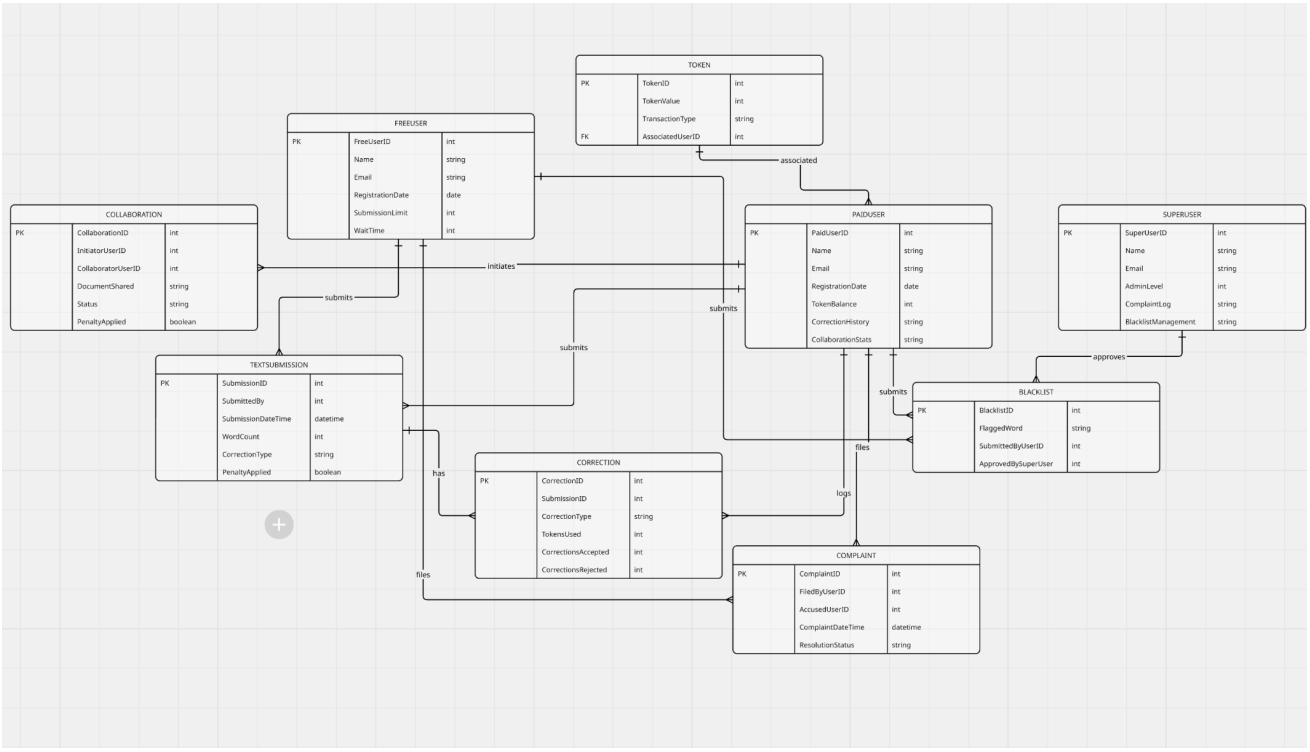
Exceptional: LLM fails to return explanation (i.e. model timeout)

