CS 306 - Term Project Phase 3

Group 53

28th December 2025

Digital Banking Application

Introduction

In this project, we developed a Digital Banking web application that integrates a MySQL database with a PHP-based backend. The main goal is to demonstrate how the database design, stored procedures, and triggers implemented in Phase 1 and Phase 2 can be actively used through a real application.

The system consists of two main domains: User portal and admin portal. The user portal allows customers to interact with the digital banking database. Users can perform actions such as adding a new card, paying bills and creating tickets. Each operation directly interacts with the underlying MySQL database and enforces application logic through predefined stored procedures and triggers.

The second domain is the administrative environment. Administrators manage support tickets and monitor user requests through a dedicated interface. In addition to the relational ticket mechanism implemented in earlier phases, the project also introduces a second ticket system that operates on MongoDB. The MySQL based Create Ticket procedure models formal customer service cases and demonstrates transactional control and integrity constraints at the database level. The MongoDB support system, on the other hand, captures informal user questions and help requests. It is designed to be lightweight, flexible and suitable for frequent interaction between users and administrators. By maintaining both systems side by side, the project highlights the different strengths of relational and document-based models and shows how they may coexist within the same application.

User Portal

1. Triggers

1.1 Trigger 1: Decrease Account Balance After Bill Payment

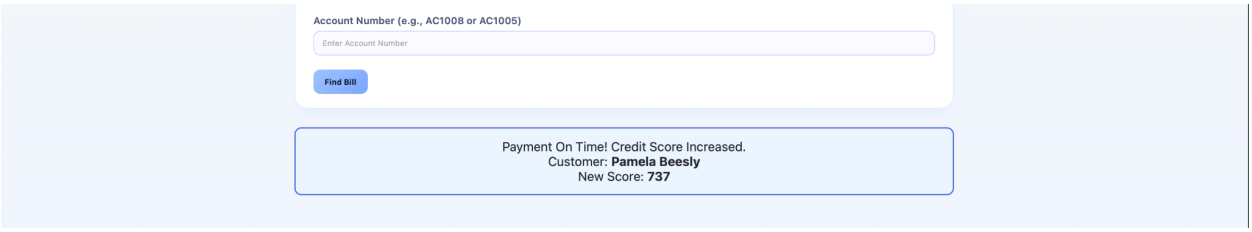
When a bill is paid, the corresponding account balance should automatically decrease. This prevents inconsistencies caused by forgetting to manually update the balance. Whenever a new entry is inserted into the Paid_Bill table (AFTER INSERT ON Paid_Bill), the trigger subtracts the bill amount from the balance of the related account.

SQL SCRIPT:

```
CREATE TRIGGER Decrease_Balance_After_Bill_Paid
AFTER INSERT ON Paid_Bill
FOR EACH ROW
BEGIN
   UPDATE Account
   SET balance = balance - NEW.bill_amount
   WHERE account_number = NEW.account_number;
END;
```

Before:

After:



Before:

	account_number	account_type	balance	created_at
Düzenle Kopyala Sil	AC1001	Personal	10000	2010-05-01
Düzenle Kopyala Sil	AC1002	Joint	25000	2012-07-15
Düzenle Kopyala Sil	AC1003	Personal	15000	2014-09-20
Düzenle Kopyala Sil	AC1004	Personal	20000	2015-02-25
Düzenle Kopyala Sil	AC1005	Joint	50000	2008-07-30
Düzenle Kopyala Sil	AC1006	Personal	12000	2013-01-22
Düzenle Kopyala Sil	AC1007	Joint	30000	2011-06-10
Düzenle Kopyala Sil	AC1008	Personal	39760	2009-10-14
Düzenle Kopyala Sil	AC1009	Personal	7000	2010-12-01
Düzenle Kopyala Sil	AC1010	Joint	9000	2007-08-25

After:

	account_number	account_type	balance	created_at
Düzenle Kopyala Sil	AC1001	Personal	10000	2010-05-01
Düzenle Kopyala Sil	AC1002	Joint	25000	2012-07-15
Düzenle Kopyala Sil	AC1003	Personal	15000	2014-09-20
Düzenle Kopyala Sil	AC1004	Personal	19950	2015-02-25
Düzenle Kopyala Sil	AC1005	Joint	50000	2008-07-30
Düzenle Kopyala Sil	AC1006	Personal	12000	2013-01-22
Düzenle Kopyala Sil	AC1007	Joint	30000	2011-06-10
Düzenle Kopyala Sil	AC1008	Personal	39760	2009-10-14
Düzenle Kopyala Sil	AC1009	Personal	7000	2010-12-01
Düzenle Kopyala Sil	AC1010	Joint	9000	2007-08-25

1.2 Trigger 2: Update Credit Score Based on Payment Timing

To encourage timely payments and simulate realistic banking behavior, credit scores should change depending on whether bills are paid on time.

• If payment is late → decrease credit score by 1%

• If payment is on time → increase credit score by 1%

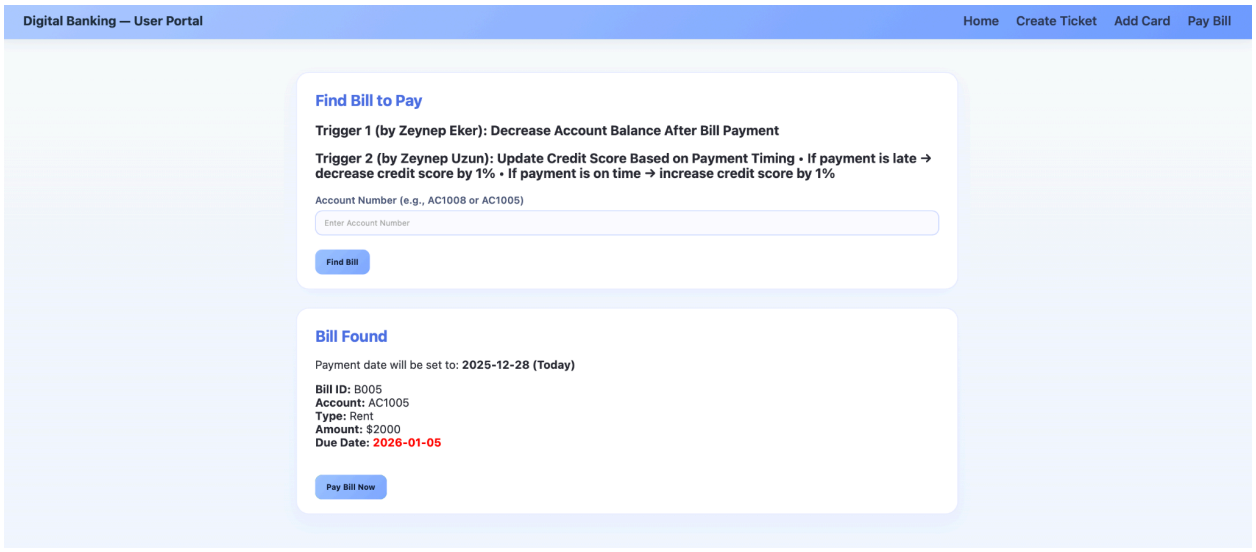
Because accounts may be jointly owned, the trigger updates the credit scores of all customers associated with that account via the Maintains table.

