Software Applications Development
WorldSkills 2026 National Competition
HUNGARY

Round 1

Submitted by:
Skills IT

WS2026_HU_TP09_R1_EN

Contents

1. Introduction

You used to work as a freelance software developer, but now you've applied for a job as a developer at a large software development company. The company's management wants to test your skills, so as part of the recruitment process, they asked you to develop prototype applications for a **Belle Croissant Lyonnais** company.

In the prototype, all you need to develop is an **import script**, a **dashboard** and a **searchable list** of customers. The dashboard helps the operators to analyse the data. The list contains the customers and allows the operators to easily search information in them.

Your API must use a JSON Server stable version (v0.17.4) as its data provider.

Your UI has to use the provided **logo**.

Finally you have to create a short **demo video** to show your presentation skills.

1.1 Description of project and tasks

Your task is divided into four parts.

1. In the first part, you will develop a script to import functions and handle errors.
2. In the second part, you have to create a web based dashboard with a custom REST API.
3. In the third part, you will develop a searchable list for customers.
4. In the fourth part, you have to create a demo video.

1.2 Database

1. Use the **`database.json`** from the assets folder
2. Run JSON Server (port 3000): **`json-server database.json`**

Field Name	Data Type	Description
id	`string` (UUID)	Unique identifier for each customer.
firstName	`string`	Customer's first name.
lastName	`string`	Customer's last name.
age	`float`	Customer's age in years (decimal format).
gender	`string`	Customer's gender (e.g., "M" for male, "F" for female, or empty if not provided).
postalCode	`string`	Postal code of the customer's location.
email	`string`	Customer's email address.
phone	`string`	Customer's phone number (may include incomplete or partial numbers).
membership	`string`	Customer's membership level (e.g., "bronze", "silver", "gold").
joinedAt	`string` (ISO Date)	Date when the customer joined (YYYY-MM-DD format).
lastPurchase	`string` (ISO Date)	Date of the customer's last purchase (YYYY-MM-DD format).
totalSpending	`float`	Total amount spent by the customer.
averageOrderValue	`float`	Average value of a single order.
frequency	`float`	Average number of purchases per year.
preferredCategory	`string`	Customer's preferred product category (e.g., "Bread", "Pastries", "Macaron", "Tarte").

1.3 How to submit your work

1. You have to share your work in a private GitHub repo as described in the README file of the test project GitHub repo (https://github.com/skillsit-hu/ws2026-s09-hu-r1).
2. Share a link with us in email (ws2026s09@skillsit.hu) from where we can download the executable files and code to test your solutions.
3. Your **README** file has to contain instructions on how to start your backend and frontend. We prefer you provide executable files or a built/deployed solution.

2. Part 1 – Import script

2.1 Command Line Tool

Create a command-line application that can read a `customers.csv` file and transform it into the desired format. The application should **log errors**, **save the cleaned data to a file**, and load it into the database using the **JSON Server API**.

2.2 Field Validations & Transformations

2.2.1 Age (age)

- Convert float to int.
- If the value is invalid, set it to an **empty string** ("").

2.2.2 Email (email)

- Convert all email addresses to **lowercase**.
- Validate email format (must contain '@' and '.').
- If invalid, set to an **empty string**.

2.2.3 Gender (gender)

- Expected values: "M", "F".
- If the value is not "M" or "F", set it to an **empty string**.

2.2.4 Phone Number (phone_number)

- Remove all non-numeric characters.
- Ensure a **minimum length of 10 digits**.
- If shorter than 10 digits, set to an **empty string**.

2.2.5 Membership (membership)

- Convert all value to **lowercase**.
- Normalize membership values as follows:

Original Value	Converted To
basic	bronze
silver	silver
gold	gold

- If the value does not match any of these, set it to an **empty string**.

2.2.6 Join Date & Last Purchase Date (joinedAt, lastPurchaseAt)

- Convert to YYYY-MM-DD format.
- Ensure the date is within the **2000-2025** range.
- **Last purchase date (lastPurchaseAt) must be later than joinedAt**.
- If any validation fails, set the date to an **empty string**.

2.2.7 Preferred Category (preferredCategory)

- If the value is one of Unknown, TBD, To Be Determined, N/A, replace it with an **empty string**.

2.2.8 Churned (churned)

- Convert the values as follows:

Original Value	Converted To
Y, yes, 1	true
N, no, 0	false
Any other value	**Empty String**

2.3. Error Handling & Reporting

2.3.1 Error Handling

- If a field fails validation, replace it with an **empty string** ("").
- Invalid rows should **not** be discarded but included in the output with empty fields.

2.3.2 Error Reporting

A separate error log file (error_report.csv) must be generated containing:

- The **original row number**.
- A **list of errors** in that row.

Example format:

Row Number	Error Description
12	Invalid email format
34	Phone number too short
50	Last purchase date earlier than join date

3. Part 2 – Dashboard

In this section, you need to create a web-based Dashboard for the UI to help operators visualize the customers' data.