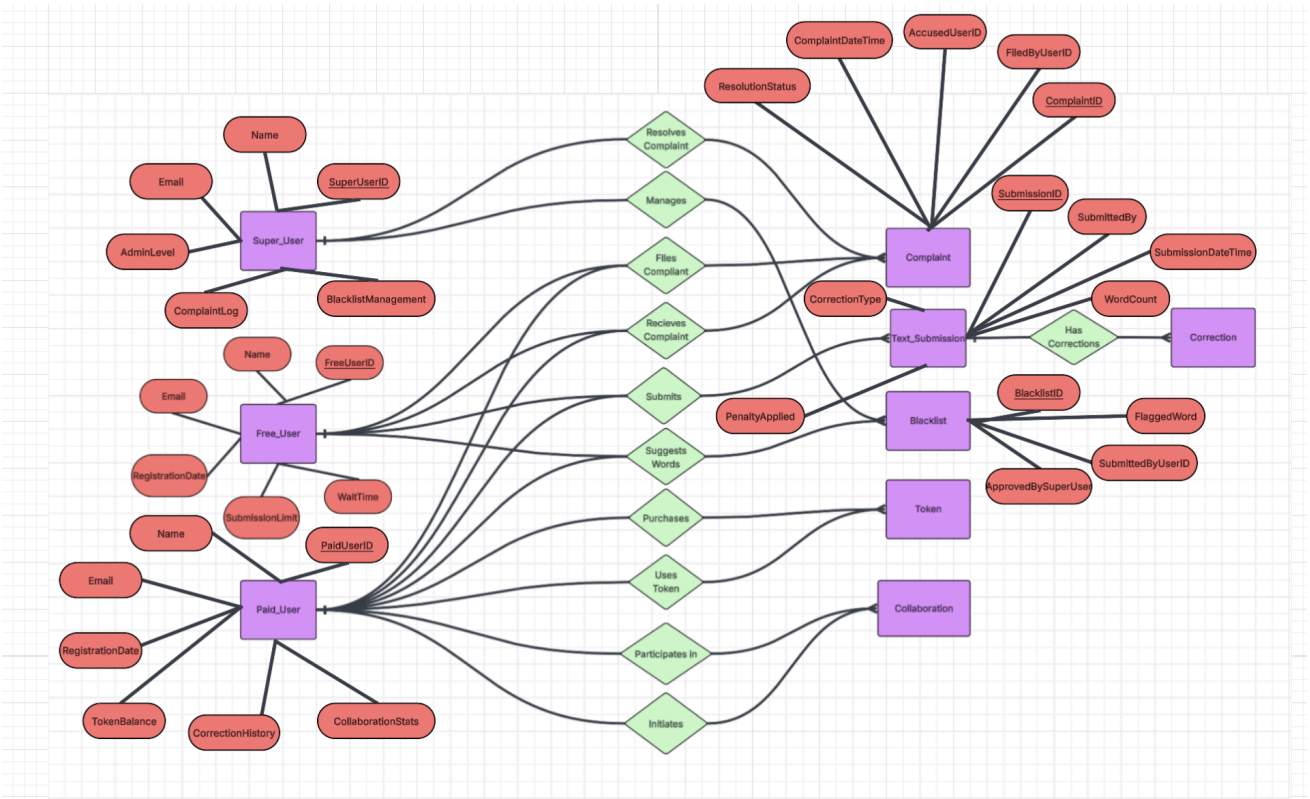


3. E/R Diagram for Entire System

This Entity-Relationship (ER) diagram represents the core database structure for the OopsFix AI-powered text editing platform. It defines relationships among various user roles-FreeUser, PaidUser, and SuperUser-and their interactions with essential system components like TextSubmission, Correction, Token, and Blacklist. Free and paid users can submit text, which links to corrections via the SubmissionID. Paid users are associated with a token balance that is tracked through the Token table. The Correction table logs the number of corrections used, accepted, or rejected per submission. Super users oversee blacklist approvals and complaint resolution, ensuring moderation and compliance with platform rules.

Collaborative editing is modeled through the Collaboration entity, connecting initiating and invited users while tracking the document's status and any penalties incurred. The Complaint entity allows users to report others, storing details like the accused user, the complaint date, and the resolution outcome. Blacklisted words are managed by super users and submitted by regular users via the Blacklist table. Overall, this ER diagram demonstrates a well-organized schema that supports core features of OopsFix, including submission tracking, collaborative workflows, token-based access, content moderation, and user role differentiation.


