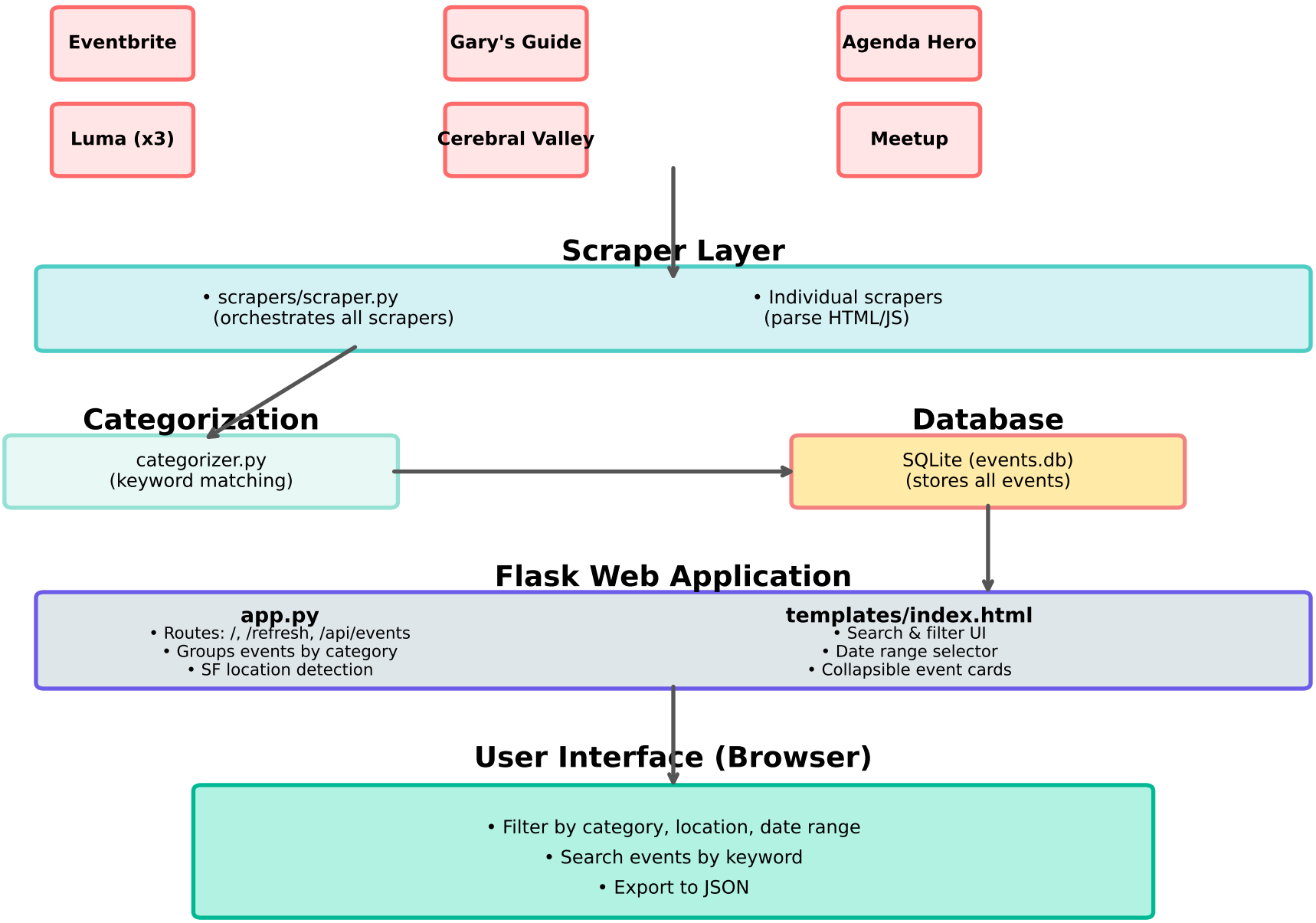
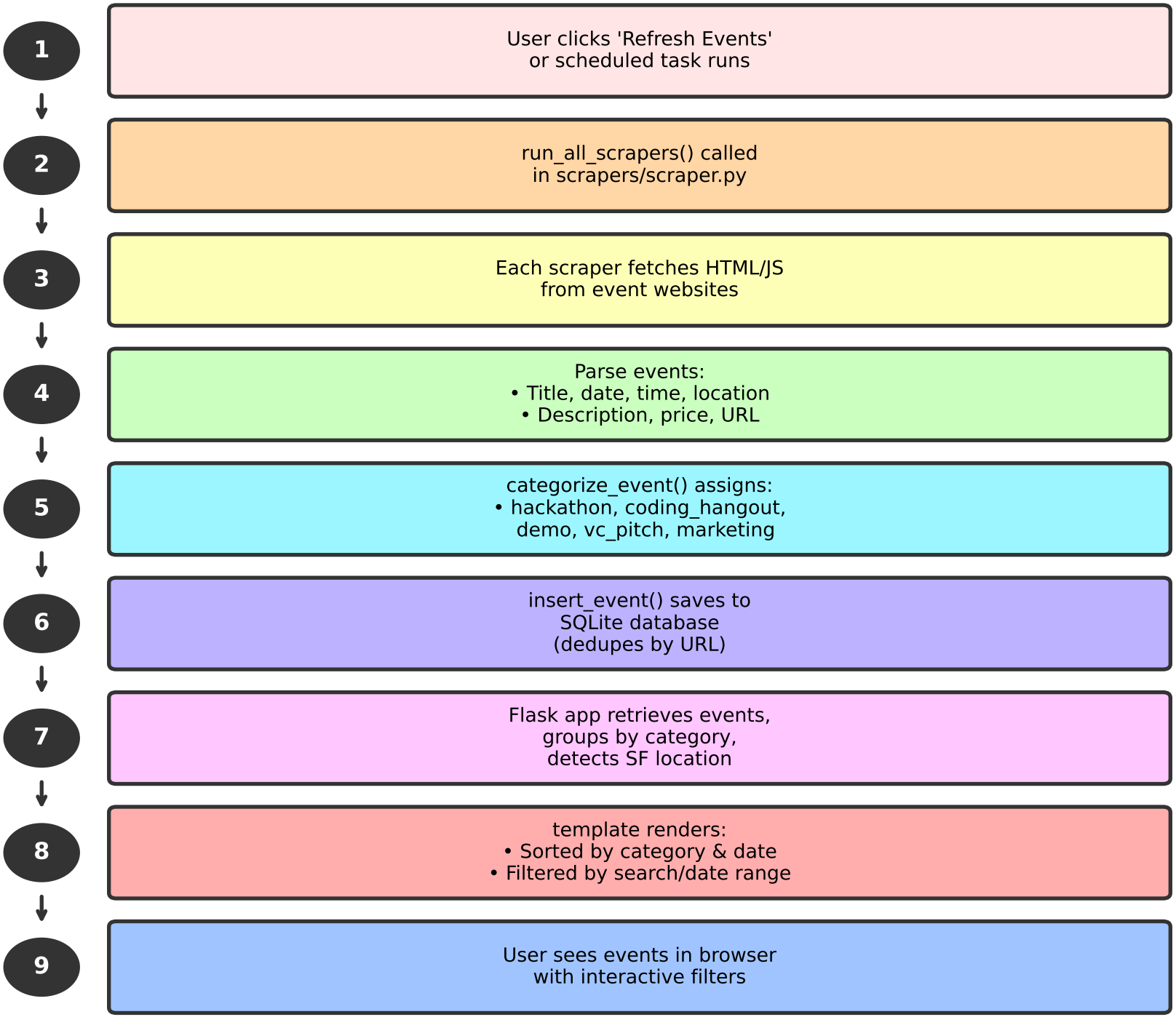
Tech Event Scraper - System Architecture



