UTILITY MANAGEMENT SYSTEM - PROJECT DOCUMENTATION

CAPSTONE PROJECT REPORT

January 2026

Author: Armaan Pandey
Project Title: Utility Management System
Website Name: Utility Billing Centre

TABLE OF CONTENTS

1. PROJECT OVERVIEW

The Utility Management System (Utility Billing Centre) is a comprehensive full-stack web application designed to streamline the management of utility services including electricity, water, and gas connections. The system provides an integrated platform for utility companies to manage consumers, connections, meter readings, billing, payments, and reporting.

The application follows a Role-Based Access Control (RBAC) model with four distinct user roles: Admin, Billing Officer, Account Officer, and Consumer. Each role has specific permissions and access to relevant modules.

Key Highlights:
- Full-stack web application with responsive design
- Real-time notifications system
- Comprehensive reporting and analytics
- Secure authentication using JWT tokens
- RESTful API architecture
- Modern Material Design UI

2. OBJECTIVES

Primary Objectives:

1. Digitize utility management operations
2. Automate billing cycle and bill generation
3. Provide real-time tracking of consumption and payments
4. Enable consumers to manage their utility connections online
5. Generate comprehensive reports for decision-making

Secondary Objectives:

---
1. Reduce manual paperwork and errors
2. Improve payment collection efficiency
3. Provide transparent billing to consumers
4. Enable data-driven insights through analytics
5. Implement secure role-based access control

3. TECHNOLOGY STACK

FRONTEND:

- Framework: Angular 21 (TypeScript)
- UI Components: Angular Material Design
- Styling: SCSS, Bootstrap
- Charts: Chart.js with ng2-charts
- HTTP Client: Angular HttpClient
- State Management: RxJS Observables
- Notifications: ngx-toastr
- JWT Handling: jwt-decode

BACKEND:

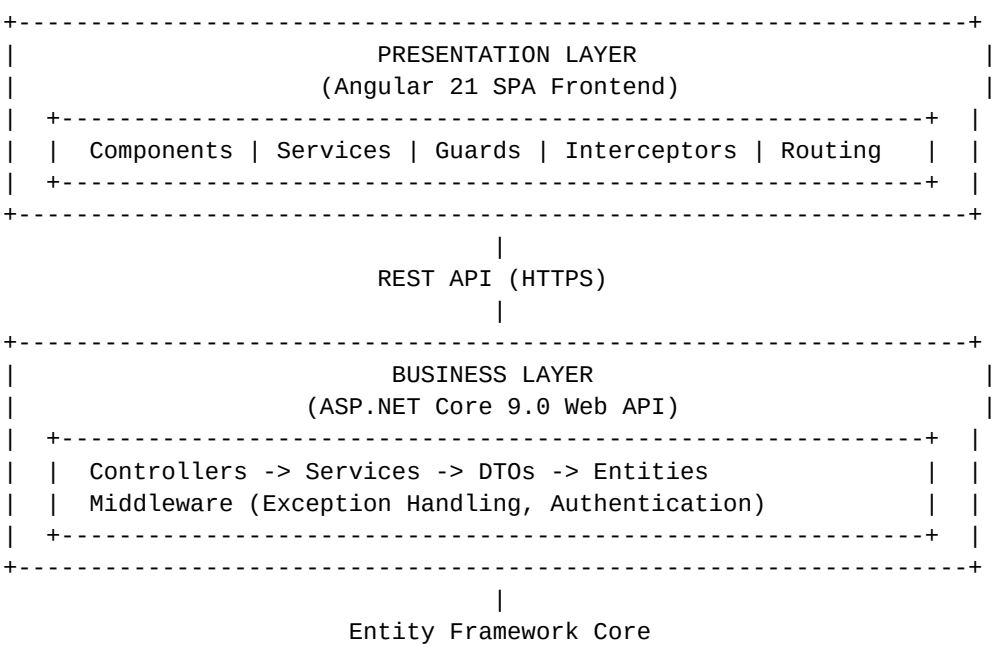
- Framework: ASP.NET Core 9.0 Web API
- ORM: Entity Framework Core 9.0
- Authentication: ASP.NET Core Identity + JWT Bearer
- Password Hashing: BCrypt.Net
- API Documentation: Swagger/OpenAPI (Swashbuckle)
- Database: Microsoft SQL Server (LocalDB)