TOKEN
PK
TokenID int
TokenValue int
TransactionType string
FK
AssociatedUserID int
FREEUSER
PK
FreeUserID int
Name string
Email string
RegistrationDate date
SubmissionLimit int
WaitTime int
COLLABORATION
PK
CollaborationID int
InitiatorUserID int
CollaboratorUserID int
DocumentShared string
Status string
PenaltyApplied boolean
PAIDUSER
PK
PaidUserID int
Name string
Email string
RegistrationDate date
TokenBalance int
CorrectionHistory string
CollaborationStats string
SUPERUSER
PK
SuperUserID int
Name string
Email string
AdminLevel int
ComplaintLog string
BlacklistManagement string
TEXTSUBMISSION
PK
SubmissionID int
SubmittedBy int
SubmissionDateTime datetime
WordCount int
CorrectionType string
PenaltyApplied boolean
BLACKLIST
PK
BlacklistID int
FlaggedWord string
SubmittedByUserID int
ApprovedBySuperUser int
CORRECTION
PK
CorrectionID int
SubmissionID int
CorrectionType string
TokensUsed int
CorrectionsAccepted int
CorrectionsRejected int
COMPLAINT
PK
ComplaintID int
FiledByUserID int
AccusedUserID int
ComplaintDateTime datetime
ResolutionStatus string
associated
initiates
submits
submits
submits
submits
approves
files
has
logs
files


Name
Email
SuperUserID
AdminLevel
Super_User
ComplaintLog
BlacklistManagement
Name
FreeUserID
Email
Free_User
RegistrationDate
WaitTime
SubmissionLimit
Name
PaidUserID
Email
Paid_User
RegistrationDate
TokenBalance
CorrectionHistory
CollaborationStats
Resolves Complaint
Manages
Files Compliant
Recieves Complaint
Submits
Suggests Words
Purchases
Uses Token
Participates in
Initiates
ComplaintDateTime
AccusedUserID
FiledByUserID
ResolutionStatus
ComplaintID
Complaint
SubmissionID
SubmittedBy
SubmissionDateTime
CorrectionType
Text_Submission
WordCount
Has Corrections
Correction
PenaltyApplied
Blacklist
BlacklistID
FlaggedWord
SubmittedByUserID
ApprovedBySuperUser
Token
Collaboration

4. PSEUDO CODE

In [ ]:
```
USER MANAGEMENT MODULE:

            FUNCTION register_user(username, password, role)
                IF role == "free"
                    CREATE FreeUser(username, password)
                ELSE IF role == "paid"
                    IF payment_verified()
                        CREATE PaidUser(username, password)
                    ELSE
                        RETURN "Payment failed"
                ELSE IF role == "super"
                    REQUIRE super_user_credentials()
                    CREATE SuperUser(username, password)
            END FUNCTION
```

In [ ]:
```
TEXT- SUBMISSION MODULE:

                    FUNCTION submit_text(user, text_input)
                        word_count = COUNT_WORDS(text_input)

                        IF user.role == "free" AND word_count > 20
                            RETURN "Free user limit exceeded. Upgrade to submit more."

                        IF user.role == "paid"
                            IF user.tokens < word_count
                                penalty = user.tokens / 2
                                user.tokens -= penalty
                                RETURN "Insufficient tokens. Penalty applied: " + penalty
                            user.tokens -= word_count

                        flagged_words = CHECK_BLACKLIST(text_input)
                        IF flagged_words NOT EMPTY
                            text_input = REPLACE_WITH_ASTERISKS(text_input, flagged_words)
                            deduction = CALCULATE_BLACKLIST_COST(flagged_words)
                            user.tokens -= deduction

                        STORE_SUBMISSION(user, text_input)
                        RETURN "Text submitted successfully."
                    END FUNCTION
```

