Industrial Training Report on Flexgrid in PureJS

A PROJECT REPORT

Submitted in Partial fulfilment of the Requirement for the Degree of

MASTER OF COMPUTER APPLICATION

by

Vinay Singh
(1900290149109)

Under the Supervision of
Prof. Ankit Verma
(Assistant Professor)

Submitted to

Department of Computer Applications
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

DR. A. P. J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW (Formerly Uttar Pradesh Technical University, Lucknow)
July 2021

DECLARATION

I hereby declare that the work presented in this report entitled “**FlexGrid in PureJS**", was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.
I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Vinay Singh

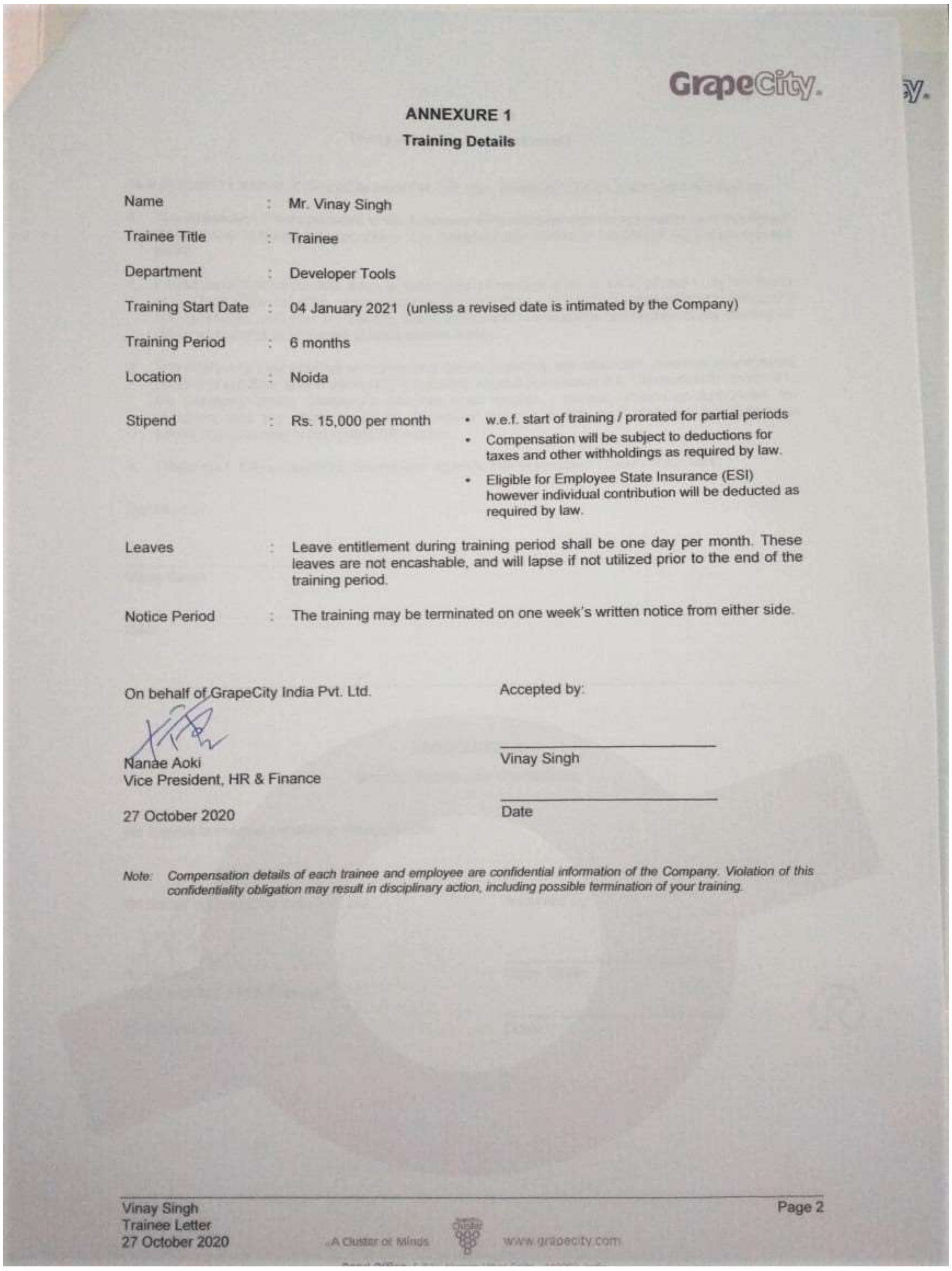
Roll.No.:1900290149109

Branch: MCA

Vinay Singh

(Candidate Signature)

Training Certificate



GrapeCity.

