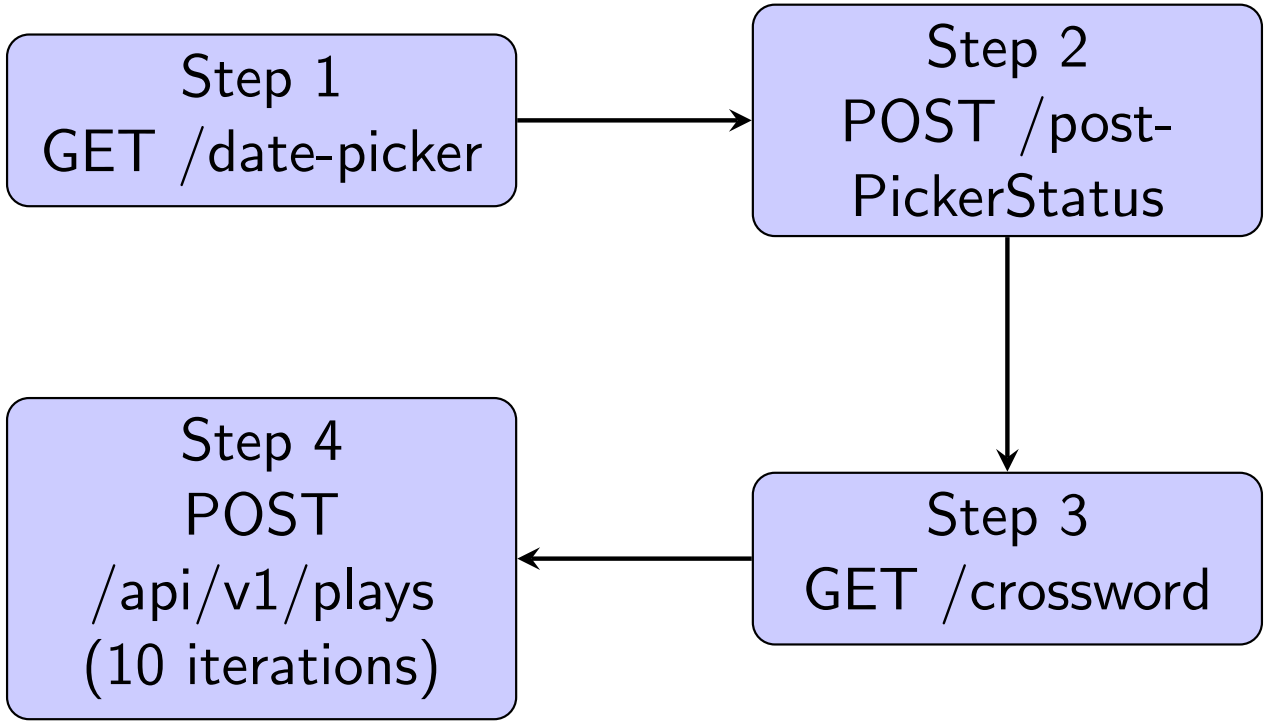
# Performance Testing Framework

## Java Implementation for Crossword API Load Testing

Saumya and Vansh

December 30, 2025

# API Flow: High-Level Architecture




**Simulates complete user journey:** Opening puzzle picker → Selecting puzzle → Playing the crossword

# Step 1: Date Picker Page

**Endpoint:** `GET /date-picker`

**Purpose:**

- Fetches the puzzle date picker HTML page
- Extracts `loadToken` from embedded JSON params
- Token required for all subsequent API calls

**Parameters:**

- `set` - Puzzle series identifier
- `uid` - User identifier (random or fixed)

**V3 Extension:** Also fetches CDN resources:

- `date-picker-min.css`, `picker-min.js`
- Font Awesome CSS and WOFF2 fonts

# Step 2: Post Picker Status

**Endpoint:** `POST /postPickerStatus`

**Purpose:**

- Notifies server that user is viewing the picker
- Validates the `loadToken`
- Records ad duration and verification status

**Request Body:**

```
{
    "loadToken": "<from step1>",
    "isVerified": true,
    "adDuration": 0,
    "reason": "displaying puzzle picker"
}
```

# Step 3: Load Crossword

**Endpoint:** `GET /crossword`

**Purpose:**

- Loads the actual crossword puzzle page
- Extracts puzzle parameters and `playId`
- Uses hardcoded puzzle ID for consistent testing

**Parameters:**

- `id` - Puzzle identifier
- `set`, `picker`, `uid`, `loadToken`

**V3 Extension:** Also fetches:

- `crossword-player-min.css`
- `c-min.js` (crossword player script)

# Step 4: Simulate Gameplay

**Endpoint:** `POST /api/v1/plays` (10 iterations)

**Purpose:**

- Simulates user solving the puzzle
- Sends state updates with progress

**Play States:**

1. **playState=1**: Game started
2. **playState=2** (iterations 2-9): In progress with mutations
3. **playState=4**: Game completed

**State Management:**

- `primaryState`: Current letter entries
- `secondaryState`: Fill status bitmap
- Random mutations simulate real user behavior

# Technology Stack

**Build System:**

- **Maven** - Dependency management & packaging
- **Java 17** - Modern language features
- **Maven Shade Plugin** - Uber JAR packaging

**Dependencies:**

| Library | Version | Purpose |
|---|---|---|
| Gson | 2.10.1 | JSON parsing and serialization |
| OpenCSV | 5.9 | CSV result file generation |
| JCommander | 1.82 | CLI argument parsing |
| SLF4J | 2.0.9 | Logging framework |

# Project Structure

**Source Files:**

- `ApiFlowV2.java` - Main entry
- `ApiFlow.java` - HTTP flow logic
- `ApiFlowV3.java` - With CDN fetches
- `WaveExecutor.java` - RPS control
- `CsvResultWriter.java` - Output
- `HtmlReportWriter.java` - Dashboard

# HTML Dashboard: Overview

**Metrics Cards:**

- **Total Threads** - Number of test runs
- **Success Rate** - Percentage of successful flows
- **Avg Latency** - Mean response time
- **P95 Latency** - 95th percentile
- **Min / Max** - Latency range

**Color Coding:**

- **Green** - Success rate $\geq 95\%$
- **Yellow** - Success rate 80-95%
- **Red** - Success rate $< 80\%$

# V2 Results

| RPS | Waves | Errors | Avg Latency (s) |
|---|---|---|---|
| 7 | 900 | 0 | 3.6 |
| 30 | 900 | 0 | 3.79 |
| 90 | 900 | 75 | 5.9 |
| 150 | 90 | 14 | 9.12 |
| 180 | 90 | 25 | 10.9 |

**Note:** 300 RPS did not work — thread pool exceeded max allowed threads on test devices.

*Observed issues running from utility node; will retry from non-GCP Linux server.*

# V3 Results

| RPS | Waves | Errors | Avg Latency (s) |
|---|---|---|---|
| 7 | 900 | 1 (502) | 4.8 |
| 15 | 900 | 3 | 6.0 |
| 90 | 100 | 13 (502) | 8.9 |

**Note:** Hit thread count limit on higher RPS values.

# Error Types Observed

- **Timeout** - Request exceeded configured timeout
- **502 Bad Gateway** - Server returned error response
- **Required settings not passed** - Missing configuration
- **Broken pipe** - Connection closed unexpectedly
- **Connection reset** - Server terminated connection