In [ ]:
```
LLM - CORRECTION MODE:

                FUNCTION llm_correction(user, text_input)
                    suggestions = AI_GENERATE_CORRECTIONS(text_input)

                    FOR suggestion IN suggestions
                        user_decision = DISPLAY_AND_GET_DECISION(suggestion)
                        IF user_decision == "accept"
                            user.tokens -= 1
                        ELSE
                            reason = GET_REJECTION_REASON()
                            IF IS_REASON_VALID(reason)
                                user.tokens -= 1
                            ELSE
                                user.tokens -= 5
                                ISSUE_WARNING(user, "Invalid rejection")

                    RETURN "LLM correction complete."
                END FUNCTION
```

In [ ]:
```
SAVING - TEXT MODULE

                FUNCTION save_text(user, edited_text)
                    IF user.tokens >= 5
                        user.tokens -= 5
                        SAVE_DOCUMENT(user, edited_text)
                        RETURN "Document saved successfully."
                    ELSE
                        RETURN "Not enough tokens to save the document."
                END FUNCTION
```

In [ ]:
```
COLLABORATION MODULE:

                FUNCTION invite_collaborator(inviter, invitee)
                    SEND_INVITE(invitee)

                    IF invitee.response == "reject"
                        inviter.tokens -= 3
                        RETURN "Invitation rejected. Penalty applied."
                    ELSE
                        GRANT_SHARED_ACCESS(inviter, invitee)
                        RETURN "Collaboration started."
                END FUNCTION
```

In [ ]:
```
REWARDS & BONUSES

                FUNCTION check_bonus(user, text_input)
                    IF LENGTH(text_input) >= 10 AND NO_ERRORS_FOUND(text_input)
                        user.tokens += 3
                        RETURN "Bonus awarded!"
                    RETURN "No bonus"
                END FUNCTION
```

In [ ]:
```
ERROR LOGS AND LOGISTICS:

                FUNCTION update_correction_log(user, correction_type, word_count)
                    LOG_ENTRY = {
                        "user": user.id,
                        "type": correction_type,
                        "count": word_count,
                        "timestamp": CURRENT_TIME()
                    }
                    ADD_TO_USER_LOG(user, LOG_ENTRY)
                END FUNCTION
```

In [ ]:
```
WARNINGS & COMPLAINTS:

                FUNCTION issue_warning(user, reason)
                    user.warnings += 1
                    ADD_TO_WARNING_LOG(user, reason)
                    NOTIFY_USER(user, reason)
                END FUNCTION

                FUNCTION resolve_complaint(complainant, accused)
                    decision = REVIEW_CASE(complainant, accused)
                    IF decision == "complainant at fault"
                        issue_warning(complainant, "False complaint")
                    ELSE
                        issue_warning(accused, "Valid complaint")
                END FUNCTION
```

