**IMS-App**

*Developed by IT482 (Steve Quinn, Chris Hunter, Tim Guerin)*

**Table of Contents**

Introduction

*1.1 Purpose*

*1.2 Features*

*1.3 Technology Stack*

Installation

*2.1 Prerequisites*

*2.2 Setup*

User Guide

*3.1 Account Creation and Login*

*3.2 Dashboard*

*3.3 Barcode Scanning*

*3.4 Reorder Notification*

*3.5 Trending Analysis and Forecasting*

*3.6 User Administration*

AWS Integration

*4.1 Database Configuration*

*4.2 Storage and Security*

Troubleshooting

*5.1 Backend Issues (Laravel)*

*5.2 Frontend Issues (Angular)*

*5.3 AWS Services*

A Message From the Developers

**1. Introduction**

**1.1 Background and Purpose**

The Laravel-based Angular web application for inventory management is designed to provide first responders with an intuitive and efficient solution to track, analyze, and manage inventory levels. This documentation serves as a comprehensive guide for users, administrators, and developers.

The primary objective of the Inventory Management System is to offer first responders a robust platform for overseeing their equipment and supplies. The system aims to streamline the inventory management process, enhance decision-making through data analysis, and ensure the timely replenishment of essential items.

**1.2 Features**

*User Authentication:*

Users can create accounts securely, and the system employs industry-standard authentication practices to protect sensitive information.

*Dashboard:*

The intuitive dashboard provides an overview of current inventory levels through visually appealing graphs and charts. This feature enables users to quickly assess the status of their resources.

*Barcode Scanning:*

The application supports barcode scanning functionality, allowing users to expedite inventory management by triggering reorder notifications with a simple scan. This feature enhances efficiency, reducing the time required to identify and address low-stock situations.

*Trending Analysis:*

The trending analysis section empowers users to delve into historical data, identifying patterns and trends in inventory usage. Additionally, the system incorporates forecasting algorithms to predict future inventory needs, facilitating proactive decision-making.

*User Administration:*

Administrators have the authority to manage user roles and permissions. This feature ensures that different team members have appropriate levels of access, maintaining data integrity and security.

**1.3 Technology Stack**

The technology stack combines the power of Angular for the frontend, Laravel for the backend, and AWS services for database storage and deployment. This robust stack ensures scalability, security, and optimal performance.

**2. Installation**

**2.1 Prerequisites**

Before initiating the installation process, ensure that PHP and Composer are installed for Laravel, and Node.js along with npm for Angular. Additionally, set up an AWS account to leverage cloud services for database storage and deployment.

**2.2 Setup**

* *Clone the Repository:*
  + Execute the following command: gh repo clone squinn8810/ims-app
* *Install Laravel Dependencies:*
  + Navigate to the Laravel project directory and run: composer install
* *Configure Laravel Environment Variables:*
  + Set up the necessary environment variables, including database connection details.
* *Install Angular Dependencies:*
  + Navigate to the Angular project directory and run: npm install
* *Build Angular Project:*
  + Generate the Angular production build: ng build --prod
* *Configure AWS Services:*
  + Utilize AWS RDS for database configuration, ensuring Laravel is set up to connect to the AWS database.
  + Deploy the application to AWS using appropriate services such as Elastic Beanstalk or EC2.

**3. User Guide**

**3.1 Account Creation and Login**

To get started with the Inventory Management System:

* *Visit the Application URL:*
  + Open your web browser and enter the URL provided.
* *Register an Account:*
  + Click on the "Register" button.
  + Complete the registration form with the required details.
* *Login:*
  + Use your registered credentials to log in securely.

**3.2 Dashboard**

Upon successful login, users can navigate to the comprehensive dashboard:

* *Real-Time Inventory Levels:*
  + The dashboard displays real-time inventory levels using intuitive charts and graphs.
* *Visual Representation:*
  + Graphical elements provide a quick and visually appealing overview of inventory status.

**3.3 Barcode Scanning**

The Barcode Scanning feature simplifies inventory management:

* *Access from Dashboard:*
  + Navigate to the designated section from the dashboard.
* *Scan Barcodes:*
  + Utilize a device camera to scan barcodes on items.
* *Reorder Trigger:*
  + Efficiently trigger reorder notifications based on scanned data.

**3.4 Reorder Notification**

Receive timely notifications for low inventory levels:

* *Notification Prompt:*
  + Instant notifications alert users about low-stock situations.
  + Identifies items that need to be reordered as well as vendor information.
* *Reorder Confirmation:*
  + Confirm reorders directly from the notification interface.

