**ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY**

**DEPARTMENT OF SOFTWARE ENGINEERING**

****

**MOBILE APPLICATION DESIGN AND DEVELOPMENT**

**PROJECT TOPICS :**

* **BAKERY MANAGEMENT SYSTEM**

**MEMBERS NAME ID**

**ABDULBASIT NEZIF UGR/30026/15**

**ABDULHAFIZ MUHAMMED UGR/30030/15**

**SALHADIN KADI UGR/31167/15**

**ABDURAHMAN MUHAMMED UGR/30041/15**

**HUSSEIN BESHIR UGR/30697/15**

**SUBMITTED TO : Mr. YARED TEKALEGN**

**DATE :MAY 2, 2025**

**Chapter 1**

**Introduction**

* 1. **Background**

Managing a bread shop involves daily activities such as production tracking, sales recording, and revenue calculation. These tasks can be error-prone and inefficient when handled manually. Small bakery businesses, including those run by individuals like your father, often face challenges in managing their operations effectively while balancing time and accuracy.

**1.2 Problem Statement**

The manual management of daily production, sales, and revenue in bread shops is time-consuming and prone to errors. This creates inefficiencies, reduces productivity, and makes it difficult for shop owners to make informed decisions.

**1.3 Project Objectives**

**1.3.1 General Objective**  
To develop a user-friendly mobile application that simplifies the daily management of a bread shop's production, sales, and revenue tracking.

**1.3.2 Specific Objectives**

* Automate the calculation of daily earnings based on sales data.
* Provide clear input fields for production tracking and sales distribution.
* Allow bakery owners to record monthly and annual wheat purchases for better resource management.
* Enable efficient tracking of bread distribution to various outlets such as hotels, markets, and internal shop sales.

**1.4 Scope and Limitations**

**Scope:**  
The application will support small bakery businesses by focusing on daily production tracking, sales recording, and revenue calculation. It will also allow users to input data about wheat purchases and bread distribution.

**Limitations:**  
The app is designed for small-scale operations and does not include advanced features like supply chain management or integration with external accounting systems.

**Chapter 2**

**Requirements Specification**

**2.1 Functional Requirements**

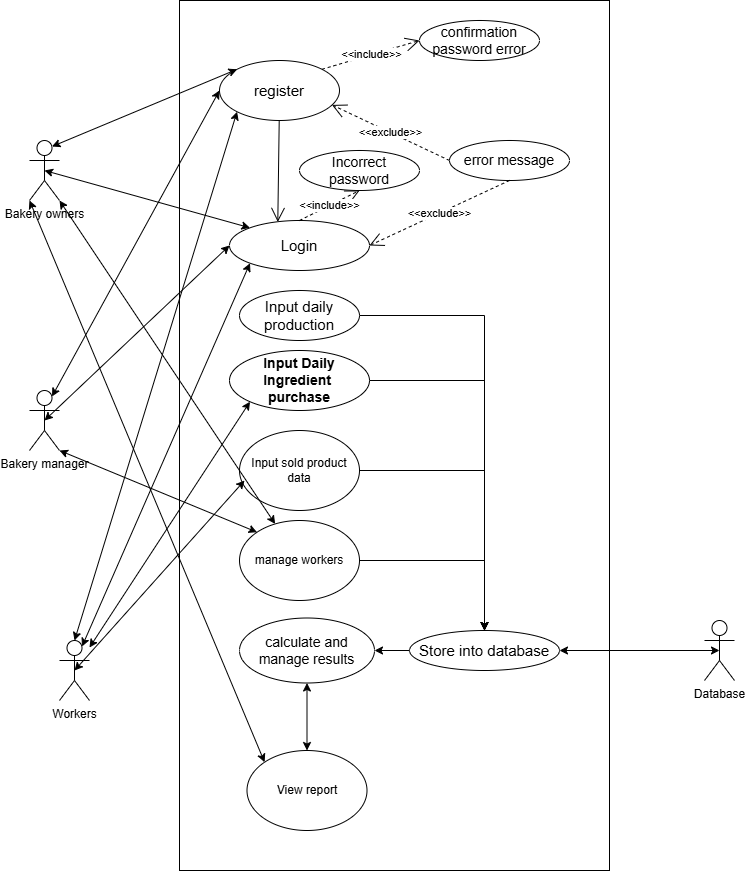
**2.1.1 Use Case Analysis**

**2.1.1.1 Use Case Diagram**

The actors and use cases should include:

* **Actor:** Bakery Owner
* **Use Cases:**
  + Input Daily Production and daily wheat consumption
  + Track Sales Distribution
  + View Revenue and Earnings