ANNEXURE 1

Training Details

Name	:	Mr. Vinay Singh
Trainee Title	:	Trainee
Department	:	Developer Tools
Training Start Date	:	04 January 2021 (unless a revised date is intimated by the Company)
Training Period	:	6 months
Location	:	Noida
Stipend	:	Rs. 15,000 per month • w.e.f. start of training / prorated for partial periods • Compensation will be subject to deductions for taxes and other withholdings as required by law. • Eligible for Employee State Insurance (ESI) however individual contribution will be deducted as required by law.
Leaves	:	Leave entitlement during training period shall be one day per month. These leaves are not encashable, and will lapse if not utilized prior to the end of the training period.
Notice Period	:	The training may be terminated on one week's written notice from either side.

On behalf of GrapeCity India Pvt. Ltd.

[illegible]

Nanae Aoki
Vice President, HR & Finance

27 October 2020

Accepted by:

Vinay Singh

Date

Note: Compensation details of each trainee and employee are confidential information of the Company. Violation of this confidentiality obligation may result in disciplinary action, including possible termination of your training.

Vinay Singh
Trainee Letter
27 October 2020

A Cluster of Minds www.grapecity.com

Page 2

CERTIFICATE

Certified that **Vinay Singh** **(**enrollment no 190029014005294) has carried out the project work presented in this thesis entitled **"FlexGrid in PureJS"** for the award of **Master of Computer Applications** from Dr. APJ Abdul Kalam Technical University, Lucknow under my supervision. The thesis embodies results of original work, and studies are carried out by the student herself and the contents of the thesis do not formthe basis for the award of any other degree to the candidate or to anybody else from this or any other University.

Mr. Ankit Verma	**External Examiner**
(Internal Examiner)	
Assistant Professor	
Dept. of Computer Applications	
KIET Group of Institutions, Ghaziabad	

Date:

ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Mr. Ankit Verma** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Ajay Kumar Shrivastava**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot in many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kinds of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Vinay Singh
Enrollment no-19002901400529

TABLE OF CONTENTS

LIST OF Tables

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION TO REPORT

FlexGrid for JavaScript is a lightweight data grid control designed on a flexible object model. Based on the popular WinForms version, **FlexGrid** offers many unique features such as unbound mode, flexible cell merging, and multi-cell row and column headers that make it a proven solution for data management and tabulation.
Like other data visualization controls, FlexGrid supports bound mode and can be populated with data from various data sources.
The FlexGrid class provides the **ItemsSource** property to populate data in the grid. The ItemsSource property binds FlexGrid to **ICollectionView** interface.
The following image shows a bound FlexGrid.
The following code examples illustrate how to bind a list of customer objects to FlexGrid control. The code below causes the grid to scan the data source and automatically generate columns for each public property of the items in the data source. You can also customize automatically created columns using code, or disable automatic column generation altogether and create custom columns through code or XAML.
FlexGrid can also be bound directly to a list (customer list). However, binding to **DataCollection** is usually better as it retains a lot of the data configuration for the application, which can be shared across controls.
The FlexGrid control is designed to work with various data sources and collections such as ObservableCollection and DataCollection to leverage its full capabilities. However, the control is not restricted to data sources and can be used in unbound mode
To create an unbound grid, add rows and columns to the grid using the Add method. The following code illustrates adding rows and columns in a grid and populating it with an indexing notation that specifies a cell by corresponding row and column index.

FlexGrid offers several advanced data visualization features that are beyond simple grids. These features are listed below:

- **Flexible data binding**
  FlexGrid can be used in bound mode, where it displays data from a data source, or in unbound mode, where the grid itself manages the data.

- **Advance**
  FlexGrid supports advanced grid features including cell merging, data filtering, sorting, editing, aggregation etc. You can merge contiguous, like-valued cells to make the data span across multiple cell; calculate totals averages, and other statistics for ranges of cells; and apply filters to each column on the grid.

