

Case Study #1

Memi Lavi
www.memilavi.com





A Real World Application

Application Introduction

```
graph TD; A[Application Introduction] --> B[Defining Requirements]; B --> C[Components Mapping]; C --> D[Technology Stack Selection]; D --> E[Architecture Design];
```

Defining Requirements

Components Mapping

Technology Stack Selection

Architecture Design

Dunderly

Your Paper Source

Dunderly

- Sells Paper Supplies
 - Printer paper, Envelopes, etc.
- Needs a new HR system
- Managing employees,
salaries, vacations, payments



Requirements

```
graph TD; Requirements[Requirements] --> Functional[Functional]; Requirements --> NonFunctional[Non-Functional];
```

Functional

What the system should do

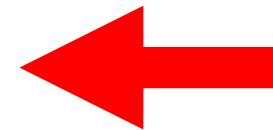
1. Web Based
2. Perform CRUD operations on employees
3. Manage Salaries:
 1. Allow manager to ask for employee's salary change
 2. Allow HR manager to approve / reject request
4. Manage vacation days
5. Use external payment system

Non-Functional

What the system should deal with

NFR - What We Know

1. Classic Information System
2. Not a lot of users
3. Not a lot of data
4. Interface to external system



NFR - What We Ask

1. *"How many expected concurrent users?"* 10
2. *"How many employees?"* 250
3. *"What do we know about the external
Payment system?"*

Payment System

- Legacy system, written in C++
- Hosted in the company's servers farm
- Input – only files ☹️
- File received once a month

Data Volume

- 1 Employee = ~1MB in data
- Each employee has ~10 scanned documents (contract, reviews etc.)
- 1 Scanned Document = ~5MB
- Total storage for 1 employee = ~51MB

Data Volume – Cont.

- Company expects to grow to 500 employees in 5 years
- Total storage: $51\text{MB} \times 500 \text{ employees} = 25.5\text{GB}$
- Not a lot, but:
 - Need to consider document storage

SLA

4. *"How critical is the system?"*

Not Very Critical

Requirements

```
graph TD; Requirements[Requirements] --> Functional[Functional]; Requirements --> NonFunctional[Non-Functional];
```

Functional

What the system should do

1. Web Based
2. Perform CRUD operations on employees
3. Manage Salaries:
 1. Allow manager to ask for employee's salary change
 2. Allow HR manager to approve / reject request
4. Manage vacation days
5. Use external payment system

Non-Functional

What the system should deal with

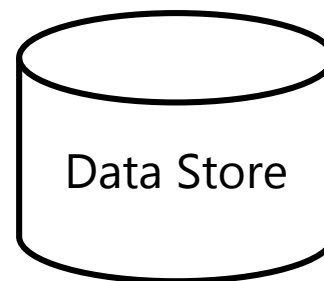
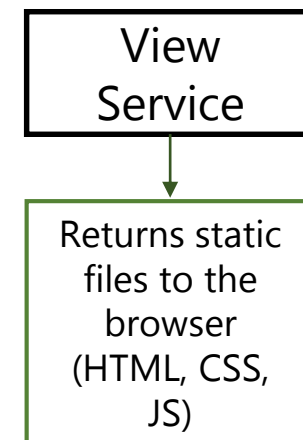
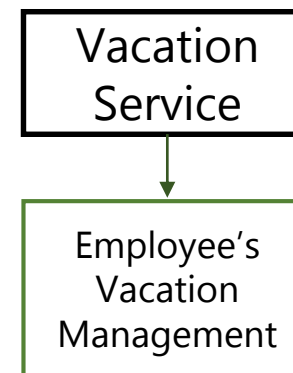
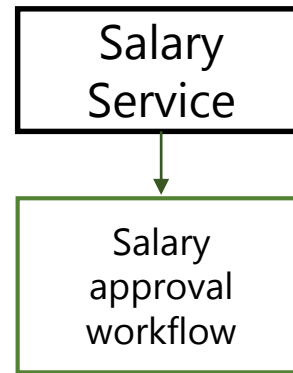
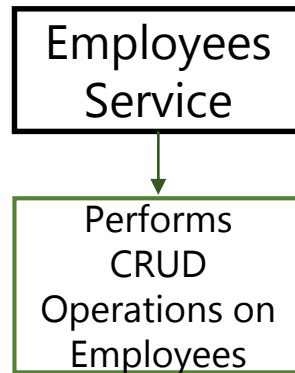
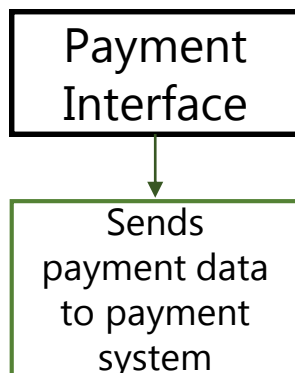
1. 10 Concurrent users
2. Manages 500 users
3. Data volume forecast: 25.5GB
 1. Relational & Unstructured
4. Not mission critical
5. File-based interface

Components

Based on requirements:

