

Aleksi Leinonen

## **ADVANTAGES OF ADHERING TO CODING STANDARDS**

# **ADVANTAGES OF ADHERING TO CODING STANDARDS**

Aleksi Leinonen  
Bachelor's thesis  
Autumn 2023  
Information Technology  
Oulu University of Applied Sciences

## ABSTRACT

Oulu University of Applied Sciences  
Information technology, Option of Software Development

---

Author: Aleksi Leinonen  
Title of thesis: Advantages of adhering to coding standards  
Supervisors: Meija Lohiniva, Esa-Matti Sarjanoja  
Term and year when the thesis was submitted: Autumn 2023  
Number of pages: 30

---

This thesis follows the author's work for an eight-week period at Nokia, mainly focusing on the development of the "Global Output" project. The project was developed using React and TypeScript for the front-end, while Python was used for data mining and validation. The project is documented through a diary-style narrative.

It also investigates the advantages of adhering to coding standards and best practices, which are briefly reflected in the author's diary entries. The work also explores how these standards contribute to improved code quality.

While the project was not finished as part of this thesis work, substantial progress was achieved, bringing it close to completion. The thesis concludes with the author's thoughts on the personal and professional growth gained throughout the process.

---

Keywords: Nokia, React, TypeScript, Python, MongoDB, diary, software development

# TABLE OF CONTENTS

VOCABULARY .....	6
1 INTRODUCTION .....	7
2 CODING STANDARDS & BEST PRACTICES .....	8
2.1 Importance of Coding Standards .....	8
2.2 Common Coding Standards .....	8
2.2.1 Commenting.....	9
2.2.2 Reusability .....	9
2.2.3 Naming conventions for variables .....	10
2.2.4 Indentation, formatting and spacing .....	11
2.2.5 Error handling .....	12
3 DIARY ENTRIES .....	13
3.1 WEEK 1.....	13
3.1.1 Sprint planning .....	13
3.1.2 UI & UX Improvement .....	13
3.2 WEEK 2.....	14
3.2.1 Weekly meeting .....	14
3.2.2 Global Output project meeting .....	15
3.2.3 Making the python scripts .....	15
3.3 WEEK 3.....	19
3.3.1 Sprint planning .....	19
3.3.2 Data parsing and validation.....	19
3.4 WEEK 4.....	20
3.4.1 Plan for the week .....	20
3.4.2 Developing the data miner .....	20
3.4.3 Aggregation Pipeline .....	21
3.5 WEEK 5.....	22
3.5.1 Plan for the week .....	22
3.5.2 Kaizen task .....	23
3.6 WEEK 6.....	23
3.6.1 Finishing the Kaizen task .....	23
3.6.2 Invalid data.....	25

3.7	WEEK 7.....	25
3.7.1	Global Output prototype view.....	25
3.8	WEEK 8.....	26
3.8.1	Filter component .....	26
3.8.2	Global Output table view .....	26
4	REFLECTION.....	28
	REFERENCES .....	29

## VOCABULARY

JavaScript	A programming language
TypeScript	Superset of JavaScript
React	A JavaScript library
MongoDB	Mongo Database
UI	User Interface
UX	User Experience
GraphQL	A query language
SQL	Structured query language
CSV	Comma separated values

# 1 INTRODUCTION

When working within a team, teamwork becomes an important factor. A key aspect of effective teamwork is developing code that is understandable by other team members. This is where following the same coding standards and practices becomes essential. This thesis explores the impact of coding standards on code improvement and how they contribute to teamwork.

In the diary section of the thesis, the author's work is described over an eight-week period. The diary section is divided and summarized by weeks, consisting of smaller day to day work tasks while simultaneously following development of the "Global Output" -project.

The author had worked at Nokia as a summer trainee for 4 months before the start of the thesis. Throughout this period, the work primarily focused on developing Nokia's internal webtool, named Foresight. Mainly the tasks consisted of improvement requests made by end-users and addressing minor bug fixes.

## 2 CODING STANDARDS & BEST PRACTICES

Coding standards consist of a set of guidelines and best practices that software developers should follow when writing code. Coding standards are usually specific to a particular programming language or framework and these standards and practices often slightly vary from team to team, but there are many universally used practices. The reason why software developers should follow the guidelines and practices is to ensure that code is written in a consistent way. [1]

A team can create their own coding standards or choose an existing one if it suits their needs. However, it is important that the standards are documented and well described in the documentation. [1]

### 2.1 Importance of Coding Standards

The goal of setting and following coding standards is to ensure that the code will be readable, expandable and maintainable. When the code is written within the standards, all team members should be able to read and follow the code, thus each team member should be able to detect possible problems within the code or prevent them completely. If the code is readable and understandable by other developers in the team, it saves time and expense when a bug needs to be fixed on older code. [2]

Following the coding standards makes onboarding of new team members more efficient. If codebase is consistent and not overly complex, it allows the new team member to understand and get used to the codebase faster. [1]

### 2.2 Common Coding Standards

The more common coding standards for JavaScript and Python programming languages are listed and briefly described below.