- **Hierarchical**
  FlexGrid summarizes data in hierarchical style like a tree. Each record can be expanded or collapsed to expose details in child grids.
- **Integrated**
  FlexGrid supports integrated printing wherein it has control over paper orientation, margins, and footer text. With a rich object model, the control provides varied printing events to handle page breaks, add repeating header rows, or add custom elements to each page. You can also show a dialog to let users select and set up the printer. FlexGrid supports grouping as a UI feature through a separate control, FlexGridGroupPanel, available as a separate assembly. Similarly, the control comes with FlexGridFilter component, which provides ad-hoc filtering and is shipped separately to achieve lower footprint.
- **Custom**
  FlexGrid supports significant customization in the grid through custom cells. The control provides CellFactory class and built-in CellTemplate and CellEditingTemplate to customize visual elements.
- **Row**
  FlexGrid provides the flexibility to show row details in a data template, which can be used to display text, images as well as data bound controls.
- **Freezing**
  This feature allows you to freeze the rows and columns using mouse drag during runtime. A column or multiple columns can be pinned to the left-hand side of the FlexGrid. Column Pinning in FlexGrid allows the you to lock column in a particular column order, this will allow you to see it while horizontally scrolling the Grid.

1.1.1 OBJECTIVE OF PROJECT

The main objective of the project is to show case the features and functionality of FlexGrid. Which is a great datagrid starts with a fundamentally sound grid. FlexGrid includes only the essential vital features and offers everything else through an extensibility model. Basic Excel-like features like sorting, grouping, and editing are built-in, while the bells and whistles are optional.

The result is a small, fast, familiar, flexible grid that includes:

- Declarative markup

- Cell templates
- Data binding
- TypeScript source and samples

1.2 PROJECT DESCRIPTION

Wijmo represents a new generation of JavaScript controls. It takes full advantage of the latest HTML5 technologies, making no compromises to support legacy browsers. The result is a set of controls that are much faster, smaller, and easier to use than what was possible before.
Wijmo has no dependencies other than EcmaScript5. You can use it without jQuery, jQueryUI, or any other frameworks.
Wijmo requires modern browsers (IE9 or better) to leverage the following technologies:

ECMAScript 5: The ECMAScript 5 standard adds support for property getters and setters. This may seem like a small change, but it makes a huge difference. The ECMAScript 5 standard adds many other significant enhancements, like the bind method that allows you to specify the value of the 'this' parameter in your callbacks. There are also new array methods that can save a lot of time
.
SVG: Modern browsers implement SVG, which makes it easier to create amazing visual representations of your data. Wijmo leverages SVG directly, without the overhead that would be required if it had to support legacy browsers.

TypeScript: We wrote Wijmo in TypeScript, taking advantage of type-checking and OOP concepts such as modules, classes, and inheritance. The output is still pure JavaScript, so you can use either language in your own development work.

Mobile Devices: Wijmo was designed with mobile browser support built in from the start. Responsive layouts and touch support were major considerations in the design and implementation of every Wijmo control.

Angular: AngularJS is one of the most popular and powerful JavaScript application frameworks today. We have supported Angular 1.x since it was released, and Angular 2 even before it was released! For more information, see the Angular Components topic.
Other frameworks: You can use Wijmo with any other JavaScript frameworks you like. In addition to AngularJS and Angular, we ship interop modules for React & Vue also. We plan to add other frameworks to this list in the future, based on customer requests.

Bootstrap: Bootstrap is one of the easiest, most powerful, and most popular CSS frameworks available. We use it in our samples and in our on-line documentation. If you use Bootstrap, be assured that Wijmo will blend right in with no extra effort required on your part.

1.2.1 Advantages of Project

Now, let's get down to some Angular-specific advantages of FlexGrid. The biggest advantage to FlexGrid in Angular is its markup: Angular components give us the ability to **declare UI controls in markup**. Declarative markup is ideal for following the MVVM design pattern, and we can fully configure our components within the View (markup).

Well, if your components support it, that is.

FlexGrid supports declaring its entire API using Angular markup. You can set properties, attach events, configure child components (like columns) entirely in markup.

1.2.2 Disadvantages of Project

1. **Use the production mode of Angular**:
In development mode, the change detection runs twice for every change, but in production mode, change detection runs only once, thereby increasing the performance.

2**. Use Paging**:
If you apply paging to the FlexGrid, the number of cells shown on the DOM will be less and performance will increase. Please refer to the link below that demonstrates how to implement paging in FlexGrid:

https://www.grapecity.com/wijmo/demos/Grid/PagingScrolling/Client-sidePaging/angular

3. **Reduce the dimensions of the grid**:
You may also reduce the width and height of the grid using CSS to decrease the number of cells shown on the DOM:

```
.wj-flexgrid {
            width: 500px;
            height: 500px;
}
```

4. **Use formatItem or CellFactory instead of CellTemplates**:
Using the formatItem event or CellFactory to customize the cells results in better performance than using cell templates.