Data Flow - Event Scraping to Display



Component Details & File Structure

Scrapers

- eventbrite.py
- garys_guide.py
- agenda_hero.py
- luma.py
- cerebral_valley.py
- meetup.py
- scraper.py

Core Application

- app.py
- config.py
- requirements.txt

Database

- db_helper.py
- schema.sql
- events.db

Key Features

Smart Scraping	Uses requests + BeautifulSoup for static sites, undetected-chromedriver for JS-rendered content
Auto-Categorization	Keyword-based categorization into 6 categories: hackathon, coding_hangout, demo, vc_pitch, marketing, other
Duplicate Detection	Prevents duplicates using unique (title, date) and URL
SF Detection	Highlights events in San Francisco with green badge
Date Filtering	Filter events by 1 week, 2 weeks, or custom date range
Search & Filter	Real-time search by keyword, category, and location

Technology Stack

Python 3 | Flask | SQLite | BeautifulSoup4 | undetected-chromedriver | requests | dateutil | APScheduler

Individual Scraper Workflow

Phase 1: Static HTML Scrapers

- requests.get(url)
- BeautifulSoup(html)
- soup.find_all() to locate events
- Extract: title, date, location, URL
- Parse dates with dateutil.parser
- Return list of event dicts

Examples:
Gary's Guide
Eventbrite (with fallback)

Phase 2: JavaScript Scrapers

- undetected_chromedriver.Chrome()
- driver.get(url)
- Wait for JS to load (sleep)
- driver.page_source
- BeautifulSoup(page_source)
- Extract events from rendered HTML
- driver.quit()

Examples:
Luma • Meetup
Cerebral Valley • Agenda Hero

Common Post-Processing (All Scrapers)

Categorization	categorize_event(title, description) → category
Date Normalization	Convert all dates to datetime.date objects
Location Detection	Check if location contains "San Francisco" or "SF"
Error Handling	Try-except blocks to handle parsing failures gracefully

Final Step: Database Insertion

insert_event() → SQLite with UNIQUE constraint on (title, event_date) and url