**Use Case Diagram**



**Fig 2.1 General 4+1 use-case diagram for the system**

**2.1.1.2 Detailed Use Case Specifications**

* **Use Case 1: Input Daily Production and daily wheat consumption**  
  **Actor:** Bakery Owner  
  **Preconditions:** User is logged or registered into the app .  
  **Main Flow:**
  1. Bakery owner enters production numbers for different bread types (e.g., 6birr, 7birr, 10birr, 12birr).
  2. Bakery owner enters daily ingredients such as wheat Flour,Water, yeast, salt, sugar, fats, additive ingredients together with their costs.
  3. App saves data to the database.  
     **Alternative Flows:**
  4. If data is incomplete, show a warning message.
* **Use Case 2: Track Sales Distribution**  
  **Actor:** Bakery Owner  
  **Preconditions:** Production and ingredient data is entered.  
  **Main Flow:**
  1. Owner enters the sales distribution or delivery (e.g., hotels, markets, internal shop).
  2. App manages the data entered.
* **Use Case 3: displays sales summary**  
  **Actor:** Bakery Owner  
  **Preconditions:** Production and ingredient data is entered, the sales distribution is managed.  
  **Main Flow:**
  1. The owner views the result of his or her shopping activity.
  2. App calculates and displays sales daily, monthly, annual summary on dashboard or report section with good user experience and UI such as graph and charts.

**2.2 Non-Functional Requirements**

**2.2.1 Quality Attributes**

* *Usability:* The app will have an intuitive user interface to simplify data entry.
* *Performance:* The app will process and display calculations within seconds to improve user experience.
* *Reliability:* Data entered by users will be securely stored and backed up.

**2.2.2 Quantitative Metrics**

* Response time for saving data should be under 2 seconds.
* System should support up to 1,000 entries per month.
* Availability of the app should exceed 99.9%.

**Chapter 3**

**Data and Transaction Design**

**4.1 Data design**

**4.1.1 Logical data model**

**Entities:**

* **Users**

Attributes: UserID (PK), FirstName, LastName, Email, RegistrationDate

* **Products**

Attributes: ProductID (PK), BreadType, CostPerUnit

* **Sales**

Attributes: SaleID (PK), SaleDate, BreadType, SoldQuantity, SalePrice

* **IngredientPurchase**

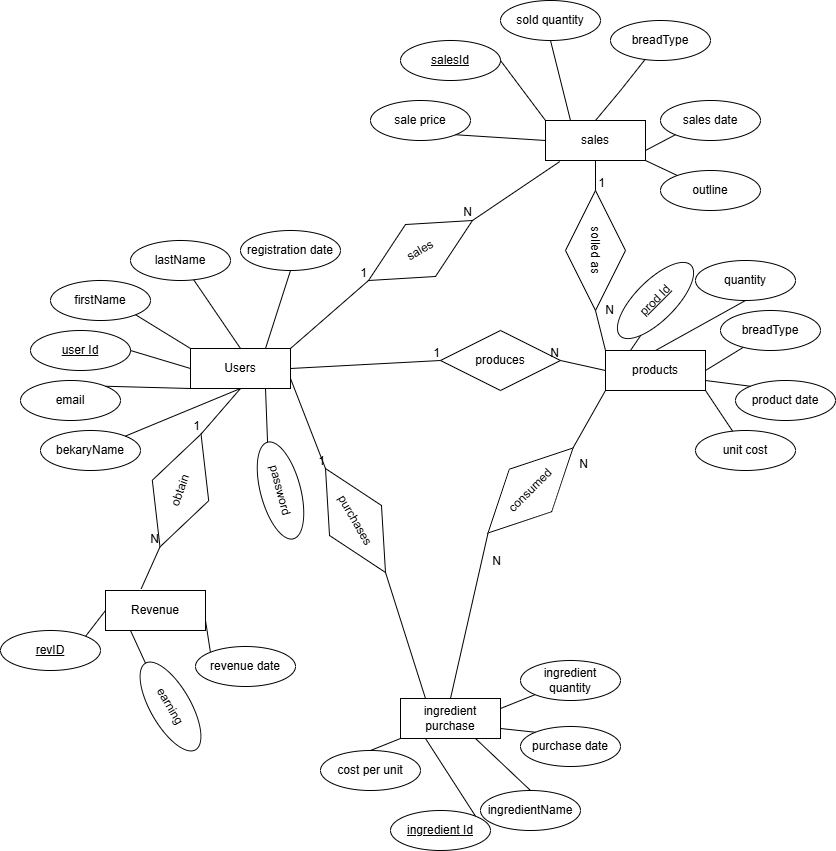
Attributes: PurchaseID (PK), QuantityPurchased, CostPerUnit, PurchaseDate