1.3 PROJECT SCOPE

The Wijmo 2019 v1 release includes several new chart features for FlexChart and FlexPie. This update adds a new chart type, a ranged area chart, and several data visualization improvements enabling enhanced chart customization capabilities. You

can now use FlexPie to visualize more of your data in a single chart by creating multiple pie chart "series" using the same data source. The multiple pie chart series feature is useful if you were building an ad-hoc reporting solution where the user builds the chart, it would be beneficial to allow the user to add more fields to the visualization without having to add additional chart controls.
In FlexGrid news, a FlexGrid.errorTip property has been added to Wijmo that allows users to specify a Tooltip to use when showing validation errors. An option called anchorCursor was also added that changes the range selection to look more like Excel. This helps users who are familiar with Excel to work within Wijmo seamlessly. The grid CSS rules in FlexGrid were also refactored and simplified for writing a single class and styling a cell background and text. FlexGrid's scrolling performance was improved further by extending the cell reordering logic to work when scrolling horizontally and on grids with frozen cells.

1.4 HARDWARE/SOFTWARE

1.4.1 SOFTWARE

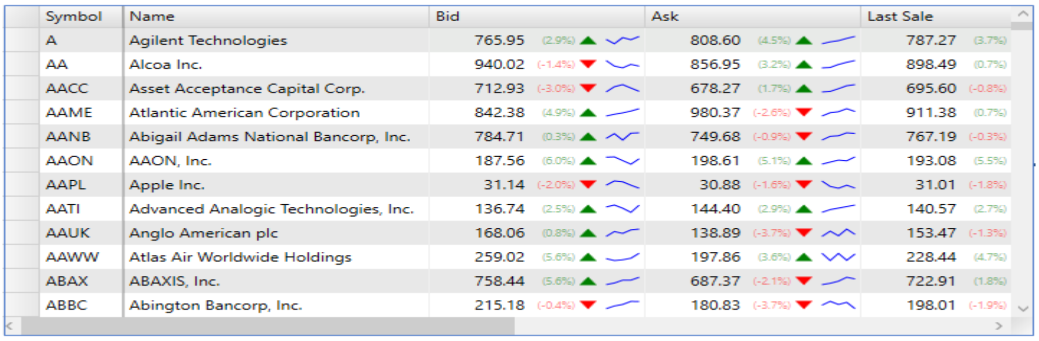
Operating System – Windows
Programming language: Javascript
Front-End: Angular, React, PureJs
Back-End: NodeJs
IDE: Visual Studio Code
API Control: Wijmo Flexgrid

1.4.2 HARDWARE

Windows Version- Windows.
2 GB Ram
1MB Cache Memory
External Memory 10 GB

1.5 Featuring and Images

FlexGrid for Javascript is a lightweight data grid control designed on a flexible object model. Based on the popular javascript version, **FlexGrid** offers many unique features such as unbound mode, flexible cell merging, and multi-cell row and column headers that make it a proven solution for data management and tabulation.



Symbol	Name	Bid			Ask			Last Sale	
A	Agilent Technologies	765.95	(2.9%)	▲	808.60	(4.5%)	▲	787.27	(3.7%)
AA	Alcoa Inc.	940.02	(-1.4%)	▼	856.95	(3.2%)	▲	898.49	(0.7%)
AACC	Asset Acceptance Capital Corp.	712.93	(-3.0%)	▼	678.27	(1.7%)	▲	695.60	(-0.8%)
AAME	Atlantic American Corporation	842.38	(4.9%)	▲	980.37	(-2.6%)	▼	911.38	(0.7%)
AANB	Abigail Adams National Bancorp, Inc.	784.71	(0.3%)	▲	749.68	(-0.9%)	▼	767.19	(-0.3%)
AAON	AAON, Inc.	187.56	(6.0%)	▲	198.61	(5.1%)	▲	193.08	(5.5%)
AAPL	Apple Inc.	31.14	(-2.0%)	▼	30.88	(-1.6%)	▼	31.01	(-1.8%)
AATI	Advanced Analogic Technologies, Inc.	136.74	(2.5%)	▲	144.40	(2.9%)	▲	140.57	(2.7%)
AAUK	Anglo American plc	168.06	(0.8%)	▲	138.89	(-3.7%)	▼	153.47	(-1.3%)
AAWW	Atlas Air Worldwide Holdings	259.02	(5.6%)	▲	197.86	(3.6%)	▲	228.44	(4.7%)
ABAX	ABAXIS, Inc.	758.44	(5.6%)	▲	687.37	(-2.1%)	▼	722.91	(1.8%)
ABBC	Abington Bancorp, Inc.	215.18	(-0.4%)	▼	180.83	(-3.7%)	▼	198.01	(-1.9%)