DEVELOPMENT TOOLS:

- IDE: Visual Studio Code / Visual Studio 2024
- Version Control: Git
- Package Manager: npm (Frontend), NuGet (Backend)
- Testing: xUnit (Backend), Vitest (Frontend)

4. SYSTEM ARCHITECTURE

The application follows a 3-tier architecture:



```
                                |
+-------------------------------------------------------------------+
|                          DATA LAYER                               |
|                  (SQL Server Database - LocalDB)                  |
|  +-------------------------------------------------------------+  |
|  |  Tables: Users, Consumers, Connections, Bills, Payments,    |  |
|  |  MeterReadings, TariffPlans, UtilityTypes, Notifications    |  |
|  +-------------------------------------------------------------+  |
+-------------------------------------------------------------------+

FOLDER STRUCTURE:
-----------------
Capstone Code/
■■■ UtilityManagementApi/          # Backend - ASP.NET Core Web API
■   ■■■ Controllers/           # API endpoint controllers (12 controllers)
■   ■■■ Data/                  # DbContext and DataSeeder
■   ■■■ DTOs/                  # Data Transfer Objects
■   ■■■ Entities/              # Database entity models (11 entities)
■   ■■■ Middleware/            # Exception handling middleware
■   ■■■ Migrations/            # EF Core database migrations
■   ■■■ Services/              # Business logic layer
■   ■   ■■■ Interfaces/        # Service contracts (12 interfaces)
■   ■   ■■■ Implementations/   # Service implementations
■   ■■■ Properties/            # Launch settings
■
■■■ UtilityManagementApp/          # Frontend - Angular Application
■   ■■■ src/app/
■       ■■■ core/              # Core services, guards, interceptors
■       ■■■ features/          # Feature modules (13 feature modules)
■       ■■■ layouts/           # Layout components
■       ■■■ shared/            # Shared components and utilities
■
■■■ Test_Cases_All/     # Unit Tests
    ■■■ Unit_Test/             # Test cases for controllers & authorization

================================================================================
                          5. DATABASE DESIGN
================================================================================

ENTITY RELATIONSHIP DIAGRAM (ERD) - Key Entities:
-------------------------------------------------

1. ApplicationUser (extends IdentityUser)
   - Id (PK), FirstName, LastName, Email, PhoneNumber, IsActive
   - Relationships: Has one Consumer

2. Consumer
   - ConsumerId (PK), UserId (FK), ConsumerNumber, Address, City, State,
     PinCode, AadharNumber, IsActive
   - Relationships: Has many Connections

3. UtilityType
   - UtilityTypeId (PK), Name (Electricity/Water/Gas), Description,
     UnitOfMeasurement
   - Relationships: Has many Connections, TariffPlans

4. TariffPlan
   - TariffPlanId (PK), UtilityTypeId (FK), Name, Description,
     RatePerUnit, FixedCharges, LatePaymentPenalty, IsActive
   - Relationships: Belongs to UtilityType, Has many Connections

5. Connection
   - ConnectionId (PK), ConsumerId (FK), UtilityTypeId (FK),
     TariffPlanId (FK), ConnectionNumber, Status, MeterNumber,
```

ConnectionDate
- Relationships: Has many MeterReadings, Bills

6. ConnectionRequest
   - RequestId (PK), ConsumerId (FK), UtilityTypeId (FK), TariffPlanId (FK), Status (Pending/Approved/Rejected), RequestDate, ProcessedDate

7. MeterReading
   - MeterReadingId (PK), ConnectionId (FK), ReadingDate, CurrentReading, PreviousReading, UnitsConsumed, ReadingStatus
   - Relationships: Has one Bill

8. BillingCycle
   - BillingCycleId (PK), Name, StartDate, EndDate, DueDate, Status

9. Bill
   - BillId (PK), ConnectionId (FK), MeterReadingId (FK), BillNumber, BillingMonth, BillingYear, TotalAmount, DueDate, Status, OutstandingBalance, PenaltyAmount
   - Relationships: Has one Payment