* **Revenue**

Attributes: RevenueID (PK), DailyEarnings, MonthlyEarnings, AnnualEarnings, RevenueDate

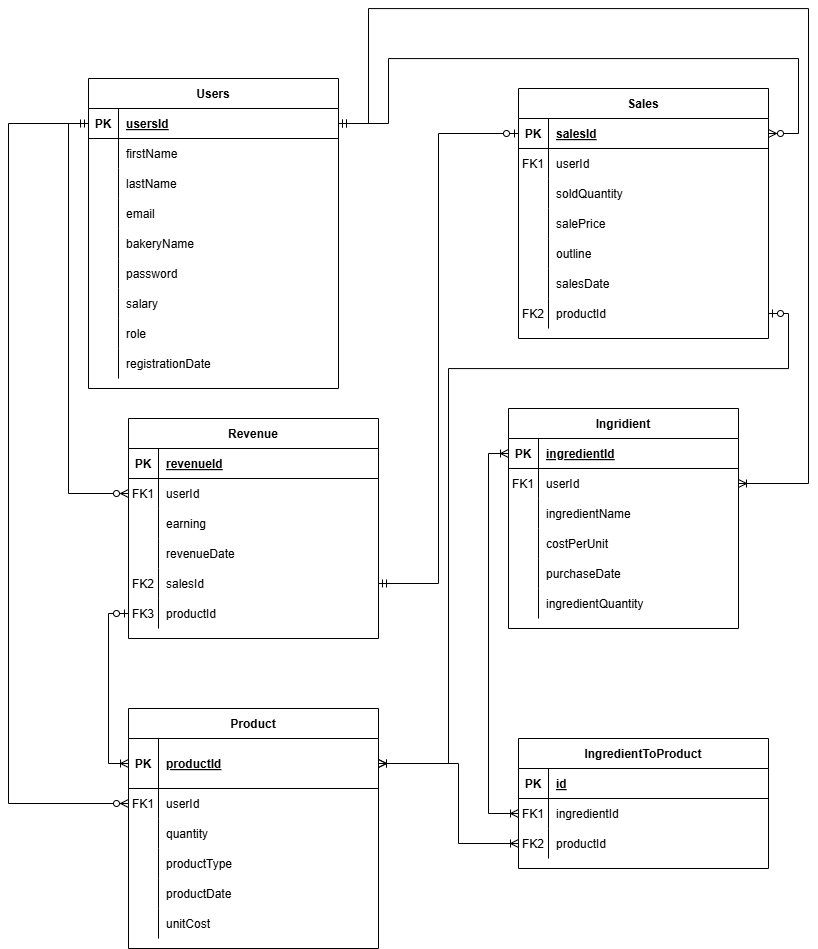
**Relationships:**

* Users can make multiple Sales.
* Each Sale is associated with one User and can relate to one or more Products.
* Each Product can have multiple IngredientPurchases.
* Revenue is calculated based on Sales data.