1. Entities: Employees, Vacation, Salary
2. Interface to the Payment System

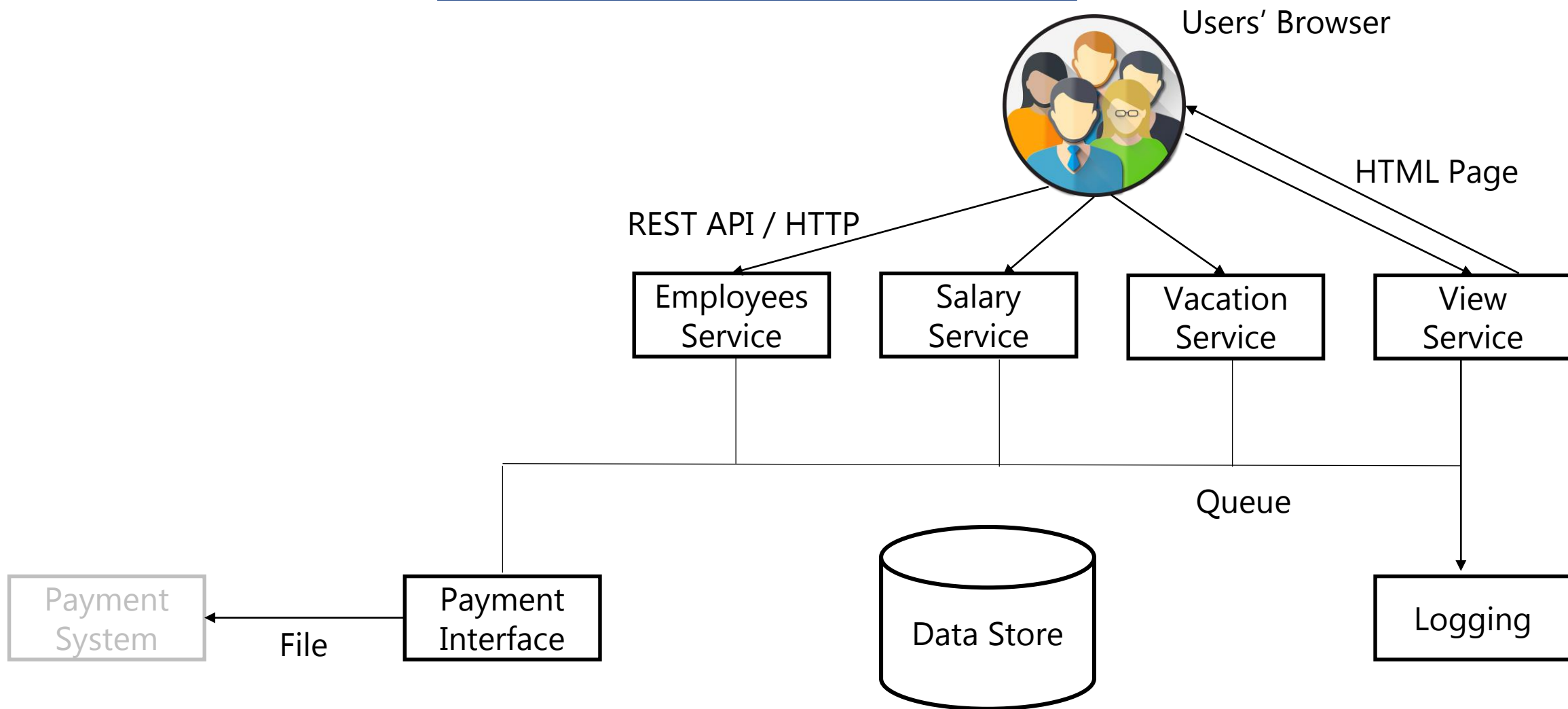
Payment System



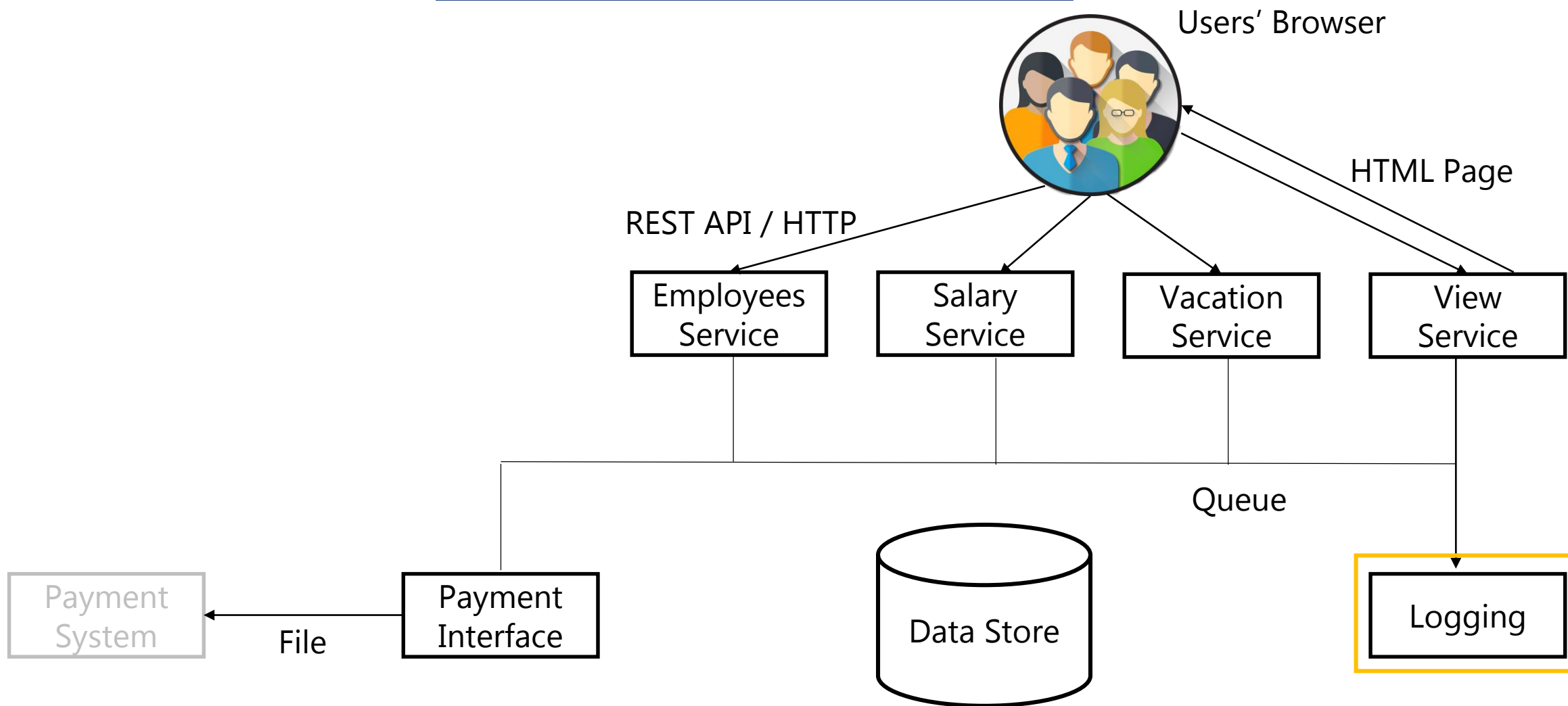
Q: Single or Per Service Data Store?

A: Data is shared between services, so a Single Data Store is better

Messaging



Components



Logging Service

- Very Important
- Other services use it

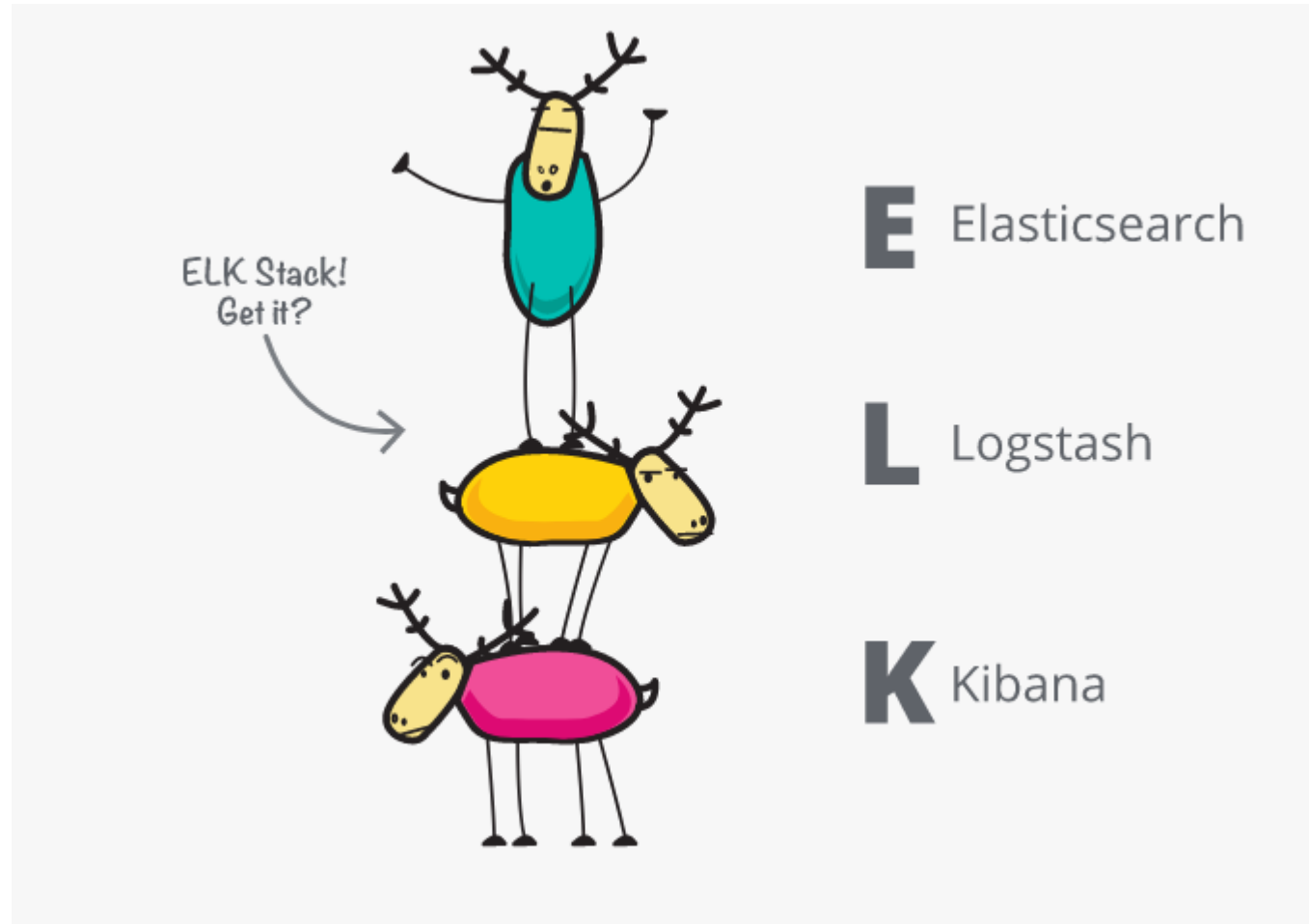
Logging - Questions

1. Is there an existing logging mechanism
in the company?

No

2. Develop our own or use 3rd party?

Logging - Alternative



Logging - Alternative

ELK:

- Powerful data store (**E**lastic)
- Import log from many sources (**L**ogstash)
- Great viewer with filter capabilities (**K**ibana)

Logging - Alternative

But:

- Requires maintenance
- Quite complicated to install and setup
- Suitable mainly for large, data-intensive systems

NO GO

Logging Service

Steps:

- Decide on Application Type
- Decide on Technology Stack
- Design the Architecture

Application Type

What it does:






- Read log records from queue
- Validate the records
- Store in data store

Application Type



What it does:

- **Read log records from queue**
- Handle the records
- Save in data store

Application Type

- Web App & Web API 
- Mobile App 
- Console 
- Service 
- Desktop App 

Application Type

- Web App & Web API 
- Mobile App 
- Console 
- Service 
- Desktop App 

Technology Stack

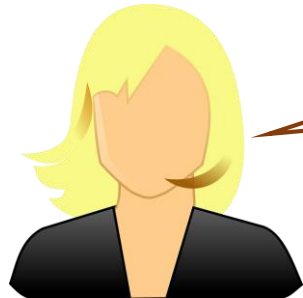
For:

- Component's Code
- Data Store

Technology Stack

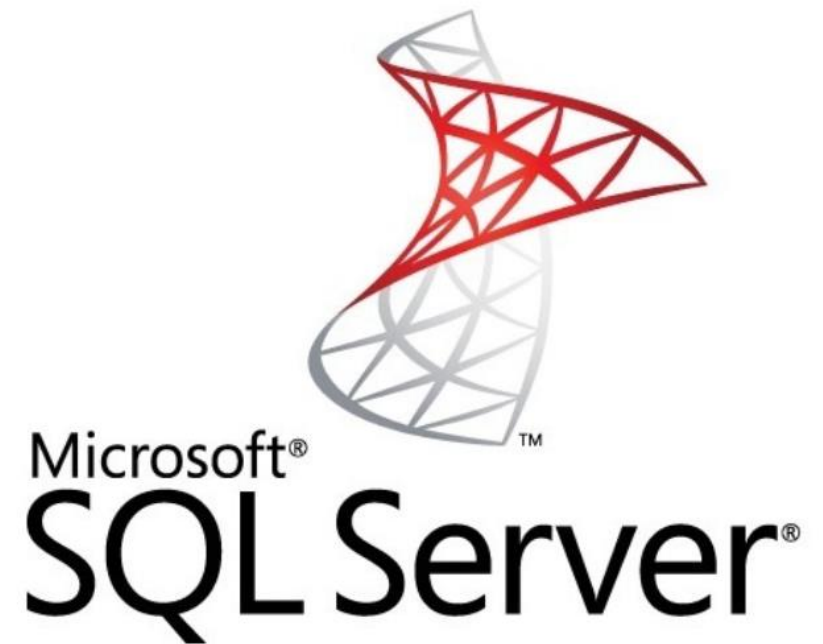
Code Should:

- Access Queue's API
- Validate the data
- Store the data



We're familiar with Microsoft stack, so we are expert in .NET and SQL Server

Technology Stack



Architecture

User Interface / Service
Interface

Business Logic

Data Access



Data Store

Architecture

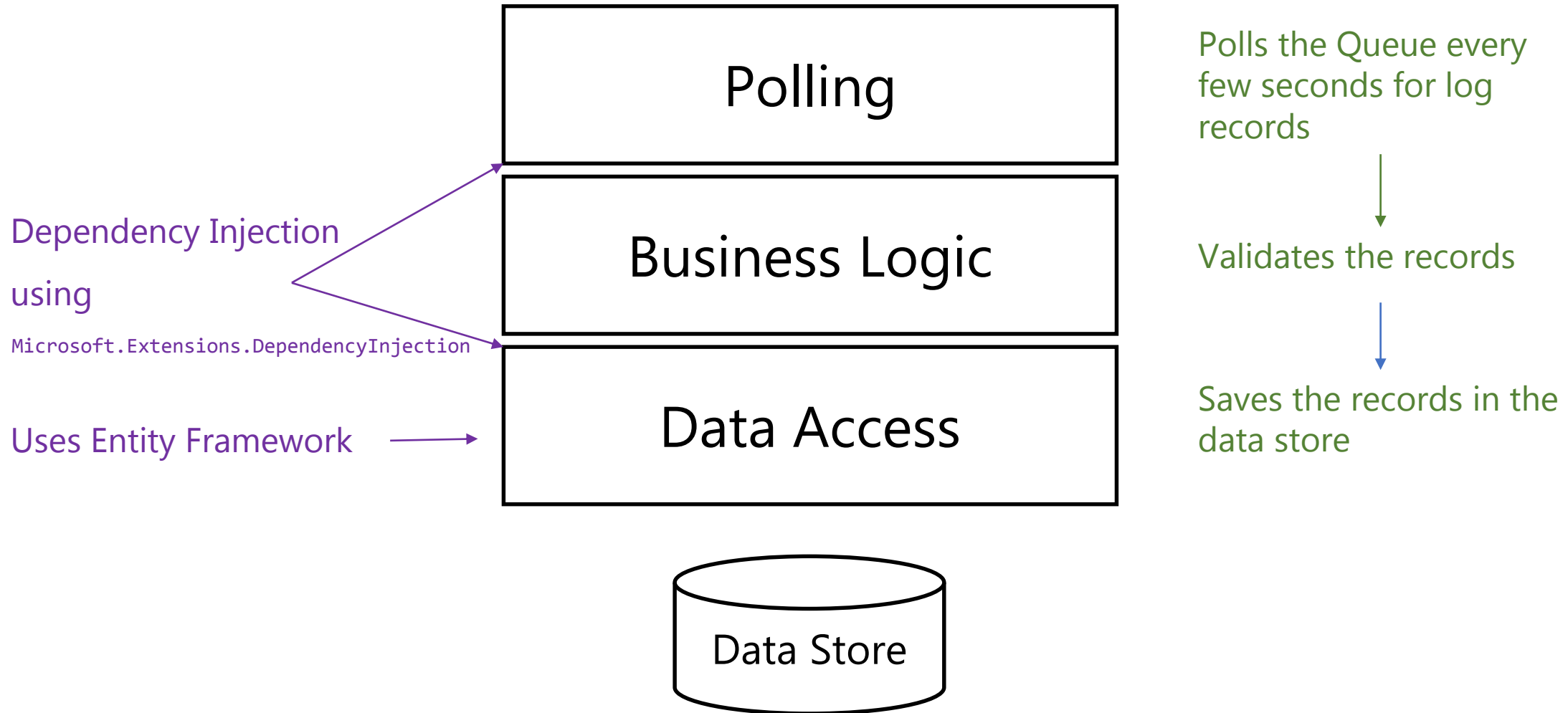
User Interface / Service
Interface

Business Logic

Data Access

Data Store

Logging Service

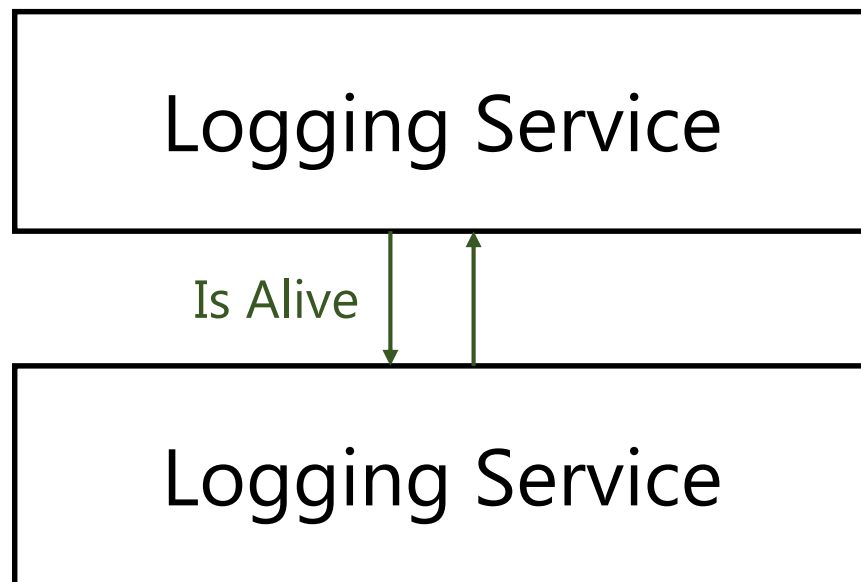


Logging Service Redundancy

Logging Service

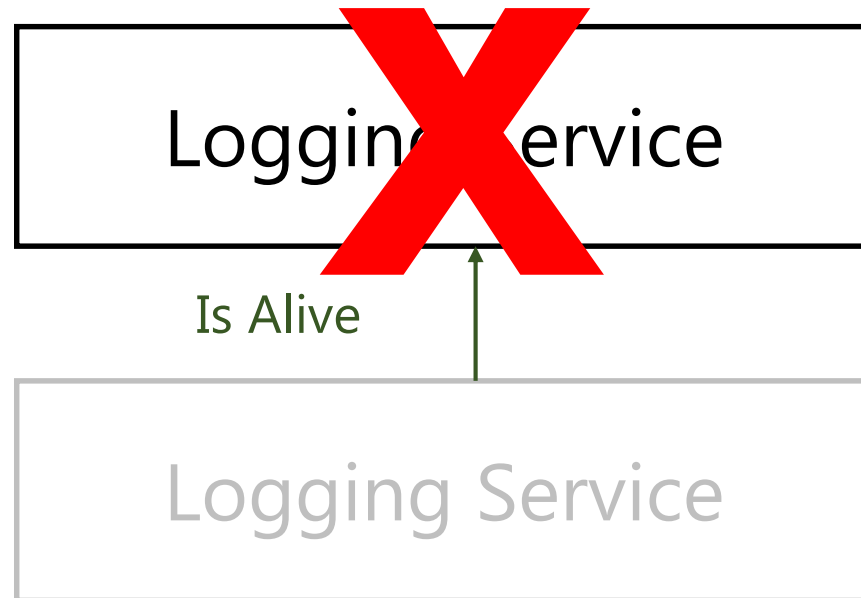


Logging Service Redundancy

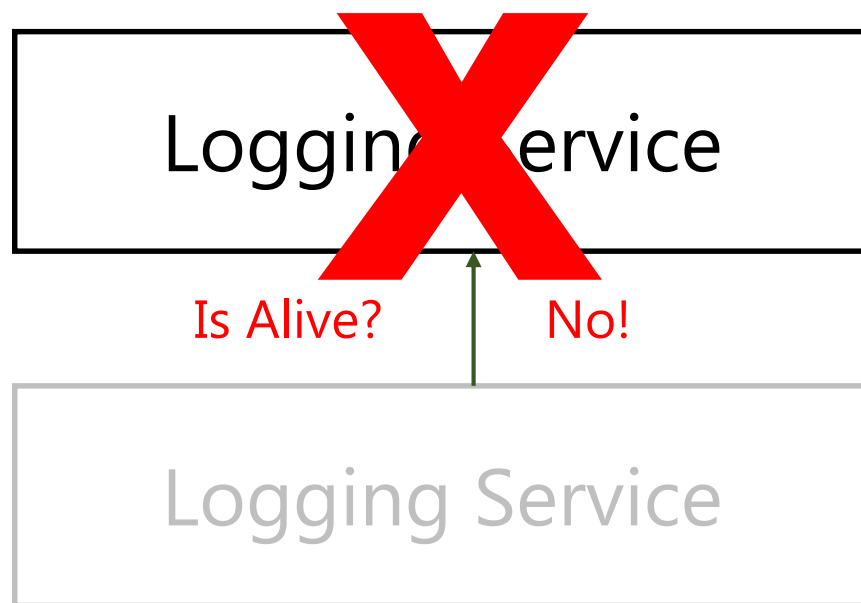


- Active / Active
- Avoid duplicate reads?

