Yangon Real Estate Price Predictor

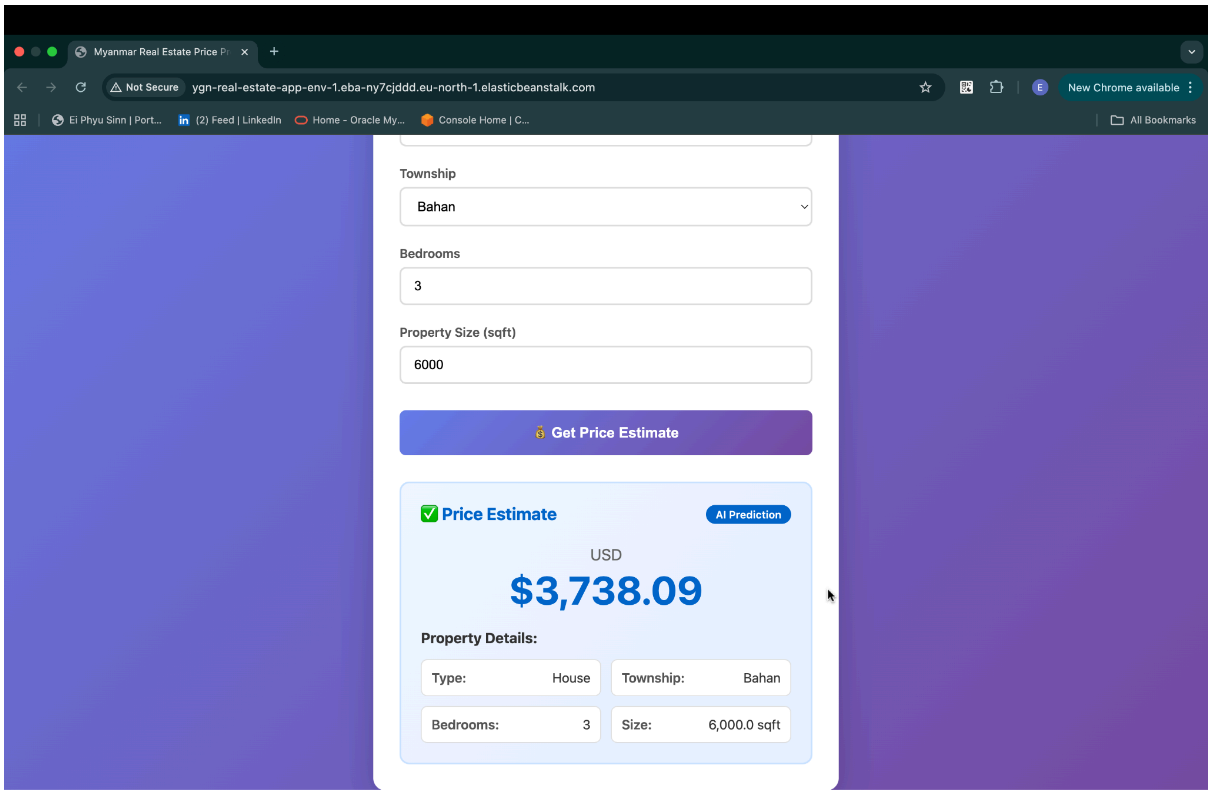
Author	Github	Project Name	Date
Ei Phyu Sinn	Data_science_end_to_end	Yangon Real Estate Price Predictor	15 Dec 2025

Project Overview

This project collects and analyzes rental property data in Yangon, Myanmar. It scrapes rental listings using Playwright, performs data cleaning and exploratory analysis, trains multiple machine learning models to predict rental prices, and deploys a full-stack web application using React, Flask, Docker, and AWS Elastic Beanstalk.

Key steps:

1. Scraping: Collect rental listings (price, size, type, bedrooms, township).
2. EDA: Clean data, analyze trends, and visualize relationships.
3. Modeling: Train and evaluate regression models; Random Forest gives the best performance.
4. Deployment: Containerize frontend and backend with Docker and deploy on AWS.