## 2.2.1 Commenting

Commenting code contributes to code readability. It helps developers to maintain the code by providing clear descriptions of the code's purpose. Comments are also helpful when debugging for bugs or errors. [3]

```
// Counts the number of items with the same item type.
function countItemsByItemType(items) {
  const countByItemType = {}
  if (Array.isArray(items)) {
    items.forEach((item) => {
      if (!item || !item.itemType) {
        console.log("Invalid item:", item)
        return
      }

      countByItemType[item.itemType] = (countByItemType[item.itemType] || 0) + 1
    })
  }
  return countByItemType;
}
```

FIGURE 1. Commenting example.

Comments should be clear and specific, while giving enough information and not making it too complicated. Refrain from adding comments stating the obvious (figure 1). [1]

## 2.2.2 Reusability

Code reusability is an important part of coding standards. Code reusability stands for creating high quality code that can be reused in different parts of the project. It helps to reduce the amount of code that needs to be written and maintained. By reusing code, developers save time and resources while still producing bug-free quality code. Reusing well-written code can also work as a learning resource for beginner developers. [4]

### 2.2.3 Naming conventions for variables

Consistent naming conventions enhance code readability, it helps developers to understand the purpose and functionality of variables and functions. Naming conventions also contribute to a unified codebase, making teamwork easier. [5]

Camel casing is the generally accepted naming convention for JavaScript. In camel casing, the first word starts with a lowercase letter, and every following word starts with an uppercase letter (figure 1). The naming convention is called camel case because the pattern of capital and lowercase letters is said to resemble the humps on the back of a camel. [6]

```
5
6  const camelCaseExample = {
7      firstName: "John",
8      lastName: "Doe",
9      placeOfBirth: "Finland"
10 }
11
```

FIGURE 2. Camel case example.

Snake case is another naming convention used in programming, it is widely used in Python, Ruby, and many other scripting languages [7]. It is a commonly used naming convention for naming variables and functions in Python [8]. In snake case, words are written in lowercase letters with underscores (\_) between the words (figure 2). Snake case is the recommended naming convention for Python variables, functions and modules in the Python's official style guide [9].

```
5
6  snake_case_example = {
7      'first_name': "John",
8      'last_name': "Doe",
9      'place_of_birth': "Finland"
10 }
11
```

FIGURE 3. Snake case example.

Often software developers are required to write code in multiple programming languages. By following the language specific coding standards, we can ensure that everyone is using the same naming conventions for each programming language. [1]

## 2.2.4 Indentation, formatting and spacing

Having common rules for code formatting is important. Consistency in code style and formatting contributes to the uniformity of the code base [4]. There are many automated formatters available that automatically format the code according to the coding standards [10].

JavaScript does not require proper indentation if the code is syntactically correct and follows the rules of JavaScript, it will run. However, it is important to follow proper indentation practices; otherwise, the code can easily become hard to read [11]. In Python proper indentation is crucial, as indentation errors will occur if not done properly as seen in Figure 4. Python uses indentation for defining the structure and scope of code blocks [12]. As seen from Figure 4 and Figure 5, a small difference in the indentation can result in errors.

```
3   name = "John Doe"
4
5   if name == "John Doe":
6       print("Hello John")
7   else:
8       print("Who are you?")
9
print("Hello John")
^
IndentationError: expected an indented block after 'if' statement on line 5
```

FIGURE 4. Indentation error example.

```
3   name = "John Doe"
4
5   if name == "John Doe":
6       |   print("Hello John")
7   else:
8       |   print("Who are you?")
9
```

*FIGURE 5. Proper indentation.*

Paying attention to indentation, formatting, and spacing is essential for writing code that is readable and maintainable. Following a consistent coding style helps to create clean and understandable code. [10]

### **2.2.5 Error handling**

Error handling is an important part of coding standards, it helps identifying and resolving errors in the code. Including guidelines for error handling as a part of coding standards helps to maintain a consistent code quality. Ensuring the code's ability to handle errors is essential for improving the overall quality of the code. [13]

## **3 DIARY ENTRIES**

### **3.1 WEEK 1**

#### **3.1.1 Sprint planning**

At Nokia, the author's team holds a sprint planning every other Monday, during the sprint planning everyone is assigned eight points worth of tasks. During the sprint planning, it was decided that part of the team would start working on a project called "Global Output", including the author. It was decided that the project would start within the second week of sprint.