**3.5 Trending Analysis and Forecasting**

Explore historical data and forecast future needs:

* *Analysis Section:*
  + Access the dedicated section for trending analysis.
* *Historical Trends:*
  + View and analyze historical trends in inventory usage.
* *Forecasting:*
  + Leverage forecasting algorithms for future inventory needs.

**3.6 User Administration**

Efficiently manage user roles and permissions:

* *Admin Privileges:*
  + Administrators can assign and modify user roles.
* *Access Levels:*
  + Ensure that different team members have appropriate levels of access for data security.

**4. AWS Integration**

**4.1 Database Configuration**

Setting up RDS for Laravel

* *Create an RDS Instance:*
  + Log in to the AWS Management Console.
  + Navigate to Amazon RDS and create a new database instance.
  + Configure the instance details, including database engine, credentials, and security group settings.
* *Database Configuration in Laravel:*
  + Update Laravel's database configuration (usually found in the .env file) with RDS connection details.
  + Specify the RDS endpoint, database name, username, and password.

|  |
| --- |
| DB\_CONNECTION=aws  DB\_HOST=ims-app.c0oaeoxlqvyj.us-east-2.rds.amazonaws.com  DB\_PORT=3306  DB\_DATABASE=ims-app  DB\_USERNAME=admin  DB\_PASSWORD=it482imsapp |

**4.2 Storage and Security**

AWS S3 for Storage

* *Create an S3 Bucket:*
  + In the AWS Management Console, navigate to Amazon S3 and create a new bucket.
  + Configure bucket settings, including region and access control.
* *Laravel Configuration for S3:*
  + Update Laravel's filesystem configuration (config/filesystems.php) to use the S3 driver.

|  |
| --- |
| 'disks' => [  's3' => [  'driver' => 's3',  'key' => env('AWS\_ACCESS\_KEY\_ID'),  'secret' => env('AWS\_SECRET\_ACCESS\_KEY'),  'region' => env('AWS\_DEFAULT\_REGION'),  'bucket' => env('AWS\_BUCKET'),  'url' => env('AWS\_URL'),  'endpoint' => env('AWS\_ENDPOINT'),  'use\_path\_style\_endpoint' => env('AWS\_USE\_PATH\_STYLE\_ENDPOINT', false),  'throw' => false,  ],  ], |

* *Security with IAM Roles:*
  + Create an IAM role with the necessary permissions for accessing RDS and S3.
  + Attach the IAM role to the EC2 instance running Laravel for secure access.

**5. Troubleshooting**

**5.1 Backend Issues (Laravel)**

Error Logs

* *Location:*
  + Laravel logs errors to the storage/logs directory.
* *Access:*
  + Check for error messages in the laravel.log file.
  + Tail the logs in real-time: tail -f storage/logs/laravel.log
* *Common Issues:*
  + Database connection errors: Verify credentials in the .env file.
  + Incorrect configuration: Double-check environment variables.

**5.2 Frontend Issues (Angular)**

Browser Console

* *Access:*
  + Open the browser's developer tools (usually by pressing F12).
  + Navigate to the "Console" tab.
* *Common Issues:*
  + Missing assets: Ensure correct paths in Angular configuration.
  + Network errors: Check API requests for errors.
  + Syntax errors: Review JavaScript errors in the console.

**5.3 AWS Services**

CloudWatch

* *Logs:*
  + Check AWS CloudWatch logs for Lambda functions, if applicable.

RDS and S3

* *AWS Console:*
  + Review RDS and S3 dashboards for any reported issues.
* *Security Groups:*
  + Confirm that security groups allow necessary traffic.

**6. A Message from the Developers**

This documentation provides a comprehensive guide for users and administrators of the Laravel-based Angular web application for inventory management for first responders. For further assistance, refer to the troubleshooting section or contact the development team.

The Laravel-based Angular web application for inventory management represents a significant advancement in the operational capabilities of first responders. By combining the strengths of Laravel, Angular, and AWS services, the system provides a reliable and scalable solution for addressing the unique challenges faced in managing critical resources.

We extend our gratitude to the developers, administrators, and first responders who contribute to the success of this system. As the needs of the users evolve, this application stands ready to adapt and support those on the front lines, ensuring they have the resources necessary to carry out their vital missions.

Thank you for choosing IMS-App Inventory Management System, and we remain committed to supporting and enhancing this application for the benefit of first responders worldwide.

*Note: Ensure that the documentation is updated regularly to reflect any changes or updates to the application*