CraftVerify

User Management Low-Level Design

Team Natural Selection

GitHub Link: <https://github.com/Natural-Selection491/CraftVerify>

Members:

Kameron Ferrer

Aiden Hocks

Jacob Phillips

Parth Thanki

An Nguyen

Khuong Nguyen

12/03/2023

| **Version** | **Who Edited** | **Date** | **What Is Edited/Changed?** |
| --- | --- | --- | --- |
| **1.0** | **An** | **12/02/2023** | * **Document Creation** |
| **1.1** | **Parth** | **12/03/2023** | * **Added Draft For Profile Update Low-Level Design** |
|  |  |  |  |

User Management Low-Level Design

## Low-Level Design Document Draft for Registration: Account Creation

## Low-Level Design Document Draft for Profile Modification

## Low-Level Design Document Draft for Profile Modification

**1. Introduction**

This section of the document describes the purpose, scope, and functionality of the Profile Modification component in the User Management System. It provides an overview of the feature's capabilities, such as updating user profiles with new information.

1.1 Purpose

The purpose of this document is to provide a detailed technical specification for the Profile Modification feature of our User Management System.

1.2 Scope

The Profile Modification component will allow users to update their personal information, including name, date of birth, bio, profile photo, and additional information.

**2. UserProfile Class Design**

This section outlines the structure of the UserProfile class, which is central to the Profile Modification feature.

2.1 Attributes

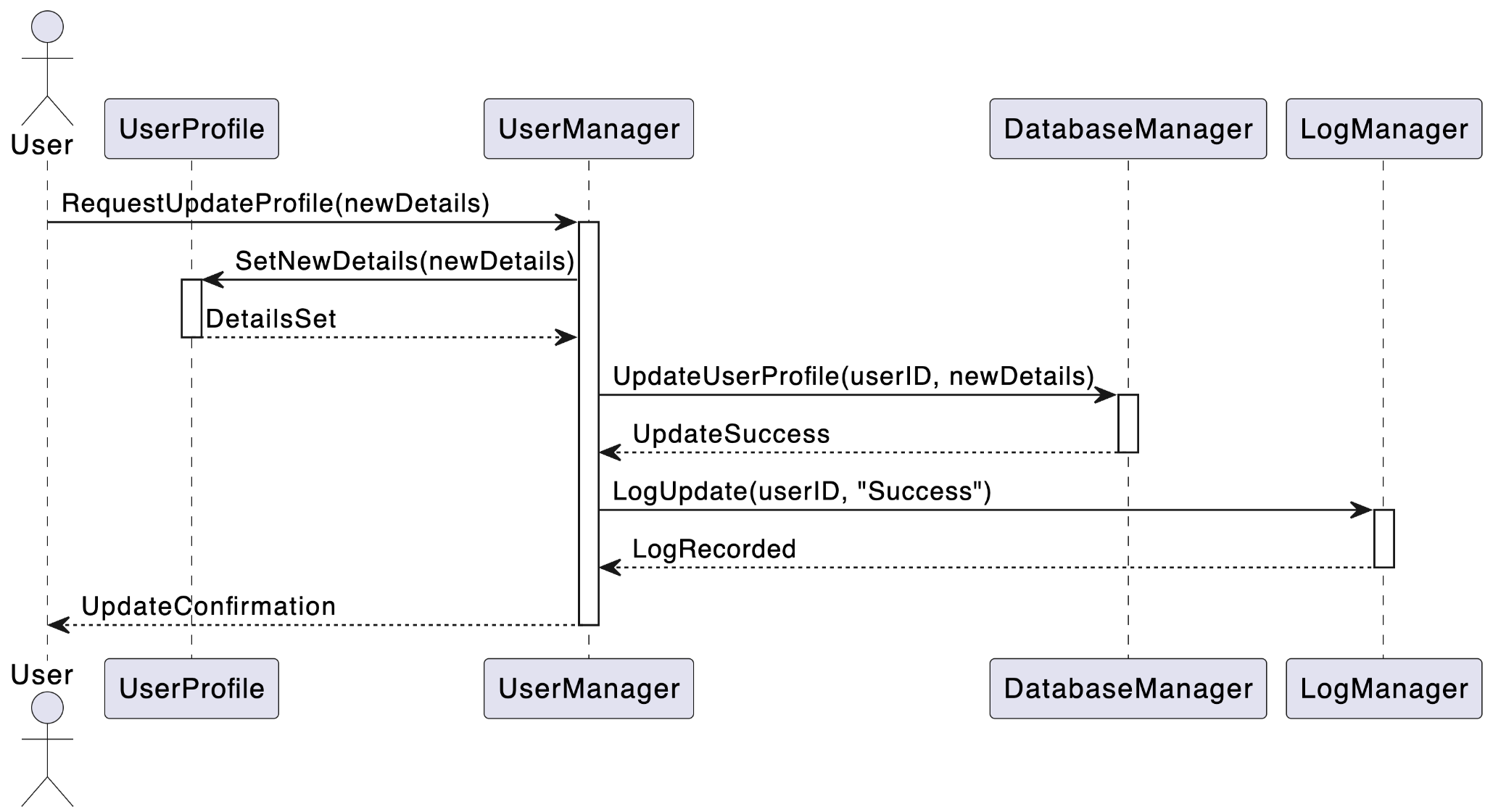
* name: String
* dateOfBirth: DateTime
* bio: String
* photo: ImageFile
* additionalInformation: Map<String, String>