#### **3.1.2 UI & UX Improvement**

The week started with a small task, designated to improve the user experience on Foresight. The goal was to improve readability on two different views.

The main view has a large list of Digital Twin models, where each model contains a list of properties and their attributes. The listed attributes take up a lot of space, and the user rarely wants to see the attributes. A toggle button that hides the attributes was created, as seen on top of figure 6. Originally the plan was to create one toggle button for the list, but later it was decided that there should be similar buttons for individual properties that have attributes as well (figure 7).

<span>Hide Attributes</span> <span>Create New Model</span>		
Contents *		
Property	Example ( <b>example</b> )	Object (test,test2,date,asdasd)
Property	Hex Color ( <b>hex_color</b> )	hex-color
	Hex Color: Data Timestamp ( <b>__hex_color__dateTimeStamp</b> )	dateTime
Property	Integer ( <b>integer</b> )	integer
Property	Property ( <b>property</b> )	Array(Object)
	Property: Data Timestamp ( <b>__property__dateTimeStamp</b> )	dateTime
	Property: KPI ( <b>__property__kpi</b> ) = true	boolean

FIGURE 6. Attributes visible.

<span>Show Attributes</span> <span>Create New Model</span>		
Contents *		
Property	Example ( <b>example</b> )	Object (test,test2,date,asdasd)
Property	<input checked="" type="checkbox"/> Hex Color ( <b>hex_color</b> )	hex-color
Property	Integer ( <b>integer</b> )	integer
Property	<input checked="" type="checkbox"/> Property ( <b>property</b> )	Array(Object)

FIGURE 7. Attributes hidden (toggle button now visible for individual properties).

As the other view was built in a similar way, the author was able to reuse the code he had written earlier. This is a good example of the importance of writing reusable code, as mentioned in the coding standards section.

## 3.2 WEEK 2

### 3.2.1 Weekly meeting

As every other week starts with a sprint planning, correspondingly every other Monday starts with a team meeting. In the meeting the team goes through what was done during the previous week, and what each individual plan for the current week is. Because only part of the team is involved in

the Global Output project, a decision was made to hold another meeting with everyone included in the project from the team.

### **3.2.2 Global Output project meeting**

Development of the Global Output project was started from scratch. Some preliminary planning regarding the project had been done, but nothing concrete. The goal of the Global Output project is to create a new view to Foresight, which displays statistics comparing the quantity of shipped items with the planned amount for each day. At the beginning of the project, a lot of data validation and research had to be done, to qualify what data is accurate enough for the use case.

During the meeting each team member's role for the project was discussed, and what would be the author's first task regarding it. Since the project was still in the early stage, it was decided that the first task would be comparing and analyzing datasets exported from SQL database. This had to be done in the early stages of the project, to ensure that the data is valid and accurate before continuing to other stages.

At the end of the meeting, rough planning was done regarding a data miner, that would extract the data from the SQL database. The extracted data will then be pushed into MongoDB, which will be utilized by the web tool.

### **3.2.3 Making the python scripts**

The first step in the data validation process was to compare two datasets exported from the SQL table over two consecutive days. The goal was to identify the differences between the datasets, by checking for serial numbers that exist in the previous dataset, but not in the newer one. After the distinctive serial numbers were found, they were stored into a separate dataset, as their absence from the newer dataset means that the item is complete. As the initial data validation showed promise for the data quality, the next step was to develop a Python script for automating the process.

It was decided that Pandas Python library would be used for the scripts, as it is already used in other Python scripts of the team. Pandas library is aimed specifically for data analysis and manipulation. As the author had limited experience in Python and data analyzing, the documentation of

Pandas was read before proceeding. After reading for a while, a function that filters out unnecessary data from the dataset was written, returning a list of dictionaries (figure 8). As seen in Figure 8, the Python code uses camel casing, which is not the recommended naming convention for Python. The author had been exclusively using JavaScript for the past year, which is why the wrong naming convention was used.

```
def fileReading2(file):
    units=[]
    with open(file) as file:
        # if wip_snapshot_details, then separator is , NOT ; and i»¿SN is SN
        datafile = pd.read_csv(file, sep=',', encoding='unicode_escape', low_memory=False)
        datafile.fillna('', inplace=True)
    for index, row in datafile.iterrows():
        if re.findall('47\d*', row["BasePartNumber"]):
            units.append(row)
    return units

units1=fileReading2(fileDay1)
units2=fileReading2(fileDay2)
```

FIGURE 8. Function to filter data.

