Fullstack Product Walkthrough

# 1. Project Setup

* Create a new Ruby on Rails project.
* Set up Git repository and push to GitHub.

# 2. Development Environment

* Use Dockerfile and docker-compose.yml to containerize the app.
* Install required gems including RSpec, Capybara, Searchkick, cassandra-driver, and Redis gems.

# 3. Database Configuration

* Configure PostgreSQL or MySQL in config/database.yml.
* Set up Redis for session, cache, and queue handling.
* Integrate ScyllaDB using the Cassandra driver.

# 4. Testing Setup

* Set up RSpec for unit testing.
* Use Capybara for integration and system tests.
* Write and validate test cases.

# 5. CI/CD Pipelines

* Configure .circleci/config.yml for CircleCI pipelines.
* Configure .github/workflows/ci.yml for GitHub Actions.
* Ensure all tests run and Docker image builds.

# 6. Search Integration

* Integrate ElasticSearch using Searchkick.
* Configure indexing for searchable models.

# 7. Kubernetes Deployment

* Create deployment and service YAML files.
* Use Helm charts to manage deployment configurations.
* Deploy app to Kubernetes cluster.

# 8. Monitoring & Observability

* Initialize Sentry in sentry\_initializer.rb for error tracking.
* Deploy DataDog agent in Kubernetes using a DaemonSet.

# 9. Infrastructure Management

* Use Terraform scripts to provision AWS infrastructure.
* Define instances, VPCs, security groups, etc.

# 10. Production Ready

* Ensure all services (DBs, Redis, ScyllaDB, ElasticSearch) are networked.
* Run CI pipelines, deploy with Helm, and monitor with Sentry and DataDog.  
    
    
    