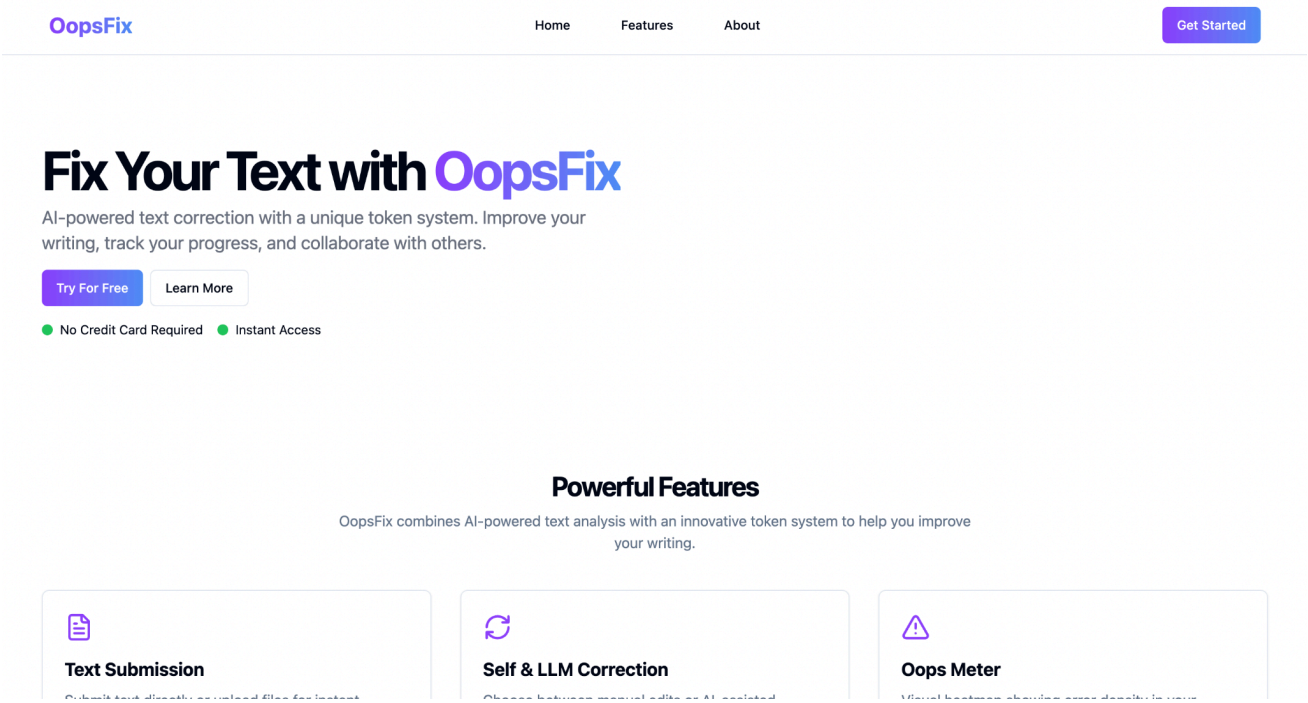
In [ ]:
```
BLACKLIST MANAGEMENT:

                FUNCTION manage_blacklist(su, word, action)
                    IF action == "add"
                        ADD_TO_BLACKLIST(word)
                    ELSE IF action == "remove"
                        REMOVE_FROM_BLACKLIST(word)
                END FUNCTION
```