SQL SCRIPT:

```
DELIMITER //

CREATE TRIGGER Update_Credit_Score_After_Bill_Paid
AFTER INSERT ON Paid_Bill
FOR EACH ROW
BEGIN
  IF NEW.payment_date > NEW.bdue_date THEN
    UPDATE Customer
    SET credit_score = ROUND(credit_score * 0.99)
    WHERE customer_id IN (
      SELECT customer_id
      FROM Maintains
      WHERE account_number = NEW.account_number
    );
  ELSE
    UPDATE Customer
    SET credit_score = ROUND(credit_score * 1.01)
    WHERE customer_id IN (
      SELECT customer_id
      FROM Maintains
      WHERE account_number = NEW.account_number
    );
  END IF;
END //
DELIMITER ;
```

Before (Increase Case):

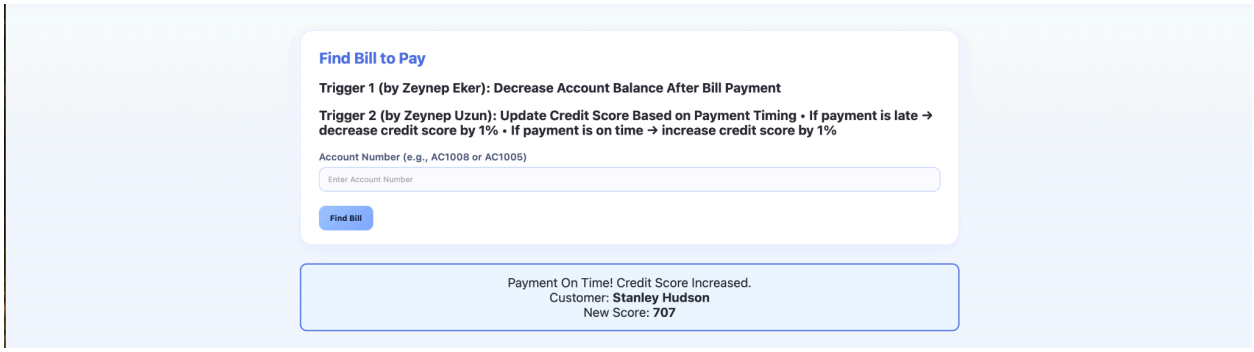
Düzenle	Kopyala	Sil	B004	2025-12-31	AC1004	2025-12-15	Water	50	2025-12-28
Düzenle	Kopyala	Sil	B005	2026-01-05	AC1005	2025-12-20	Rent	2000	*NULL*
Düzenle	Kopyala	Sil	B006	2026-01-10	AC1006	2025-12-22	Subscription	20	*NULL*
Düzenle	Kopyala	Sil	B007	2025-12-24	AC1007	2025-12-05	Electricity	100	2025-12-22



	customer_id	full_name	age	credit_score	customer_since	phone_number
Düzenle Kopyala Sil	C001	Michael Scott	45	720	2010-04-21	00000000000
Düzenle Kopyala Sil	C002	Dwight Schrute	42	760	2012-03-05	11111111111
Düzenle Kopyala Sil	C003	Jim Halpert	38	740	2014-09-17	22222222222
Düzenle Kopyala Sil	C004	Pamela Beesly	36	730	2015-02-11	33333333333
Düzenle Kopyala Sil	C005	Stanley Hudson	55	700	2008-07-30	44444444444
Düzenle Kopyala Sil	C006	Kevin Malone	40	610	2013-01-22	55555555555
Düzenle Kopyala Sil	C007	Angela Martin	41	750	2011-06-10	66666666666
Düzenle Kopyala Sil	C008	Oscar Martinez	43	760	2009-10-14	77777777777
Düzenle Kopyala Sil	C009	Toby Flenderson	47	680	2010-12-01	88888888888
Düzenle Kopyala Sil	C010	Creed Bratton	60	650	2007-08-25	99999999999

After:

	customer_id	full_name	age	credit_score	customer_since	phone_number
Düzenle Kopyala Sil	C001	Michael Scott	45	720	2010-04-21	00000000000
Düzenle Kopyala Sil	C002	Dwight Schrute	42	760	2012-03-05	11111111111
Düzenle Kopyala Sil	C003	Jim Halpert	38	740	2014-09-17	22222222222
Düzenle Kopyala Sil	C004	Pamela Beesly	36	730	2015-02-11	33333333333
Düzenle Kopyala Sil	C005	Stanley Hudson	55	707	2008-07-30	44444444444

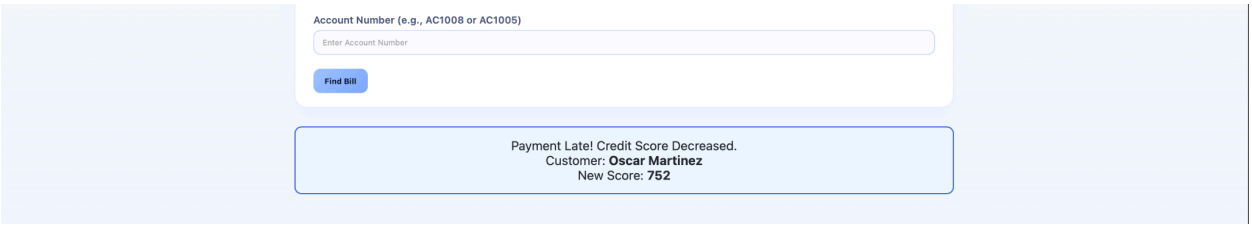


Düzenle	Kopyala	Sil	B004	2025-12-31	AC1004	2025-12-15	Water	50	2025-12-28
Düzenle	Kopyala	Sil	B005	2026-01-05	AC1005	2025-12-20	Rent	2000	2025-12-28
Düzenle	Kopyala	Sil	B006	2026-01-10	AC1006	2025-12-22	Subscription	20	*NULL*
Düzenle	Kopyala	Sil	B007	2025-12-24	AC1007	2025-12-05	Electricity	100	2025-12-22