10. Payment
    - PaymentId (PK), BillId (FK), Amount, PaymentDate, PaymentMethod, TransactionId, Status

11. Notification
    - NotificationId (PK), UserId (FK), Title, Message, Type, IsRead, CreatedAt

6. USER ROLES & PERMISSIONS

ROLE	PERMISSIONS
Admin	- Full system access - Manage all users and roles - Configure utility types and tariff plans - View all reports and analytics - Process connection requests - Manage system settings
Billing Officer	- View dashboard summary - Manage consumers - Process connection requests - Record meter readings (individual/bulk) - Generate bills (individual/bulk/by billing cycle) - Manage billing cycles - View tariff plans (read-only)
Account Officer	- View dashboard summary - Process and record payments - Generate revenue reports (monthly/yearly) - View outstanding dues reports - View collection analytics - Manage payment records
Consumer	- View personal dashboard - View own connections and status - Request new connections - View bills and payment history - Make online payments - Track consumption history

	- Update profile information	
	- View notifications	

7. MODULES & FEATURES

A. AUTHENTICATION MODULE

- User registration with email verification
- Secure login with JWT token authentication
- Role-based access control
- Password hashing with BCrypt
- Token refresh mechanism
- Logout functionality

B. CONSUMER MANAGEMENT MODULE

- Consumer registration and profile management
- Consumer search and filtering
- View consumer details and connection history
- Activate/deactivate consumer accounts
- Consumer number generation (auto)

C. CONNECTION MANAGEMENT MODULE

- Request new utility connections (Electricity/Water/Gas)
- Connection request approval workflow
- View connection status and details
- Connection number and meter number assignment
- Multiple connections per consumer support
- Connection activation/deactivation

D. METER READING MODULE

- Record individual meter readings
- Bulk meter reading upload
- Reading validation (current > previous)
- Automatic consumption calculation
- Reading history tracking
- Pending readings dashboard

E. BILLING MODULE

- Automatic bill generation based on meter readings
- Bulk bill generation by billing cycle
- Bill calculation: (Units × Rate) + Fixed Charges
- Late payment penalty application
- Bill status tracking (Pending/Due/Overdue/Paid)
- Outstanding balance management
- Bill number auto-generation

F. BILLING CYCLE MODULE

- Create and manage billing cycles
- Set billing period dates (start, end, due)
- Link bills to billing cycles
- Billing cycle status management

G. PAYMENT MODULE

- Record payment against bills
- Multiple payment methods (Online/Cash/Cheque/Card)

- Transaction ID tracking
- Payment status management (Pending/Completed/Failed)
- Partial payment support
- Payment history and receipts

H. TARIFF PLAN MODULE

- Create and manage tariff plans per utility type
- Configure rate per unit
- Set fixed charges
- Define late payment penalty percentage
- Activate/deactivate tariff plans

I. UTILITY TYPE MODULE

- Manage utility types (Electricity, Water, Gas)
- Define units of measurement (kWh, Liters, Cubic Meters)
- Associate tariff plans with utility types

J. REPORTS & ANALYTICS MODULE
---
1. Dashboard Summary
   - Total consumers count
   - Active connections count
   - Pending/overdue bills
   - Revenue this month
   - Total outstanding
   - Recent activities
   - Consumption by utility type (pie chart)
   - Revenue by utility type (bar chart)

2. Revenue Reports
   - Monthly revenue report
   - Yearly revenue report (aggregated)
   - Revenue breakdown by utility type
   - Collection rate calculation
   - Billed vs Collected comparison

3. Outstanding Dues Report
   - Age-wise bucketing (0-30, 31-60, 61-90, 90+ days)
   - Top defaulters list
   - Total outstanding amount
   - Overdue accounts count

4. Consumption Report
   - Consumption by utility type
   - Top consumers by consumption
   - Average consumption metrics

K. NOTIFICATION MODULE

- Real-time notifications
- Notification categories (Bill, Payment, Connection, System)
- Mark as read functionality
- Notification history
- Unread count badge

L. CONSUMER PORTAL

- Personal dashboard with summary
- View active connections
- Request new connections
- View and download bills
- Make online payments

- View payment history
- Track consumption trends
- Profile management

==
8. API ENDPOINTS
==

BASE URL: https://localhost:5001/api

AUTHENTICATION ENDPOINTS:

POST   /api/auth/register   - Register new user
POST   /api/auth/login   - Login and get JWT token
POST   /api/auth/refresh-token   - Refresh JWT token
GET   /api/auth/profile   - Get current user profile
PUT   /api/auth/profile   - Update user profile
POST   /api/auth/change-password   - Change password