The function “value\_comparison” in Figure 9, returns distinctive items whose serial number exists in the previous dataset but not in the newer dataset. This indicates that the item has been shipped. The function is written while using snake casing as the naming convention, which is the recommended naming convention for variables, functions and modules.


```
def value_comparison(file1, file2):
    units=[]
    df1 = pd.DataFrame(file1, columns = [SERIALNUMBER])
    df2 = pd.DataFrame(file2, columns = [SERIALNUMBER])
    set_a = set(df1[SERIALNUMBER].values)
    set_b = set(df2[SERIALNUMBER].values)
    values = set_a.difference(set_b)
    return list(values)
```

FIGURE 9. Function for value comparison.

After the script was run with data from multiple different dates, the results were promising. Hence, the decision was made to proceed with the next step, getting the ordered items amounts and adding them to a MongoDB collection. The item order amounts are listed in a large excel table, however the table structure is more complicated than in the previous one. A problem was encountered soon after the task was started. In the excel table, some of the columns and rows are grouped (figure 10). The Python script can access the grouped data only if it is toggled visible. The author searched



for a solution if it would be possible to have the grouped data visible by default when reading the excel file in Python. However, no viable solution was found. For now, it was decided that the excel table would be manually ungrouped before running the script, as it takes only a few seconds and is done once or twice per month.



FullYear	MonthandY	WeekNum												
2022														
10.2022			w41	w42	w43	w44	w45	w46	w47	w48	w49	w50	w51	w52
11.2022														
12.2022														

FIGURE 10 Excel table layout for years, months and weeks.

The layout of the excel table caused other challenges as well. The problem was to figure a way to get the month and year for each table cell, as in some columns they are empty (figure 10). The problem was solved by iterating through the table multiple times, which was inefficient. It was concluded that there must be a more efficient way to accomplish this. The author set up a meeting with the technical lead of the author's team and was instructed to read about lookup tables. The problem was quickly solved after the meeting, by utilizing lookup tables (figure 11). With populated lookup tables, the month and year of each item can be obtained by using the column index of the item.

```
test = datafile.drop(datafile.columns[[0, 1, 2, 3, 4]],axis = 1).iloc[0]
year_list = test.index.to_list()
month_list = test.to_list()

month_value = None
for month in month_list:
    if not pd.isnull(month):
        month_value = int(month.split(".")[0])
        month_lookup_table.append(month_value)
    else:
        if month_value is not None:
            month_lookup_table.append(month_value)

year_value = None
for year in year_list:
    if isinstance(year, int):
        year_value = year
        year_lookup_table.append(year)
    else:
        if year_value is not None:
            year_lookup_table.append(year_value)
```

FIGURE 11. Mockup code for showcasing the usage of lookup tables.

In some cases, one item description has multiple version numbers (figure 12). Each version number of the item is pushed into the MongoDB separately. Since the table is iterated row-by-row, in some cases the item description was empty.

245	ITEM DESCRIPTION 1	837321
246		732192
247		432123
248		843292
249		498432
250	ITEM DESCRIPTION 2	321342
251		543321
252		432123
253		433212

FIGURE 12. Example of the issue faced with item descriptions.

At first, the issue was attempted to be solved by overly complicated implementations. Eventually a simple solution to the problem was found (figure 13). By creating a variable outside of the iteration and reassigning a new value to it if the item description is not null. This way the variable updates only if the description changes.

```

for index, row in data.iterrows():
    if pd.notnull(row["Part Description"]):
        rowVal = row["Part Description"]
    if row[LOCATION] != location + ' Total':
        weekvalues = row.iloc[5:13]
        for week, value in weekvalues.items():
            if not pd.isnull(value):
                returnData.append({
                    'location': location,
                    'depot_desc': depot_desc,
                    'part_desc': rowVal,
                    'item_code': row["Item/Component"],
                    'week': week,
                    'amount': value })

```

FIGURE 13. Part description.

To finish the week, a function was implemented to calculate the number of weeks of data for each dictionary (figure 14).

```

def inner_func(table_index, week_number):
    if table_index < (len(week_lookup_table) - 1):
        if year_lookup_table[table_index] == year_lookup_table[table_index + 1]:
            return int(week_lookup_table[table_index + 1][1:]) - week_number
        else:
            weeks_in_year = isoweek.Week.last_week_of_year(year_lookup_table[table_index]).week
            return (weeks_in_year + 1) - week_number
    elif table_index == len(week_lookup_table) - 1:
        return 1

```

FIGURE 14. Function for calculating the weeks of data in one column group.

