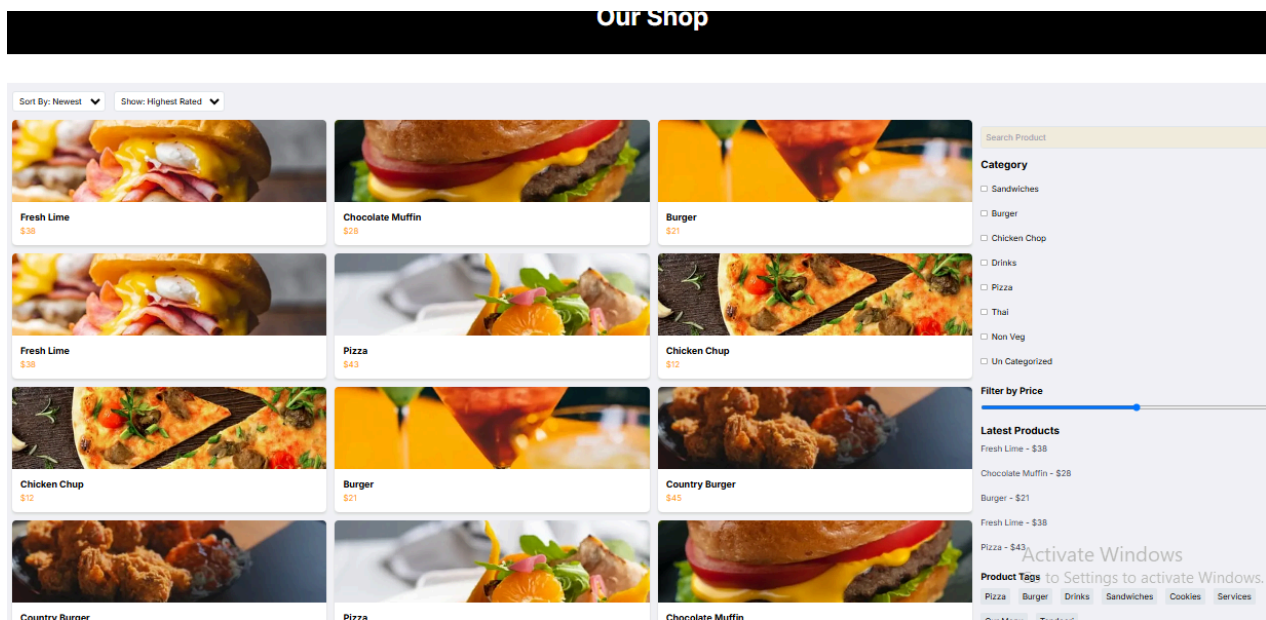
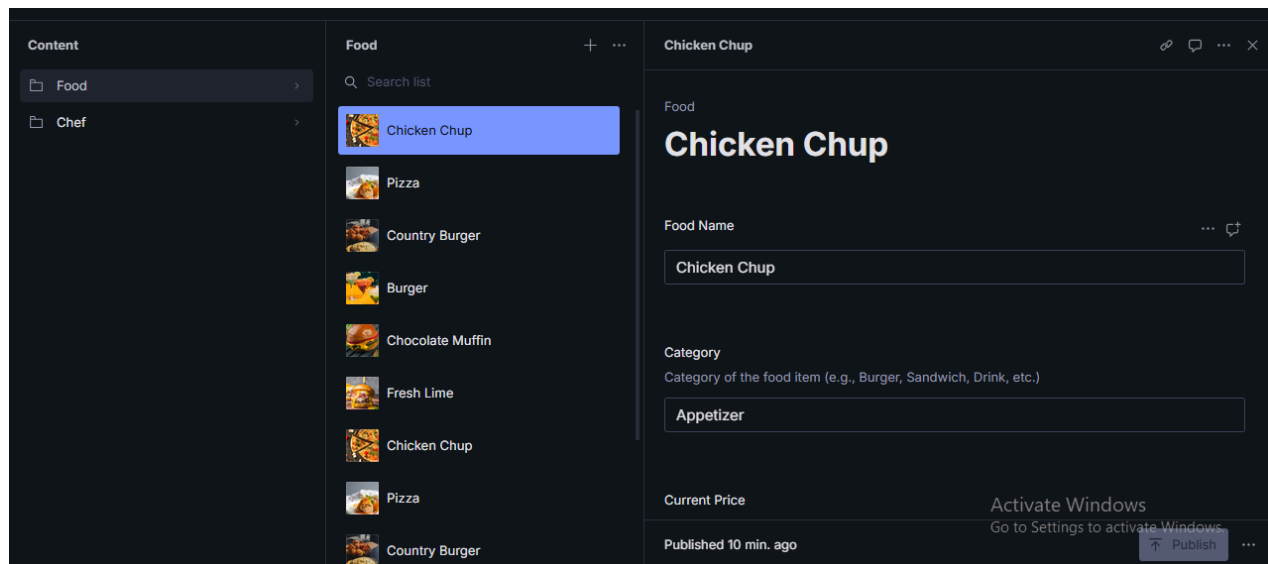