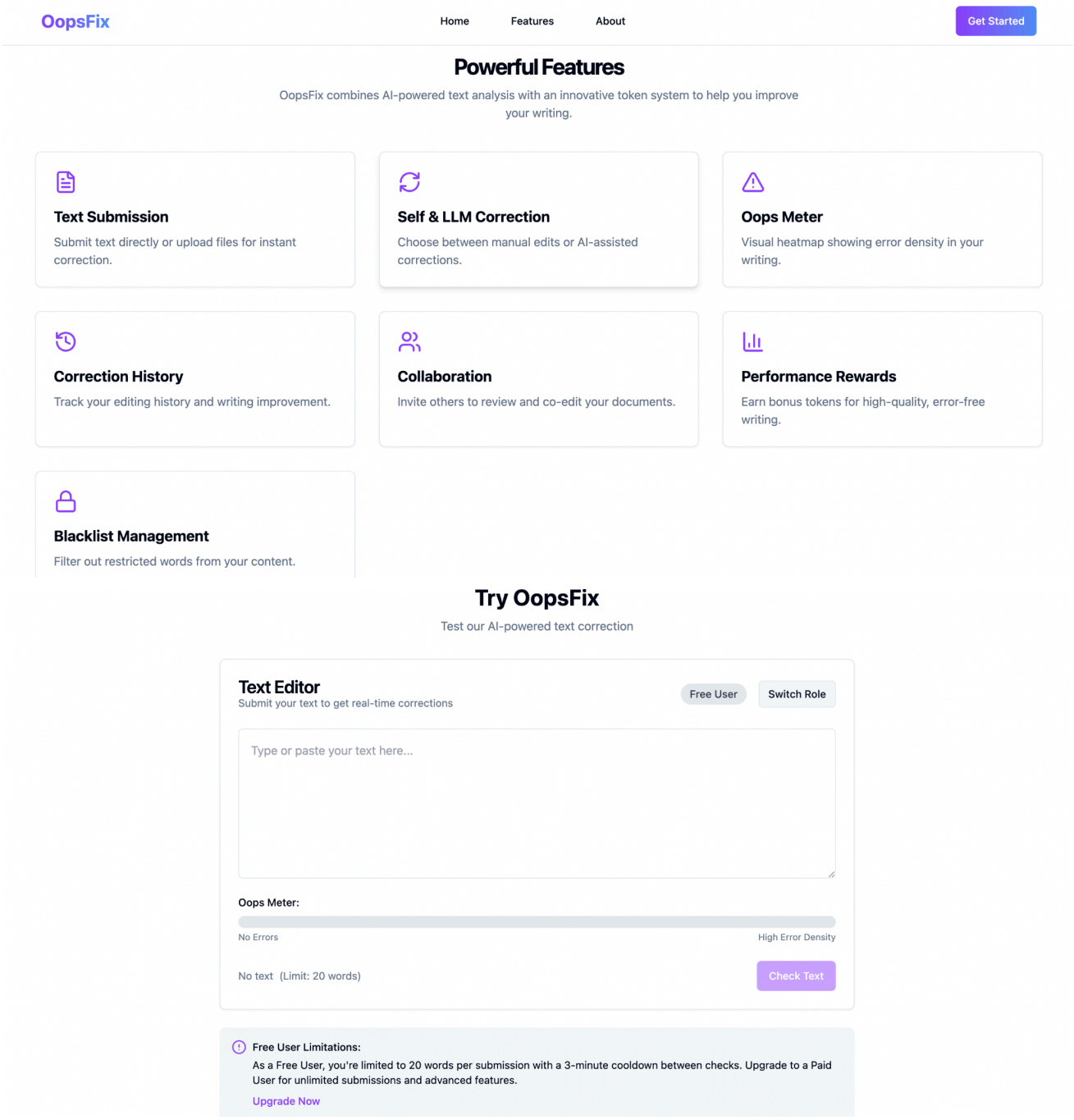
In [ ]:
```
TOEKN PURCHASES:

                FUNCTION purchase_tokens(user, amount)
                    price = CALCULATE_PRICE(amount)
                    DISPLAY_PAYMENT_FORM(price)

                    IF PROCESS_PAYMENT(user, price) == SUCCESS
                        user.tokens += amount
                        LOG_TRANSACTION(user, "purchase", amount)
                        RETURN "Tokens purchased successfully"
                    ELSE
                        RETURN "Payment failed. Try again."
                END FUNCTION
```