In the Excel table most of the expendable month columns had 4 sub-columns, for each week within the month. In some scenarios, the sub-columns are missing, and instead the data is summed into one column. This information is crucial when comparing and analyzing the data.

### 3.3 WEEK 3

#### 3.3.1 Sprint planning

The week started with a sprint planning. The status of the Global Output project, as well as the status of other ongoing projects within the team, was reviewed. It was decided that the author would continue working on the data validation process.

#### 3.3.2 Data parsing and validation

To further validate the SQL data, it was compared to an existing MongoDB data that is confirmed to be valid. The reason why the MongoDB data was not used in the Global Output project is that the database only contains data from the Oulu factory. With the list of serial numbers returned by the function in Figure 8, the data from MongoDB collection was manually exported for all the matching serial numbers. With the dataset exported from MongoDB, a function was developed to merge the two datasets together, creating a CSV file using the merged data. This was done with the “merge” method, found from the Pandas library [14]. From the returned Excel table, it can be seen if the MongoDB datetime falls between the two SQL datetimes (figure 15).

Excel Save DateTime	Serial Number	Factory	MongoDB DateTime	SQL DateTime 1	SQL DateTime 1
2023-09-05T12:44:58	SN23446902	Oulu	9/4/2023 13:13	2023-09-04 07:51:42:3610000 -04:00	2023-09-05 09:44:23:3610000 -04:00
2023-09-05T12:44:58	SN26324902	Oulu	9/4/2023 18:10	2023-09-04 07:51:42:3610000 -04:00	2023-09-05 09:44:23:3610000 -04:00
2023-09-05T12:44:58	SN25324921	Oulu	9/4/2023 16:20	2023-09-04 07:51:42:3610000 -04:00	2023-09-05 09:44:23:3610000 -04:00
2023-09-05T12:44:58	SN23447321	Oulu	9/4/2023 15:20	2023-09-04 07:51:42:3610000 -04:00	2023-09-05 09:44:23:3610000 -04:00
2023-09-05T12:44:58	SN23445423	Oulu	9/4/2023 13:15	2023-09-04 07:51:42:3610000 -04:00	2023-09-05 09:44:23:3610000 -04:00

FIGURE 15. Merged dataset.

The function still needed improvement, as the datetimes were not using the same time zone. It was easily fixed, by using the datetime module in Python. After fixing the issue, the author spent some time commenting, reformatting and cleaning the code. While the code was not meant to be used for anything other than testing the data, it was still important for the code to be clear for other team members of the author, in case of absence.

To finish the week, a Python script was developed. The script creates an excel file with shipped item amounts from the SQL table, along with completed item amounts from the MongoDB. The script will be run daily, and accuracy of the returned data will be manually verified.

### 3.4 WEEK 4

#### 3.4.1 Plan for the week

A great deal of validation has been done for the SQL data, ensuring its accuracy. As the data has proven to be accurate so far, it was decided that a data miner for the SQL data will be developed.

#### 3.4.2 Developing the data miner

From the beginning of the week, the author began developing a new data miner. The data miner checks if the SQL table has updated, and if so, it gets the data from the last 7 days. Queried data then is upserted to MongoDB collection. Upsert is combination of the terms "update" and "insert". Upsert updates the document if found in the collection, and if not, it creates a new document [15]. The data inserted to MongoDB by the miner will be used in the front-end. A problem was encountered shortly after, requiring permissions and an account for the SQL database to proceed, which will take some time. The data miner was developed as much as possible without access to the SQL database.

A local MongoDB environment was set up initially, to minimize the risks of faulty data getting inserted into the database. Quickly after, a problem was encountered, the author would need permissions and an account to the SQL database in order to continue, which would take some time. The data miner was developed as much as possible without having the access to the SQL database. By Tuesday the miner was at a point where it could push manually exported SQL data into the MongoDB. Technically no data mining was done at this point, but everything was working on the MongoDB side.

With data in the MongoDB collection used by the Global Output back-end, the author started working on the queries. A problem was faced when upserting the data, the upsert mutation always inserted a new document to the MongoDB, even if document for the same item already existed. The cause for the issue was that after the item document was inserted in the MongoDB, the datetime format changed. So, the next time the same item is upserted, the mutation fails to match the item, and inserts a new one instead. The issue was fixed by turning the datetimes into epoch timestamps.