3.1 Dashboard API

Design and develop your own API that can generate the specified values below and return them in a format that is understandable for the frontend. The API should query data through the interface provided by **JSON Server**.

3.1.1. Overview Metrics (Top Panel)

- **Total Customers**
- **Average Age**
- **Most Frequent Purchase Category**
- **Total Purchase Value**
- **Average Order Value**
- **Purchase Frequency (per year)**

3.1.2. Demographics

- **Gender Distribution** (Pie Chart)
- **Membership Levels Distribution** (Pie Chart)

3.1.3. Purchase Behavior

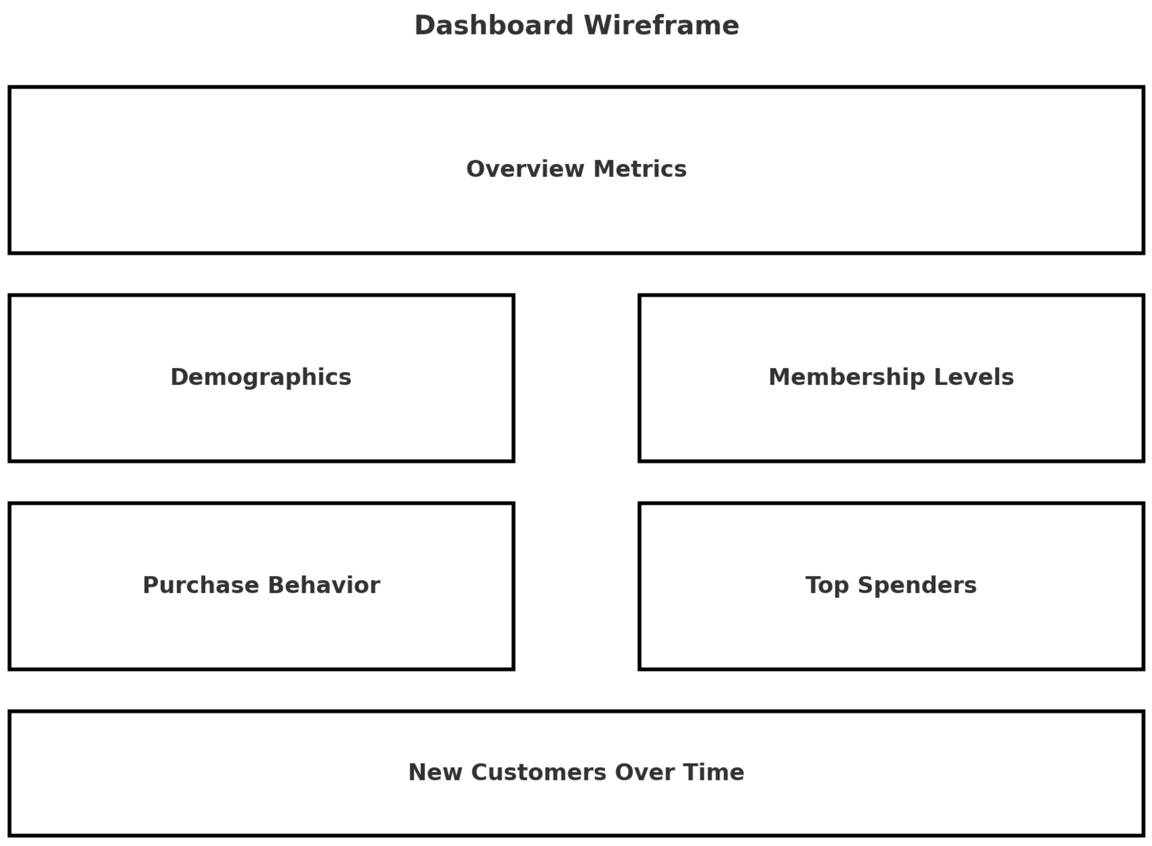
- **Most Purchased Categories** (Bar Chart)
- **Top Spenders (Highest Total Spend)** (Top 10 Table)

3.1.4. Trends Over Time

- **New Customers Over Time** (Line Chart showing yearly trends)

3.2 Dashboard UI

Design and develop the Dashboard interface based on the provided wireframe.



4. Part 3 – Customers list

In this section, you have to extend the dashboard application to create a list where operators can easily search customers.

The list should contain all available information for each customer. The operator may be able to **filter** and **sort** the data. You also have to implement **pagination** functionality where the page size has to be **25**.

The solution is in your hands, there is no visual/technical guideline to follow, but you have to use **JSON Server existing endpoints**.

5. Part 4 – Demo video

Create a demo video showcasing the completed solution. The video should include the following sections:

- Import process and error handling
- Dashboard API
- Dashboard UI demonstration
- Customer list interface with sorting and filtering

Make the demo video available to us through a video-sharing platform.

6. Additional information

- Some media, icons and text have been provided for you in the media files. You are free to use these, but you can also create your own, as long as the application is still fit for purpose. **You should not use any other media files (e.g. downloaded videos, images, icons, etc.)**.
- Clean code and user interface accessibility are also important considerations.
- Do not hardcode API responses as another database will be used for testing.