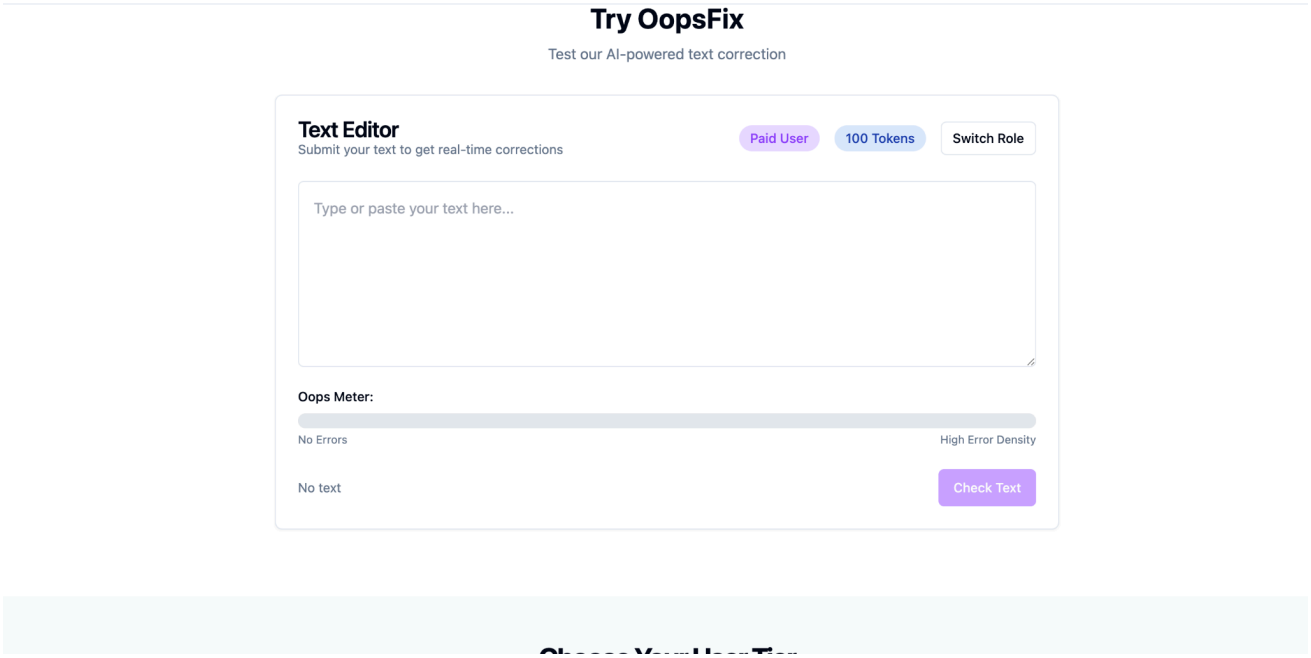
In [ ]:
```
EMERGANCY TOKEN BORROWING:

                FUNCTION borrow_tokens(user)
                    IF user.borrowed_tokens < 10
                        user.tokens += 10
                        user.borrowed_tokens += 10
                        SET_REPAYMENT_TIMER(user, 48_hours)
                        RETURN "Borrowed 10 tokens"
                    RETURN "Borrow limit reached"
                END FUNCTION
```

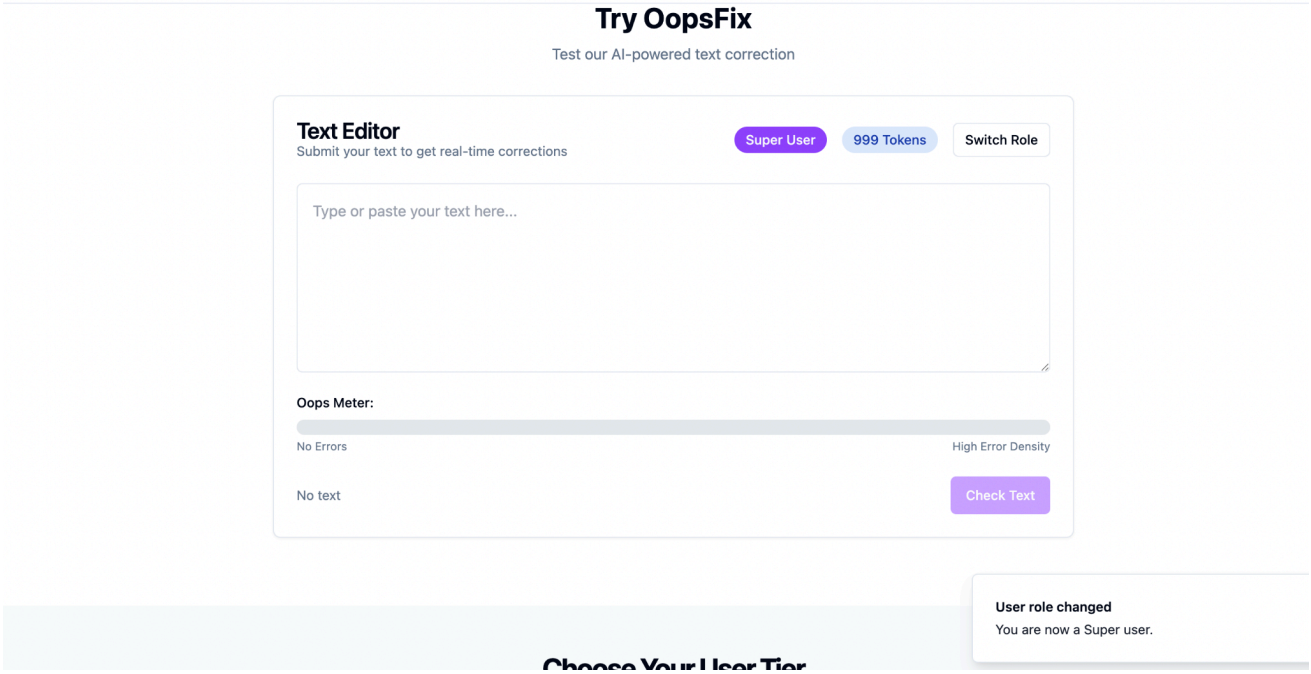
In [ ]:
```
UNDO~FEATURES:
                FUNCTION undo_last_correction(user)
                    IF TIME_SINCE_LAST_CORRECTION < 30_seconds
                        RESTORE_PREVIOUS_STATE(user)
                        RETURN "Undo successful"
                    RETURN "Undo period expired"
                END FUNCTION
```