2.2 Methods

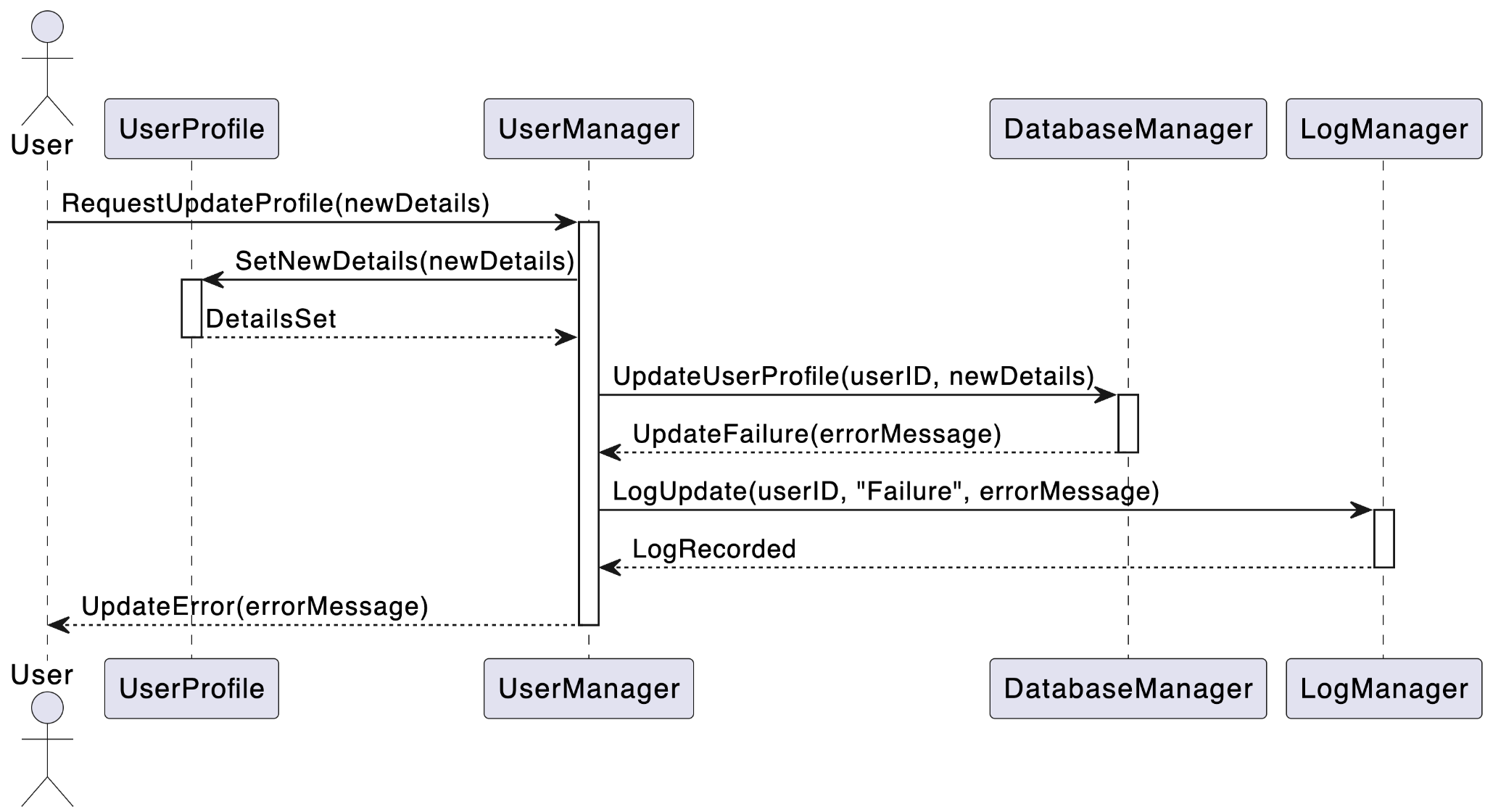
* updateName(String newName): Boolean
* updateDOB(DateTime newDOB): Boolean
* updateBio(String newBio): Boolean
* updatePhoto(ImageFile newPhoto): Boolean
* updateAdditionalInfo(String key, String value): Boolean