(1.1 *Flexgrid*)

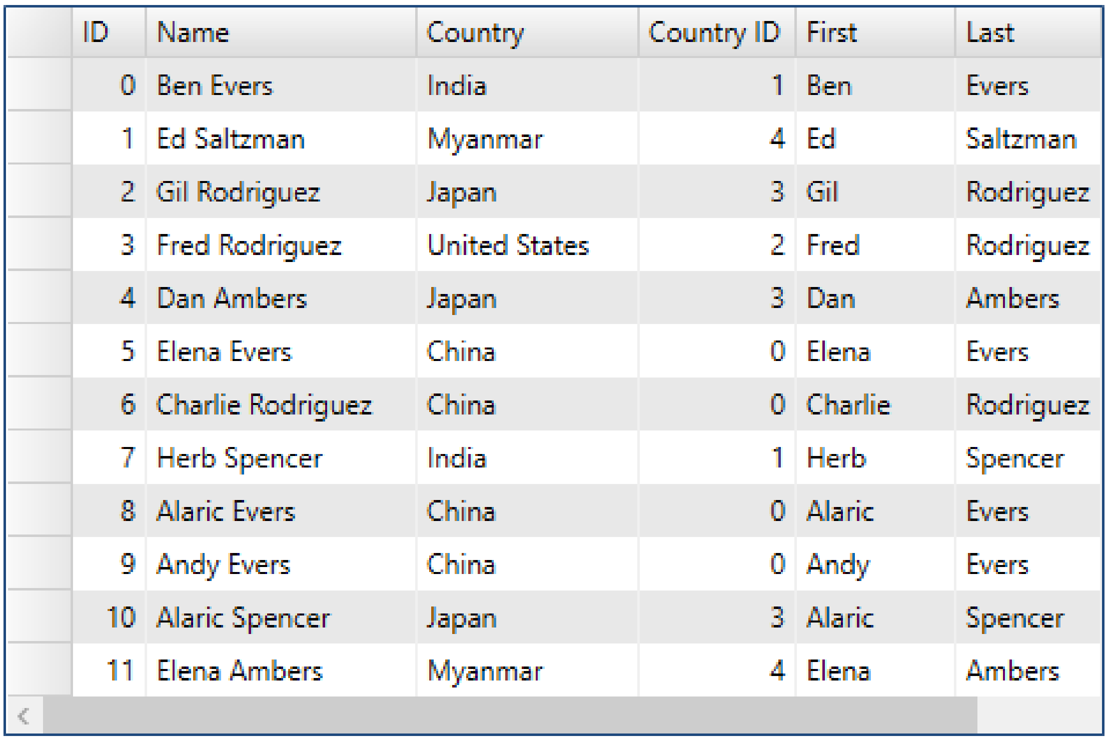
API References	**Product**
Wijmo FlexGrid	Wijmo Javascript

(1.1 *Product Detail*)

Like other data visualization controls, FlexGrid supports bound mode and can be populated with data from various data sources. The below section discusses bound FlexGrid.

TheFlexGrid class provides the **ItemsSource** property to populate data in the grid. The ItemsSource property binds FlexGrid to **IEnumerable** interface, or to DataCollection interface.

The following image shows a bound FlexGrid.



ID	Name	Country	Country ID	First	Last
0	Ben Evers	India	1	Ben	Evers
1	Ed Saltzman	Myanmar	4	Ed	Saltzman
2	Gil Rodriguez	Japan	3	Gil	Rodriguez
3	Fred Rodriguez	United States	2	Fred	Rodriguez
4	Dan Ambers	Japan	3	Dan	Ambers
5	Elena Evers	China	0	Elena	Evers
6	Charlie Rodriguez	China	0	Charlie	Rodriguez
7	Herb Spencer	India	1	Herb	Spencer
8	Alaric Evers	China	0	Alaric	Evers
9	Andy Evers	China	0	Andy	Evers
10	Alaric Spencer	Japan	3	Alaric	Spencer
11	Elena Ambers	Myanmar	4	Elena	Ambers