Township

Bahan

Bedrooms

3

Property Size (sqft)

6000

✅ Price Estimate

USD

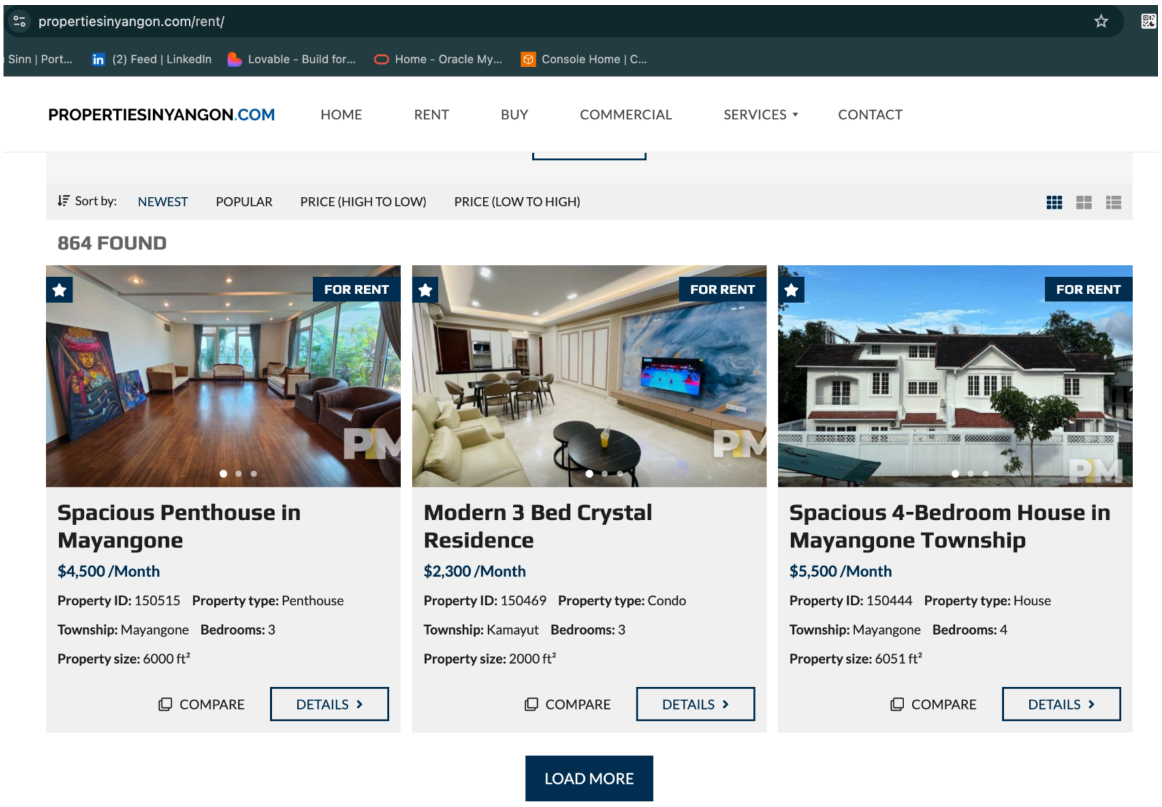
$3,738.09

Property Details:

Type:	House	Township:	Bahan
Bedrooms:	3	Size:	6,000.0 sqft

1 — Scraping Yangon(Myanmar) Rental Data with Playwright

- I used the website **propertiesinyangon.com/rent** to scrape rental prices in Yangon across different townships.
- I used **Playwright** as the scraping tool because it can load dynamic websites, scroll automatically, and handle pages that use JavaScript.
- First, I checked robots.txt and navigated to the Rent page because I wanted to scrape rental listings.
- Then I scrolled through the page to load all property items and extracted the information (price, property ID, type, bedroom count, size, etc.).
- It takes about 1 hour to scrape all data.
- Finally, I saved all the scraped data into a JSON file.


propertiesinyangon.com/rent/

PROPERTIESINYANGON.COM HOME RENT BUY COMMERCIAL SERVICES CONTACT

Sort by: NEWEST POPULAR PRICE (HIGH TO LOW) PRICE (LOW TO HIGH)

