A cartoon of cars and buildings

AI-generated content may be incorrect.

# Table of Contents

[Scenario and Scope: Define the task (Task 1A) (**±** 200 words) 1](#_Toc159454338)

[1. Topic 1](#_Toc159454339)

[2. Purpose of program 1](#_Toc159454340)

[3. Possible solution 1](#_Toc159454341)

[4. Scope 1](#_Toc159454342)

[User Requirements (TASK 1B) 2](#_Toc159454343)

[Use a table OR a 'use case diagram' 2](#_Toc159454344)

[Design the Database (TASK 2) 3](#_Toc159454345)

[Database Tables 3](#_Toc159454346)

[Relationship between the tables 3](#_Toc159454347)

[Data Dictionary (TASK 3A) 4](#_Toc159454348)

[Classes and Objects 4](#_Toc159454349)

[Data Dictionary (TASK 3B) 5](#_Toc159454350)

[(Text Files and Arrays) or Advanced Concepts 5](#_Toc159454351)

[ Text File 5](#_Toc159454352)

[ Arrays 5](#_Toc159454353)

[ Advanced programming constructs 5](#_Toc159454354)

[Navigation/Description of Flow Diagram (TASK 4A) 6](#_Toc159454355)

[Option 1 6](#_Toc159454356)

[Option 2 7](#_Toc159454357)

[Design the Graphical User Interface (GUI) (TASK 4B) 8](#_Toc159454358)

[Screen 1: <screen name> 8](#_Toc159454359)

[Screen 2: <screen name> 8](#_Toc159454360)

[Screen 3… 8](#_Toc159454361)

[IPO – Software Design Tool (TASK 5) 9](#_Toc159454362)

[Data Input validation (TASK 5A) 9](#_Toc159454363)

[Screen 1: Screen name 10](#_Toc159454364)

[INPUT 10](#_Toc159454365)

[Screen 2: Screen name 10](#_Toc159454366)

[INPUT 10](#_Toc159454367)

[Data Processing (TASK 5B) 11](#_Toc159454368)

[Data Output (TASK 5C) 12](#_Toc159454369)

# Scenario and Scope: Define the task (Task 1A)

# 1. Topic

# The topic I selected is “Vehicle License Management,” which is based on creating a desktop application that simplifies the handling of motor vehicle licenses. My project, *LicenseLink*, aims to assist both motorists and licensing departments by digitizing and streamlining tasks such as license generation, renewals, traffic fine checking, and access to services like test applications and routing to nearest licensing stations.

# 2. Purpose of Program

# Many South Africans face challenges with manually managing vehicle licensing – including forgetting renewal dates, paying fines late, or not knowing how or where to apply for license-related services. This often leads to unnecessary penalties, expired licenses, and time-consuming queues. *LicenseLink* is designed to centralize and simplify these processes in one user-friendly program. It saves time, improves compliance, and reduces confusion for users.

# 3. Possible Solution

# *LicenseLink* will consist of three main tab screens:

# Tab 1 – License Management: For generating new licenses and renewing existing ones. This screen will capture vehicle and owner data and store it in a database.

# Tab 2 – Traffic Fines and Test Applications: Users can check for outstanding fines and be redirected to online learner’s or driving license test applications. Fine data will be retrieved from a database.

# Tab 3 – Station Locator: Uses location input to display nearby licensing stations using routing logic. A text file may be used to store static data about station addresses and operating hours.

# All data will be managed through an ADO-connected database that stores user profiles, licenses, and fines.

# 4. Scope

# The application will not:

# process actual payments,

# provide real-time government updates,

# or integrate live geolocation services.

# It also will not interface directly with the Department of Transport systems

# but instead simulate their functionality with stored data.

# User Requirements (TASK 1B)

1. Vehicle Owner(User)

|  |  |
| --- | --- |
| Role | Regular user / vehicle owner |
| Activity | -Create an account  - Delete their own account  - Generate a new vehicle license  - Renew an existing license  - Check outstanding fines  - View licensing station locations |
| Limitations | * Cannot manage other users * Cannot process or approve test applications * Cannot perform admin maintenance tasks |

1. Admin App Manager (Administrator)

|  |  |
| --- | --- |
| Role | Admin/official at the licensing department docs |
| Activity | * Create and delete user accounts manually   Access all user data for maintenance   * Perform data cleanup or updates |
| Limitations | * Cannot generate or renew licenses for users * Cannot pay fines on behalf of users unless simulating the process for testing |

|  |  |  |
| --- | --- | --- |
| Task | Administrator | User |
| Log in to the system | Yes | Yes |
| Add a new User | Yes | Yes (self-registration) |
| Delete a User | Yes | Yes (delete own account) |
| Generate a new vehicle license | No | Yes |
| Renew an existing vehicle license | No | Yes |
| Check outstanding fines | Yes | Yes |
| View licensing station locations | Yes | Yes |
| Manipulate Database (Create fines, delete accounts, change user data) | Yes | No |