### **3.4.3 Aggregation Pipeline**

To finish the week, a new aggregation pipeline for the MongoDB query was developed (figure 16). The aggregation pipeline enables stages to be chained together, with each stage processing the documents. The output of the last stage, is passed on to the next stage [16].

```

const pipeline: object[] = [
  {
    "$match": {
      "year": params.year,
      "month": params.month,
      "week": params.week
    }
  },
  {
    "$lookup": {
      from: 'globalOutput',
      let: {
        "year": "$year",
        "month": "$month",
        "week": "$week",
        "siteID": "$siteID",
        "item": "$item"
      },
      pipeline: [
        {
          "$match": {
            "expr": {
              "and": [
                { "$eq": [ "$year", "$$year" ] },
                { "$eq": [ "$month", "$$month" ] },
                { "$eq": [ "$week", "$$week" ] },
                { "$eq": [ "$item", "$$item" ] },
              ]
            }
          }
        },
        {
          "as": 'shippedAmount'
        }
      ]
    }
  },
  {
    "$group": {
      "_id": "$_id",
      "item": { $first: "$item" },
      "description": { $first: "$description" },
      "siteID": { $first: "$siteID" },
      "site": { $first: "$site" },
      "year": { $first: "$year" },
      "month": { $first: "$month" },
      "week": { $first: "$week" },
      "shippedAmount": {
        "$sum": { $sum: "$shippedAmount.shippedCount" }
      },
      "orderedAmount": { $first: "$amount" }
    }
  }
]

```

FIGURE 16. Aggregation pipeline example.

Purpose of the aggregation pipeline, was to streamline the process, instead of using two different queries for shipped item amounts and ordered item amounts.

## 3.5 WEEK 5

### 3.5.1 Plan for the week

Fixing the shipment summary Python script, that was developed during week three. The issue with the script is that there are items in the MongoDB that do not exist in the SQL data. As the script queries only matching items with the SQL data from the MongoDB.

Another task is regarding a Kaizen that was made related to Foresight. Kaizen is a business approach originating from Japan, focused on continuous improvement through changes [17]. At Nokia, Kaizen event has three steps: improvement suggestion, implementing the improvements, and evaluating the results.

### 3.5.2 Kaizen task

Foresight has web tools for production planning, and some planned items need to be traced. Currently the tracing information of a planned item is stored in Excel. The goal of the Kaizen task was simple, a new feature for adding board tracing information in the planned item modal, instead of in Excel. The modal opens when specific planning item is clicked.

To begin the task, new fields were added to the plan item modal (figure 17). The new fields had to be also added to the resolver and the GraphQL schema of the board document.

Board Trace:		
LOT:	Manufacturer:	Board code:
LOT1	MANUFACTURER1	BOARDCODE1
LOT2	MANUFACTURER2	BOARDCODE2
LOT3	MANUFACTURER3	BOARDCODE3

[+ Add new](#)

FIGURE 17. Board Tracing UI.

The UI was then refined, and time was spent on making sure that the new extension to the modal is working properly.

## 3.6 WEEK 6

### 3.6.1 Finishing the Kaizen task

The author finished the Kaizen task by making a small change to the styling of the plan item box. The coloring of the plan item is partially grey if board tracing data is not added (figure 18). With this

small change, the production planners can see if the planned item has board tracing data or not, without opening the modal.

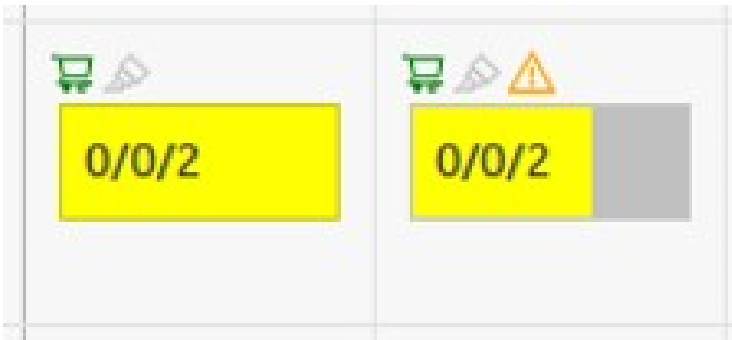


FIGURE 18. Plan items with and without board tracing.