Before (Decrease Case):

	customer_id	full_name	age	credit_score	customer_since	phone_number
Düzenle Kopyala Sil	C001	Michael Scott	45	720	2010-04-21	00000000000
Düzenle Kopyala Sil	C002	Dwight Schrute	42	760	2012-03-05	11111111111
Düzenle Kopyala Sil	C003	Jim Halpert	38	740	2014-09-17	22222222222
Düzenle Kopyala Sil	C004	Pamela Beesly	36	730	2015-02-11	33333333333
Düzenle Kopyala Sil	C005	Stanley Hudson	55	700	2008-07-30	44444444444
Düzenle Kopyala Sil	C006	Kevin Malone	40	610	2013-01-22	55555555555
Düzenle Kopyala Sil	C007	Angela Martin	41	750	2011-06-10	66666666666
Düzenle Kopyala Sil	C008	Oscar Martinez	43	760	2009-10-14	77777777777
Düzenle Kopyala Sil	C009	Toby Flenderson	47	680	2010-12-01	88888888888
Düzenle Kopyala Sil	C010	Creed Bratton	60	650	2007-08-25	99999999999

After:



Before:

	bill_id	bdue_date	account_number	bstart_date	bill_type	bill_amount	payment_date
Düzenle Kopyala Sil	B001	2025-12-20	AC1001	2025-12-01	Rent	1200	2025-12-19
Düzenle Kopyala Sil	B002	2025-12-26	AC1002	2025-12-05	Electricity	150	2025-12-26
Düzenle Kopyala Sil	B003	2025-12-28	AC1003	2025-12-10	Internet	80	2025-12-28
Düzenle Kopyala Sil	B004	2025-12-31	AC1004	2025-12-15	Water	50	NULL
Düzenle Kopyala Sil	B005	2026-01-05	AC1005	2025-12-20	Rent	2000	NULL
Düzenle Kopyala Sil	B006	2026-01-10	AC1006	2025-12-22	Subscription	20	NULL
Düzenle Kopyala Sil	B007	2025-12-24	AC1007	2025-12-05	Electricity	100	2025-12-22
Düzenle Kopyala Sil	B008	2025-12-27	AC1008	2025-12-12	Internet	60	NULL
Düzenle Kopyala Sil	B009	2026-01-02	AC1009	2025-12-18	Water	45	NULL
Düzenle Kopyala Sil	B010	2025-12-26	AC1010	2025-12-22	Rent	1500	NULL

After: (Account AC1008 is customer C008's personal account)

	bill_id	bdue_date	account_number	bstart_date	bill_type	bill_amount	payment_date
Düzenle Kopyala Sil	B001	2025-12-20	AC1001	2025-12-01	Rent	1200	2025-12-19
Düzenle Kopyala Sil	B002	2025-12-26	AC1002	2025-12-05	Electricity	150	2025-12-26
Düzenle Kopyala Sil	B003	2025-12-28	AC1003	2025-12-10	Internet	80	2025-12-28
Düzenle Kopyala Sil	B004	2025-12-31	AC1004	2025-12-15	Water	50	NULL
Düzenle Kopyala Sil	B005	2026-01-05	AC1005	2025-12-20	Rent	2000	NULL
Düzenle Kopyala Sil	B006	2026-01-10	AC1006	2025-12-22	Subscription	20	NULL
Düzenle Kopyala Sil	B007	2025-12-24	AC1007	2025-12-05	Electricity	100	2025-12-22
Düzenle Kopyala Sil	B008	2025-12-27	AC1008	2025-12-12	Internet	60	2025-12-28
Düzenle Kopyala Sil	B009	2026-01-02	AC1009	2025-12-18	Water	45	NULL
Düzenle Kopyala Sil	B010	2025-12-26	AC1010	2025-12-22	Rent	1500	NULL

←T→	customer_id	full_name	age	credit_score	customer_since	phone_number
Düzenle Kopyala Sil	C001	Michael Scott	45	720	2010-04-21	00000000000
Düzenle Kopyala Sil	C002	Dwight Schrute	42	760	2012-03-05	11111111111
Düzenle Kopyala Sil	C003	Jim Halpert	38	740	2014-09-17	22222222222
Düzenle Kopyala Sil	C004	Pamela Beesly	36	730	2015-02-11	33333333333
Düzenle Kopyala Sil	C005	Stanley Hudson	55	700	2008-07-30	44444444444
Düzenle Kopyala Sil	C006	Kevin Malone	40	610	2013-01-22	55555555555
Düzenle Kopyala Sil	C007	Angela Martin	41	750	2011-06-10	66666666666
Düzenle Kopyala Sil	C008	Oscar Martinez	43	752	2009-10-14	77777777777
Düzenle Kopyala Sil	C009	Toby Flenderson	47	680	2010-12-01	88888888888
Düzenle Kopyala Sil	C010	Creed Bratton	60	650	2007-08-25	99999999999

1.3 Test Cases and Web Interface Buttons

Each trigger is connected to interactive buttons on the web interface. These buttons simulate realistic database events and allow users to observe how the trigger behaves in different scenarios. The purpose is to verify that the trigger executes automatically and enforces the intended business logic.

Triggers are tested through two buttons on the Pay Bill page:

- **Find Bill**

  This button retrieves the bill and displays it on the screen without executing any database update. Its purpose is only to show the initial state.
- **Pay Bill**

  This button inserts a record into the Paid_Bill table. Once the record is inserted, the trigger automatically decreases the account balance. The updated balance is then shown to the user. When the payment is late, it decreases the customer(s)' credit score(s) linked to that account. However, when the payment is on time; it increases by %1.

2. Stored Procedures

2.1 Stored Procedure 1: Create Ticket

The Create Ticket feature allows customers to submit support requests through the user portal. The ticket ID is automatically generated by the system. Before creating a new ticket, the system retrieves the maximum existing ticket_id from the Customer_Ticket table and increments it by one. The ticket IDs follow the format T001, T002, T003, …, ensuring uniqueness and sequential order.

Before a ticket is created, the system verifies whether the entered customer_id exists in the Customer table. If the customer does not exist, the ticket creation process is stopped and an error message is displayed to the user. If the customer validation is successful, the **CreateTicket** stored procedure is executed.