# Design the Database (TASK 2)

## 

## **Database Tables**

## **Table Name: tblUsers**

|  |  |  |  |
| --- | --- | --- | --- |
| **PK/FK** | **Field Name** | **Field Type** | **Field Size** |
| PK | UserID | Short Text | 20 |
|  | Username | Short Text | 25 |
|  | Password | Short Text | 25 |
|  | FirstName | Short Text | 25 |
|  | LastName | Short Text | 25 |
|  | emailAddress | Short Text | 25 |
|  | ContactNumber | Short Text | 15 |
|  | DateOfBirth | DateTime | Short |
|  | AccountStatus | Short Text | 20 |
|  | UserRoles | Short Text | 20 |
|  | CreatedAt | Date/Time | Short |
|  | UpdatedAt | Date/Time | Short |

## **Table Name: tblFines**

|  |  |  |  |
| --- | --- | --- | --- |
| **PK/FK** | **Field Name** | **Field Type** | **Field Size** |
| PK | FineID | Short Text | 20 |
|  | FineAmount | Currency | Currency |
|  | FineDate | Date/Time | Short |
|  | CreatedAt | Date/Time | Short |
|  | UpdatedAt | Date/Time | Short |
|  | FineType | Short Text | 20 |
| FK | LicenseID | Short Text | 50 |

## **Table Name: tblLicenses**

|  |  |  |  |
| --- | --- | --- | --- |
| **PK/FK** | **Field Name** | **Field Type** | **Field Size** |
| PK | LicenseID | Short Text | 50 |
|  | LicenseType | Short Text | 20 |
|  | ExpirationDate | Date/Time | Short |
|  | Province | Short Text | 20 |
|  | CreatedAt | Date/Time | Short |
|  | UpdatedAt | Date/Time | Short |
| FK | ownerID | Short Text | 20 |

## **Table Name: tblLicensingStations**

|  |  |  |  |
| --- | --- | --- | --- |
| **PK/FK** | **Field Name** | **Field Type** | **Field Size** |
| PK | StationID | Integer | 2 |
|  | StationName | Short Text | 100 |
|  | StationAddress | Long Text | - |
|  | StationContact | Short Text | 30 |
|  | CreatedAt | Date/Time | Short |
|  | UpdatedAt | Date/Time | Short |
|  | Coordinates | Short Text | 255 |

## **Table Name: tblPayments**

|  |  |  |  |
| --- | --- | --- | --- |
| **PK/FK** | **Field Name** | **Field Type** | **Field Size** |
| PK | PaymentID | Short Text | 20 |
|  | AmountPaid | Currency | Currency |
|  | PaymentDate | Date/Time | Short |
|  | PaymentMethod | Short Text | 20 |
|  | Status | Short Text | 20 |
| FK | FineID | Short Text | 20 |

## **Relationship between the tables**

tblLicenses

tblFines

tblUsers

tblLicenses

incurs

has

\

TblFines

tblPayments

has

One User owns many licenses

One license incurs many fines

One fine has one payment

# Data Dictionary (TASK 3A)

## **Classes and Objects**

## **Class Description**

## **The TFine class is used to represent an individual fine issued to a user in the LicenseLink system. It contains all necessary data about the fine, such as the amount, due date, license number, and fine type. This class allows easy manipulation and display of fine information throughout the program. It adds value by encapsulating all related attributes and methods in a single object, reducing the need for repeated database access and simplifying code reuse.**

## **How the Class Will Be Used**

## **Objects of the TFine class will be created when a new fine is generated or loaded. The object will store the fine’s data, allow for updates (e.g., changing the amount), and present formatted output (e.g., summary strings shown to users).**

## **Where the Object Will Be Used in the Program**

## **Fines Screen (frmFines): To load and display user fines.**

## **Payment Screen (frmPayment): To show a summary of the fine during payment.**

## **Reporting or Alerts: To format and send reminders using the toString method.**

## **Testing/Demo Data: To simulate fines without writing to the database.**

|  |
| --- |
| **Class Name:** TUser |
| ***Attributes*** |
| * fFineID: string * fUserID: string * fAmount : double * fDueDate: string * fLicense: string * fType : string |
| ***Methods*** |
| + Create(sFineID, sUserID, sDueDate,sLicense,sType: string; dAmount: double;)  + getDate(): string  + SetAmount(dAmnt: double): procedure  + toString(): string  + getFineID(): string  + getAmt(): double |

- 🡪 private

+ 🡪 public

# 

# Data Dictionary (TASK 3B)

## **Text Files and (Arrays or Advanced Concepts)**

### Text File

**Text File: LoginTrail.txt**

**Purpose**

The purpose of the LoginTrail.txt text file is to **record each user login** attempt. It stores a history of logins including the user's unique ID, the date and time of login, and their user type (Admin or Car Owner). This file is useful for auditing and tracking access without placing unnecessary load on the database.