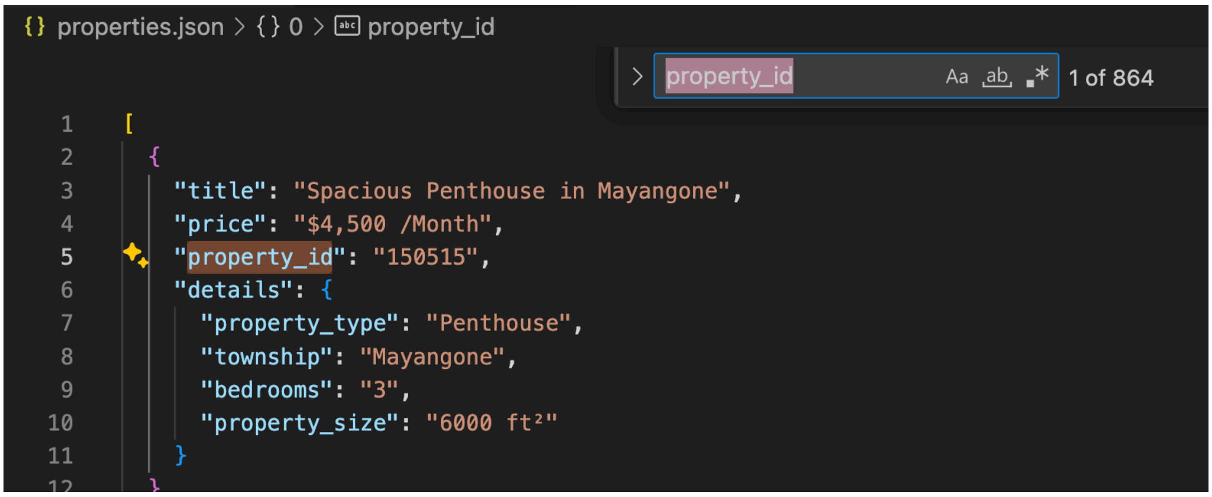
864 FOUND

FOR RENT
Spacious Penthouse in Mayangone
$4,500 /Month
Property ID: 150515 Property type: Penthouse
Township: Mayangone Bedrooms: 3
Property size: 6000 ft²
COMPARE DETAILS

FOR RENT
Modern 3 Bed Crystal Residence
$2,300 /Month
Property ID: 150469 Property type: Condo
Township: Kamayut Bedrooms: 3
Property size: 2000 ft²
COMPARE DETAILS

FOR RENT
Spacious 4-Bedroom House in Mayangone Township
$5,500 /Month
Property ID: 150444 Property type: House
Township: Mayangone Bedrooms: 4
Property size: 6051 ft²
COMPARE DETAILS

LOAD MORE


properties.json > {} 0 > property_id

property_id 1 of 864

```
[
  {
    "title": "Spacious Penthouse in Mayangone",
    "price": "$4,500 /Month",
    "property_id": "150515",
    "details": {
      "property_type": "Penthouse",
      "township": "Mayangone",
      "bedrooms": "3",
      "property_size": "6000 ft²"
    }
  }
```

2 — EDA (Exploratory Data Analysis)

1. Data Loading and Cleaning

- Loaded JSON data from previous scraped data.
- Focused only on relevant columns: property type, township, bedrooms, property size, and price.
- Handled missing values:

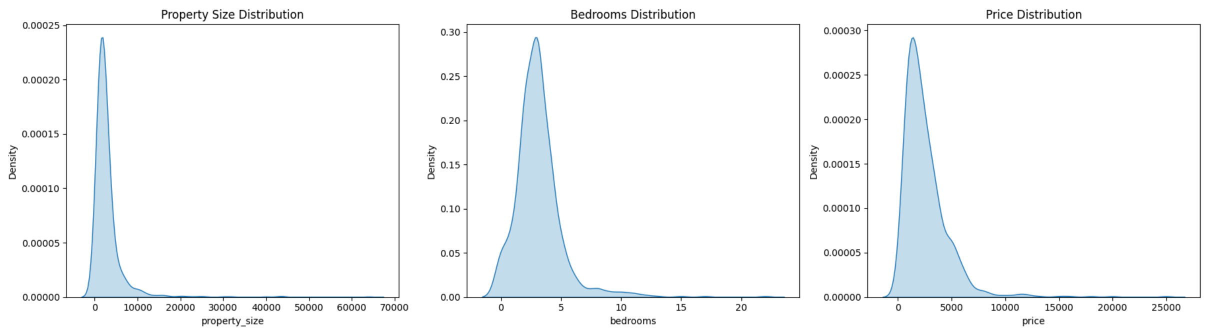
    - Dropped rows without property size (essential for analysis).
    - Filled missing bedroom values with 0.

2. Data Transformation

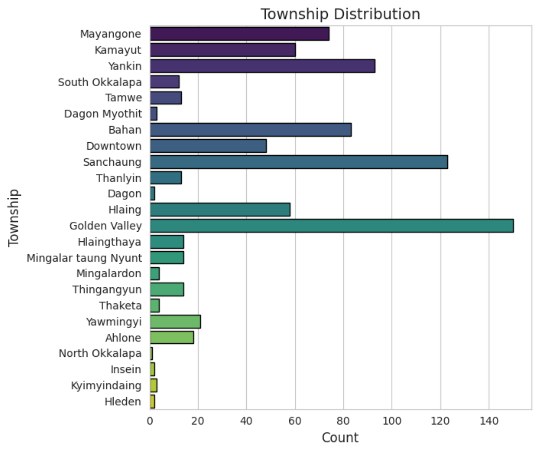
- Property size:
  - Removed "ft²"
  - For ranges like "400 to 710 ft²", replaced with the average.
- Price:
  - Removed rows where price was "Contact for price" or contained "for sale".
  - Cleaned text by removing $, ,, /Month.
  - For price ranges like $2,000 - $3,700 /Month, replaced with the average.
- Property type:
  - Standardized names, e.g., "Condo, Penthouse" → "Penthouse".
  - Removed rare categories (less than 10 occurrences).
- Township:
  - Removed rare townships (less than 10 occurrences) to reduce class imbalance.

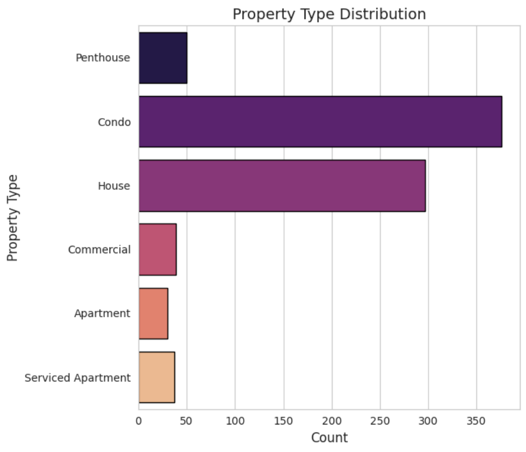
3. Data Exploration

- **Numerical Columns** (property_size, bedrooms, price):
  - Plotted KDE plots to visualize distribution and detect skewness/outliers.