After successful ticket creation, the system displays a confirmation message along with the generated ticket details on the same page. If an error occurs (such as an invalid customer ID), an appropriate error message is shown to the user.

SQL SCRIPT:

```
DELIMITER //
CREATE PROCEDURE CreateTicket (
   IN p_ticket_id CHAR(20),
   IN p_customer_id CHAR(20),
   IN p_topic CHAR(50),
   IN p_message VARCHAR(250),
   IN p_type CHAR(20),
   IN p_extra_info VARCHAR(50)
)
BEGIN
   INSERT INTO Customer_Ticket (ticket_id, customer_id, topic, message)
   VALUES (p_ticket_id, p_customer_id, p_topic, p_message);

   IF p_type = 'IT' THEN
      INSERT INTO IT_Ticket(ticket_id, device)
```

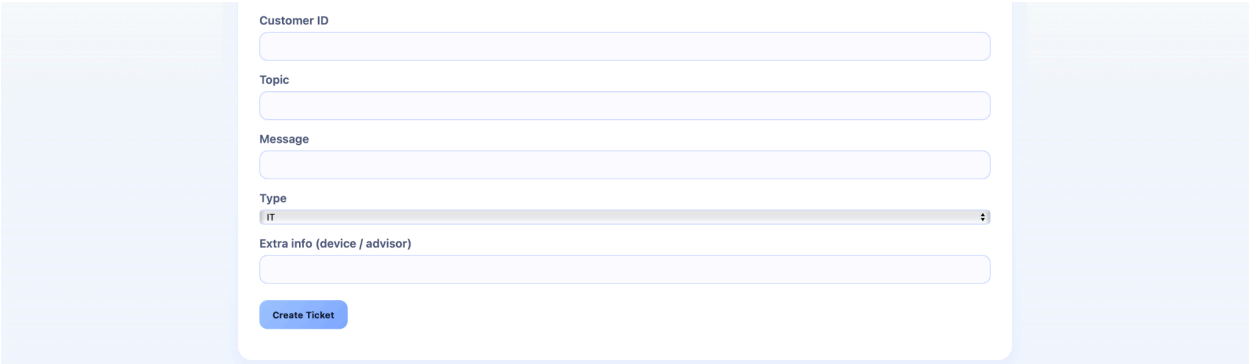
```
        VALUES (p_ticket_id, p_extra_info);

    ELSEIF p_type = 'Consultation' THEN
        INSERT INTO Consultation_Ticket(ticket_id, advisor)
        VALUES (p_ticket_id, p_extra_info);
    END IF;
END //
DELIMITER ;
```

Input Parameters and Corresponding Form Fields

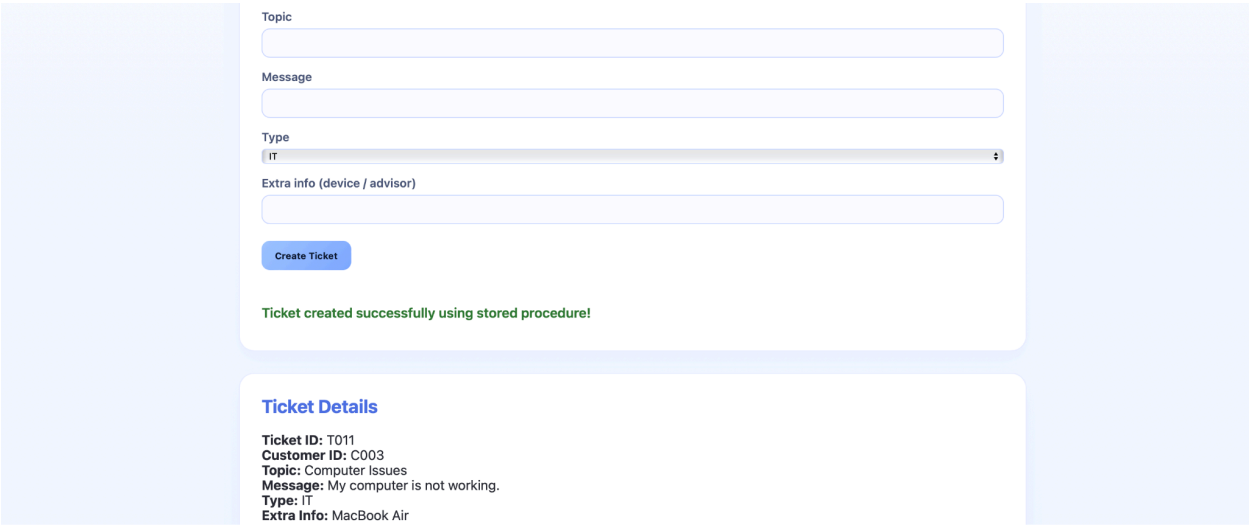
The CreateTicket stored procedure receives six parameters. Each parameter corresponds directly to one form input field in the web page:

Stored Procedure Parameter	Web Form Field Label	Description
p_ticket_id	Generated automatically	Unique ticket identifier
p_customer_id	Customer ID	ID of the user creating the ticket
p_topic	Topic	Short description of the issue
p_message	Message	Detailed explanation of the problem
p_type	Type	Ticket category (IT or Consultation)
p_extra_info	Extra Info	Device or advisor, depending on type

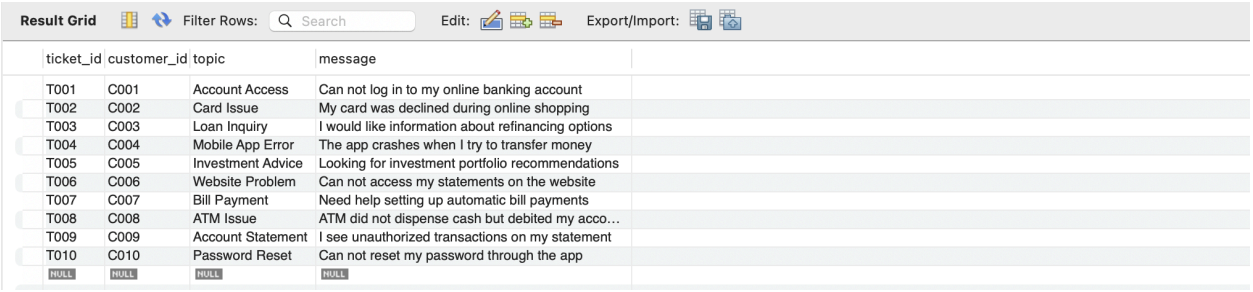
Before:



After:

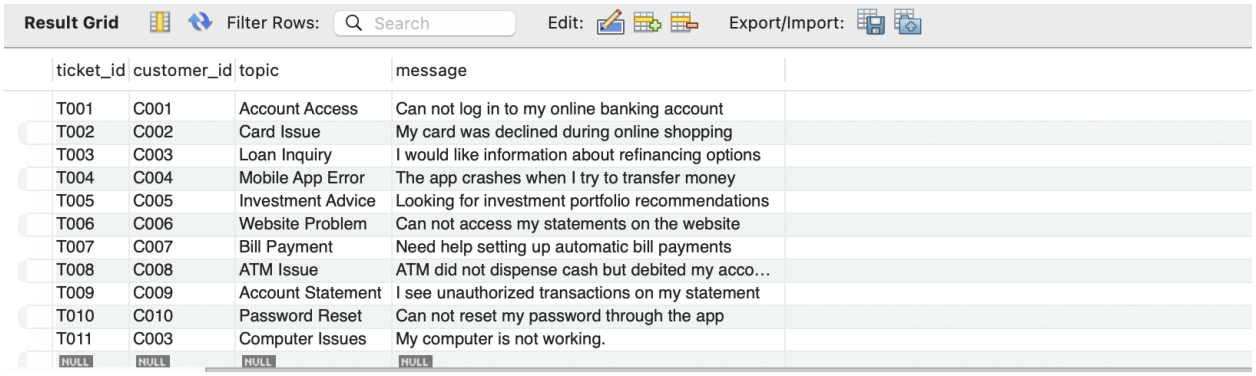


Before:



ticket_id	customer_id	topic	message
T001	C001	Account Access	Can not log in to my online banking account
T002	C002	Card Issue	My card was declined during online shopping
T003	C003	Loan Inquiry	I would like information about refinancing options
T004	C004	Mobile App Error	The app crashes when I try to transfer money
T005	C005	Investment Advice	Looking for investment portfolio recommendations
T006	C006	Website Problem	Can not access my statements on the website
T007	C007	Bill Payment	Need help setting up automatic bill payments
T008	C008	ATM Issue	ATM did not dispense cash but debited my acco…
T009	C009	Account Statement	I see unauthorized transactions on my statement
T010	C010	Password Reset	Can not reset my password through the app
NULL	NULL	NULL	NULL

After:



Result Grid | Filter Rows: Search | Edit: | Export/Import:

ticket_id	customer_id	topic	message
T001	C001	Account Access	Can not log in to my online banking account
T002	C002	Card Issue	My card was declined during online shopping
T003	C003	Loan Inquiry	I would like information about refinancing options
T004	C004	Mobile App Error	The app crashes when I try to transfer money
T005	C005	Investment Advice	Looking for investment portfolio recommendations
T006	C006	Website Problem	Can not access my statements on the website
T007	C007	Bill Payment	Need help setting up automatic bill payments
T008	C008	ATM Issue	ATM did not dispense cash but debited my acco…
T009	C009	Account Statement	I see unauthorized transactions on my statement
T010	C010	Password Reset	Can not reset my password through the app
T011	C003	Computer Issues	My computer is not working.
NULL	NULL	NULL	NULL

2.2 Stored Procedure 2: Add New Card

Cards are modeled as weak entities dependent on customers, meaning a card can not exist without a customer. The stored procedure is responsible for inserting a new card into the Has_Card table, ensuring that the card is always associated with an existing customer. To keep the interface simple and realistic, the user only provides:

- Customer ID
- Card Type (Physical / Virtual)

Instead of asking users to manually type security-sensitive values, the system generates these to remove the risk of invalid inputs:

- card number (9-digit)
- expiration date (four years from the current date)
- CVV (3-digit)

SQL SCRIPT:

```
DELIMITER //
CREATE PROCEDURE AddNewCard (
    IN p_card_number INT,
    IN p_customer_id CHAR(20),
    IN p_exp DATE,
    IN p_type CHAR(20),
```

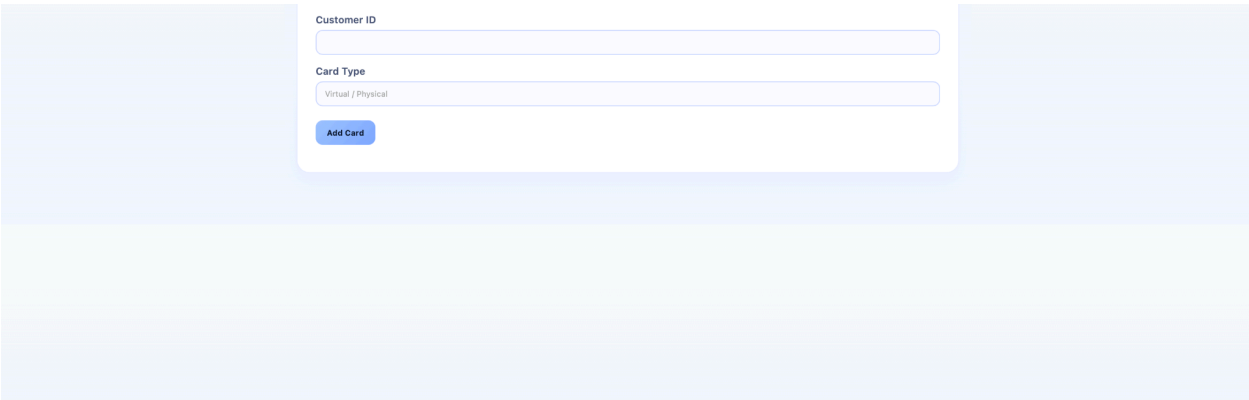
```
    IN p_cvv INT
)
BEGIN
    INSERT INTO Has_Card (card_number, customer_id, expiration_date, card_type, cvv)
    VALUES (p_card_number, p_customer_id, p_exp, p_type, p_cvv);
END //
DELIMITER ;
```