**Read/Write Usage**

* **Write to File:**  
  The program writes to the file during a successful login event (inside the imgLoginButtonClick procedure). It appends a new line each time someone logs in.
* **Read from File:**  
  In this specific implementation, the file is only written to, but it could be read from in the future for displaying login history to the admin or tracking misuse.

**When This Happens:**

* The file is written to **immediately after a successful login** (both for Admins and regular users).
* The Append() function is used so that existing entries are preserved, and each new login is added to the end of the file.

**Text File Format**

Each line in the file is formatted as follows:

<UserID>#<Login Date>#<Login Time>#<User Type>

**Example:**

USR123#2025/05/01#09:45:32#Car Owner

* UserID links the text file to the tblUsers table in the database (foreign key style).
* The rest of the line stores extra data not found in the database (login time & user type label).
  + - **Extract of data:** Place three lines as an example of your text file here.

### Arrays

Used in frmRouting with arrLocs

* Used to store the names of the locations for permanent use

# Navigation/Description of Flow Diagram (TASK 4A)

## A diagram of a software company AI-generated content may be incorrect.

# Design the Graphical User Interface (GUI) (TASK 4B)

## **Screen 1: Login Screen**

## **A screenshot of a login screen AI-generated content may be incorrect.**

## **Screen 2: Sign up**

*A screenshot of a login form

AI-generated content may be incorrect.*

## Screen 3: Main Screen

A screenshot of a computer

AI-generated content may be incorrect.

Profile:

A screenshot of a computer

AI-generated content may be incorrect.

# 

# Routing

# A screenshot of a phone AI-generated content may be incorrect.

# Generation

# A screenshot of a computer screen AI-generated content may be incorrect.

# Fines

# A screenshot of a computer AI-generated content may be incorrect.

# IPO – Software Design Tool (TASK 5)

## Data Input validation (TASK 5A)

Input interfaces and validation.

You need to validate at least

* + - 4 different data types
    - At least FOUR inputs validated, including:

- Validate for NULL/empty field AND

- Test if value was selected in a selection component

* + - Associated error messages

For example:



<https://youtu.be/5fjW_AtGGXM>

* Test if an Edit has been left empty.
* Test if the user selected something from the ComboBox / RadioGroup / ListBox.
* Use Try .. catch StrToFloat for real input (real data type).
* Test the length of a string entered (string data type).
* Test if an integer number is in a certain range. (Integer data type).
* Test if the file exists (Text file data type).

For the *DateTimePicker*, you can test if they selected a date before or after a certain date depending on your scenario. This will give you another data type check should you need one.

Specify the format, data types, source of input, validation of input and error checking mechanisms of at least TWO INPUT interfaces.

Use the following format:

**Screen 1: License Generation Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | Source | Data Type | Format | GUI Component | Validation |
| sProv | ComboBox | String | Province abbreviation | cmbProvince | Check selection using `itemindex` |
| sType | ComboBox | String | License type | cmbLicenseType | Check selection using `itemindex` |
| ExpirationDate | System Date | Date | YYYY/MM/DD | Generated automatically | Set using `incYear(Date, 1)` |
| License Limit | Programmatic | Integer | Max 2 Licenses | Internal check | If record count >= 2 then block |

**Screen 2: Sign Up Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | Source | Data Type | Format | GUI Component | Validation |
| sUser | Keyboard | String | General Text | edtUsername | Check if empty using `TValidation.notEmpty` |
| sPass / Confirm | Keyboard | String | Password | edtPassword / edtConfirmPassword | Check if empty and match |
| sEmail | Keyboard | String | Email format | edtEmail | Check if empty and contains @ and . |
| sContact | Keyboard | String | 10 digits only | edtContact | Check length = 10 and not empty |
| DOB | Date Picker | Date | Valid past date | dtpDOB | Ensure selected date is before today |

## Data Processing (TASK 5B)

**WHAT** and **HOW** the processing will need to be done

Specify the processing that needs to be done and provide algorithm(s)/formulae to show how the processing will be done.

You need to **list** a total of **8** processes in your program (You need to write **algorithms** for **4** of these processes. Ensure that the FOUR algorithms you write have **significant** processing. Reading from the text files, string handling, your array manipulation planned earlier, or Database manipulation using Delphi code, should provide you with the proper algorithms to write.