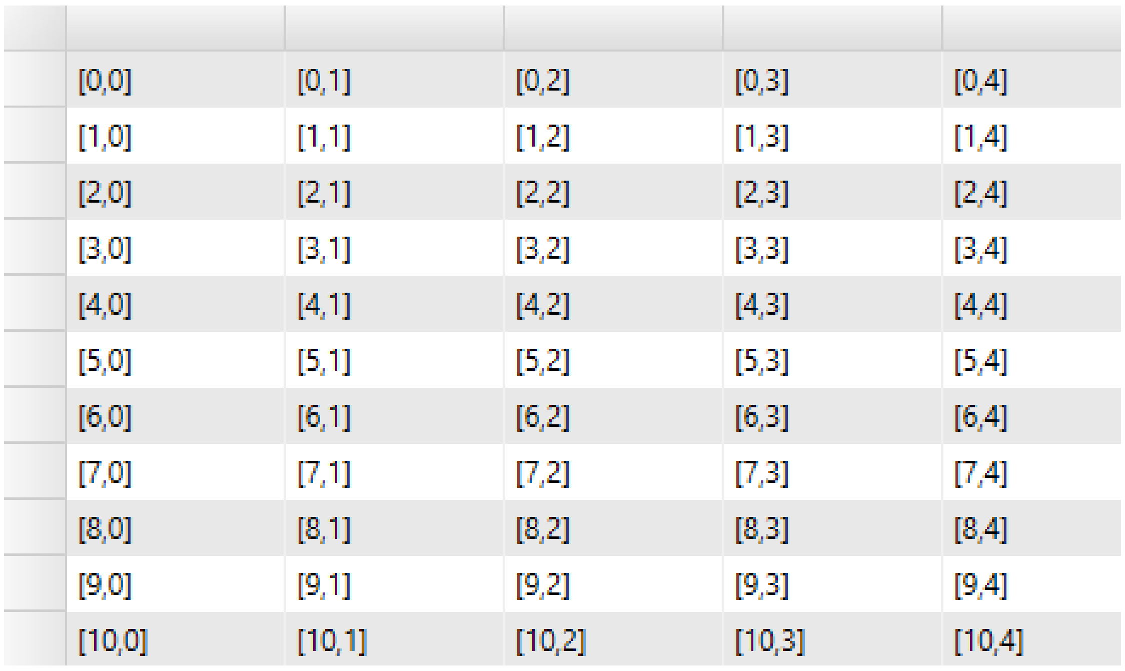
(1.2 *Flexgrid Data Visualization*)

The following code examples illustrate how to bind a list of customer objects to FlexGrid control. The code below causes the grid to scan the data source and automatically generate columns for each public property of the items in the data source. You can also customize automatically created columns using code, or disable automatic column generation altogether and create custom columns through code or XAML.

```
grid.ItemsSource = Customer.GetCustomerList(12);
```

FlexGrid can also be bound directly to a list (customer list). However, binding to **DataCollection** is usually better as it retains a lot of the data configuration for the application, which can be shared across controls.

The FlexGrid control is designed to work with various data sources and collections such as ObservableCollection and DataCollection to leverage its full capabilities. However, the control is not restricted to data sources and can be used in unbound mode.
The image given below shows an unbound grid populated with cell index notation.



	[0,0]	[0,1]	[0,2]	[0,3]	[0,4]
	[1,0]	[1,1]	[1,2]	[1,3]	[1,4]
	[2,0]	[2,1]	[2,2]	[2,3]	[2,4]
	[3,0]	[3,1]	[3,2]	[3,3]	[3,4]
	[4,0]	[4,1]	[4,2]	[4,3]	[4,4]
	[5,0]	[5,1]	[5,2]	[5,3]	[5,4]
	[6,0]	[6,1]	[6,2]	[6,3]	[6,4]
	[7,0]	[7,1]	[7,2]	[7,3]	[7,4]
	[8,0]	[8,1]	[8,2]	[8,3]	[8,4]
	[9,0]	[9,1]	[9,2]	[9,3]	[9,4]
	[10,0]	[10,1]	[10,2]	[10,3]	[10,4]

(1.3 *Flexgrid Data Collection*)

To create an unbound grid, add rows and columns to the grid using the Add method. The following code illustrates adding rows and columns in a grid and populating it with an indexing notation that specifies a cell by corresponding row and column index.

The indexing notation displayed in the grid specifies a cell by row and column index. The cells can also be specified by row index and column name, or by row index and column name. The indexing notation works in bound and unbound modes. In bound mode, the data is retrieved or applied to the items in the data source. In unbound mode, the data is stored internally by the grid.