Input Parameters and Corresponding Form Fields

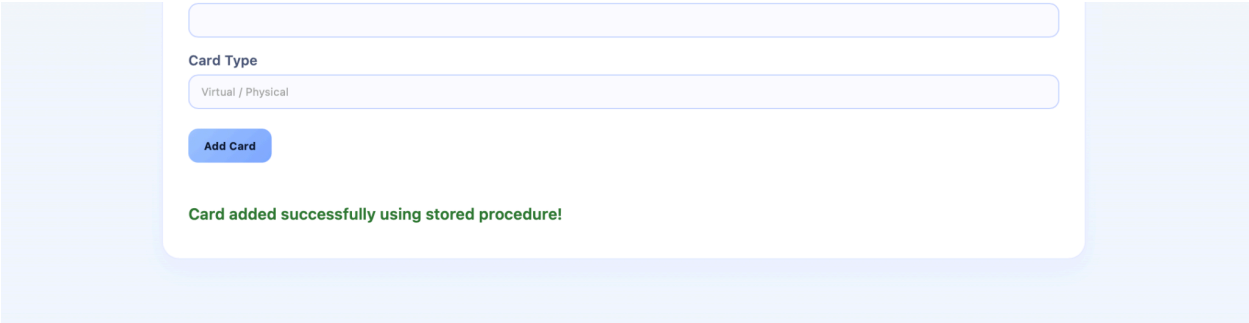
Values generated automatically are never typed by users.

Stored Procedure Parameter	Web Form Field Label	Description
p_card_number	Generated automatically	Unique card number
p_customer_id	Customer ID	Owner of the card
p_exp	Generated automatically	Expiration date
p_type	Card Type	Physical or Virtual
p_cvv	Generated automatically	Security code

Before:



After:



Before:

	card_number	customer_id	expiration_date	card_type	cvv
Düzenle Kopyala Sil	1001	C001	2027-04-30	Physical	237
Düzenle Kopyala Sil	1002	C002	2026-03-15	Virtual	374
Düzenle Kopyala Sil	1003	C003	2028-09-17	Physical	397
Düzenle Kopyala Sil	1004	C004	2027-02-11	Virtual	913
Düzenle Kopyala Sil	1005	C005	2029-07-30	Physical	467
Düzenle Kopyala Sil	1006	C006	2026-01-22	Virtual	365
Düzenle Kopyala Sil	1007	C007	2027-06-10	Physical	574
Düzenle Kopyala Sil	1008	C008	2028-10-14	Virtual	562
Düzenle Kopyala Sil	1009	C009	2026-12-01	Physical	525
Düzenle Kopyala Sil	1010	C010	2025-08-25	Virtual	753

After:

	card_number	customer_id	expiration_date	card_type	cvv
Düzenle Kopyala Sil	1001	C001	2027-04-30	Physical	237
Düzenle Kopyala Sil	1002	C002	2026-03-15	Virtual	374
Düzenle Kopyala Sil	1003	C003	2028-09-17	Physical	397
Düzenle Kopyala Sil	1004	C004	2027-02-11	Virtual	913
Düzenle Kopyala Sil	1005	C005	2029-07-30	Physical	467
Düzenle Kopyala Sil	1006	C006	2026-01-22	Virtual	365
Düzenle Kopyala Sil	1007	C007	2027-06-10	Physical	574
Düzenle Kopyala Sil	1008	C008	2028-10-14	Virtual	562
Düzenle Kopyala Sil	1009	C009	2026-12-01	Physical	525
Düzenle Kopyala Sil	1010	C010	2025-08-25	Virtual	753
Düzenle Kopyala Sil	496441912	C001	2029-12-27	Virtual	851

Admin Portal

The admin portal provides a dedicated environment for managing the MongoDB-based support ticket system. While the user interface focuses on submitting requests, the administrative interface enables monitoring, responding to, and resolving those requests in a controlled and traceable manner. The admin side consists of two pages: the dashboard and the ticket management page.

1. Admin Dashboard

The dashboard acts as an entry point for administrators. It summarizes the purpose of the administrative environment and provides navigation to the ticket manager. To support awareness, the dashboard also presents a brief overview of system activity such as the number of active tickets and the number of resolved tickets. This allows the administrator to quickly assess the current workload before moving to detailed operations.

2. Ticket Management Page

The ticket management page is the core administrative workspace. All interactions with the MongoDB ticket collections are implemented through this interface.

Only tickets whose status field is set to true are displayed, meaning that administrators see only unresolved and active issues. Tickets are sorted by creation time in descending order so that the most recent issues appear first.

For each ticket the following information is shown:

• subject of the ticket
• customer identifier
• body of the message
• creation timestamp
• current status (active or closed)
• history of previous admin replies

Each ticket box also includes an interactive update form. Administrators may either add a new reply or mark the ticket as solved. When a response is submitted, the reply is appended to the admin_reply array with the prefix “admin:” to clearly distinguish administrative messages from user-generated content. If a ticket is marked as solved, its status is updated to false and it disappears from both the user and admin interfaces, since both views filter only active records.

The page also supports informative status messages. After an update, the system displays a confirmation that the operation has completed successfully. This helps prevent ambiguity and ensures administrators receive feedback for each action.

The Flow

1. Insert New Ticket (submit_help.php)

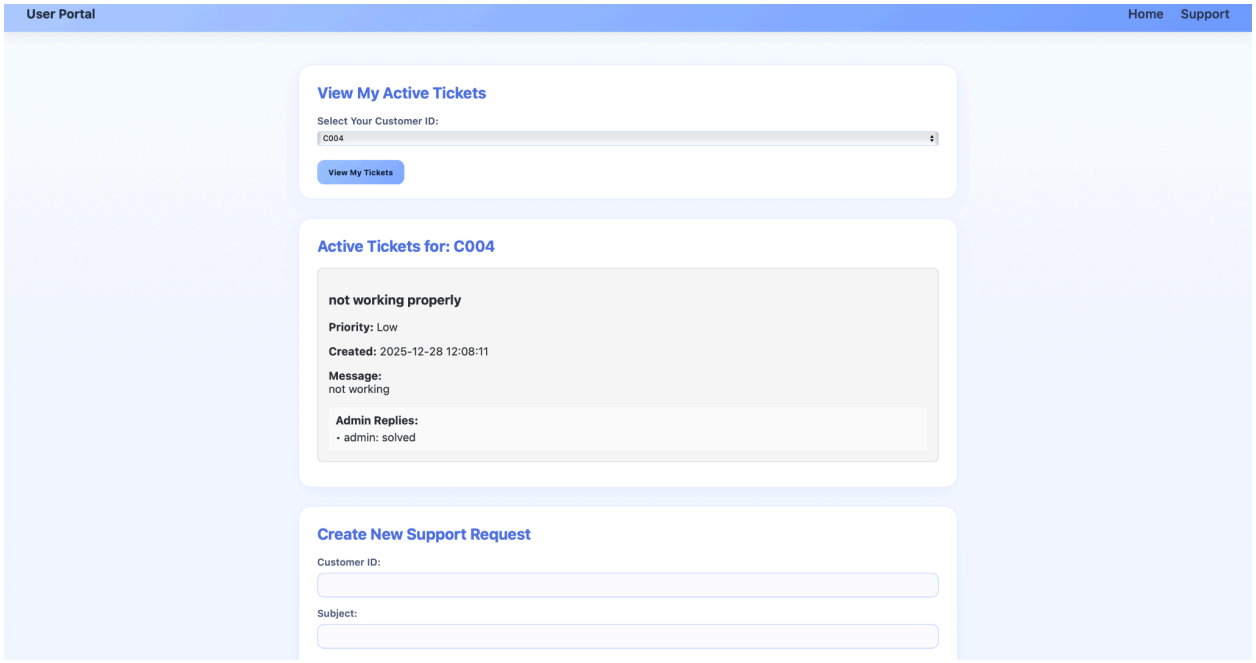
When a user submits the help form from user portal, the system inserts a new document:

```
$bulk = new MongoDB\Driver\BulkWrite();

$bulk->insert([

    'username' => $username,

    'message' => $message,

    'created_at' => date("Y-m-d H:i:s"),

    'status' => true,

    'admin_reply' => []

]);

$manager->executeBulkWrite('cs306.tickets', $bulk);
```

