Below is a **technical documentation / project specification** written in **English**, based strictly on your requirements. This can be used as a **README.md** or **Technical Test Documentation**.

Tablelink Technical Test Documentation

1. Overview

This project is a **Laravel 12** web application that implements **user authentication, authorization, and management**, following **REST API standards**, **MVC architecture**, and **Dockerized deployment**.

The application supports two roles:

- **Admin**
- **User**

Admins have access to dashboards and user management features, while normal users can only access a basic dashboard.

---

2. Technology Stack

- **Backend Framework**: Laravel 12
- **Database**: MySQL (via Docker)
- **Authentication**: Laravel Authentication (session-based or token-based)
- **Authorization**: Role-based access control (Admin, User)
- **Charts**: Chart.js (based on provided Gist references)
- **Containerization**: Docker & Docker Compose
- **Testing**: PHPUnit (Unit Tests)
- **Architecture**: MVC (Model-View-Controller)
- **API Standard**: RESTful API

---

3. User Data Model

User Table Fields

Field Name	Type	Description
name	string	User full name
email	string	Unique user email
password	string	Hashed password
created_at	timestamp	Record creation time

Field Name	Type	Description
updated_at	timestamp	Record update time
last_login	timestamp	Last successful login
deleted_at	timestamp	Soft delete timestamp

Notes

- `email` must be **unique**
- `deleted_at` is used for **soft deletes**
- Passwords are stored using Laravel hashing

---

4. Authentication & Authorization

4.1 Registration

- Users can register using **email and password**
- Email must be **unique**
- Default role assigned: **User**

4.2 Authentication

- Both **Admin** and **User** can log in
- On successful login:
  - `last_login` is updated

4.3 Authorization

- **User**
  - Redirected to an empty dashboard
- **Admin**
  - Access to:
    - Dashboard with charts
    - User management features

Authorization is enforced using:

- Middleware
- Policies or Gates

---

5. User Management (Admin Only)

5.1 Read Users

- Endpoint returns paginated user data
- Pagination: **10 users per page**

- Soft-deleted users are excluded by default

5.2 Update User

- Admin can update user data
- Email validation:
  - Must be unique
  - Current user's email is excluded from uniqueness check

5.3 Delete User

- Admin-only access
- Uses **soft delete**
- Sets `deleted_at` timestamp instead of permanently removing data

---

6. Dashboard (Admin)

The Admin Dashboard displays the following charts:

6.1 Line Chart

- Source reference: https://gist.github.com/rachmanlatif/323bd55b284774bf98e11225ce2374e1

6.2 Vertical Bar Chart

- Source reference: https://gist.github.com/rachmanlatif/51277a2070e6cd240bf471d9aead29d7

6.3 Pie Chart

- Source reference: https://gist.github.com/rachmanlatif/ad0290b004c1bfa9ded5f872f680fea8

Notes

- Charts are modularized into reusable view components
- Data is provided via REST API endpoints

---

6. Flight Infomation (Admin)

The Flight Information page collects flight ticket information from **Tiket.com** based on specific search criteria:

Notes

- Target: https://www.tiket.com
- Search Criteria:
  - Search for **one-way flight tickets**.
  - Depature city: Jakarta (CGK)

  - Destination city: Bali (DPS)
  - Departure time: Before 5:00 PM (17:00 local time)
  - Flight type: Economy class
  - Trip type: One-way
- Data to Collect:
  - Airline name
  - Flight number
  - Departure time
  - Price
  - Departure airport
  - Arrival airport
- Output Requirement:
  - Data is provided via REST API endpoints.
  - At Flight Information page, make as data-table.

---

7. Project Structure (MVC Best Practices)

Best Practices Applied

- Request validation separated using **Form Request**
- Business logic kept inside services or controllers
- Views split into reusable components
- Clean separation between API and UI logic

---

8. Dockerization

Dockerized Components

- Laravel Application
- Database (MySQL / PostgreSQL)
- Web Server (Nginx)

Docker Compose

- Single `docker-compose.yml` file
- Environment variables managed via `.env`
- One command setup:

```
docker-compose up -d
```

---

9. Testing

Unit Tests

- Each major function has corresponding unit tests:

- Authentication
- Authorization
- User CRUD
- Dashboard data generation

Testing Tools

- PHPUnit
- Laravel testing utilities

Run Tests

```
php artisan test
```

---

10. Security Considerations

- Password hashing using Laravel Hash
- Role-based middleware
- Soft delete to prevent accidental data loss
- Validation on all incoming requests