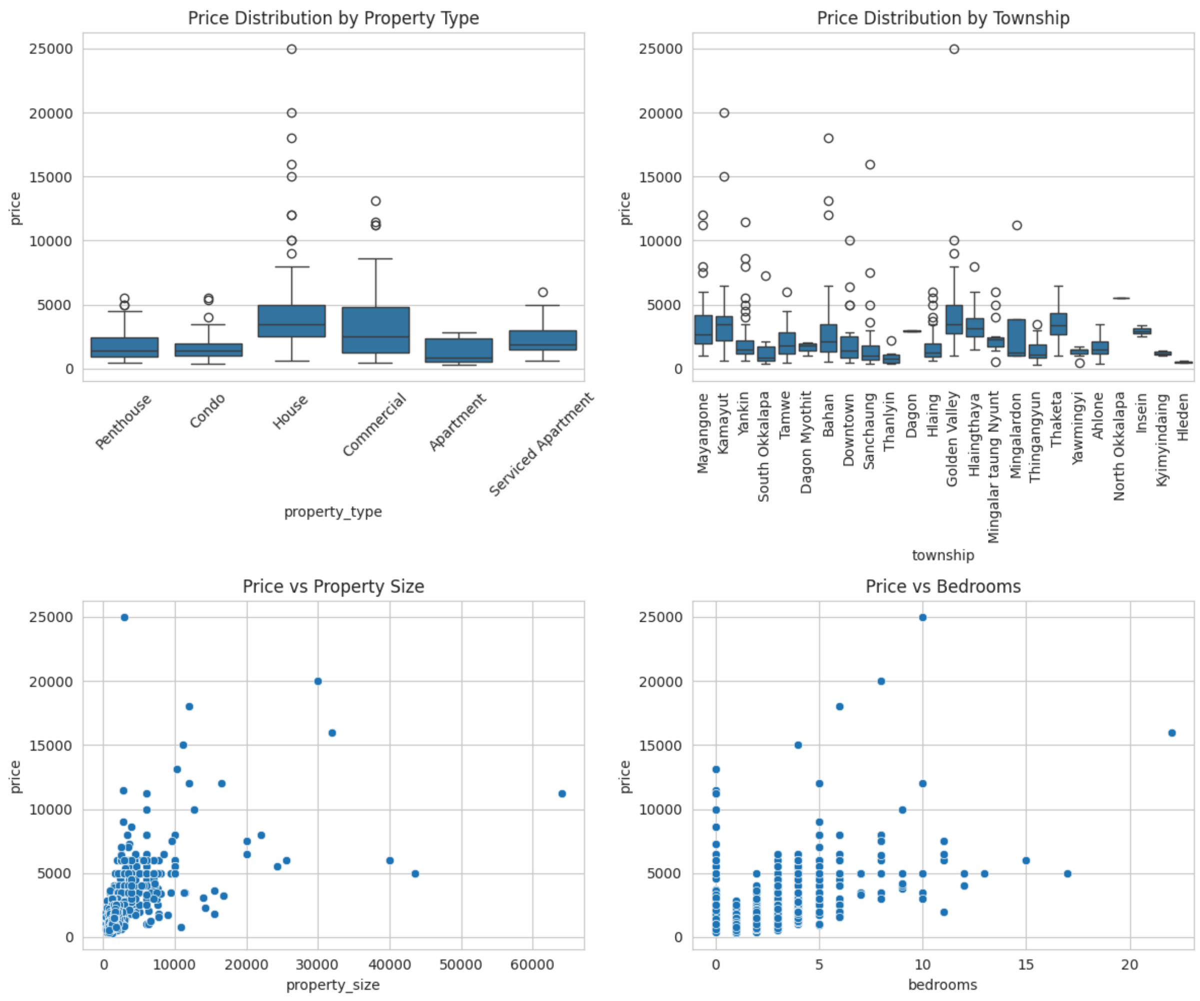
- **Categorical Columns** (township, property_type):
  - Used countplots to see distribution and class imbalance.