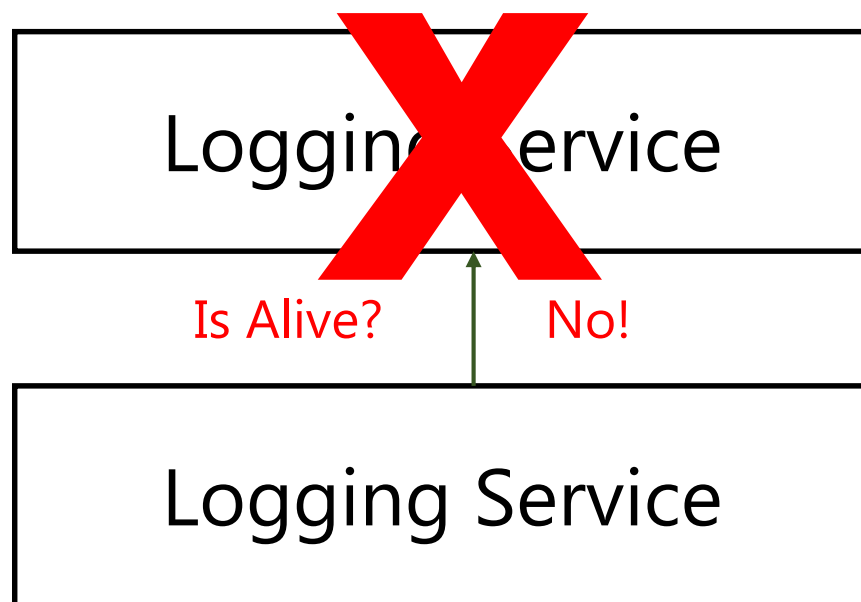
Logging Service Redundancy



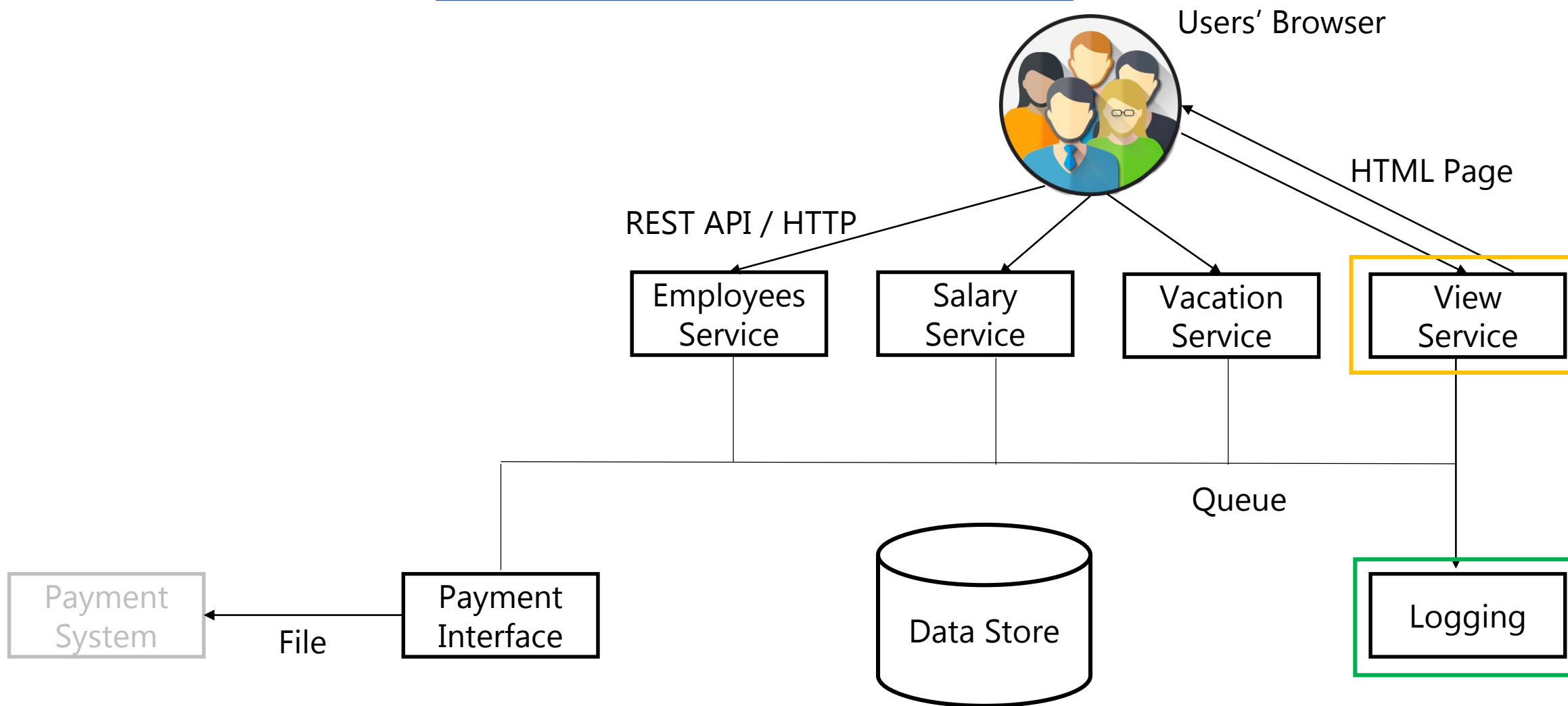
Logging Service Redundancy



Logging Service Redundancy



Components



View Service

What it does:

- Get requests from the end users' browsers
- Returns static files (HTML / CSS / JS)

Application Type

- Web App & Web API ✓
- Mobile App ✗
- Console ✗
- Service ✗
- Desktop App ✗



Technology Stack

.NET Core has a great support for Web Apps

So...

Technology Stack



Architecture

User Interface / Service
Interface

Business Logic

Data Access



Data Store

Architecture

User Interface

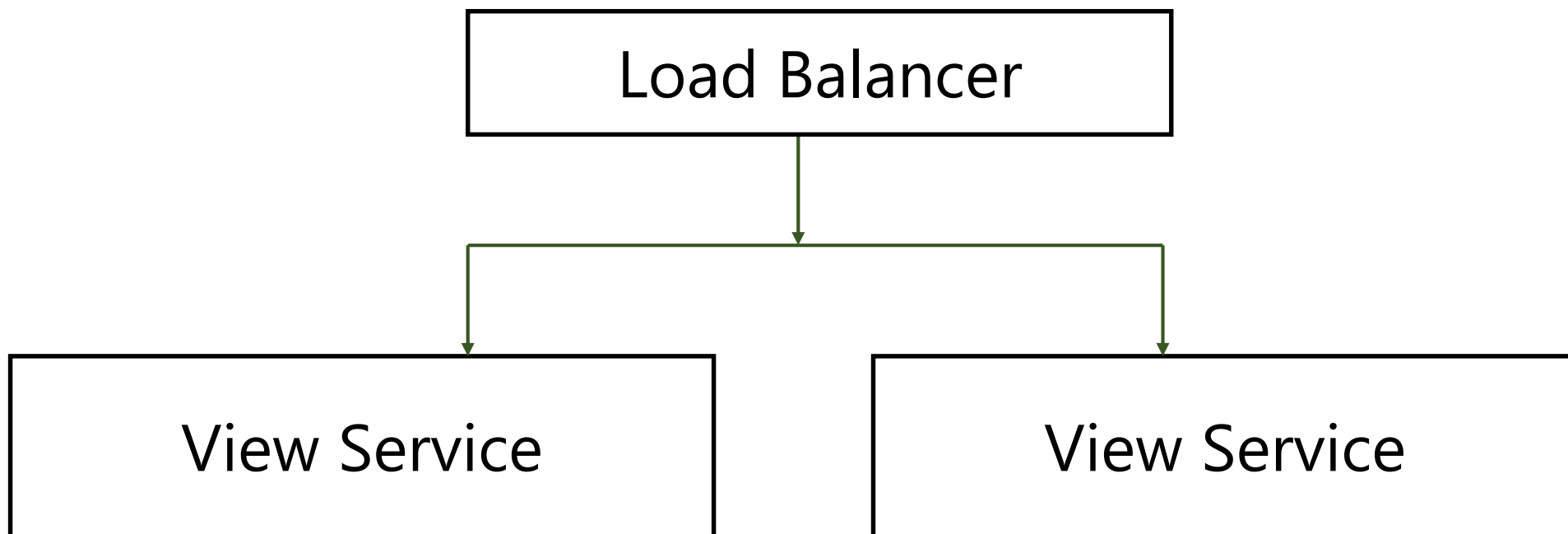
Business Logic

Data Access

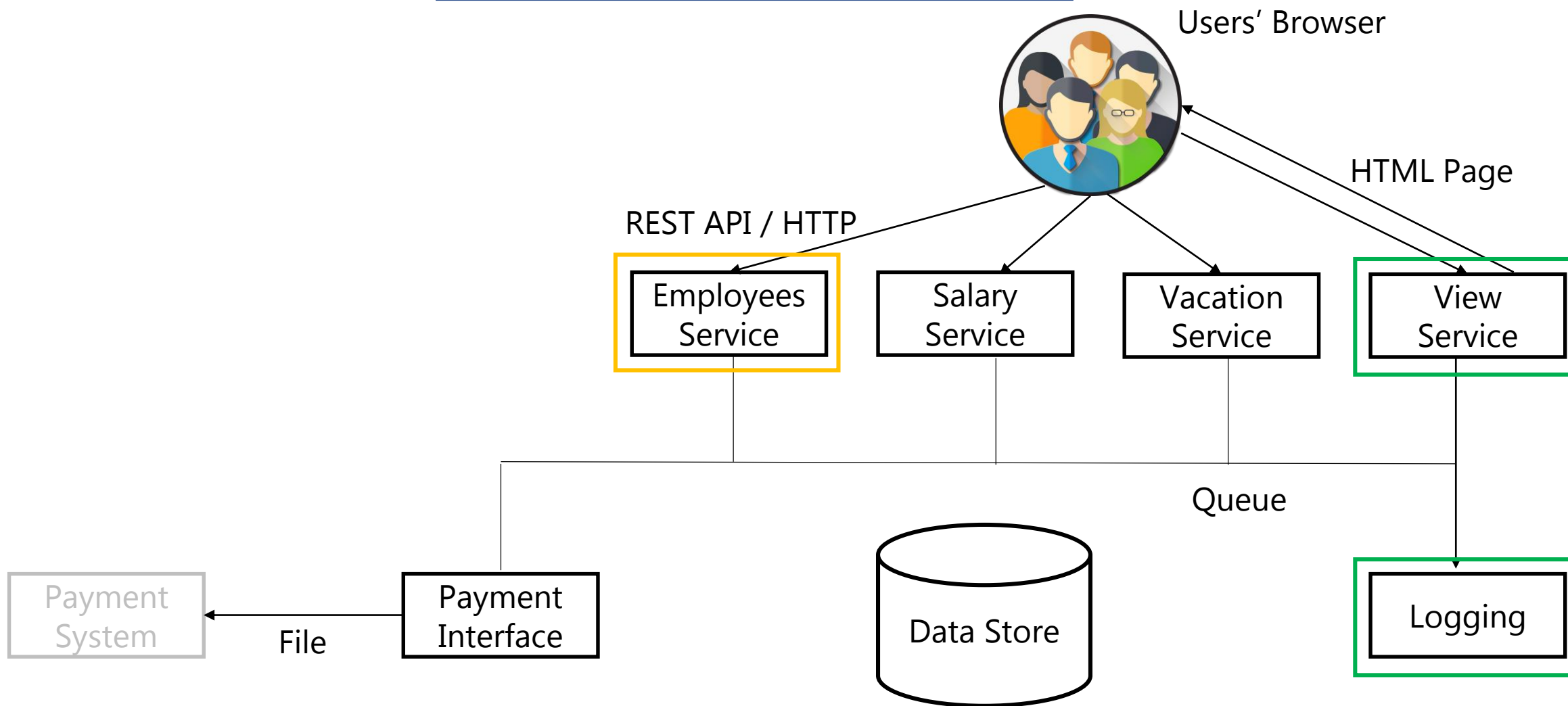
Data Store



View Service Redundancy



Components



Employees Service

What it does:

- Allows end users to query employees' data
- Allows performing actions on data (CUD)

What it doesn't:

- Displays the data

Application Type

- Web App & Web API ✓
- Mobile App ✗
- Console ✗
- Service ✗
- Desktop App ✗

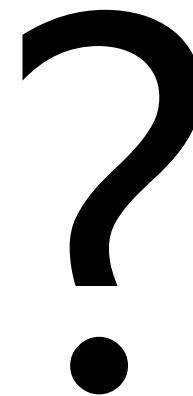
Technology Stack – Dev Platform



Technology Stack – Database

Employee Data (Relational)