<https://youtu.be/oakEThuYQ6Y>

**NOTE:** (SQLs are NOT considered as algorithms).

PROCESSING

|  |  |
| --- | --- |
| **What processing needs to be done** | **How processing will be done** |
| *<List AT LEAST EIGHT processes (****brief explanation) over 2 (or more) IPO tables>*** | *<Give Pseudo code or example code/algorithm of FOUR out of the EIGHT processes listed>* |
| Generate new license (creates a new record in tblLicenses) | 1. Get selected province and license type from combo boxes.  2. Query database for the most recent license for the selected province.  3. If a record exists:  Extract digits from LicenseID.  Convert digits to integer and increment by 1.  Else:  Set starting number to 1.  4. Based on province, construct a formatted LicenseID using prefix/suffix and incremented number.  5. Insert new license into tblLicenses with generated LicenseID, selected LicenseType, current date, and expiration one year from today.  6. Refresh licenses on screen. |
| Renew License | 1. Show input box asking user to confirm renewal.  2. If user confirms:  a. Set new expiration date to current date + 1 year.  b. Execute UPDATE statement on tblLicenses to update ExpirationDate where LicenseID matches.  c. Show success message.  d. Reload licenses.  3. Else: Exit. |
| Sign up | 1. Validate that password and confirm password fields are filled and match.  2. Check if username already exists using helper function.  3. If not:  a. Validate all required fields: username, names, email, contact, DOB, user type.  b. Check email contains "@" and ".", and contact is 10 digits.  c. Generate random unique user ID using initials and random number.  d. Ensure generated ID is not already in tblUsers.  e. Insert new user record into tblUsers with all values.  f. Show success message and redirect to login screen.  4. Else:  Show appropriate error messages. |
| Account deletion | 1. Query tblFines to check if any fines exist for the given LicenseID.  2. If record count is 0:  a. Ask user for confirmation.  b. If confirmed:  i. Delete license record from tblLicenses where LicenseID matches.  ii. Show deletion message and reload licenses.  3. Else:  Show error message preventing deletion due to existing fines. |
| Loading Licenses |  **WHAT:** Loads the user’s licenses (max 2) and updates the visual blocks and labels.   **HOW:** Queries DB, checks record count, then loads details and color codes blocks based on expiration. |
| Google Maps Routing |  **WHAT:** Opens Google Maps with coordinates from the DB.   **HOW:** After selecting a station, coordinates are extracted and passed into a ShellExecute call with the Google Maps URL. |
| Location Array Manipulation |  **WHAT:** Updates the UI image and name depending on arrow button clicked.   **HOW:** arrLocs and arrLocImgs arrays are used with an index that wraps around based on arrow direction. Visual feedback is shown by loading different image assets. |
| License ID Parsing and Number Extraction |  **WHAT:** Extracts numeric characters from the latest license ID (e.g., "CA 54 GP") to determine the next available license number.   **HOW:** Uses a for loop to iterate over each character in the LicenseID string and appends digits to a number string, which is then converted to an integer and incremented. |



<https://youtu.be/oakEThuYQ6Y>

## Data Output (TASK 5C)

Provide a clear description to indicate the output requirements of the system for at least TWO of the main interfaces.

You need to describe **all** the output of **TWO** of your screens / Tab Sheetswith **SIGNIFICANT** output. Create a table as below to describe 2 of your screens / your Tab Sheets.

**NOTE**: when you describe the output in the DB grid, make sure you add the field names and format of the output of everything that will be displayed.

**Screen: Fines Screen**

|  |  |  |
| --- | --- | --- |
| Output | Format (type, size) | Output Component |
| Fine Cost | You owe: <Currency> (e.g., R125.00), with two decimal places | Label |
| Fine Summary | Displayed as: "You (License Number: X) owe: RXXX.XX for "<FineType>" due by <Date>" | RichEdit |
| Fine Due Date | Short date format (dd/mm/yyyy) | Label |
| License Number | String format (e.g., CA 123 GP) | Label |

**Screen: License Generation Screen**

|  |  |  |
| --- | --- | --- |
| Output | Format (type, size) | Output Component |
| License ID (Slot 1 & 2) | String format (e.g., CA 125 GP) | Label (lblID1, lblID2) |
| Expiration Date | Short date format (dd/mm/yyyy) | Label (lblExp1, lblExp2) |
| Block Colour Indicator | Image path: "RedRec.png" for expired, "GreenRec.png" for valid licenses | Image (imgBlock1, imgBlock2) |
| Max Licenses Warning | String message popup if user exceeds 2 licenses | ShowMessage dialog |
| License Type Selected | Dropdown selection | ComboBox (cmbLicenseType) |
| Province Selected | Dropdown selection (e.g., Gauteng, Western Cape) | ComboBox (cmbProvince) |

**Resources**:

* Text to Speech by Mr Long: <https://youtu.be/TqCZUioQDeg>
* Add a video by Mr Long: <https://youtu.be/lbPIwgzhWCE>
* All QR codes extracted from: ***PAT 2023*** by Lilané le Grange & Karen Andersen
* <http://www.delphibasics.co.uk/>
* <http://festra.com/>