****

**Fig 4.1 ER diagram of bakery management web and mobile app**

**Relational schema (Table)**

****

**Fig 4.2 Relation diagram of bakery management web and mobile app.**

**4.2.1 Physical schema**

**Database tables/collections with constraints:**

*-- Users Table*

*CREATE TABLE Users (*

*userId INT PRIMARY KEY AUTO\_INCREMENT,*

*firstName VARCHAR(255),*

*lastName VARCHAR(255),*

*bakeryName VARCHAR(255),*

*password VARCHAR(255),*

*email VARCHAR(255) UNIQUE,*

*Salary DECIMAL(10,2),*

*Role ENUM (“manager”,”owner”,”worker”),*

*regDate DATETIME DEFAULT CURRENT\_TIMESTAMP*

*);*

*-- Product Table*

*CREATE TABLE Product (*

*productId INT PRIMARY KEY AUTO\_INCREMENT,*

*userId INT,*

*quantity INT,*

*productType VARCHAR(255),*

*productDate DATETIME DEFAULT CURRENT\_TIMESTAMP,*

*unitCost DECIMAL(10, 2),*

*FOREIGN KEY (userId) REFERENCES Users(userId)*

*);*

*-- Sales Table*

*CREATE TABLE Sales (*

*salesId INT PRIMARY KEY AUTO\_INCREMENT,*

*userId INT,*

*soldQuantity INT,*

*salePrice DECIMAL(10, 2),*

*outline TEXT,*

*salesDate DATETIME DEFAULT CURRENT\_TIMESTAMP,*

*productId INT,*

*FOREIGN KEY (userId) REFERENCES Users(userId),*

*FOREIGN KEY (productId) REFERENCES Product(productId)*

*);*

*-- Revenue Table*

*CREATE TABLE Revenue (*

*revenueId INT PRIMARY KEY AUTO\_INCREMENT,*

*userId INT,*

*earning DECIMAL(10, 2),*

*revenueDate DATETIME DEFAULT CURRENT\_TIMESTAMP,*

*salesId INT,*

*productId INT,*

*FOREIGN KEY (userId) REFERENCES Users(userId),*

*FOREIGN KEY (salesId) REFERENCES Sales(salesId),*

*FOREIGN KEY (productId) REFERENCES Product(productId)*

*);*

*-- Ingredient Table*

*CREATE TABLE Ingredient (*

*ingredientId INT PRIMARY KEY AUTO\_INCREMENT,*

*userId INT,*

*ingredientName VARCHAR(255),*

*costPerUnit DECIMAL(10, 2),*

*purchaseDate DATETIME DEFAULT CURRENT\_TIMESTAMP,*

*ingredientQuantity DECIMAL(10, 2),*

*FOREIGN KEY (userId) REFERENCES Users(userId)*

*);*

*-- IngredientToProduct Table (Many-to-Many Relationship between Ingredient and Product)*

*CREATE TABLE IngredientToProduct (*

*id INT PRIMARY KEY AUTO\_INCREMENT,*

*ingredientId INT,*

*productId INT,*

*FOREIGN KEY (ingredientId) REFERENCES Ingredient(ingredientId),*

*FOREIGN KEY (productId) REFERENCES Product(productId),*

*UNIQUE (ingredientId, productId) -- Ensures a unique combination of ingredient and product*