Documents



Technology Stack – Database

Document (BLOB) Storage Alternatives

Relational Database

File System

Object Store

Cloud Storage



Document (BLOB) Storage Alternatives

Alternative	Description	Examples	Pros	Cons
Relational Database	Store the document in a specialized column type designed for BLOBs	SQL Server's FILESTREAM, Oracle's BLOB type	Part of the app transaction Part of the DB's backup / DR	Clunky syntax, Limited size



Document (BLOB) Storage Alternatives

Alternative	Description	Examples	Pros	Cons
Relational Database	Store the document in a specialized column type designed for BLOBs	SQL Server's FILESTREAM, Oracle's BLOB type	Part of the app transaction Part of the DB's backup / DR	Clunky syntax, Limited size
File System	Store the document in a file, and hold a pointer to it in the DB	File System (duh...)	Unlimited size Easy to execute	Not part of transaction, Unmanageable



Document (BLOB) Storage Alternatives

Alternative	Description	Examples	Pros	Cons
Relational Database	Store the document in a specialized column type designed for BLOBs	SQL Server's FILESTREAM, Oracle's BLOB type	Part of the app transaction Part of the DB's backup / DR	Clunky syntax, Limited size
File System	Store the document in a file, and hold a pointer to it in the DB	File System (duh...)	Unlimited size Easy to execute	Not part of transaction, Unmanageable
Object Store	Use special type of store mechanism that specializes in BLOBs	CEPH	Great scale Unlimited size	Complex setup Dedicated knowledge New product in the mix



Document (BLOB) Storage Alternatives

Alternative	Description	Examples	Pros	Cons
Relational Database	Store the document in a specialized column type designed for BLOBs	SQL Server's FILESTREAM, Oracle's BLOB type	Part of the app transaction Part of the DB's backup / DR	Clunky syntax, Limited size
File System	Store the document in a file, and hold a pointer to it in the DB	File System (duh...)	Unlimited size Easy to execute	Not part of transaction, Unmanageable
Object Store	Use special type of store mechanism that specializes in BLOBs	CEPH	Great scale Unlimited size	Complex setup Dedicated knowledge New product in the mix
Cloud Storage	Store the documents in one of the public cloud storage mechanisms	Azure's Storage Account AWS's S3	Great scale Easy to execute	Requires internet connection Cost

Technology Stack – Database

Employee Data (Relational)

Documents



Technology Stack – Database

Employee Data (Relational)



Microsoft®
SQL Server®

- Documents are small (~1MB)
- Already exists
- Part of the app

Documents



Microsoft®
SQL Server®

Architecture

Service Interface


Business Logic

Data Access

Data Store



API

- Get full employee details by ID
- List of employees by parameters
- Add employee
- Update employee details
- Remove employee  **Not physical delete!**

API – Cont.

- Add document
- Remove document
- Get document
- Retrieve documents by parameters

Q: Do we need a separate
Document Handler
service?

A: Since only the
Employee entity requires
docs, then no.

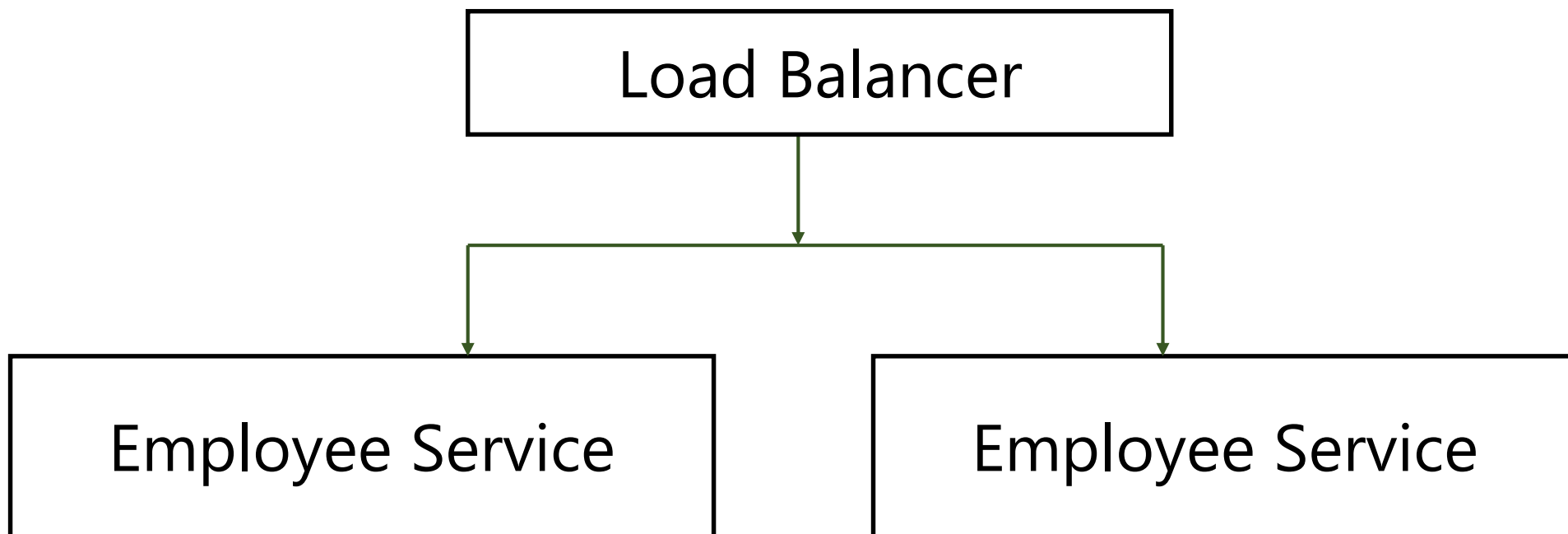
API

Functionality	Path	Return Codes
Get employee details by ID	GET /api/v1/employee/{id}	200 OK 404 Not Found
List employees by parameters	GET /api/v1/employees?name=...&birthdate=...	200 OK 400 Bad Request
Add employee	POST /api/v1/employee	201 Created 400 Bad Request
Update employee details	PUT /api/v1/employee/{id}	200 OK 400 Bad Request 404 Not Found
Remove employee	DELETE /api/v1/employee/{id}	200 OK 404 Not Found

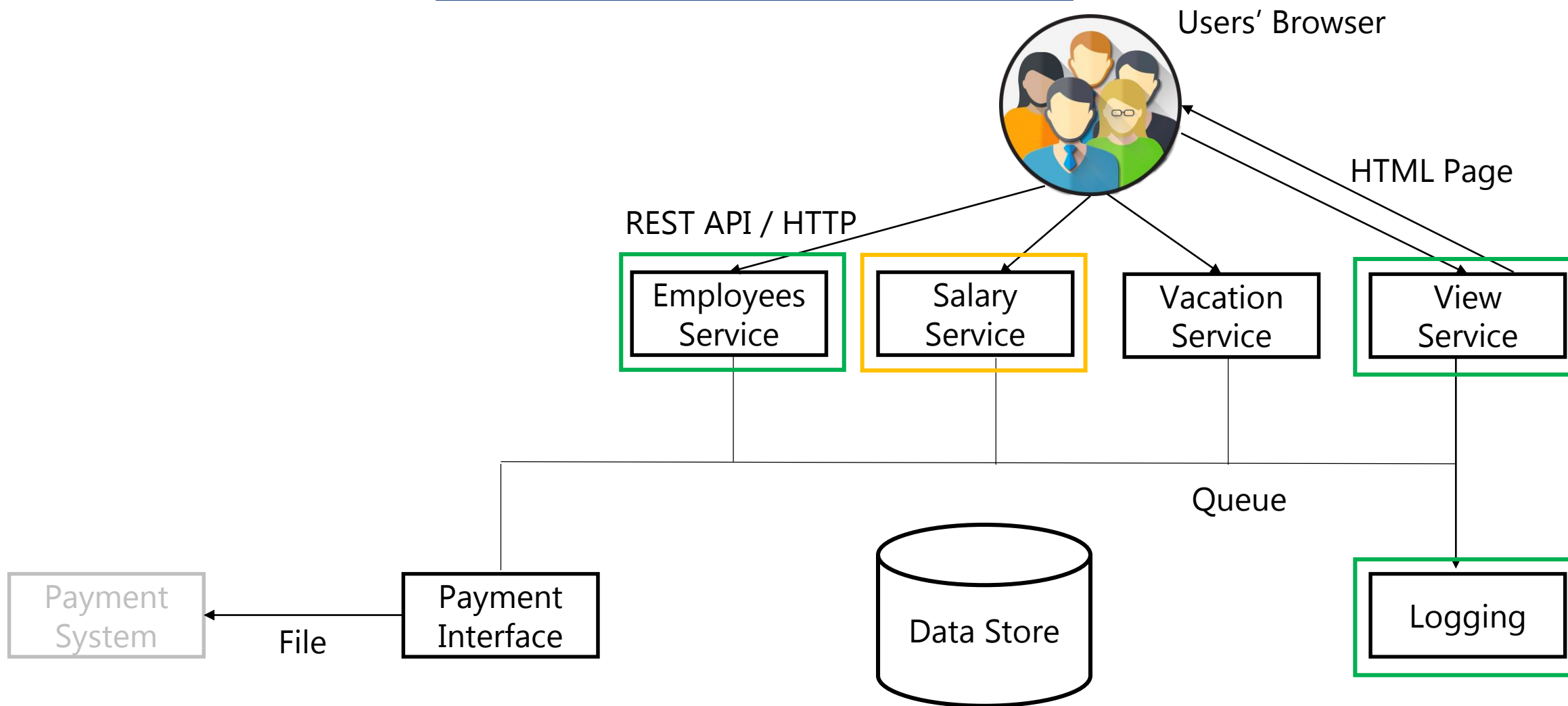
API

Functionality	Path	Return Codes
Add document	POST /api/v1/employee/{id}/document	201 Created 404 Not Found
Remove document	DELETE /api/v1/employees/{id}/document/{docid}	200 OK 404 Not Found
Get document	GET /api/v1/employees/{id}/document/{docid}	200 OK 404 Not Found
Retrieve documents for employee	GET /api/v1/employees/{id}/documents	200 OK 404 Not Found

Employee Service Redundancy



Components



Salary Service

What it does:

- Allows managers to ask for an employee's salary change
- Allows HR representative to approve / reject the request

Application Type

- Web App & Web API ✓
- Mobile App ✗
- Console ✗
- Service ✗
- Desktop App ✗

Technology Stack



Architecture

Service Interface

Business Logic

Data Access

Data Store



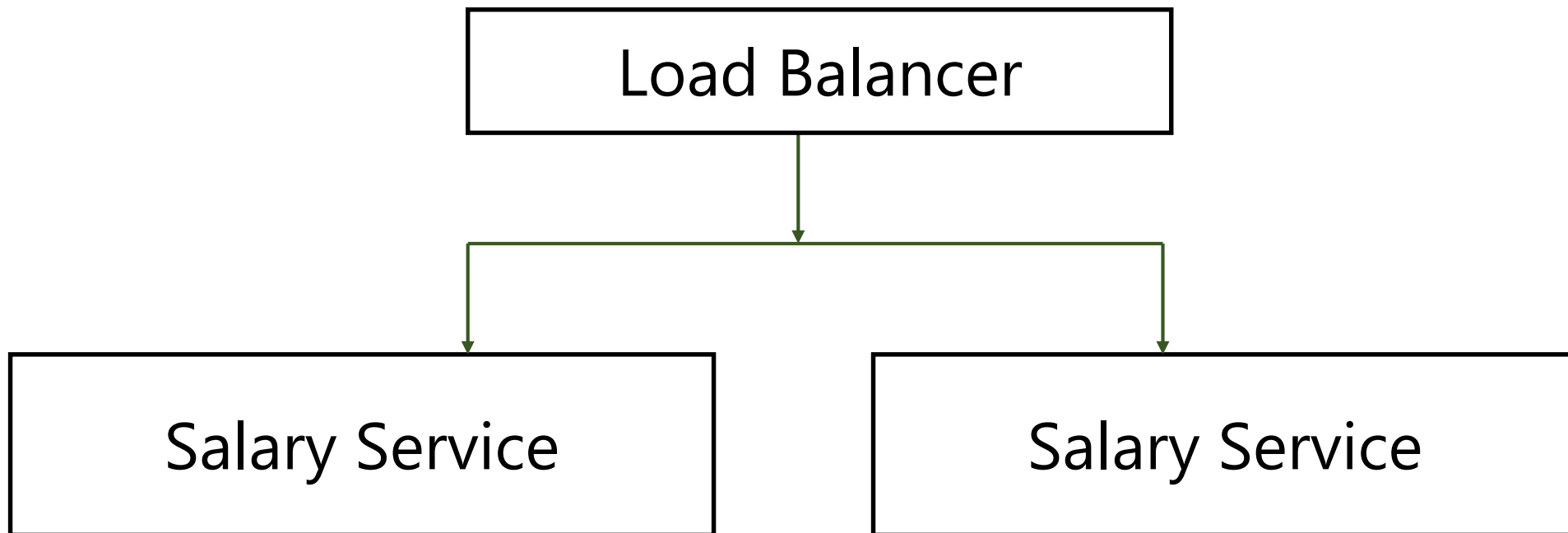
API

- Add salary request
- Remove salary request
- Get salary requests
- Approve salary request
- Reject salary request

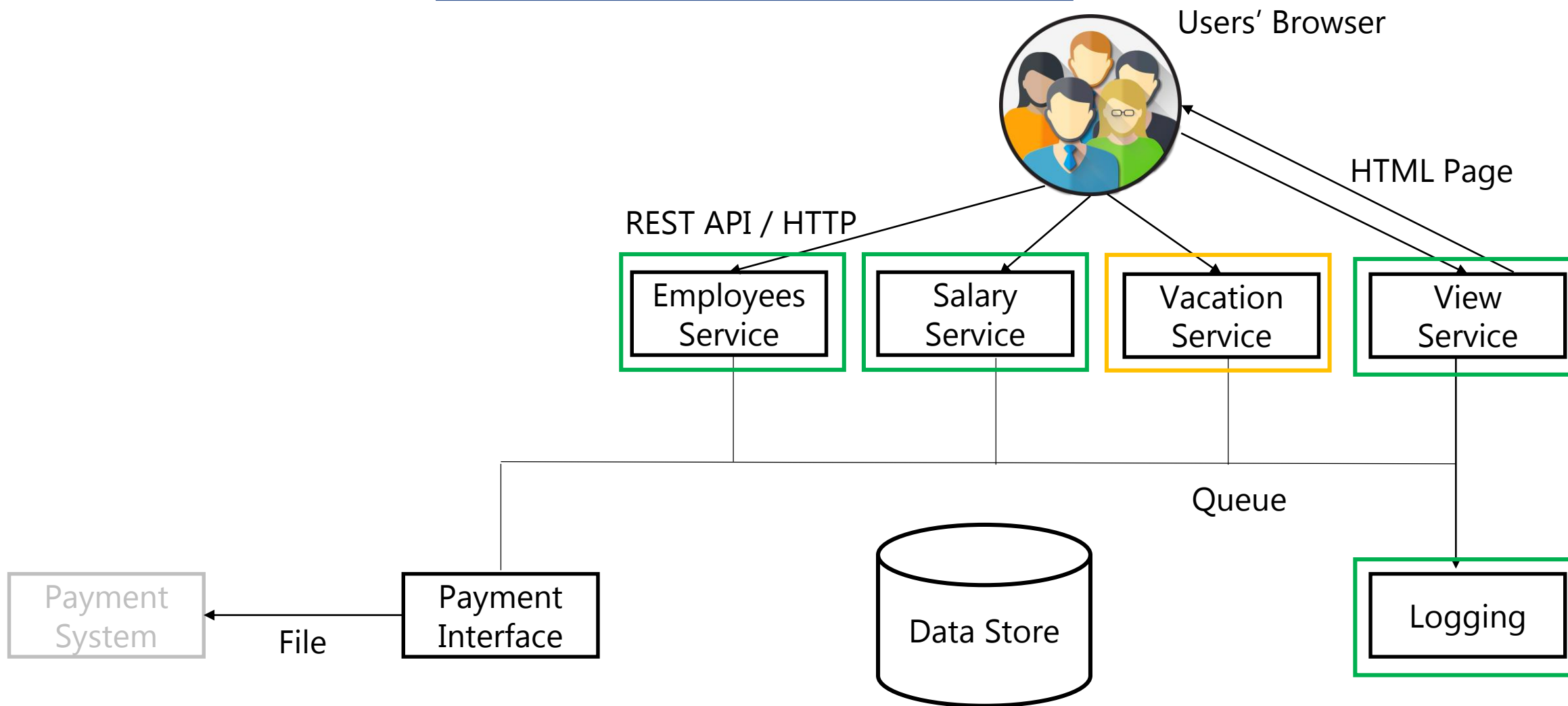
API

Functionality	Path	Return Codes
Add salary request	POST /api/v1/salaryRequest/	200 OK 400 Bad Request
Remove salary request	DELETE /api/v1/salaryRequest/{id}	200 OK 404 Not Found
Get salary requests	GET /api/v1/salaryRequests	200 OK
Approve salary request	POST /api/v1/salaryRequest/{id}/approval	200 OK 404 Not Found
Reject salary request	POST /api/v1/salaryRequest/{id}/rejection	200 OK 404 Not Found

Salary Service Redundancy



Components



Vacation Service

What it does:

- Allows employees to manage their vacation days
- Allows HR to set available vacation days for employees

Application Type

- Web App & Web API ✓
- Mobile App ✗
- Console ✗
- Service ✗
- Desktop App ✗

Technology Stack



Architecture

Service Interface

Business Logic

Data Access

Data Store



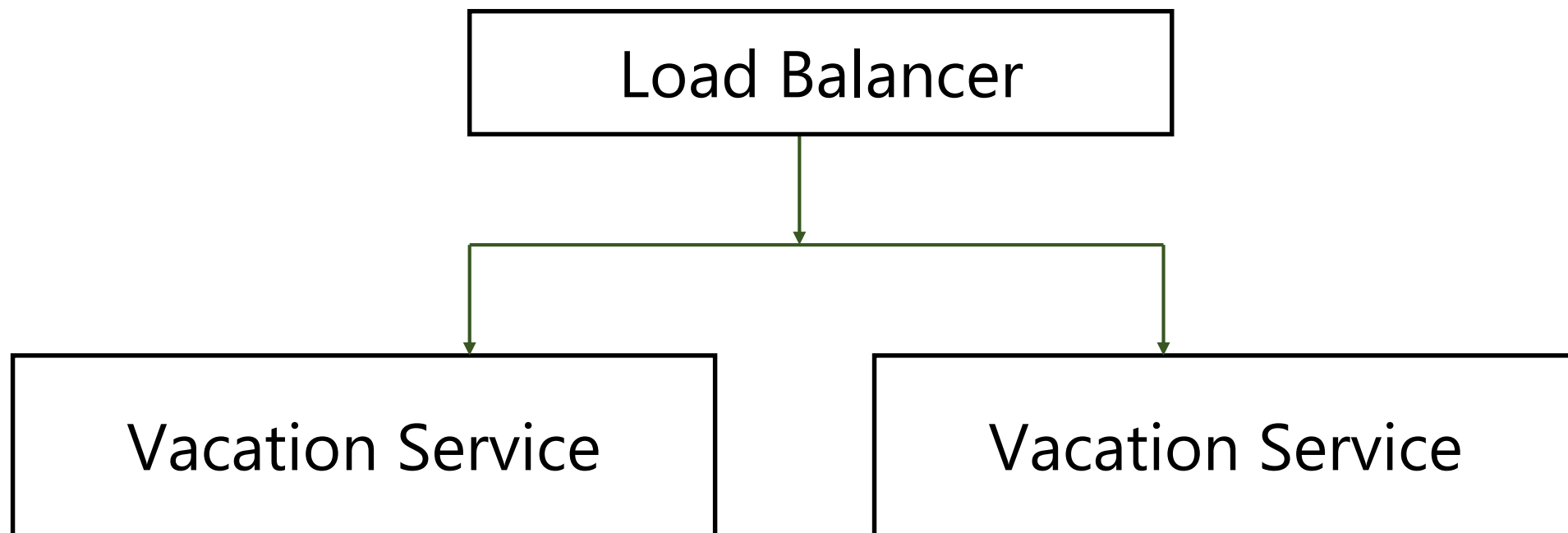
API

- Set available vacation days (by HR)
- Get available vacation days
- Reduce vacation days (by employees)