CONSUMER ENDPOINTS:

GET   /api/consumers   - Get all consumers (Admin/BillingOfficer)
GET   /api/consumers/{id}   - Get consumer by ID
POST   /api/consumers   - Create new consumer
PUT   /api/consumers/{id}   - Update consumer
DELETE /api/consumers/{id}   - Delete/deactivate consumer
GET   /api/consumers/my-profile   - Get current consumer profile

CONNECTION ENDPOINTS:

GET   /api/connections   - Get all connections
GET   /api/connections/{id}   - Get connection by ID
GET   /api/connections/my-connections - Get consumer's connections
POST   /api/connections   - Create connection (Admin)
PUT   /api/connections/{id}   - Update connection

CONNECTION REQUEST ENDPOINTS:

GET   /api/connection-requests   - Get all requests
GET   /api/connection-requests/{id}   - Get request by ID
GET   /api/connection-requests/my-requests - Get consumer's requests
POST   /api/connection-requests   - Submit new request
PUT   /api/connection-requests/{id}/approve - Approve request
PUT   /api/connection-requests/{id}/reject - Reject request

METER READING ENDPOINTS:

GET   /api/meter-readings   - Get all readings
GET   /api/meter-readings/{id}   - Get reading by ID
POST   /api/meter-readings   - Record new reading
POST   /api/meter-readings/bulk   - Bulk record readings
PUT   /api/meter-readings/{id}   - Update reading

BILL ENDPOINTS:

GET   /api/bills   - Get all bills
GET   /api/bills/{id}   - Get bill by ID
GET   /api/bills/my-bills   - Get consumer's bills
POST   /api/bills/generate   - Generate bill for reading
POST   /api/bills/generate-bulk   - Generate bills for billing cycle
PUT   /api/bills/{id}   - Update bill

BILLING CYCLE ENDPOINTS:

```
GET    /api/billing-cycles         - Get all billing cycles
GET    /api/billing-cycles/{id}    - Get billing cycle by ID
POST   /api/billing-cycles         - Create billing cycle
PUT    /api/billing-cycles/{id}    - Update billing cycle
DELETE /api/billing-cycles/{id}    - Delete billing cycle

PAYMENT ENDPOINTS:
------------------
GET    /api/payments               - Get all payments
GET    /api/payments/{id}          - Get payment by ID
GET    /api/payments/my-payments   - Get consumer's payments
POST   /api/payments               - Record payment
PUT    /api/payments/{id}          - Update payment

TARIFF PLAN ENDPOINTS:
----------------------
GET    /api/tariff-plans           - Get all tariff plans
GET    /api/tariff-plans/{id}      - Get tariff plan by ID
POST   /api/tariff-plans           - Create tariff plan (Admin)
PUT    /api/tariff-plans/{id}      - Update tariff plan (Admin)
DELETE /api/tariff-plans/{id}      - Delete tariff plan (Admin)

UTILITY TYPE ENDPOINTS:
-----------------------
GET    /api/utility-types          - Get all utility types
GET    /api/utility-types/{id}     - Get utility type by ID
POST   /api/utility-types          - Create utility type (Admin)
PUT    /api/utility-types/{id}     - Update utility type (Admin)

REPORT ENDPOINTS:
-----------------
GET    /api/reports/dashboard              - Get dashboard summary
GET    /api/reports/revenue/monthly        - Monthly revenue report
GET    /api/reports/revenue/yearly         - Yearly revenue report
GET    /api/reports/outstanding-dues       - Outstanding dues report

NOTIFICATION ENDPOINTS:
-----------------------
GET    /api/notifications          - Get user notifications
GET    /api/notifications/unread-count - Get unread count
PUT    /api/notifications/{id}/read    - Mark as read
PUT    /api/notifications/mark-all-read - Mark all as read

================================================================================
                          9. FRONTEND STRUCTURE
================================================================================

ANGULAR APPLICATION STRUCTURE:
------------------------------
src/app/
■■■ core/                          # Core functionality
■   ■■■ guards/                    # Route guards (AuthGuard, RoleGuard)
■   ■■■ interceptors/              # HTTP interceptors (Auth, Error)
■   ■■■ models/                    # TypeScript interfaces/models
■   ■■■ services/                  # Core services (Auth, API)
■
■■■ features/                      # Feature modules
■   ■■■ admin/                     # Admin management
■   ■■■ auth/                      # Login/Register components
■   ■■■ billing/                   # Bill management
■   ■■■ billing-cycles/            # Billing cycle management
■   ■■■ connection-requests/       # Connection request handling
■   ■■■ connections/               # Connection management
■   ■■■ consumer-portal/           # Consumer dashboard
```

```
■   ■■■ consumers/                # Consumer management
■   ■■■ dashboard/                # Main dashboard
■   ■■■ meter-readings/           # Meter reading module
■   ■■■ notifications/            # Notifications
■   ■■■ payments/                 # Payment management
■   ■■■ reports/                  # Reports & analytics
■
■■■ layouts/                      # Layout components
■   ■■■ admin-layout/             # Admin/Officer layout
■   ■■■ consumer-layout/          # Consumer portal layout
■
■■■ shared/                       # Shared components
    ■■■ components/               # Reusable UI components
    ■■■ pipes/                    # Custom pipes
```