2. List Active Tickets (user side)

```
$filter = ['status' => true];

$query = new MongoDB\Driver\Query($filter);

$result = $manager->executeQuery('cs306.tickets', $query);
```

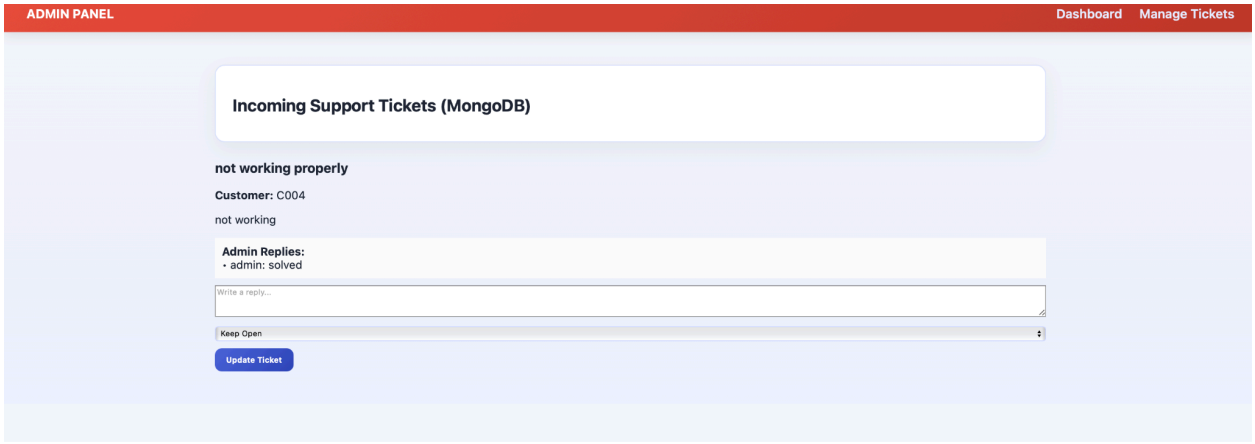
Only active tickets are displayed.



3. Add Admin Reply

```
$bulk->update(
        ['_id' => $mongoID],
         ['$push' => ['admin_reply' => "admin: $admin_reply"]]
);
```

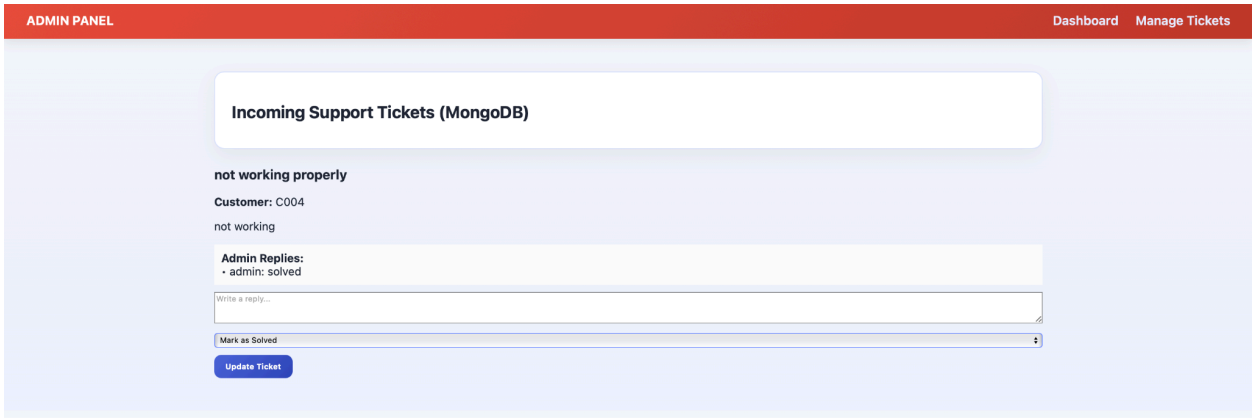
The new reply is appended to the comment history.