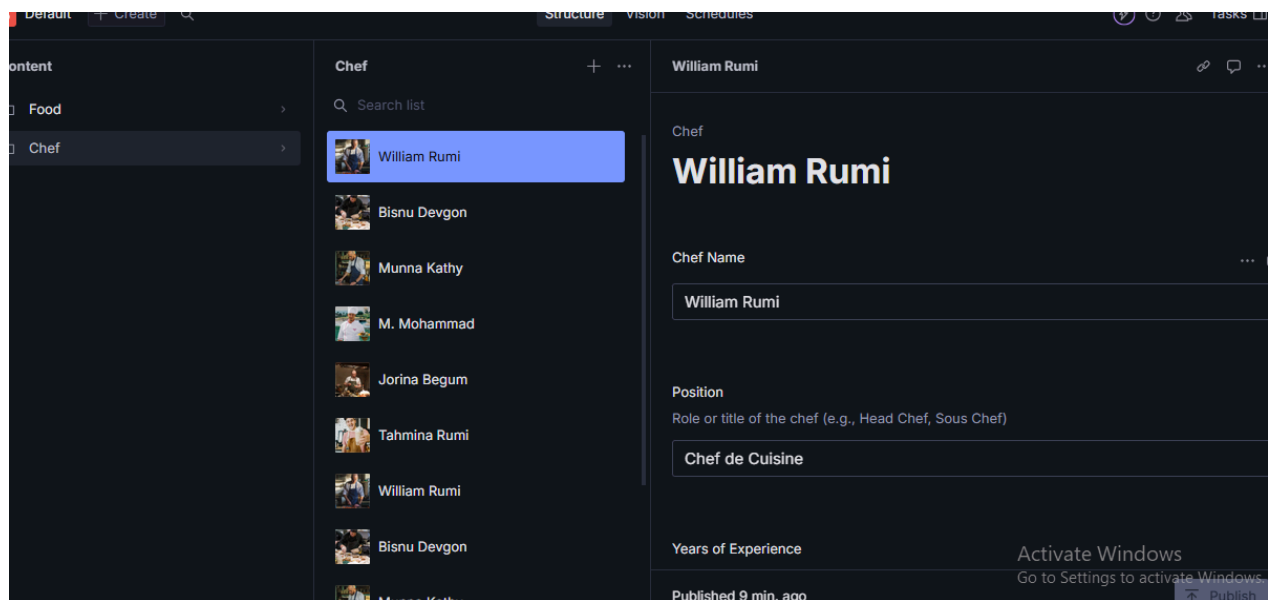