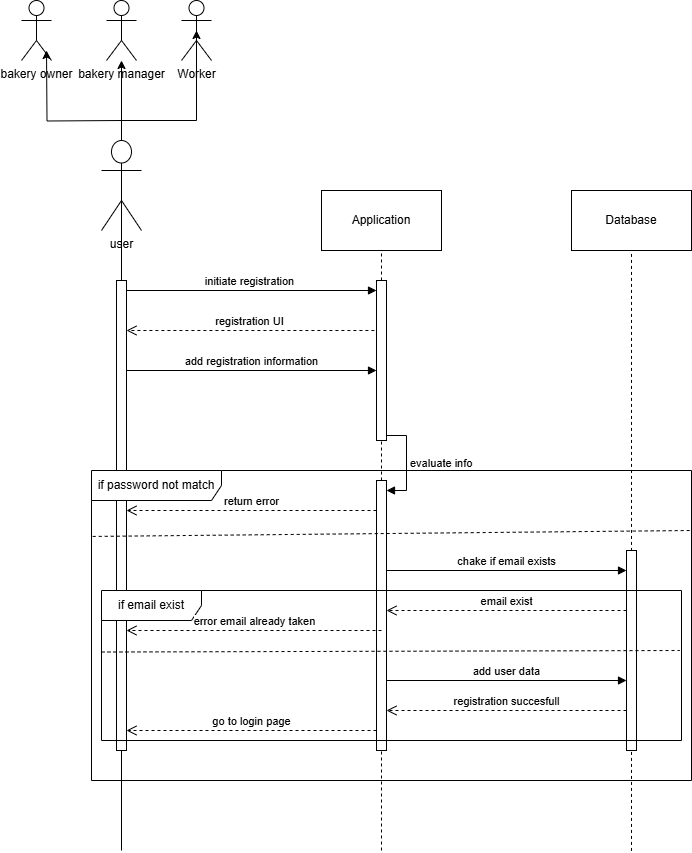
*);*

**4.2 Transaction design**

**4.2.1 Transaction flows**

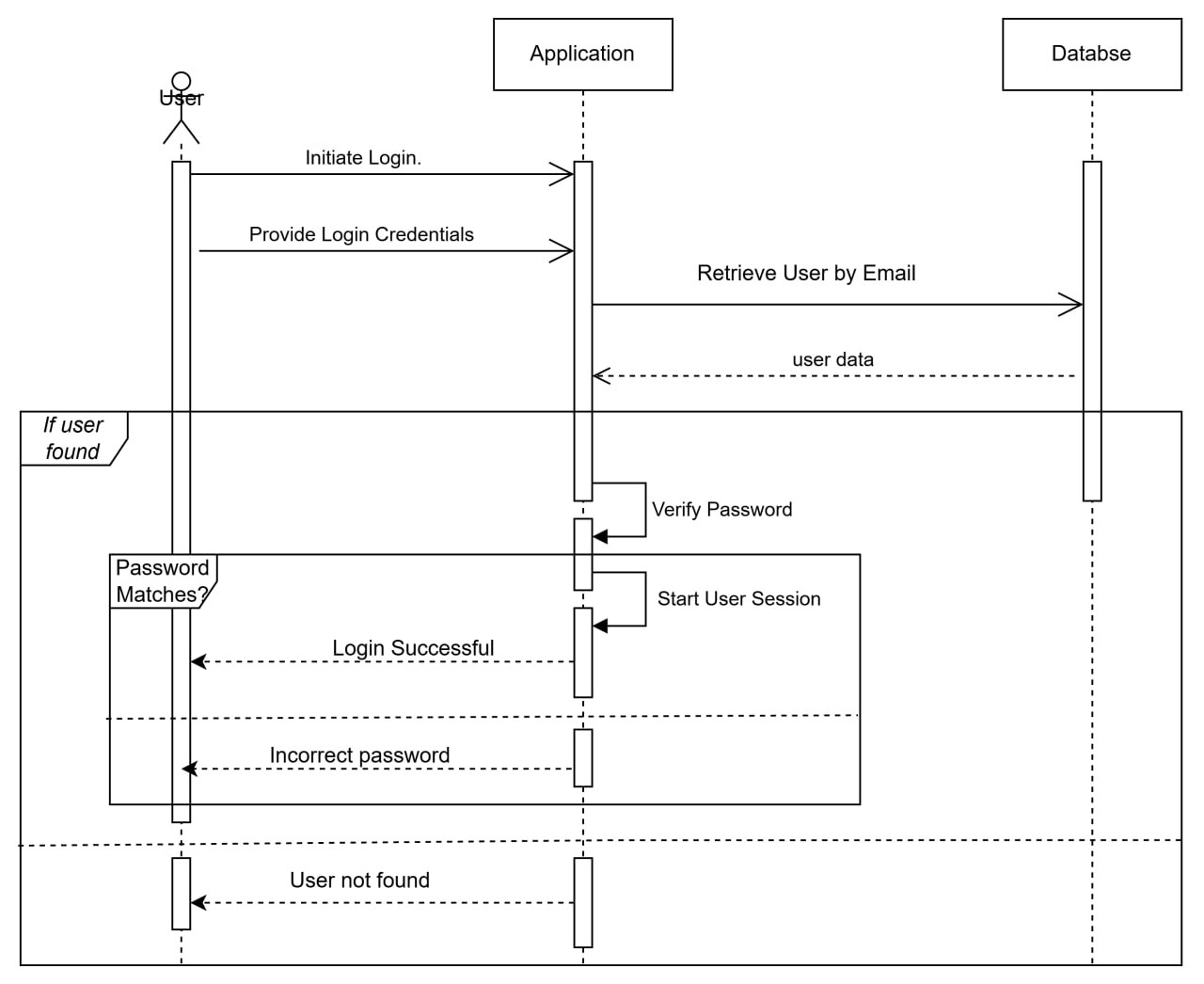
* **Sequence diagram for core operation**

**Sequence diagram for registration**

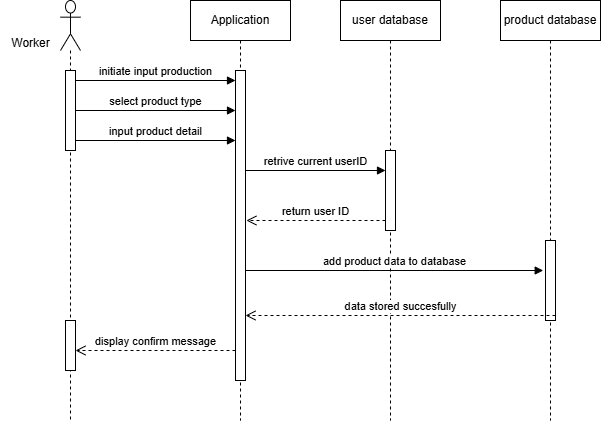
****

**Fig 4.3 sequence diagram for registration operation.**

**Login sequence diagram**

****

**Fig 4.3 sequence diagram for registration operation.**

****

**Fig 4.4 sequence diagram for adding product detail operation.**