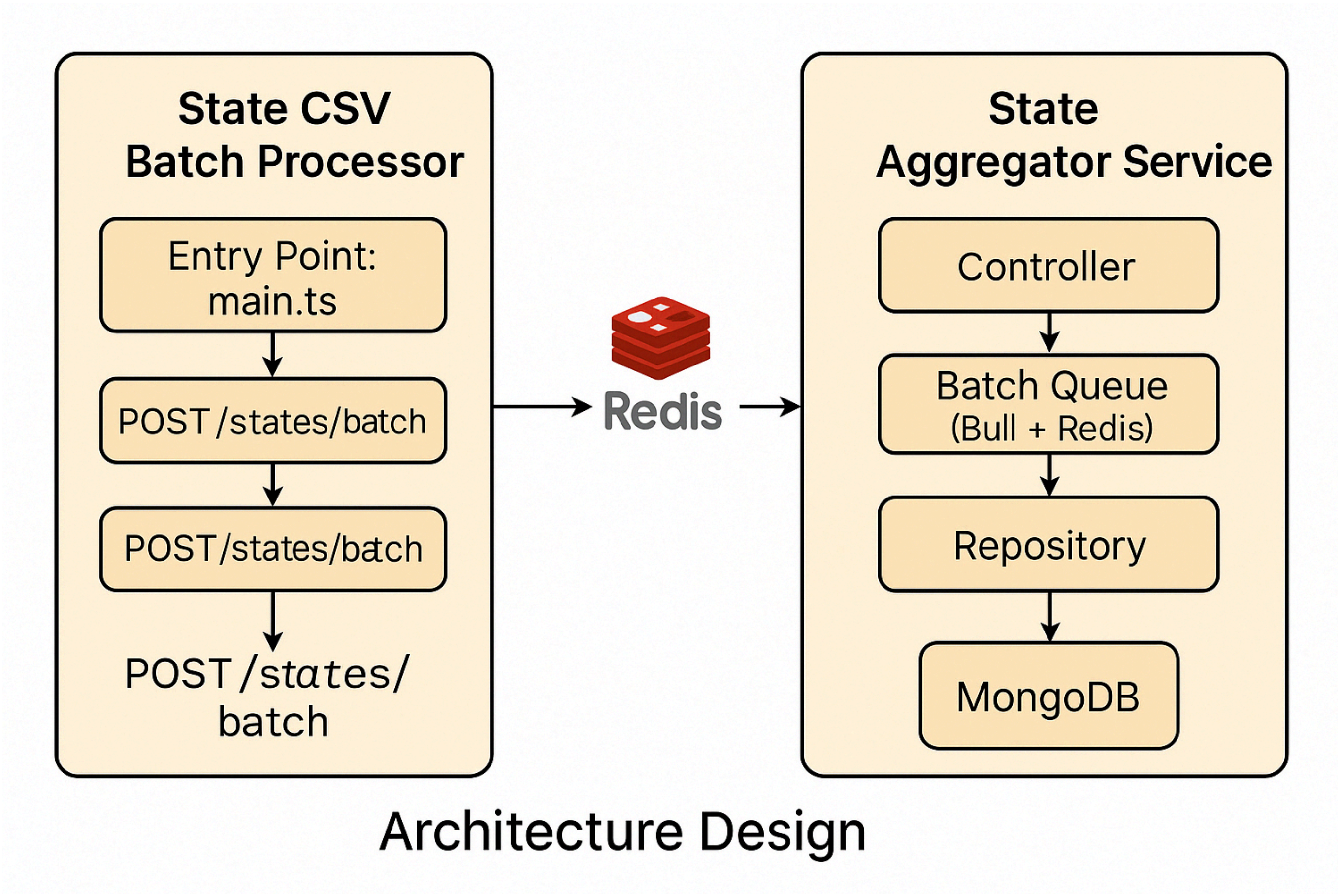
# Architecture Design Documentation

## Architecture Design (Diagram)



## Layered View

```
State CSV Batch Processor
-------------------------
[Entry Point: main.ts]
    |
[CsvReaderService]
    |
[Batching & Validation]
    |
[HTTP Client (axios)]
    |
[POST /states/batch]
    |
State Aggregator Service
------------------------
```

```
[Controller]
     |
[Service]
     |
[Batch Queue (Bull + Redis)]
     |
[Repository]
     |
[MongoDB]
```

## Objective

The objective of this architecture is:

- Automated Data Ingestion: Efficiently process a large CSV file containing phone records, extracting and batching state population data.
- Reliable Data Transfer: Send this data in manageable batches to a backend service using HTTP, ensuring scalability and fault tolerance.
- Centralized Aggregation & Storage: The backend ( State Aggregator Service ) receives these batches, aggregates the population by state, and persists the results in a MongoDB database.
- Asynchronous & Scalable Processing: Use Redis as a queue backend (via Bull) to decouple HTTP request handling from heavy batch processing, enabling horizontal scaling and reliability.
- Easy Data Access: Expose REST API endpoints so users or other systems can query the aggregated state population data, monitor ingestion, and manage records.

### Redis in the Architecture

- Purpose: Redis is used as a fast, in-memory data store to back the Bull queue system. This enables the State Aggregator Service to queue incoming batch jobs and process them asynchronously.
- How it works: When a batch is received via the /states/batch endpoint, it is placed into a Redis-backed queue. Worker processes (which can be scaled horizontally) pull jobs from this queue and perform aggregation and storage in MongoDB.
- Benefits:
  - Decouples HTTP requests from processing, improving API responsiveness.
  - Supports retry, delayed, and scheduled jobs.
  - Enables monitoring of job status and queue health.
  - Facilitates horizontal scaling—multiple workers can process jobs from the same queue.

## In summary:

- State CSV Batch Processor automates reading and batching of CSV data, sending it to the backend.
- State Aggregator Service receives, queues (via Redis), aggregates, and stores the data, providing APIs for access and management.
- Redis (with Bull) acts as a reliable, scalable queue for batch processing, ensuring that heavy workloads do not block API requests and can be processed efficiently in the background.
- MongoDB stores the final, aggregated state population data.
  This separation ensures each part of the system is focused, maintainable, scalable, and robust for production workloads.