**Renewly**

**“Because staying updated should be effortless.”**

**Abstract**:

Renewly is a full-stack document management web application designed to help individuals effortlessly track and manage important time-sensitive documents. Whether it’s a driver’s license, insurance, ID proof, or any personal paperwork with an expiry date, Renewly acts as a smart assistant, ensuring that users are always notified before deadlines hit.

In today’s fast-paced life, people often forget renewal dates, leading to unnecessary penalties, service disruptions, or loss of validity. Renewly solves this problem with a clean interface and intelligent reminders through mobile OTP-based login, ensuring document security, personalized management, and timely alerts.

Built using Angular for the frontend, Python (Flask/Django) for the backend, and SQL Server for persistent data storage, this system offers a smooth and reliable experience. Renewly uses mobile number-based OTP login, with email being optional, making it accessible to everyone, even those without email access.

**Overview**:

**Key Modules:**

1. **Home**
   * Displays a personalized dashboard of the user’s documents
   * Highlights documents that are:
     + Expiring soon (within 7 days)
     + Already expired
     + Safe (valid for over a week)
   * Color-coded badges (Red, Yellow, Green) for quick glance
2. **Add Documents**
   * Form to input document title, type, expiry date, optional notes, and upload file (PDF/Image)
   * Category selection (e.g., ID, License, Insurance)
   * Secure file handling and validation
3. **Editable Profile**
   * View and update user details (mobile, optional email)
   * Reverify mobile number using OTP
   * Set reminder preferences (e.g., SMS only or SMS + email)
   * Option to delete account and data

**Tech Stack:**

|  |  |
| --- | --- |
| Layer | Technology |
| Frontend | Angular |
| Backend | Python (Flask or Django) |
| Database | SQL Server |
| Auth System | OTP via Mobile (JWT) |
| Optional | Email Reminders (SMTP) |

**Database Schema for Renewly:**

**1. Users Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| UserID | INT | Primary Key, Identity | Unique user identifier |
| MobileNumber | VARCHAR(15) | Unique, Not Null | User’s mobile number (for OTP login) |
| Email | VARCHAR(100) | Nullable | Optional email address |
| IsVerified | BIT | Default = 0 | Indicates if OTP is verified |
| CreatedAt | DATETIME | Default = GETDATE() | Account creation timestamp |

**2. Documents Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| DocumentID | INT | Primary Key, Identity | Unique document ID |
| UserID | INT | Foreign Key → Users | Owner of the document |
| Title | VARCHAR(100) | Not Null | Name/title of the document |
| DocType | VARCHAR(50) | Not Null | Type/category (e.g., License, ID) |
| ExpiryDate | DATE | Not Null | Expiry date of the document |
| Notes | TEXT | Nullable | Optional additional info |
| FilePath | VARCHAR(255) | Nullable | Path to the uploaded file |
| CreatedAt | DATETIME | Default = GETDATE() | When the document was added |

**3. ReminderLogs Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| ReminderID | INT | Primary Key, Identity | Unique reminder entry |
| DocumentID | INT | Foreign Key → Documents | Associated document |
| ReminderDate | DATETIME | Not Null | When reminder was sent |
| ReminderType | VARCHAR(20) | Not Null | Type of notification sent (SMS/Email) |

**Notes for Implementation:**

**✔ Frontend with Angular:**

* Use Angular CLI for project setup
* Implement components for Home, Add Documents, and Profile
* Use services to handle API calls with proper error handling

**✔ Backend with Python:**

* Use Flask or Django REST Framework for APIs
* Implement endpoints for user authentication, document management, and reminders
* Secure APIs using JWT for token-based authentication

**✔ Database with SQL Server:**

* Use pyodbc or SQLAlchemy for Python-SQL Server interaction
* Ensure encryption and proper indexing for performance

**✔** **Authentication**:

* Generate OTP using third-party SMS APIs (e.g., Twilio, MSG91)
* Secure login with token expiration and refresh mechanisms

**✔ File Handling:**

* Store uploaded files securely in server directories or cloud storage
* Validate file types (PDF, images) and sizes

**✔ Notifications:**

* Schedule reminders using Python’s Celery or APScheduler
* Optionally integrate email notifications via SMTP