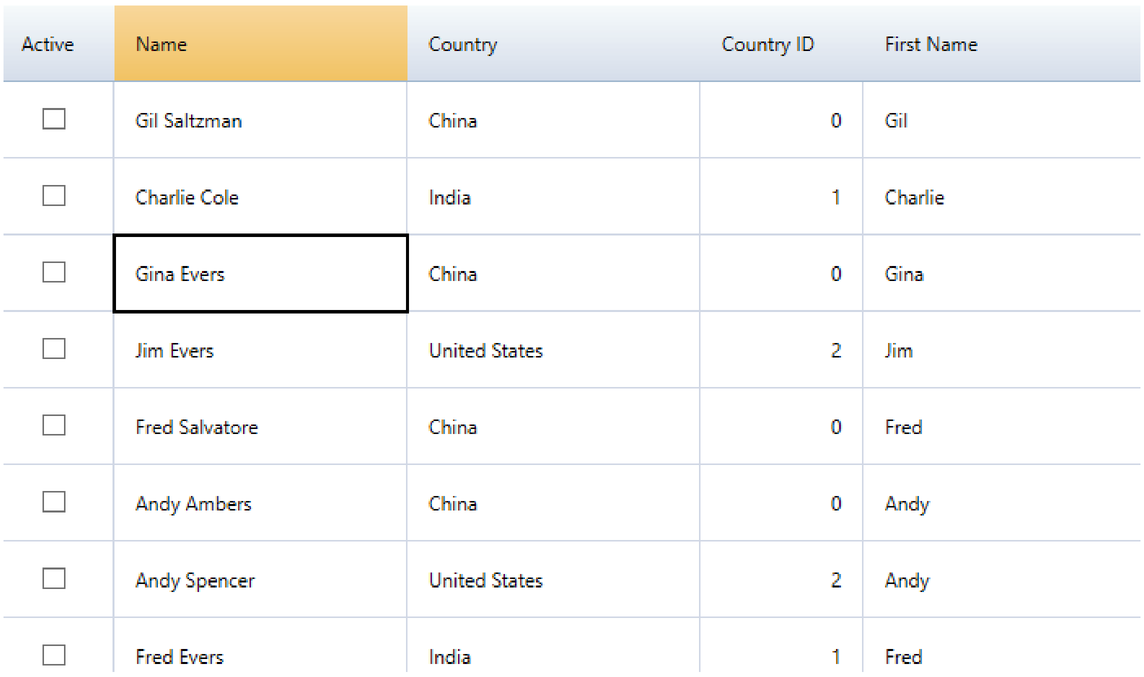
The new indexing notation displayed in the grid contains no items in the 0th row. This notation makes indexing easier as the indices match the index of data items and the column count matches the number of displayed properties. The only drawback of this notation is that a new method is required to access the content of fixed cells

Most grid controls allow users to select parts of the data using the mouse and the keyboard.

- **Cell**: To select a single cell.

- **CellRange**: To select a cell range (block of adjacent cells).
- **Row**: To select an entire row.
- **RowRange**: To select a set of contiguous rows.
- **ListBox**: To select an arbitrary set of rows (not necessarily contiguous).

The default **SelectionMode** is **CellRange**, which provides an Excel-like selection behavior. The row-based options are also useful in scenarios where it makes sense to select whole data items instead of individual cells. Regardless of the selection mode, FlexGrid exposes the current selection with the Selection property. This property gets or sets the current selection as a **CellRange** object.

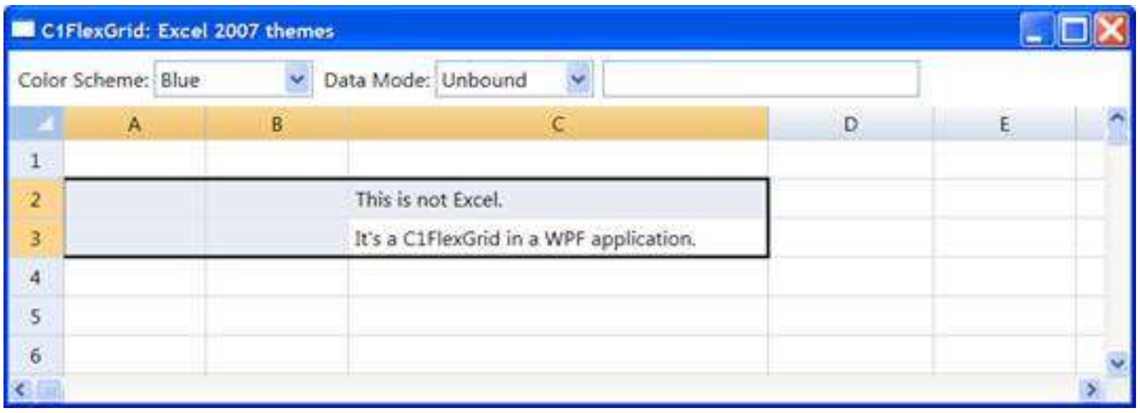


Active	Name	Country	Country ID	First Name
☐	Gil Saltzman	China	0	Gil
☐	Charlie Cole	India	1	Charlie
☐	Gina Evers	China	0	Gina
☐	Jim Evers	United States	2	Jim
☐	Fred Salvatore	China	0	Fred
☐	Andy Ambers	China	0	Andy
☐	Andy Spencer	United States	2	Andy
☐	Fred Evers	India	1	Fred