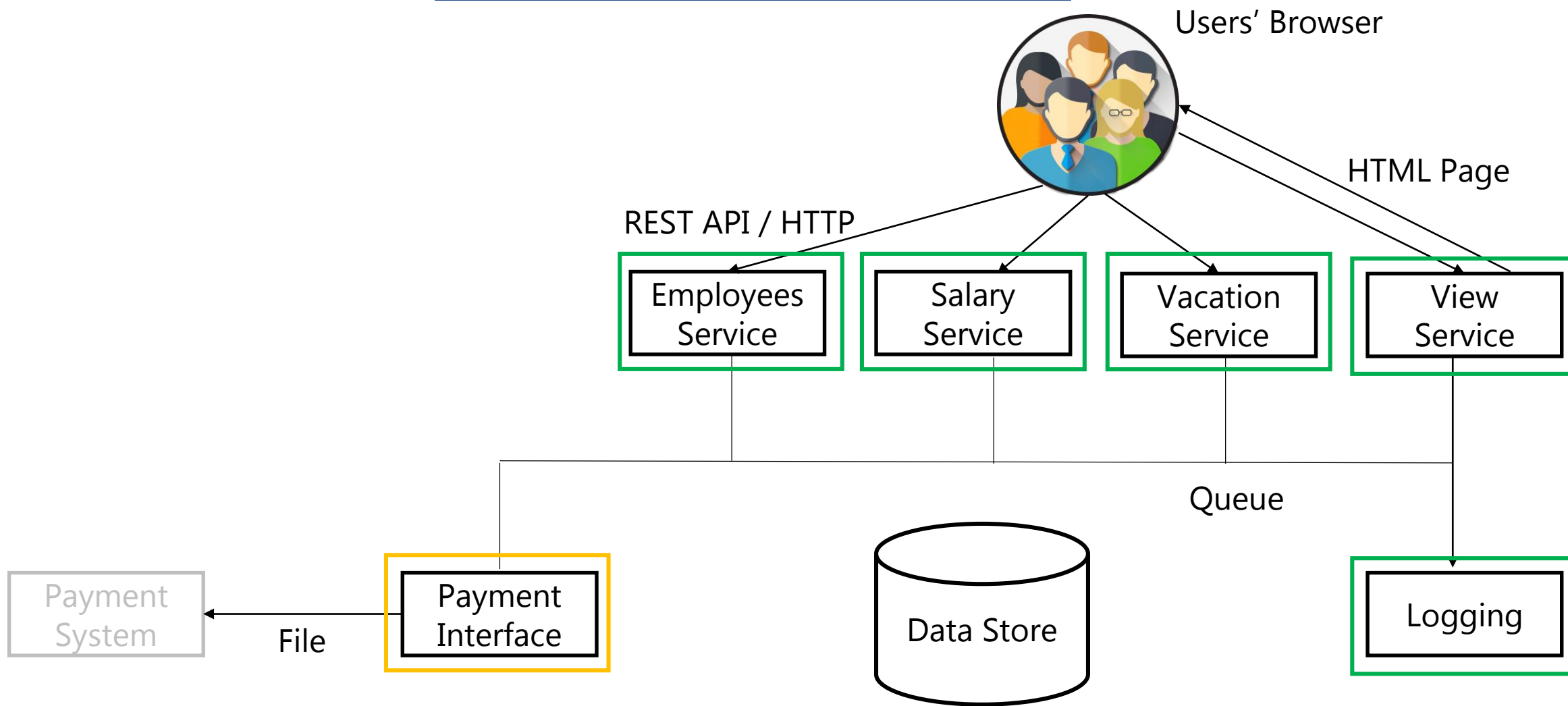
API

Functionality	Path	Return Codes
Set available vacation days	PUT /api/v1/vacation/{ <i>empid</i> }	200 OK 404 Not Found
Get available vacation days	GET /api/v1/vacation/{ <i>empid</i> }	200 OK 404 Not Found
Reduce vacation days	POST /api/v1/vacation/{ <i>empid</i> }/reduction	200 OK

Vacation Service Redundancy



Components








Payment Interface

What it does:

- Queries the database once a month for salary data
- Passes payment data to the external payment system