Screen Shots

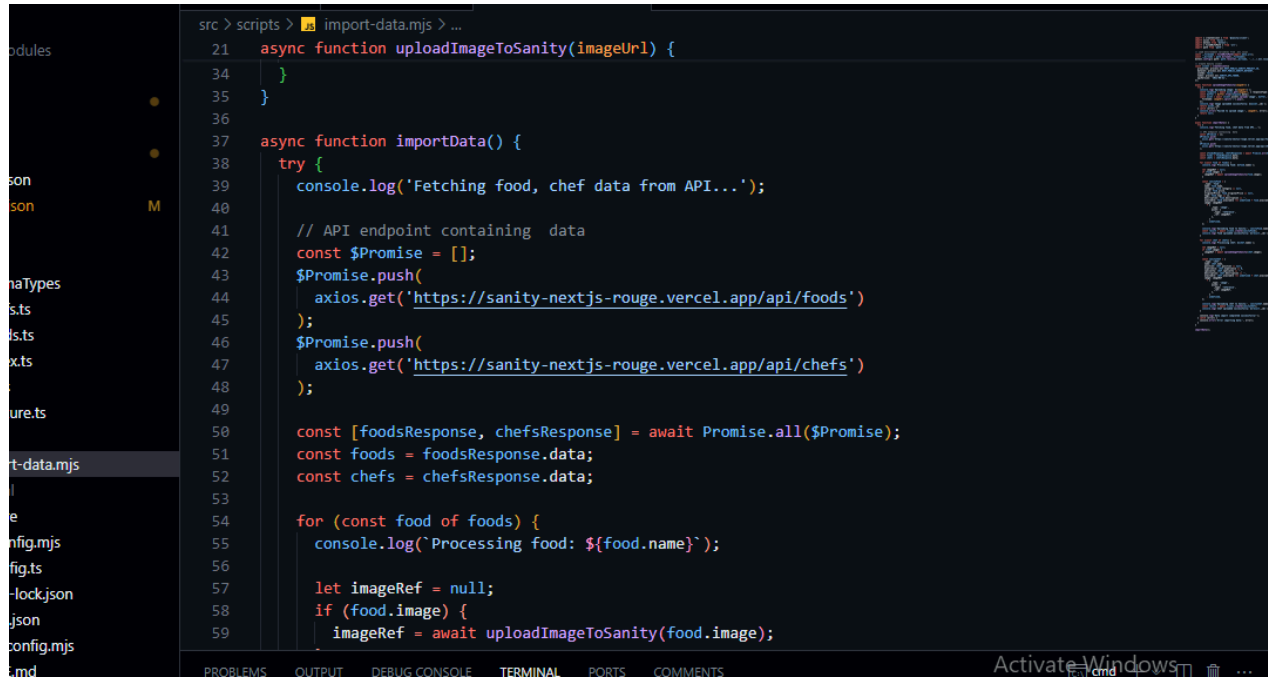
1)Data successfully displayed in the frontend:



2)Populated Sanity CMS fields:



3)Data Migration Script:



The image shows a screenshot of a Visual Studio Code editor window. The left sidebar displays a file explorer with a list of files including 'modules', 'son', 'son', 'naTypes', 's.ts', 's.ts', 'x.ts', 'ure.ts', 't-data.mjs', 'e', 'nfig.mjs', 'fig.ts', 'lockjson', 'json', 'config.mjs', and 'md'. The main editor area shows the content of 'import-data.mjs'. The script defines an asynchronous function 'uploadImageToSanity' and another asynchronous function 'importData'. The 'importData' function uses 'axios' to fetch data from two API endpoints: 'https://sanity-nextjs-rouge.vercel.app/api/foods' and 'https://sanity-nextjs-rouge.vercel.app/api/chefs'. It then processes the fetched data, logging the food names and uploading their images to Sanity. The bottom status bar shows 'Activate Windows' and 'cmd'.

```
src > scripts > import-data.mjs > ...
21  async function uploadImageToSanity(imageUrl) {
34  }
35  }
36
37  async function importData() {
38    try {
39      console.log('Fetching food, chef data from API...');
40
41      // API endpoint containing data
42      const $Promise = [];
43      $Promise.push(
44        axios.get('https://sanity-nextjs-rouge.vercel.app/api/foods')
45      );
46      $Promise.push(
47        axios.get('https://sanity-nextjs-rouge.vercel.app/api/chefs')
48      );
49
50      const [foodsResponse, chefsResponse] = await Promise.all($Promise);
51      const foods = foodsResponse.data;
52      const chefs = chefsResponse.data;
53
54      for (const food of foods) {
55        console.log(`Processing food: ${food.name}`);
56
57        let imageRef = null;
58        if (food.image) {
59          imageRef = await uploadImageToSanity(food.image);
60        }
61      }
62    } catch (error) {
63      console.error('Error importing data:', error);
64    }
65  }
66
67  importData();
```