MenuMap: Restaurant Menu Management System

Team Member Roles:
First Phase:

- Team Leader, Alfonso Oramas
- Document Editor & Diagram Coordinator, Andre Lewis
- Minute Keeper, Evelio Gonzalez
- Time Keeper, Alexandra Cozar Fraus

Second Phase:

- Team Leader, Alexandra Cozar Fraus
- Document Editor & Diagram Coordinator, Andre Lewis
- Minute Keeper, Alfonso Oramas
- Time Keeper, Kamal Ayoub

Third Phase:

- Team Leader, Andre Lewis
- Architecture and Design Lead, Alexandra Cozar Fraus

MenuMap Purpose and Scope of System

Purpose:

- MenuMap provides customers with an efficient way of finding their favorite restaurants and menu items.
- Customers can browse restaurants and menus, all while filtering for their dietary preferences.
- Restaurants can use MenuMap to reach a wider range of customers, through restaurant profiles and menus.

System Scope:

- MenuMap includes features such as:
    - User accounts, account authentication, menu verification, profile and preference management, and user data storage.
- MenuMap caters to both customers and restaurants.

Updated Project Schedule

Phase 1: Requirements (Weeks 1-3)

- Requirements gathering and documentation
- Use case development
- Deliverable 1 submission

Phase 2: Design (Weeks 4-10)

- Architecture design
- UML diagram creation
- Design document completion
- Deliverable 2 submission

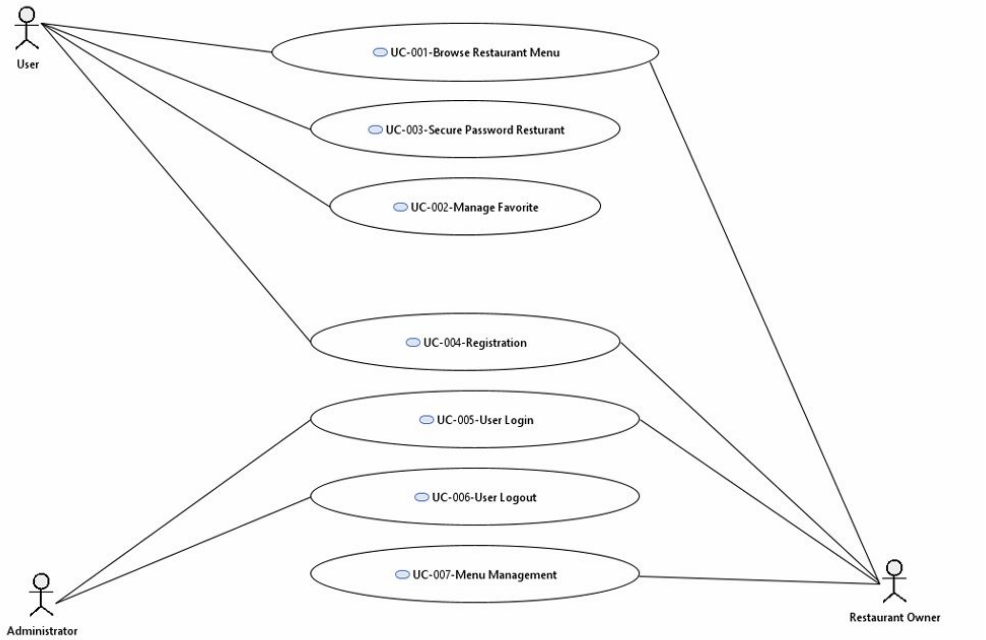
Phase 3: Implementation (Weeks 11-15)

- System implementation
- Comprehensive testing (33 test cases)
- Final documentation
- Deliverable 3 submission