Application Type

- Web App & Web API 
- Mobile App 
- Console 
- Service 
- Desktop App 

Technology Stack



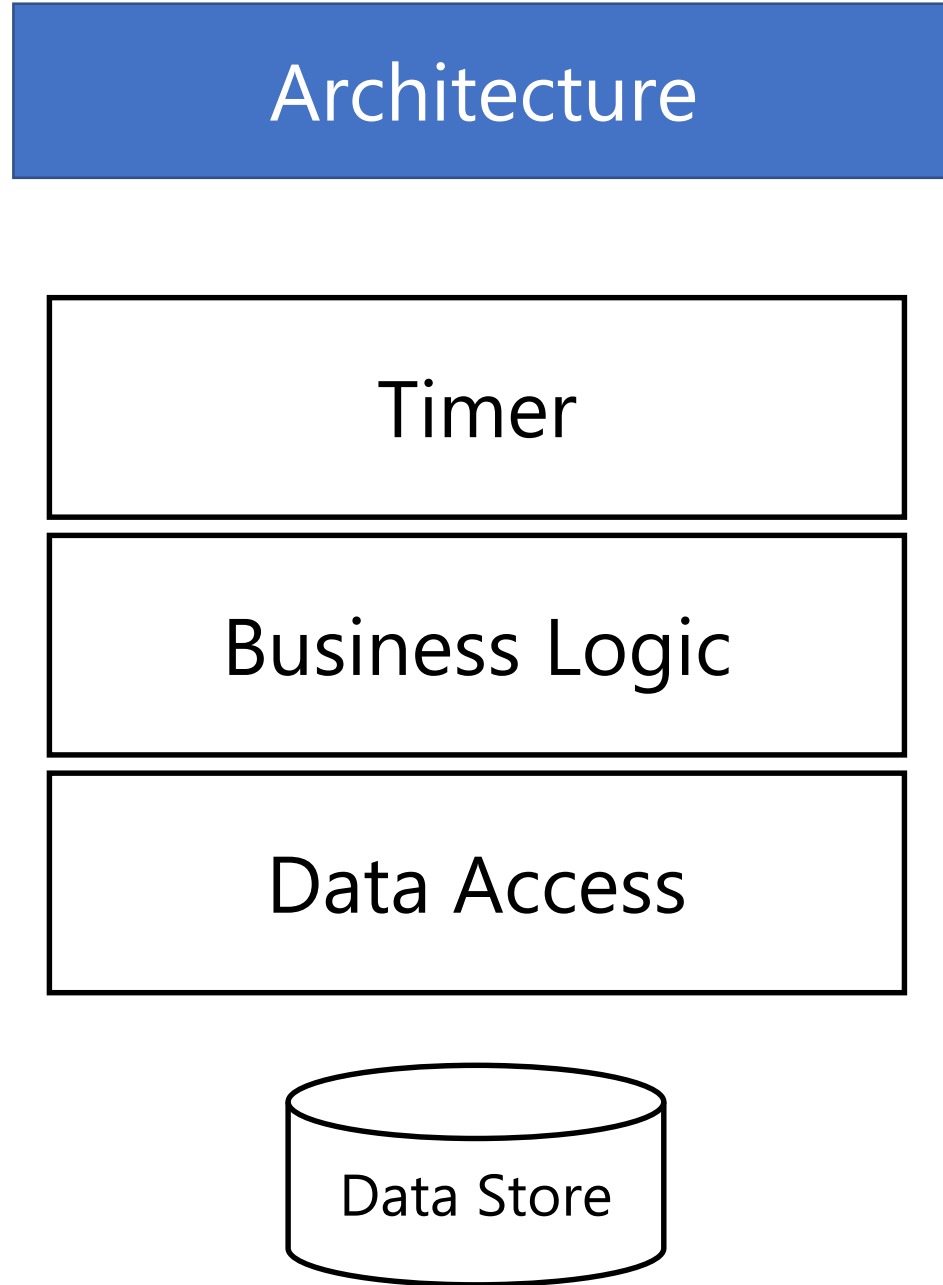
Architecture

Timer

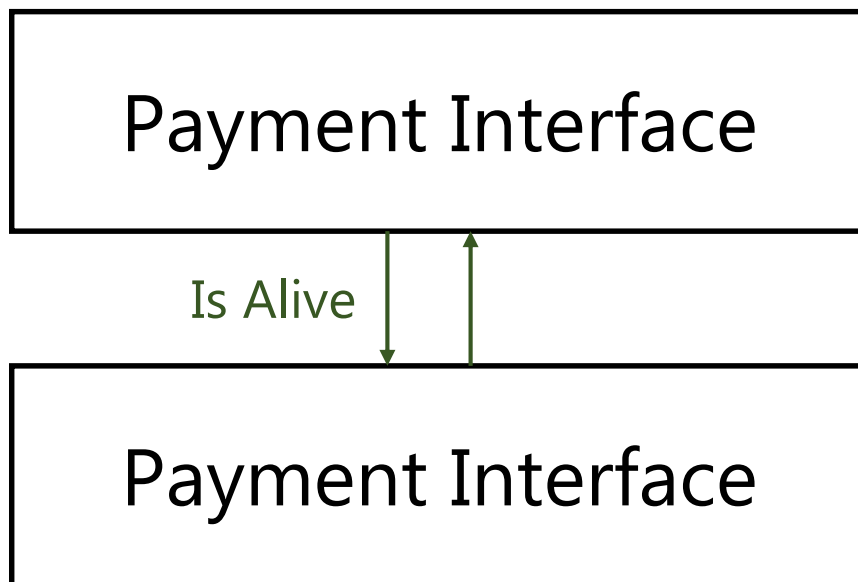
Business Logic

Data Access

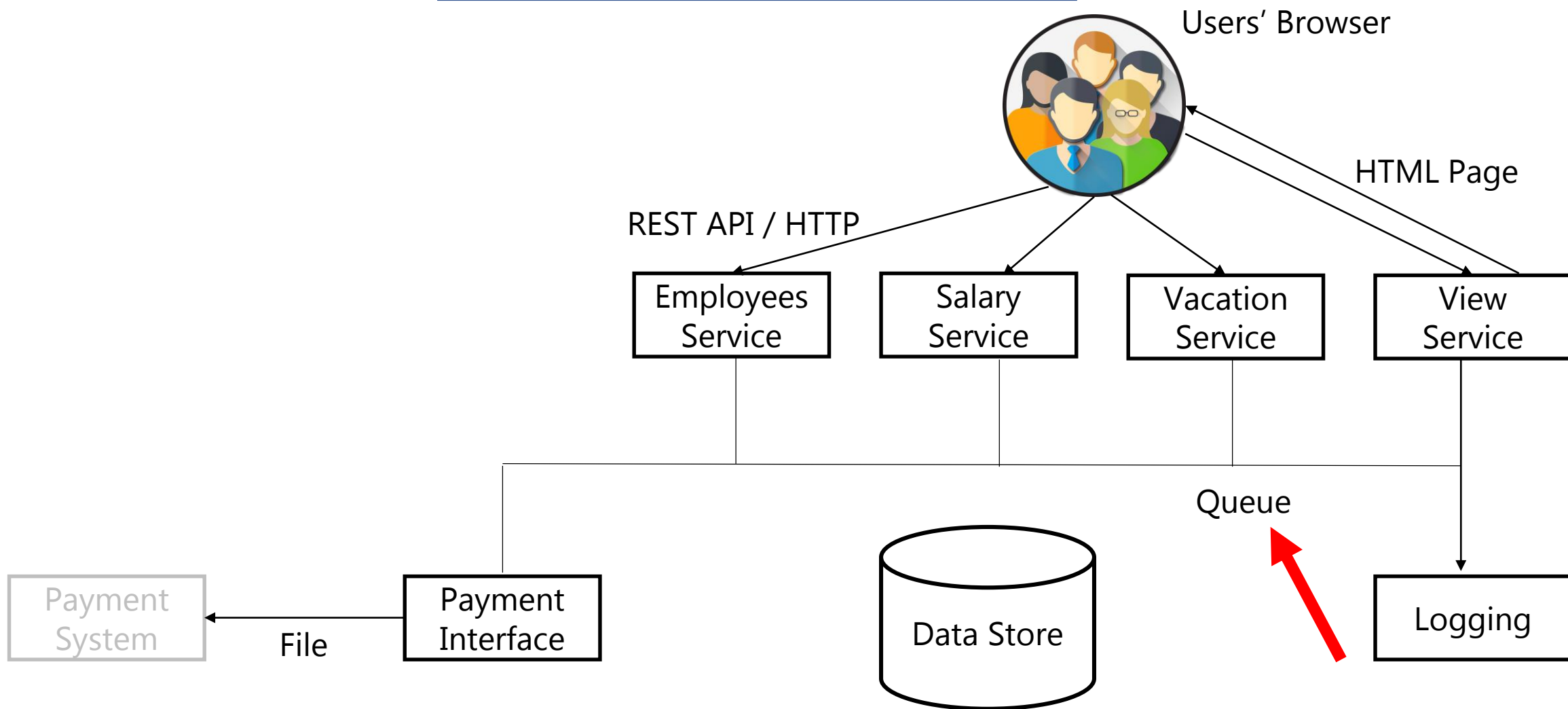
Data Store



Payment Interface Redundancy



Messaging



Technology Stack – Queue

Queue Alternatives:

~~Self Developed~~



RabbitMQ



Kafka

Alternative	Description	Pros
Rabbit MQ	General purpose message-broker engine	Easy to setup Easy to use

Alternative	Description	Pros
Rabbit MQ	General purpose message-broker engine	Easy to setup Easy to use
Apache Kafka	Stream processing platform	Perfect for data intensive scenarios

Technology Stack – Queue

Queue Alternatives:

Self Developed



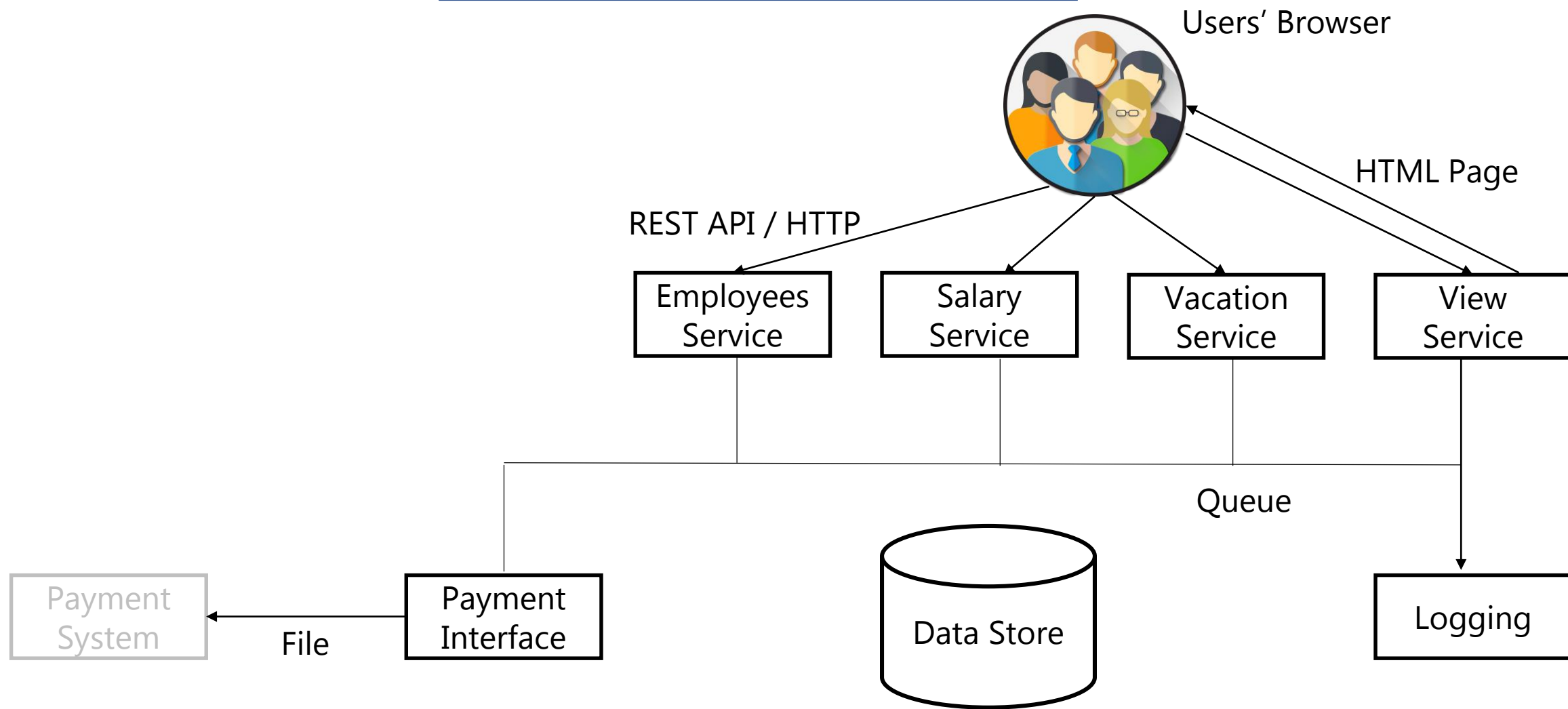
RabbitMQ



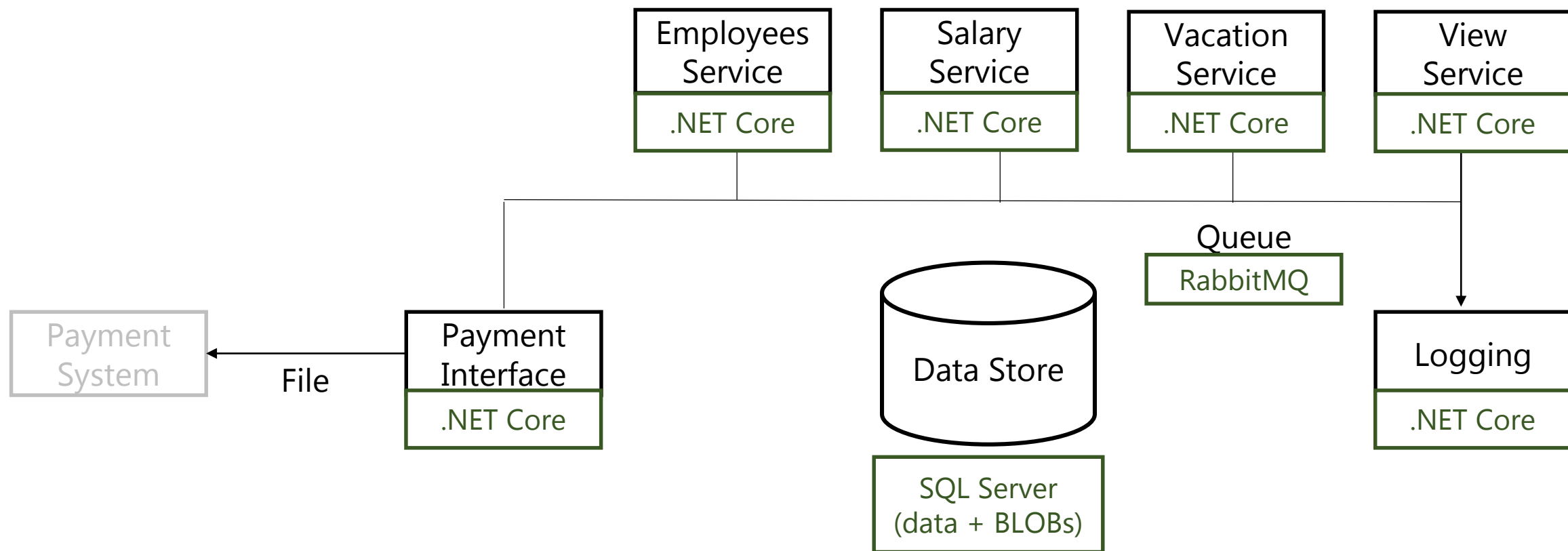
Kafka

- No data stream involved
- Easy to use

Logic Diagram



Technical Diagram



Physical Diagram