**3. Sequence Diagrams for Profile Modification**

3.1 Success Scenario



3.2 Failure Scenario



**4. Database Design**

This section details the database schema required to support profile updates, including modifications to existing tables and the introduction of new tables.

4.1 Profile Table

The Profile table stores user profile information. The structure of the table includes:

* UserID (Primary Key): Integer
* Name: String
* DateOfBirth: Date
* Bio: Text
* Photo: Blob
* AdditionalInformation: Text (JSON or key-value pairs)

4.2 Logging Table

The Logging table is used to track profile updates. Its structure includes:

* LogID (Primary Key): Integer
* UserID: Integer (Foreign Key)
* ActionType: String (e.g., 'UpdateProfile')
* Timestamp: DateTime
* Status: String (e.g., 'Success', 'Failure')
* Details: Text (Description of the action and any errors)

**5. Error Handling and Logging**

This section outlines the strategies for handling errors and logging during the profile update process.

5.1 Error Handling

Error handling methods include:

* Validation Errors: Before updating, validate data formats (e.g., date formats, string lengths). Reject invalid updates with descriptive error messages.
* Database Errors: Catch exceptions during database operations. Log these errors with details for troubleshooting.
* Access Control Errors: Ensure the user has permission to update the profile. Unauthorized attempts should be logged and rejected.

5.2 Logging

Approach for logging:

* Successful Updates: Log all successful profile updates with timestamps and user IDs.
* Failed Attempts: Log failed update attempts, including error details and user IDs. Use this data to identify common issues or potential security concerns.

**6. Test Cases**

This section lists the test cases for the Profile Modification feature.

6.1 Unit Tests

Unit tests for the UserProfile class:

* Test Update Name: Verify that the updateName() method correctly updates the name.
* Test Date of Birth Update: Ensure that updateDOB() handles various date formats and rejects invalid dates.
* Test Bio Update: Test updateBio() for handling different lengths and formats of bios.
* Test Additional Information Update: Validate that updateAdditionalInfo() correctly updates key-value pairs.

6.2 Integration Tests

Integration tests cover the complete profile update process:

* Profile Update Flow: Test the entire flow from the user request to the database update and logging.
* Error Handling: Simulate various error conditions (e.g., database unavailability, invalid input) and verify that they are handled appropriately.
* Security and Permissions: Verify that only authorized users can update profiles and that unauthorized attempts are logged.