5. GUI System Screenshots:




OopsFix
Home
Features
About
Get Started
Powerful Features
OopsFix combines AI-powered text analysis with an innovative token system to help you improve your writing.
Text Submission
Submit text directly or upload files for instant correction.
Self & LLM Correction
Choose between manual edits or AI-assisted corrections.
Oops Meter
Visual heatmap showing error density in your writing.
Correction History
Track your editing history and writing improvement.
Collaboration
Invite others to review and co-edit your documents.
Performance Rewards
Earn bonus tokens for high-quality, error-free writing.
Blacklist Management
Filter out restricted words from your content.
Try OopsFix
Test our AI-powered text correction
Text Editor
Submit your text to get real-time corrections
Free User
Switch Role
Type or paste your text here...
Oops Meter:
No Errors
High Error Density
No text (Limit: 20 words)
Check Text
Free User Limitations:
As a Free User, you're limited to 20 words per submission with a 3-minute cooldown between checks. Upgrade to a Paid User for unlimited submissions and advanced features.
Upgrade Now


Try OopsFix
Test our AI-powered text correction
Text Editor
Submit your text to get real-time corrections
Paid User
100 Tokens
Switch Role
Type or paste your text here...
Oops Meter:
No Errors
High Error Density
No text
Check Text


Try OopsFix
Test our AI-powered text correction
Text Editor
Submit your text to get real-time corrections
Super User
999 Tokens
Switch Role
Type or paste your text here...
Oops Meter:
No Errors
High Error Density
No text
Check Text
User role changed
You are now a Super user.

6. Group memo

Project Name: OopsFix – AI-Powered Cooperative Text Editor

Date: April 13, 2025

To: Prof. Jie Wei

From: Team G

Subject: Group Memo – Design Report Submission

Throughout the development of the OopsFix design report, our team maintained active collaboration through regular group meetings and task coordination. The team worked together to break down the project requirements and assign responsibilities based on individual strengths and availability. We held weekly planning sessions—both in person and online—where we discussed project goals, clarified requirements, and set internal deadlines for each phase of the report.

Each team member contributed to different sections of the project: some focused on technical modeling (ER diagram and collaboration diagrams), while others worked on the system logic and GUI layout. We collectively reviewed all materials to ensure consistency and quality across the document. One challenge we encountered was coordinating our work across varying schedules, especially when developing the pseudo-code and system diagrams. We resolved this by using shared documents and asynchronous check-ins. Overall, the project was a strong example of balanced teamwork, and we are proud of the system design and documentation we've delivered together.

Please let us know if further clarification is needed, and we look forward to your feedback.

Sincerely,
Team G

7. Github Repo

https://github.com/ysujaana2004/OppsFix