KEY ANGULAR FEATURES USED:

- Standalone Components (Angular 17+)
- Lazy Loading for feature modules
- Route Guards for authentication/authorization
- HTTP Interceptors for token handling
- Reactive Forms for form handling
- Angular Material for UI components
- RxJS for state management
- Server-Side Rendering (SSR) support

10. SECURITY IMPLEMENTATION

AUTHENTICATION:

- JWT (JSON Web Token) based authentication
- Tokens stored in localStorage
- Token expiration: 24 hours
- Automatic token refresh mechanism

AUTHORIZATION:

- Role-based access control (RBAC)
- [Authorize] attribute on API endpoints
- Role-specific route guards on frontend
- Policy-based authorization for complex scenarios

PASSWORD SECURITY:

- BCrypt hashing algorithm
- Minimum password requirements enforced
- Password change functionality

API SECURITY:

- HTTPS enforcement
- CORS policy configuration
- Input validation on all endpoints
- Exception handling middleware
- SQL injection prevention (parameterized queries via EF Core)

11. INSTALLATION & SETUP

PREREQUISITES:

- Node.js 18.x or higher
- npm 9.x or higher
- .NET SDK 9.0
- SQL Server (LocalDB or SQL Server Express)
- Visual Studio Code or Visual Studio 2024

STEP-BY-STEP INSTALLATION:
---

1. Clone/Download the Project
   > Extract to a folder (e.g., "Capstone Code")

2. Setup Backend (ASP.NET Core API)
   > cd UtilityManagementApi
   > dotnet restore
   > dotnet ef database update
   > dotnet run

   API will run at: https://localhost:5001

3. Setup Frontend (Angular App)
   > cd UtilityManagementApp
   > npm install
   > ng serve

   App will run at: http://localhost:4200

4. Access the Application
   > Open browser: http://localhost:4200
   > Login with test credentials (see below)

DATABASE CONNECTION STRING:

Located in: UtilityManagementApi/appsettings.json

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=UtilityBillingDb;
                          Trusted_Connection=True;MultipleActiveResultSets=true"
  }
}
```

==========

12. TEST CREDENTIALS

==========

The system comes pre-seeded with test accounts:

ROLE	EMAIL	PASSWORD
Admin	admin@utilitybilling.com	Admin@123
Billing Officer	billing@utilitybilling.com	Billing@123
Account Officer	account@utilitybilling.com	Account@123
Consumer (Sample)	Various seeded consumers	Consumer@123

Note: Additional test consumers are created with sample connections,
meter readings, bills, and payments during database seeding.

==========

13. FUTURE ENHANCEMENTS

==

Planned Features:
----
1. Email/SMS notifications for bill generation and due dates
2. Online payment gateway integration (Razorpay/PayU)
3. Mobile application (React Native/Flutter)
4. Advanced analytics with predictive consumption
5. Document management (bill PDFs, receipts)
6. Audit trail and logging
7. Multi-language support
8. Export reports to Excel/PDF
9. Customer support ticketing system

Technical Improvements:
----
1. Implement caching (Redis) for performance
2. Add real-time updates using SignalR
3. Implement CQRS pattern for complex queries
4. Add automated testing (integration tests)
5. CI/CD pipeline setup
6. Docker containerization
7. Cloud deployment (Azure/AWS)

==
                              END OF DOCUMENT
==