When a new plan item was created without board tracing, the MongoDB document still had the field for board tracing, with value of an empty array. Generally, a field with the value of null is left out of the document when inserting to the database. Mongoose documentation states that arrays are special because they implicitly have a default value of an empty array, and to overwrite it, the default value needs to be set to undefined [18]. A small change to the board trace schema was made, which resolved the issue (figure 19). When no value is provided, the default value is now undefined instead of an empty array.

```
// OLD
boardTrace: [BoardTraceItemSchema],

// NEW
boardTrace: {
  type: [BoardTraceItemSchema],
  default: undefined
}
```

FIGURE 19. Mongoose schema before and after the changes.

With these two changes, the Kaizen task was finished. The rest of the week was spent planning and working on the Global Output front-end.



### 3.6.2 Invalid data

The SQL data that the author had been using turned out to be potentially unsuitable for its intended use case. A new Python script had to be developed for additional data validation, with a new data source used for the comparison. The main functionality of the new script was similar to the old one, so the author was able to reuse most of the code in the previous script. The only exception was the MongoDB aggregation pipeline, which had to be completely remade. When developing the initial script, coding practices were followed, enabling the reuse of the old code.

## 3.7 WEEK 7

### 3.7.1 Global Output prototype view

The week was spent mainly working on the Global Output front-end. A week picker for controlling the time range of the data was developed. When the week is changed, the Global Output data is refetched with the updated parameters.

As there was uncertainty about the final UI design, the project team chose to develop a quick prototype. This allowed presenting something concrete to the end-user and proceeding based on the feedback. A prototype of a card view was developed, where each factory has its own card (figure 20). The color of the card changes depending on the percentage of items shipped compared to the planned quantity.

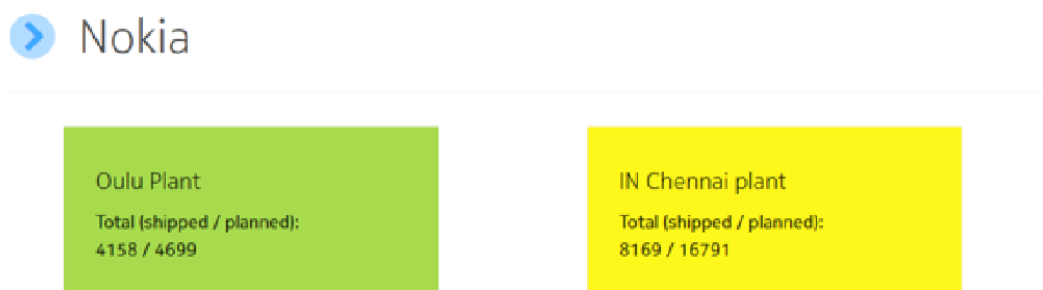


FIGURE 20. First prototype view with mockup data.

At the end of the week, a meeting with the end-user was held. The prototype view was presented, and the received feedback was positive, especially regarding the color coding. However, it was suggested that the data should be presented at a product level. Following the meeting, a decision was made to start developing a table view, which allows presenting the data in an efficient way.

## 3.8 WEEK 8

### 3.8.1 Filter component

Before the development of the table view began, a possibility to filter the data by product type and factory was added (figure 21). The decision to use a shared filter component for the card and table view allowed for testing of the filtering functionality before the card view was implemented.

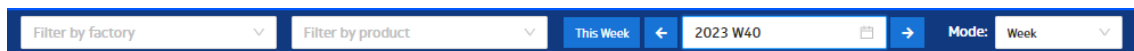


FIGURE 21. Screenshot of the filter component.

The filter component updates the selected filters in the URL as query parameters. When the user has selected data from a specific week or specific product, the applied filters are reflected in the shared link. (figure 22)

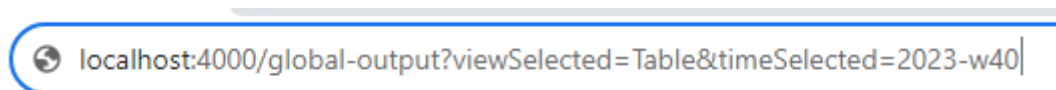


FIGURE 22. Query parameters.

This feature is important especially when developing web tools that manage and visualize data. URL parameters allow users to bookmark or share specific views of the data, which improves communication regarding the data.

### 3.8.2 Global Output table view

The remainder of the week was spent on developing the table view, mainly focusing on the UI. In the table, each row presents data for a distinct product and each column aligns with a manufacturer.

The cell coloring in the table follows the same logic as the cards, but now the logic is applied to individual products produced by each manufacturer. (figure 23)

Product	Nokia		Product Total
	Chennai	Oulu	
	65 / 0	152 / 285	0 / 147
		0 / 119	0 / 119
		12 / 0	12 / 0
	74 / 0	1413 / 1800	156 / 47
			851 / 976
			2324 / 2823
			222 / 0
			40 / 24
			0 / 120
			100 / 298
			0 / 2
			0 / 275
			0 / 31
			23 / 9
			0 / 92
Factory Total			

FIGURE 23. Global Output table view.