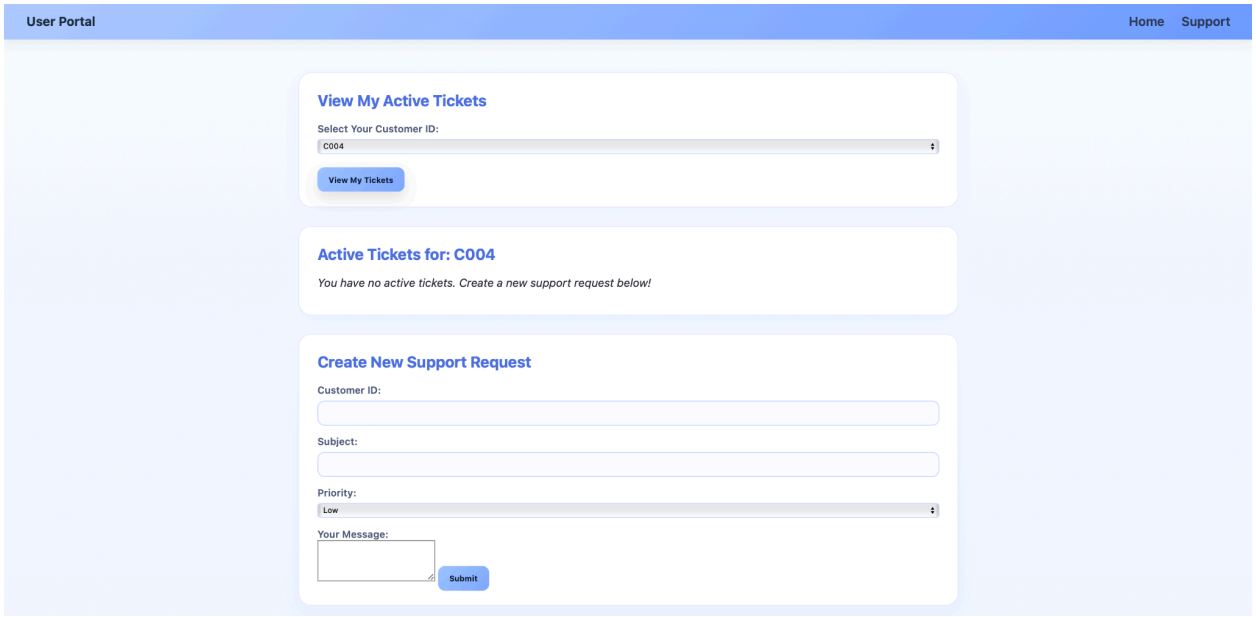


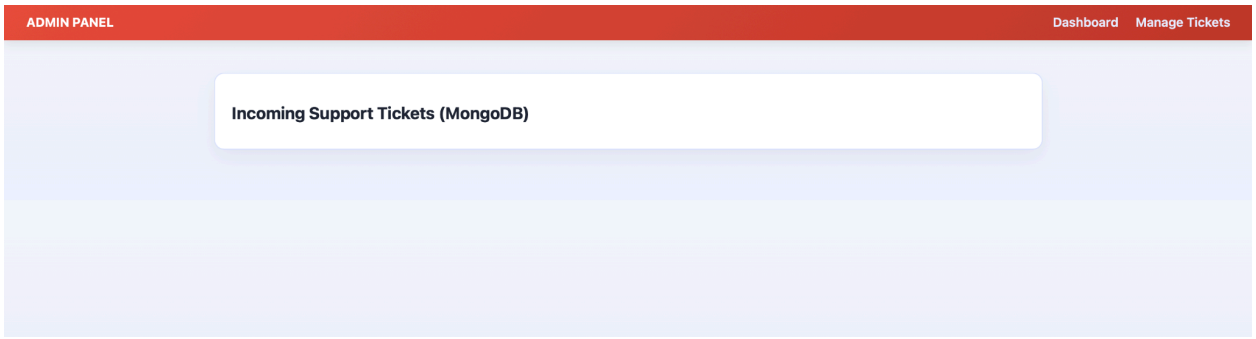
4. Mark Ticket as Resolved

```
$bulk->update(
    ['_id' => $mongoID],
    ['$set' => ['status' => false]]
);
```

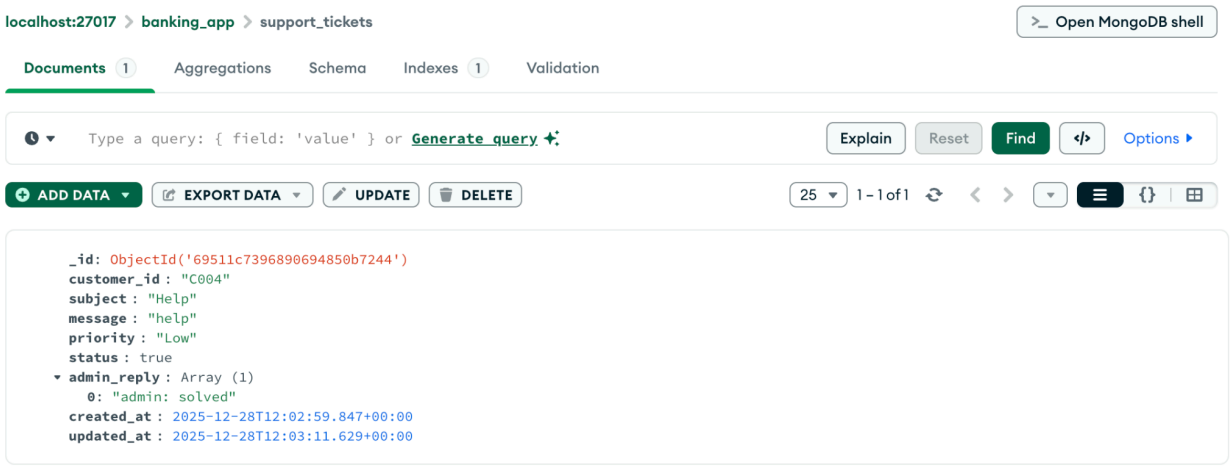
The ticket disappears from both admin and user dashboards.







5. MongoDB View



Assumptions, Challenges and Design Decisions

Several assumptions and design decisions were made during the development of the application. First, it is assumed that users provide valid identifiers corresponding to existing customer and account records in the system. In addition, users are assumed to have sufficient account balance when initiating bill payments, as balance validation is considered outside the primary scope of this phase of the project. Support tickets are also assumed to consist of short textual messages rather than large documents or file attachments. These assumptions were made to limit the project scope and to focus on demonstrating database design and integration principles.

One of the major challenges encountered was the integration of two different database management systems within a single application. MySQL is used to manage structured, transactional banking data, while MongoDB is employed to store flexible and unstructured support ticket messages. Ensuring reliable connectivity, data consistency, and seamless interaction between both databases and the PHP backend required careful configuration, testing, and error handling.

The final system design intentionally separates functional responsibilities across the two databases. Core banking logic, such as bill payments and credit score updates, is implemented using MySQL stored procedures and triggers to ensure data integrity and transactional reliability. In contrast, communication-oriented and schema-flexible data, such as customer support messages, is managed in MongoDB. MongoDB was preferred due to its schema flexibility, which allows future extensions such as message history or additional metadata without requiring schema modifications. This architectural separation improves clarity, scalability, and extensibility, while also reflecting real-world system design practices.

Although the system demonstrates the intended functionality, certain aspects such as balance validation, concurrency control and advanced security mechanisms were simplified to maintain focus on the core objectives of the project.