Task ID	Task Description	Duration	Milestones/Deliverables	Dependencies
T1	Team Role Organizing	1 week	Deliverable 1	None
T2	Initial Use Cases	3 weeks	Deliverable 1	T1
T3	Deliverable Formatting	1 week	Deliverable 1	None
T4	Deliverable Completion	1 week	Deliverable 1 Completed	T1, T2
T5	Team Role Organizing	1 week	Deliverable 2	None
T6	Update Use Cases	1 week	Deliverable 2	T5
T7	Create Diagrams	3 weeks	Deliverable 2	T5
T8	Combine Contributions	1 week	Deliverable 2	T5, T6, T7
T9	Deliverable Completion	2 weeks	Deliverable 2 Completed	T8
T10	Team Role Organizing	1 week	Deliverable 3	None
T11	Creating Test Cases	1 week	Deliverable 3	None
T12	Updating Use Cases	1 week	Deliverable 3	T10
T13	Updating Diagrams	2 weeks	Deliverable 3	T10
T14	Combine Contribution	1 week	Deliverable 3	T10, T11, T12, T13
T15	Deliverable Completion	1 week	Deliverable 3 Completed	T14

UML Use Case Diagram showing all implemented use cases in the MenuMap system.

Implemented Use Case Diagram



Diagram

Functional and Non-Functional Requirements

UC-001: Browse Restaurant Menus

Description: Customer searches for and views restaurant menus with items, descriptions, and prices.

Functional Requirements:

- FR-005: System allows users to search for restaurants by name
- FR-006: System displays restaurant menus with items, descriptions, and prices
- FR-007: System supports filtering menu items by category
- FR-008: System handles cases where restaurants or menus are not found

Non-Functional Requirements:

- NFR-001: System response time for menu browsing shall be less than 2 seconds
- NFR-006: User interface shall be intuitive and easy to navigate
- NFR-007: System shall provide clear error messages to users

UC-003: Secure Password Reset

Description: User securely resets forgotten password through email verification process.

Functional Requirements:

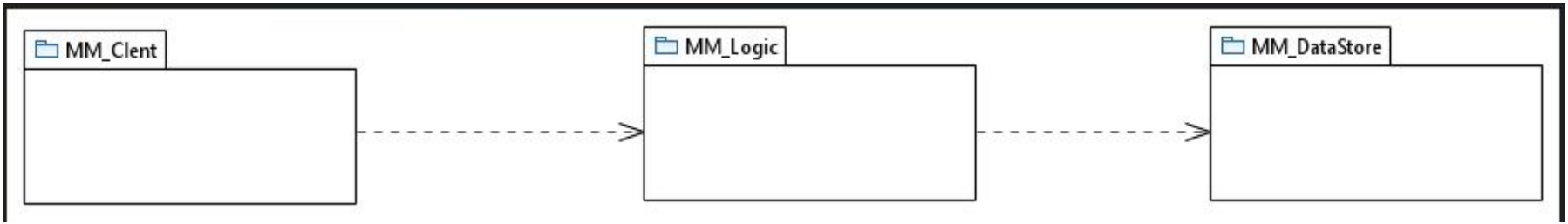
- FR-012: System allows users to request password reset
- FR-013: System validates user email before sending reset link
- FR-014: System generates secure, time-limited reset tokens
- FR-015: System validates reset tokens before allowing password change

Non-Functional Requirements:

- NFR-003: User passwords shall be hashed and stored securely
- NFR-004: System shall prevent SQL injection attacks
- NFR-005: System shall validate and sanitize all user inputs
- NFR-010: Reset tokens shall expire after 24 hours

Software Architecture Diagram

Software Architecture Diagram showing the 3-Tier Architecture of MenuMap system.



Data Management 1

Database Type: Relational Database (MySQL/PostgreSQL)
Primary Entities:

- Users: User accounts and authentication information
  - Attributes: userId, email, password (hashed), firstName, lastName, accountStatus, emailVerified
- Restaurants: Restaurant information and details
  - Attributes: restaurantId, name, address, phone, status, ownerUserId
- MenuItems: Individual menu items with descriptions and prices
  - Attributes: menuItemId, restaurantId, itemName, description, price, category, available

Relationships:

- Users → Restaurants (One-to-Many: Owner relationship)
- Restaurants → MenuItems (One-to-Many)
- Users → Sessions (One-to-Many)

Data Management 2

Data Access Objects (DAOs):