The table view provides much more detailed information compared to the card view, while providing a comprehensive overview of the overall status of a specific product or a manufacturer, in an easily readable format.

## 4 REFLECTION

Throughout the 8-week period, the author worked with Python and React for the creation of a web tool focused on data presentation and visualization. The Global Output project played a crucial role in the author's growth as a software developer. The project served as a practical learning experience, allowing the author to significantly improve their skills in Python, especially in data analyzing. Although the Global Output project was not completed within the scope of the thesis, a significant amount of progress was made.

One goal of the thesis was to explore the advantages of adhering to coding standards and best practices, while reflecting them in the practical work. Although the coding standards were studied, they were only briefly covered in the diary section. Also, most of the Python scripts developed were prototype versions, only used by the author to complete the data validation.

Outside of university, it was the author's first time working on a team project. Working together on a project with experienced software developers resulted in a significant personal improvement in programming. Most of the front-end development was done independently, while asking for opinions and help from the other team members if necessary. Many of the challenges faced during this thesis were due to the author being inexperienced in Python. The challenges faced during the development of the project were always solved together with the project team.

## REFERENCES

1. Codacy. 2023. Coding Standards: What Are They and Why Are They Important? Search date 15.11.2023. <https://blog.codacy.com/coding-standards>
2. Khan, Arooj. 2022. Importance of Coding Standards. Search date 11.09.2023. <https://medium.com/codex/importance-of-coding-standards-ac6e69d24e11>
3. Swimm. Comments in code: best practices and 4 mistakes to avoid. Search date 12.09.2023. <https://swimm.io/learn/code-collaboration/comments-in-code-best-practices-and-mistakes-to-avoid>
4. Lambdatest. 2023. Importance of Code Reusability in Software Development. Search date 15.09.2023. <https://www.lambdatest.com/learning-hub/code-reusability>
5. Cominetti, Fabrizio. 2021. Programming Naming Conventions. Search date 20.10.2023. <https://medium.com/nerd-for-tech/programming-naming-conventions-f876fbdddc92>
6. Mozilla Developer Network. Camel case. Search date 20.10.2023. [https://developer.mozilla.org/en-US/docs/Glossary/Camel\\_case](https://developer.mozilla.org/en-US/docs/Glossary/Camel_case)
7. TechTerms. Snake case. Search date 20.10.2023. [https://techterms.com/definition/snake\\_case](https://techterms.com/definition/snake_case)
8. Mozilla Developer Network. Snake case. Search date 20.10.2023. [https://developer.mozilla.org/en-US/docs/Glossary/Snake\\_case](https://developer.mozilla.org/en-US/docs/Glossary/Snake_case)
9. Warsaw, Barry & Van Rossum, Guido & Coghlan, Alyssa. PEP 8 – Style Guide for Python Code. Search date 22.10.2023. <https://peps.python.org/pep-0008/#function-and-variable-names>

10. AscenWork Technologies Private Limited. 2023. Writing Cleaner Code: Tips for Improved Readability. Search date 20.10.2023. <https://www.linkedin.com/pulse/writing-cleaner-code-tips-improved-readability-ascenwork>
11. CodeGuppy. Importance of code indentation. Search date 23.10.2023. <https://codeguppy.com/blog/importance-of-indentation/index.html>
12. GeeksforGeeks. Indentation in Python. Search date 23.10.2023. <https://www.geeksforgeeks.org/indentation-in-python/>
13. Vishwakarma, Suraj. 2023. A Guide to Coding Standards to Improve Code Quality. Search date 25.10.2023. <https://dev.to/documatic/a-guide-to-coding-standards-to-improve-code-quality-68n>
14. pandas. DataFrame.merge. Search date. 25.11.2023. <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html>
15. GeeksforGeeks. Upsert in MongoDB. Search date 25.11.2023. <https://www.geeksforgeeks.org/upsert-in-mongodb/>
16. MongoDB. Aggregation Pipeline. Search date 25.11.2023. <https://www.mongodb.com/docs/manual/core/aggregation-pipeline/>
17. Kanbachi. What is Kaizen? Search date 28.11.2023. <https://www.kanbanchi.com/what-is-kaizen>
18. Mongoose. Arrays. Search date 28.11.2023. <https://mongoosejs.com/docs/schematypes.html#arrays>