‏23 מרץ 2025

Keshet System Design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Date** | **Update** | **Contents** | **Version** |
| 1 | 21/3/2025 |  | Oron Tamir | 1.0.0 |
|  |  |  |  |  |
|  |  |  |  |  |

**Contents**

[1. General: 2](#_Toc193612826)

[2. High-level architecture: 2](#_Toc193612827)

[2.1. Main components: 2](#_Toc193612828)

[2.2. Diagram: 4](#_Toc193612829)

[2.3. To be continue: 4](#_Toc193612830)

## General:

The system supports displaying invoices from an existing platform and includes the following features:

* Efficient retrieval of invoice data.
* A list of invoices with search and filtering capabilities (e.g., by invoice name, supplier name, or dates).
* Real-time notifications when there are changes to invoice data.
* Direct navigation to the invoice details page upon clicking an invoice.

## High-level architecture:

### Main components:

#### User Interface (Frontend):

A web application can be built using frameworks like React, Angular, or Vue.js, which will display a list of invoices. It should include features for searching, filtering, and viewing specific invoices. Additionally, the application needs to support two display modes: a list view and a minimized PDF view (partial view).

#### API / Backend Layer:

The application will utilize a RESTful or GraphQL API to connect with the existing systems for data retrieval, searching, and filtering. Even though the task does not require a direct connection to NestJS at this point, the design should accommodate the future integration with a real API.

#### Databases:

Choose either an SQL or NoSQL database to hold details like invoice ID, name, date, and vendor information

#### Storage:

Save invoice PDF files using a storage solution, such as AWS S3 or local storage. For simplicity in the current task, all invoices can reference the same PDF document.

#### Notification Service:

Implement a real-time notification system using WebSocket or a pub/sub method (like Redis or Socket.IO). This feature will ensure the user interface updates automatically whenever changes are made in the invoice system.

#### Search and Filtering Engine:

The application should support a free text search by invoice or supplier name and include date-based filtering options. If the volume of data increases, consider integrating a specialized search engine like Elasticsearch to enhance search and filtering performance.

### Diagram:

A diagram of a cloud

Description automatically generated

### To be continue:

#### Caching:

Implement caching solutions like Redis to enhance search speed and minimize database load. The system can respond faster to user queries by caching frequently accessed data, significantly improving overall performance.

#### Load Balancing and Microservices:

Consider architecting the application using a microservices approach, where different functionalities are divided into separate services (such as an invoice management service and an alert service). This design facilitates easier management and scaling of individual components and makes it simpler to enhance or modify specific features without affecting the entire system.

#### Search Optimization:

As the database grows, integrating advanced search engines (like Elasticsearch) can significantly improve the response times and accuracy of the free search and filtering capabilities. This ensures that users can efficiently find the information they need, even when dealing with large datasets.