- MenuDAO: Handles menu data operations
  - Methods: findMenuByRestaurantId(), saveMenu(), updateMenu()
- UserDAO: Manages user data operations
  - Methods: findUserByEmail(), saveUser(), updateUser()
- RestaurantDAO: Handles restaurant data operations
  - Methods: findRestaurantById(), saveRestaurant(), updateRestaurant()

Data Integrity:

- Foreign key constraints ensure referential integrity
- Transaction management for data consistency
- Validation at service layer before persistence

Data Security:

- Passwords stored as hashed values (BCrypt)
- Sensitive data encrypted in transit (HTTPS)
- Parameterized queries prevent SQL injection

Access Control and Authentication

Authentication Mechanisms:

- Email/User ID and password authentication
- Secure password hashing using BCrypt algorithm
- Session management with secure tokens
- Password reset with time-limited tokens

Access Control:

- Role-based access control (RBAC)
  - Customer: Browse menus, manage favorites
  - Restaurant Owner: Manage own restaurant menus
  - Administrator: Verify menus, manage system
- Authorization checks at service layer
- Unauthorized access attempts logged

Session Management:

- Secure session tokens
- Session timeout after inactivity
- Session invalidation on logout
- Protection against session hijacking

Data Encryption and Privacy Protection

Data Encryption:

- Passwords: BCrypt hashing (one-way encryption)
- Data in transit: HTTPS/TLS encryption
- Sensitive data: Encrypted at rest in database
- Reset tokens: Cryptographically secure random generation

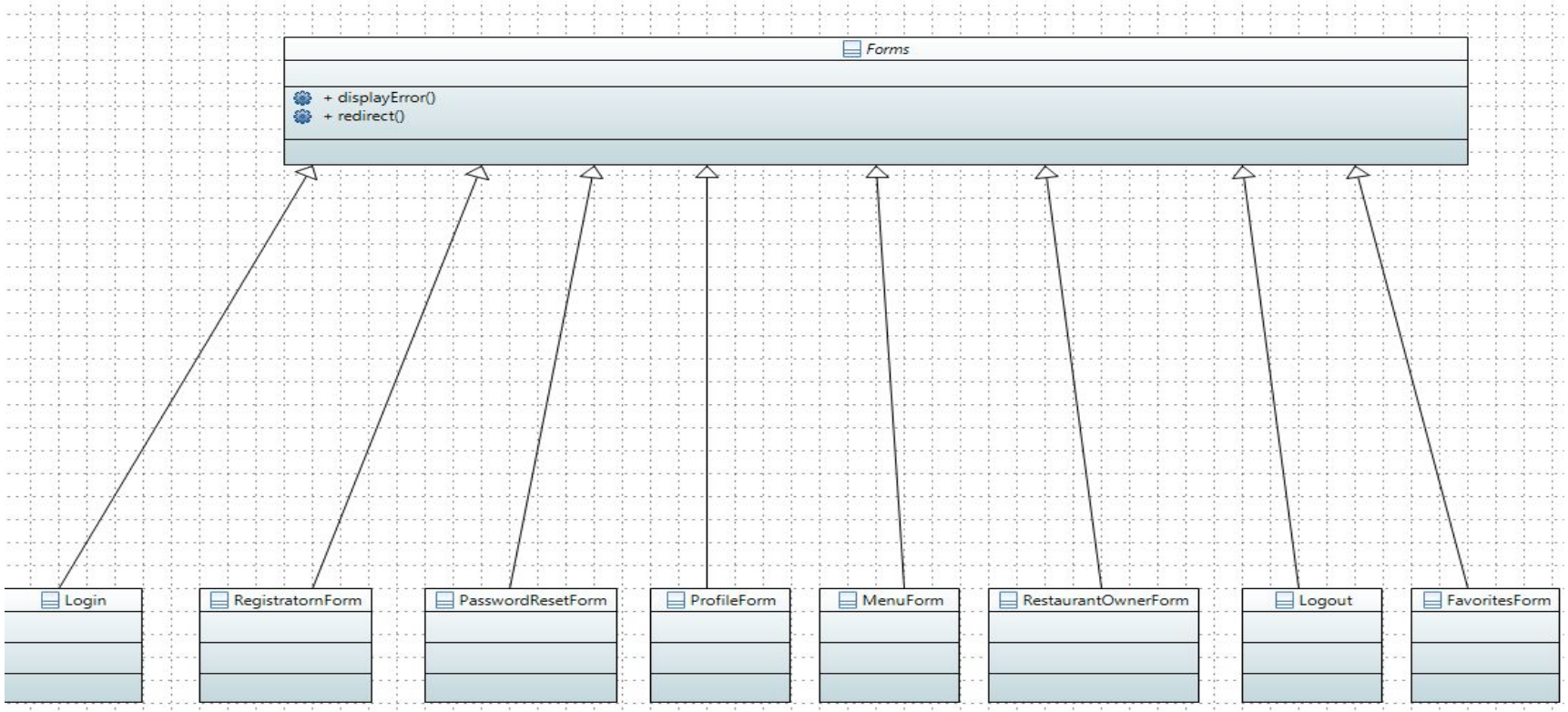
Privacy Protection:

- User data access restricted to authorized users
- No information leakage in error messages
- Secure password reset process
- Email verification required for account activation

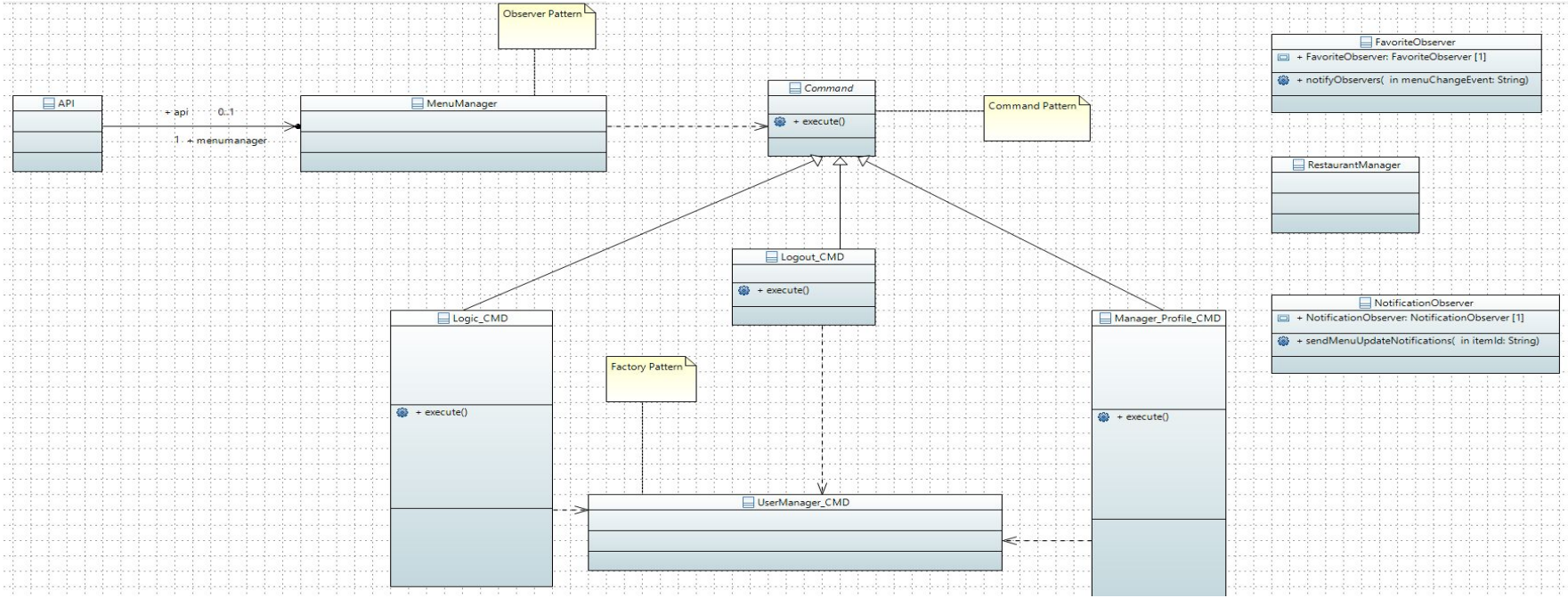
Security Measures:

- SQL injection prevention (parameterized queries)
- Input validation and sanitization
- XSS (Cross-Site Scripting) prevention
- CSRF (Cross-Site Request Forgery) protection
- Account lockout after multiple failed login attempts

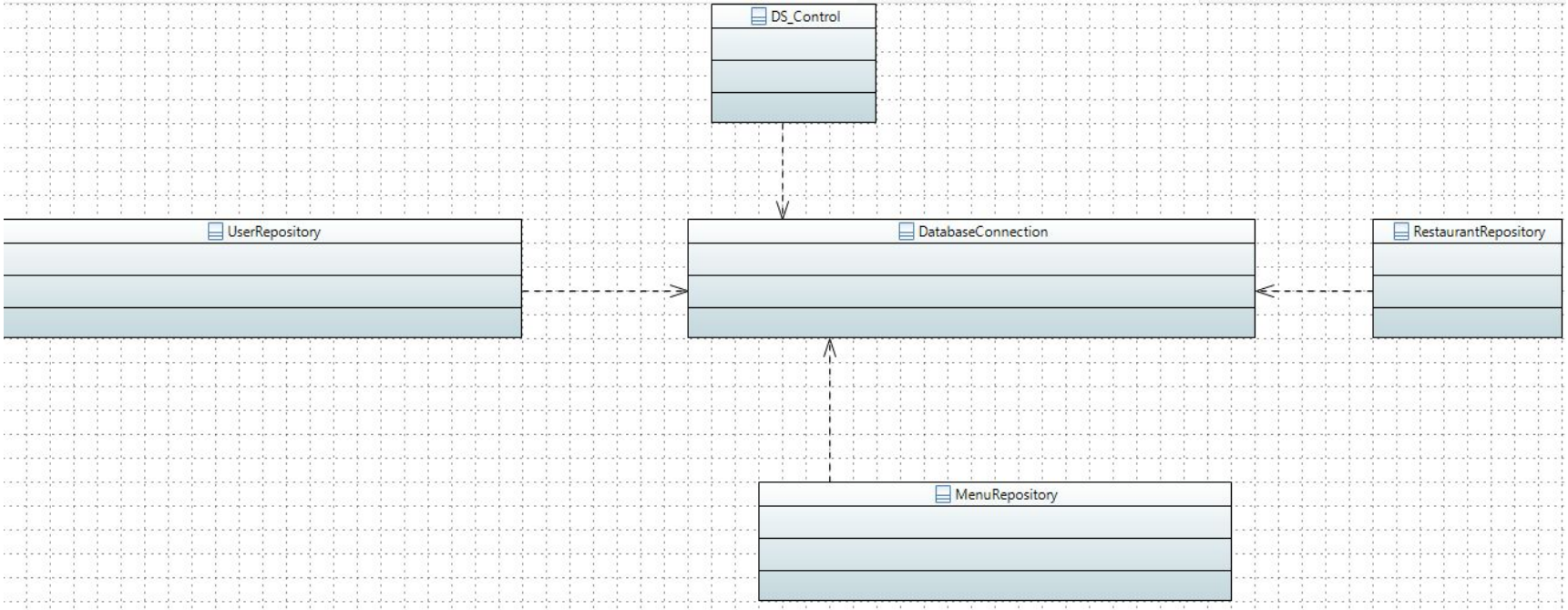
Minimal Class Diagram - Part 1



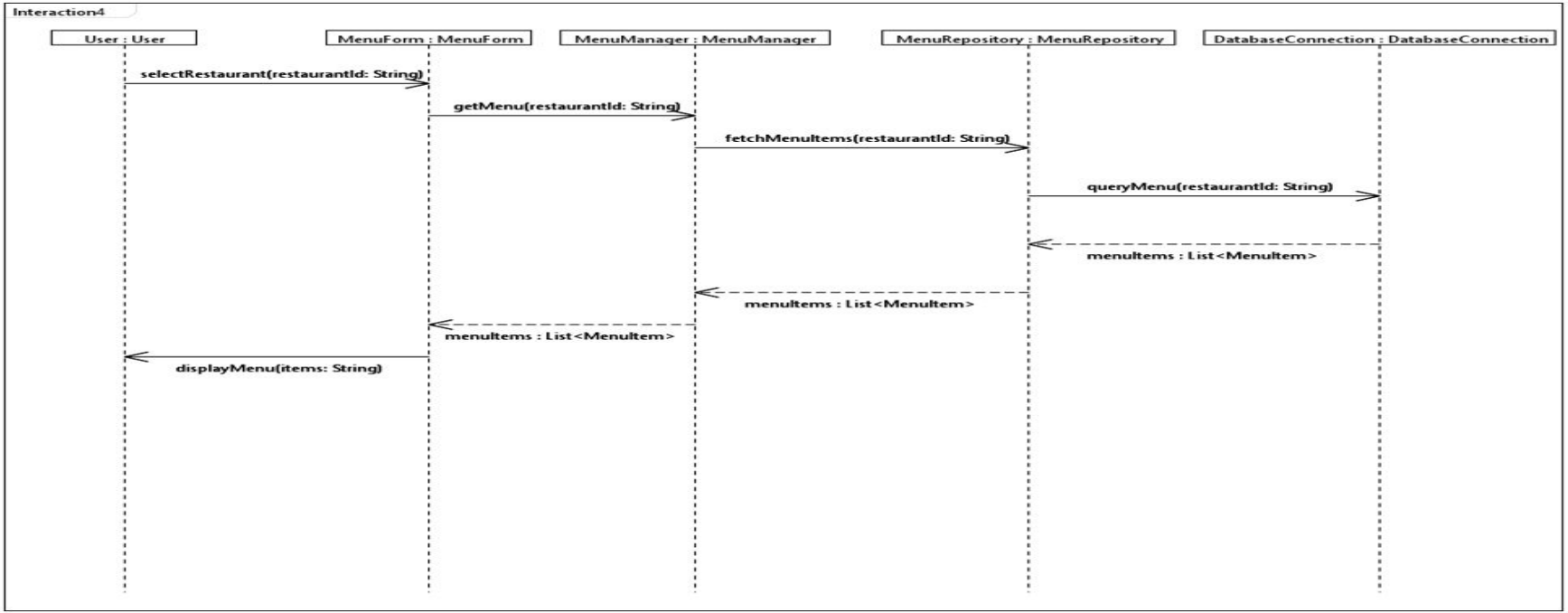
Minimal Class Diagram - Part 2



Minimal Class Diagram - Part 3



Sequence Diagram



UC-001: Browse Restaurant Menus - Sunny Day Test Case

Test Case ID: SystemTest-011-UC001

Purpose: Verify that a user can successfully browse and view a restaurant menu.

Test Setup:

- Restaurant "Joe's Pizza" exists with menu items in database
- Database state initialized (see Data Model)

Data Model (Before Test):

UC-001: Browse Restaurant Menus - Sunny Day Test Case

Test Case ID: SystemTest-011-UC001
Purpose: Verify that a user can successfully browse and view a restaurant menu.
Test Setup:

- Restaurant "Joe's Pizza" exists with menu items in database
- Database state initialized (see Data Model)

Data Model (Before Test):

Restaurant ID	Restaurant Name	Status
101	Joe's Pizza	Active

Menu Item ID	Restaurant ID	Item Name	Price	Category
1001	101	Margherita Pizza	$12.99	Entree
1002	101	Pepperoni Pizza	$14.99	Entree
1003	101	Caesar Salad	$8.99	Appetizer

UC-001: Browse Restaurant Menus - Rainy Day Test Case

Test Case ID: SystemTest-013-UC001
Purpose: Verify system handles search for non-existent restaurant.
Test Setup:

- Restaurant does not exist in database
- Database state initialized (see Data Model)

Data Model (Before Test):

Restaurant ID	Restaurant Name	Status
101	Joe's Pizza	Active
102	Burger Palace	Active

Test Steps:

1. User navigates to MenuMap application
2. User searches for "NonExistent Restaurant"
3. System processes search request

Expected Output:

Test Steps:

1. User navigates to MenuMap application
2. User searches for "Joe's Pizza"
3. User selects "Joe's Pizza" from results
4. System displays menu with all items

Expected Output:

- Menu displays correctly with all items, prices, and descriptions
- All menu items visible
- Categories properly organized

UC-003: Secure Password Reset - Sunny Day Test Case

Test Case ID: SystemTest-041-UC003
Purpose: Verify user can successfully reset password through secure process.
Test Setup:

- User "johndoe" exists with verified email
- Database state initialized (see Data Model)

Data Model (Before Test):

User ID	Email	Password (Hashed)	Email Verified	Account Status
johndoe	johndoe@example.com	$2a$10$hashed123	Yes	Active

Test Steps:

1. User navigates to password reset page
2. User enters email: johndoe@example.com
3. System validates email and generates reset token
4. System sends reset link to email
5. User clicks reset link
6. System validates token
7. User enters new password: NewSecurePass123!
8. System validates and updates password

Expected Output:

- Reset link sent successfully
- Token validated correctly
- Password updated securely
- User can log in with new password

Data Model (After Test):

User ID	Password (Hashed)	Reset Token
johndoe	$2a$10$newHash456	NULL (invalidated)

UC-003: Secure Password Reset - Rainy Day Test Case

Test Case ID: SystemTest-042-UC003
Purpose: Verify system handles expired reset token correctly.
Test Setup:

- User exists with expired reset token (older than 24 hours)
- Database state initialized (see Data Model)

Data Model (Before Test):

User ID	Email	Reset Token	Token Created	Token Expires
johndoe	johndoe@example.com	abc123token	2024-12-01 10:00	2024-12-02 10:00
Current Time			2024-12-03 15:00	(Token expired)

Test Steps:

1. User navigates to password reset page
2. User clicks expired reset link
3. System checks token expiration
4. System detects token is expired

Expected Output:

- Error message: "Reset link has expired. Please request a new password reset."
- Option to request new reset link
- User cannot proceed with expired token
- Security maintained (expired tokens cannot be used)

Data Model (After Test):

- No password changes
- Token remains expired
- User must request new reset

UC-001: Browse Restaurant Menus - Additional Test Case

Test Case ID: SystemTest-012-UC001
Purpose: Verify that user can filter menu items by category.
Test Setup:

- Restaurant exists with menu items in multiple categories
- Database state initialized (see Data Model)

Data Model (Before Test):

Menu Item ID	Restaurant ID	Item Name	Category
1001	101	Margherita Pizza	Entree
1002	101	Pepperoni Pizza	Entree
1003	101	Caesar Salad	Appetizer
1004	101	Garlic Bread	Appetizer

Test Steps:

1. User views restaurant menu
2. User selects "Appetizers" filter
3. System applies category filter

Expected Output:

- Only appetizer items displayed (Caesar Salad, Garlic Bread)
- Entree items hidden
- Filter works correctly

UC-003: Secure Password Reset - Additional Test Case

Test Case ID: SystemTest-043-UC003
Purpose: Verify system prevents password reset with invalid email.
Test Setup:

- Email does not exist in database
- Database state initialized (see Data Model)

Data Model (Before Test):

User ID	Email
johndoe	johndoe@example.com
janedoe	janedoe@example.com

Test Steps:

1. User navigates to password reset page
2. User enters non-existent email: nonexistent@example.com
3. System processes reset request

Expected Output:

- Generic message displayed: "If an account exists with this email, a reset link will be sent."
- No information leakage (does not reveal if email exists)
- Security maintained (prevents email enumeration)

Data Model (After Test):

- No reset tokens generated
- No changes to database
- Security maintained