Functional Documentation

I.Backend Features:

1. Authentication

- Login:
    - Verifies the user’s email and password using identities that are saved in the user’s table.
    - Uses (bcrypt) to securely hash and compare passwords.
    - Stores user sessions upon successful login.
    - Redirects back to the authentication page if identities are invalid.
- Registration:
    - Allows new users to create accounts.
    - Full names , email , passwords are required.
    - Hashes passwords using (bcrypt)before storing them in the user’s table.
    - Checks if the email already exists in the database.
    - Logs in the user automatically after successful registration.

2. Car Management

- View All Cars:
    - Fetches all cars marked as `AvailabilityStatus = TRUE` from the Cars table.
    - Requires the user to be logged in.
    - Displays a list of available cars.
- Search Cars:
    - Allows users to search for cars by make or model using a search query.
    - Displays cars matching the query and their availability.

Routes and Functionalities

Route	Method	Description
/	GET	Renders the homepage (index.html).
/auth	GET, POST	Renders the authentication page (auth.html).

/login	POST	Handles user login, validates identities and redirects based on success or failure.
/register	POST	Handles user registration. hashes the password and adds user details to the Users table.
/view-more	GET	Retrieves all available cars and displays them. requires user login.
/cars	GET, POST	Displays cars based on availability or search query. Requires user login.

Database Interaction

Tables:

1. Users Table:
    - Stores user details such as `Name`, `Email`, and hashed `Password`.
2. Cars Table:
    - Stores car information including `Make`, `Model`, `AvailabilityStatus`, and other car details.

Queries:

Retrieve user details by email during login:

(SELECT * FROM Users WHERE Email = %s)

Insert new user during registration:

(INSERT INTO Users (Name, Email, Password) VALUES (%s, %s, %s))

Retrieve available cars:

(SELECT * FROM Cars WHERE AvailabilityStatus = TRUE)

Search cars by make or model:

(SELECT * FROM Cars WHERE (Make LIKE %s OR Model LIKE %s) AND AvailabilityStatus = TRUE)

Error Handling

- If a database query fails, errors are logged and the user is notified with a generic message.
- For authentication errors, appropriate messages are flashed (*Invalid password*, *No user found*).

Security Features

- Passwords are hashed using (bcrypt) before storage and validated securely during login.
- Sessions are managed using Flask-Session with a secret key for encryption.
- Redirects non-logged-in users attempting to access restricted routes like /view-more or /cars.

II. Frontend Features:

Observations and Interactions

- **Data Flow:**
  - index.html sends search queries to the cars.html page using the search form.
  - auth.html manages user sessions, ensuring users can log in or register before accessing features like renting cars.
- **Styling and User Experience:**
  - Common font (Nunito, Open Sans) and styling theme across all pages.
  - Consistent branding with a logo, navigation links, and action buttons.
- **Dynamic Elements:**
  - Backend integration required for routes (url_for('auth'), url_for('cars')).
  - Backend needs to populate cars dynamically in cars.html and handle user sessions from auth.html.

Pages and Their Functionalities

A. index.html - Homepage

- **Purpose**: Introduces the service and offers search functionality.
- **Key Functionalities**:
    1. **Hero Section**:
        - Display a title and subtitle ("The easy way to take over a ride").
        - Includes a Search Form:
            - Input field for car, model, or brand.
            - Button to submit the query to the /cars route (via url_for('cars')).
    2. **Navigation Bar**:
        - Links to:
            - Home (#home).
            - Explore Cars (#featured-car).
            - About Us (placeholder).
            - Blog (placeholder).
            - Login/Register (url_for('auth')).
    3. **Responsive Design**:
        - Mobile-friendly navigation toggle button.
        - Icons and header adjustments for different screen sizes.
    4. **Actions**:
        - Button to explore cars (#featured-car).
        - Profile icon for user interaction (placeholder for future use).

B. auth.html - Authentication Page

- **Purpose**: Enable users to log in or register.
- **Key Functionalities**:
    1. **Login Form**:
        - Fields:
            - Email.
            - Password.
        - Submit button (POST request to /login).
    2. **Register Form**:
        - Fields:
            - Full Name.
            - Email.
            - Password.
        - Submit button (POST request to /register).
    3. **Validation**:
        - required attribute ensures fields are filled before submission.

4. **Styling**:
    - Both forms are styled for easy distinction.
    - Centralized layout with responsive design.
5. **Redirection**:
    - Users are redirected to homepage after successful login or registration.

C. cars.html - Cars Listing Page

- **Purpose**: Display a list of available cars with detailed features and options for renting.
- **Key Functionalities**:
    1. **Featured Cars**:
        - Each car is displayed as a card with:
            - Image.
            - Title and year.
            - Key features:
                - Seating capacity.
                - Fuel type.
                - Mileage.
                - Transmission type.
            - Price per day.
        - Action buttons:
            - Add to Favorites (icon with heart-outline).
            - Rent Now (button for proceeding to rent).
    2. **Styling**:
        - Each card is visually appealing with a consistent design.
        - Images are optimized with lazy loading (loading="lazy").
    3. **Expandable List**:
        - Placeholder to "View more" cars via a button or link.
    4. **Navigation**:
        - Header links for consistent site navigation.

III. Cross Platform :

For our car rental website, the cross-platform features are achieved through:

1. Responsive Design (Frontend)

Technologies Used: HTML5, CSS3, Bootstrap.

These tools ensure that the layout and design automatically adapt to various screen sizes and resolutions.

Outcome: the website will look and work well on devices running Windows, macOS, Linux, Android, iOS, etc.

2. Browser Compatibility

Supported Browsers: Google Chrome , Mozilla Firefox , Safari , Microsoft Edge

Ensured by: Writing standards-compliant code. Using CSS and JavaScript features that are widely supported.

3. Backend Interoperability

Backend Technologies: SQL for the database

Framework : (Flask) runs on any server supporting Python or Node.js.

Outcome: the backend can be hosted on various platforms, such as Windows Server, Linux or cloud services like AWS, Google Cloud, or Azure.

4. Cross-Platform Hosting

The website can be deployed on any platform that supports web hosting, including: Shared Hosting, Virtual Private Servers (VPS) , Cloud Platforms (AWS, Heroku, Azure)

5. Testing and Optimization: To ensure cross-platform compatibility, test the website on: Multiple devices (Android phones, iPhones, tablets, PCs). Various browsers (listed above). Available for older versions of devices or browsers.