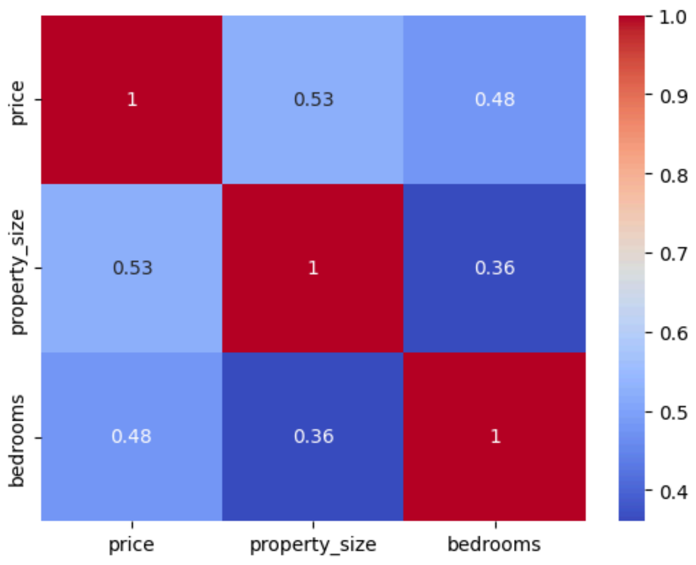


4. Relationship Analysis

- Explored how numerical variables relate to price:
  - Boxplots:
    - property_type vs price
    - township vs price
  - Scatterplots:
    - property_size vs price
    - bedrooms vs price



- Calculated correlation between numeric columns:
  - price, property_size, bedrooms to understand linear relationships.



3 — Model Training

I first loaded the processed CSV data, removed outliers in price (top and bottom 5%), separated the target variable y = price and features X, and applied a log-transform to the price for better scaling.

For feature preprocessing, numerical columns were scaled using StandardScaler and categorical columns were encoded using OneHotEncoder, combined via a ColumnTransformer. The data was then split into 80% training and 20% testing sets.

Multiple models were trained and evaluated, including below models

- LinearRegression
- Ridge
- Lasso
- DecisionTree
- RandomForest
- AdaBoost
- KNN
- SVR,

For evaluation, I used metrics R^2, MAE, MEDAE, and RMSE.
Among these, the **Random Forest Regressor** gave the best performance with R2: 0.57, MAE: 0.28, RMSE: 0.35

4 — Docker Setup

The project is fully containerized using Docker Compose, providing a consistent development and production environment. This setup runs both the frontend React application and backend Flask API as interconnected services.

Architecture Overview

- Backend Service: Flask API served with Gunicorn WSGI server for production-grade performance
- Frontend Service: React application built with production-optimized Nginx server
- Networking: Docker Compose creates a shared network allowing seamless communication between frontend and backend containers

Prerequisites

- Ensure Docker and Docker Compose are installed on your system:
- Docker Desktop (Windows/Mac) or Docker Engine (Linux)
- Docker Compose V2 or higher

Complete Setup Instructions

1. Build Docker Images

```
docker-compose build
```

This command:

- Builds the Flask backend image with Python dependencies
- Builds the React frontend image with Node.js dependencies
- Caches layers for faster subsequent builds

2. Start Containers

```
docker-compose up
```

For detached mode (running in background):

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

To view logs:

```
docker-compose logs -f
```

3. Access the Application

Once containers are running, access the services:

```
Frontend Application: http://localhost:80
Backend API Documentation: http://localhost:5000/api
Backend Health Check: http://localhost:5000/health
```

This Docker setup ensures consistent behavior across development, testing, and production environments, with all dependencies isolated and version-controlled.