    
   Overview:
* # This is a full-stack Ruby on Rails project that integrates all the listed technologies.
* # It models a powerlifting app where users can input their best lifts, and the app tells them
* # which records by federation they could potentially beat using data from OpenPowerlifting.org.
* # Directory structure
* /
* ├── app/
* │ ├── models/ # Rails models (ActiveRecord)
* │ │ ├── user.rb
* │ │ ├── lift.rb
* │ │ ├── federation\_record.rb
* │ │ └── match\_result.rb
* │ ├── controllers/
* │ │ ├── users\_controller.rb
* │ │ ├── lifts\_controller.rb
* │ │ ├── federation\_records\_controller.rb
* │ │ └── match\_results\_controller.rb
* │ ├── views/
* │ │ ├── users/
* │ │ ├── lifts/
* │ │ ├── federation\_records/
* │ │ └── match\_results/
* │ └── channels/ # For real-time updates via Redis pub/sub
* │ └── lifts\_channel.rb
* ├── public/stylesheets/application.css # UI styling (red and black theme)
* ├── lib/tasks/import\_openpowerlifting.rake # CSV importer
* ├── lib/scylladb\_feed\_logger.rb # ScyllaDB integration for activity feeds
* ├── config/initializers/redis.rb # Redis configuration
* ├── config/initializers/sidekiq.rb # Sidekiq background job setup
* ├── spec/ # RSpec tests with Capybara for integration tests
* ├── config/
* │ ├── database.yml # Config for PostgreSQL & MySQL
* │ ├── elasticsearch.yml # Config for ElasticSearch
* │ └── redis.yml # Redis config if needed
* ├── k8s/ # Kubernetes manifests
* │ ├── helm-chart/ # Helm chart for deployment
* ├── .circleci/config.yml # CircleCI pipeline definition
* ├── .github/workflows/ci.yml # GitHub Actions CI workflow
* ├── terraform/ # Infrastructure as Code for cloud provisioning
* ├── Dockerfile # Container definition
* ├── docker-compose.yml # Local development setup
* └── sentry\_initializer.rb # Sentry error monitoring config
* # Scaffold Commands
* rails g scaffold User name:string email:string password\_digest:string gender:string
* rails g scaffold Lift user:references lift\_type:integer weight:integer
* rails g scaffold FederationRecord federation:string weight\_class:string age\_class:string lift\_type:integer gender:string record\_weight:integer
* rails g scaffold MatchResult user:references federation\_record:references note:string
* # CSS Theme (public/stylesheets/application.css)
* body {
* background-color: #000;
* color: #fff;
* font-family: sans-serif;
* }
* a { color: #e10600; }
* .btn {
* background-color: #e10600;
* border: none;
* padding: 0.5em 1em;
* color: white;
* border-radius: 4px;
* text-decoration: none;
* }
* .card {
* background-color: #111;
* border: 1px solid #e10600;
* border-radius: 8px;
* padding: 1em;
* margin-bottom: 1em;
* }
* # CSV Importer (lib/tasks/import\_openpowerlifting.rake)
* namespace :import do
* desc "Import federation records from OpenPowerlifting CSV"
* task openpowerlifting: :environment do
* require 'csv'
* path = ENV['CSV'] || 'data/records.csv'
* CSV.foreach(path, headers: true) do |row|
* FederationRecord.find\_or\_create\_by(
* federation: row['Federation'],
* weight\_class: row['WeightClass'],
* age\_class: row['AgeClass'],
* lift\_type: row['LiftType'].downcase,
* gender: row['Gender']
* ) do |record|
* record.record\_weight = row['RecordKg'].to\_i
* end
* end
* puts "Import complete."
* end
* end
* # Matching logic (app/services/record\_matcher.rb)
* class RecordMatcher
* def self.match(user)
* user.lifts.each do |lift|
* RecordMatchJob.perform\_later(user.id, lift.id)
* end
* end
* end
* # Background Job (app/jobs/record\_match\_job.rb)
* class RecordMatchJob < ApplicationJob
* queue\_as :default
* def perform(user\_id, lift\_id)
* user = User.find(user\_id)
* lift = Lift.find(lift\_id)
* matches = FederationRecord.where(
* lift\_type: lift.lift\_type,
* gender: user.gender
* ).where("record\_weight < ?", lift.weight)
* matches.each do |record|
* MatchResult.find\_or\_create\_by(user: user, federation\_record: record) do |result|
* result.note = "User lift #{lift.weight}kg beats #{record.record\_weight}kg in #{record.federation}"
* end
* end
* # Log to ScyllaDB
* ScyllaDBFeedLogger.log\_lift(user, lift)
* # Broadcast real-time update
* ActionCable.server.broadcast("lifts\_channel", { user\_id: user.id, lift\_type: lift.lift\_type, weight: lift.weight })
* end
* end
* # ScyllaDB Feed Logger (lib/scylladb\_feed\_logger.rb)
* require 'cassandra'
* class ScyllaDBFeedLogger
* def self.log\_lift(user, lift)
* cluster = Cassandra.cluster(hosts: ['scylla-node1', 'scylla-node2'])
* session = cluster.connect('powerlifting\_app')
* session.execute(
* "INSERT INTO lifts\_feed (user\_id, lift\_type, weight, timestamp) VALUES (?, ?, ?, ?)",
* arguments: [user.id, lift.lift\_type, lift.weight, Time.now]
* )
* end
* end
* # Redis Setup (config/initializers/redis.rb)
* require 'redis'
* REDIS = Redis.new(url: ENV.fetch("REDIS\_URL") { "redis://localhost:6379/1" })
* # Example usage:
* # REDIS.set("user\_#{user.id}\_last\_lift", lift.to\_json)
* # REDIS.get("user\_#{user.id}\_last\_lift")
* # Sidekiq initializer (config/initializers/sidekiq.rb)
* Sidekiq.configure\_server do |config|
* config.redis = { url: ENV.fetch("REDIS\_URL") { "redis://localhost:6379/1" } }
* end
* Sidekiq.configure\_client do |config|
* config.redis = { url: ENV.fetch("REDIS\_URL") { "redis://localhost:6379/1" } }
* end
* # ActionCable Channel (app/channels/lifts\_channel.rb)
* class LiftsChannel < ApplicationCable::Channel
* def subscribed
* stream\_from "lifts\_channel"
* end
* end
* # The result is a robust, searchable app that helps powerlifters discover which records they might beat.
* # UI uses red and black for a bold, strength-focused visual identity.
* # Redis supports session caching, queuing via Sidekiq, and real-time updates.
* # ScyllaDB logs activity feed events, and ActionCable pushes updates live.