5 — Deployment

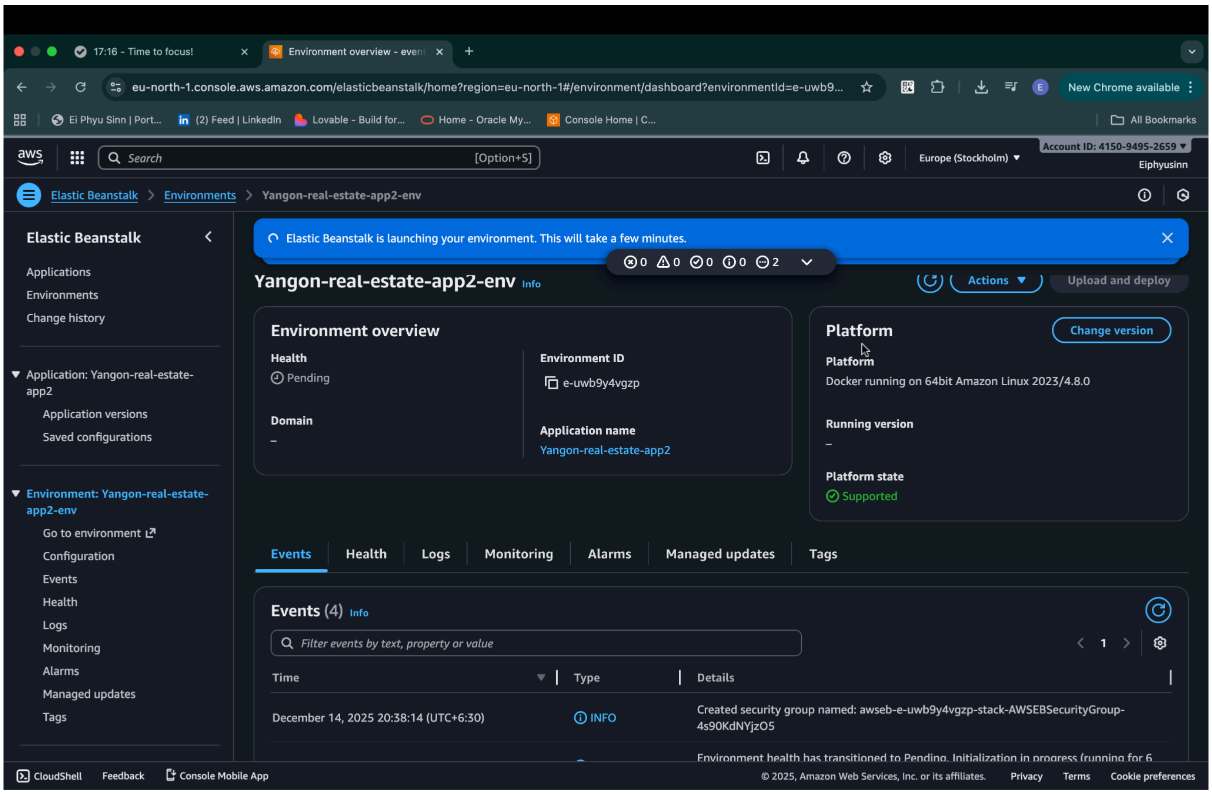
The application is deployed on AWS Elastic Beanstalk using a multi-container Docker platform, providing automated scaling, load balancing, and monitoring capabilities.

Deployment Architecture

- Platform: AWS Elastic Beanstalk Multi-Container Docker
- Load Balancer: Application Load Balancer with health checks
- Frontend: React app served via Nginx (port 80)
- Backend: Flask API served via Gunicorn (port 5000)
- Database: AWS RDS PostgreSQL instance

Deployment Process

- Build production Docker images
- Push to AWS Elastic Container Registry (ECR)
- Configure Elastic Beanstalk environment
- Deploy using Docker-compose.yml
- Configure environment variables and scaling policies



eu-north-1.console.aws.amazon.com/elasticbeanstalk/home?region=eu-north-1#/environment/dashboard?environmentId=e-uwb9...

Elastic Beanstalk > Environments > Yangon-real-estate-app2-env

Elastic Beanstalk is launching your environment. This will take a few minutes.

Yangon-real-estate-app2-env

Environment overview

Health: Pending

Environment ID: e-uwb9y4vgzp

Domain: –

Application name: Yangon-real-estate-app2

Platform

Platform: Docker running on 64bit Amazon Linux 2023/4.8.0

Running version: –

Platform state: Supported

Events (4)

Time	Type	Details
December 14, 2025 20:38:14 (UTC+6:30)	INFO	Created security group named: awseb-e-uwb9y4vgzp-stack-AWSEBSecurityGroup-4s90KdNYjzO5
		Environment health has transitioned to Pending. Initialization in progress (running for 6

Access Points

- Monitoring: AWS CloudWatch dashboard
- Logs: Accessible via Elastic Beanstalk console or AWS CLI

This deployment strategy ensures high availability, automatic scaling, and seamless updates with zero-downtime deployments using Elastic Beanstalk's blue-green deployment feature.

This comprehensive documentation provides complete instructions for local development with Docker and details about the production deployment setup. The containerized approach ensures consistency across all environments and simplifies the deployment process.