(1.4 *Flexgrid Checkbox Selection*)

FlexGrid includes two features that allow you to customize the way in which the selection is highlighted for the user.
Excel-Style Marquee
If you set the ShowMarquee property to true, the grid automatically draws a rectangle around the selection, making it extremely easy to see. By default, the marquee is a two-pixel thick black rectangle, but you can customize it using the Marquee property.
Selected Cell Headers

Together, these properties let you implement grids that have the familiar Excel look and feel. The image below shows an example:

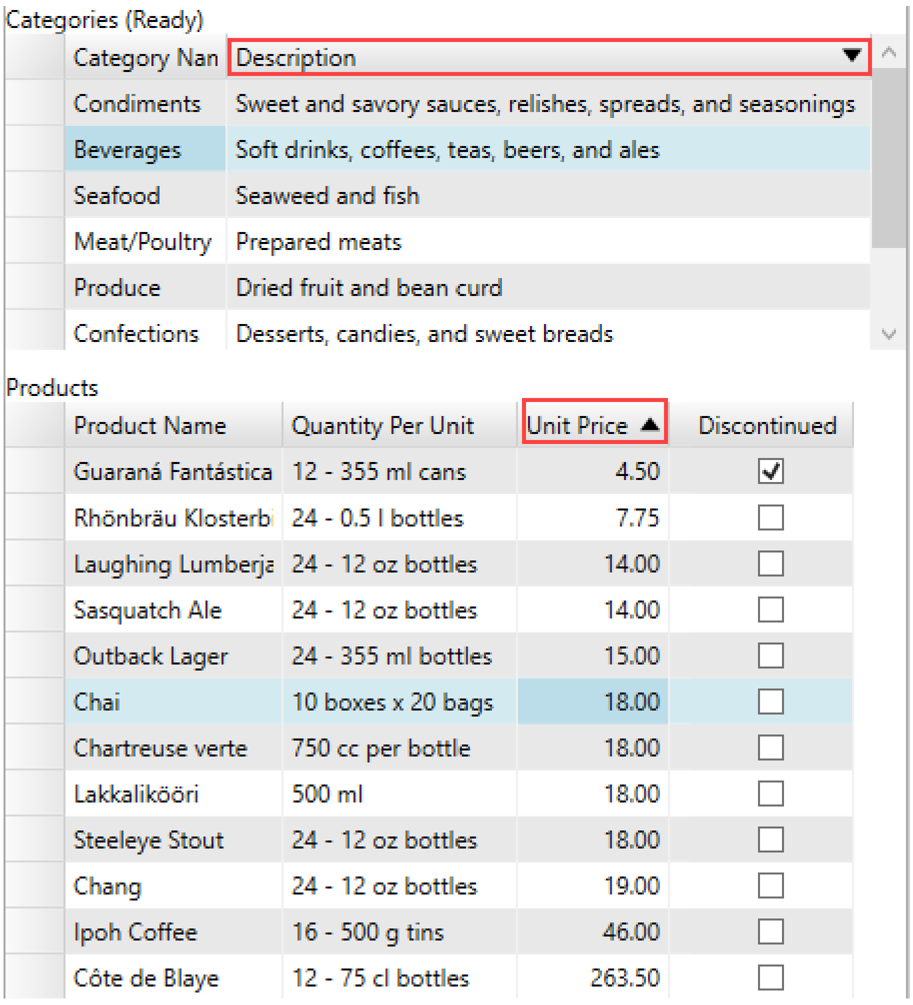

C1FlexGrid: Excel 2007 themes
Color Scheme: Blue
Data Mode: Unbound
A
B
C
D
E
1
2
3
4
5
6
This is not Excel.
It's a C1FlexGrid in a WPF application.

(1.5 *Excel-Like Marquee*)

FlexGrid designer provides a context menu with options to select Excel-like color schemes (Blue, Silver, Black). In addition, you can easily copy the XAML generated by the designer into reusable style resources.

FlexGrid control by default lets you to **sort** by any column. The user simply has to click on the header of a column to toggle between the sorting states, ascending and descending. If a user clicks the column header, it sorts by that column. In this case, the column header shows an upward pointing arrow indicating ascending order. Likewise, if a user clicks the column header once more, that is, the second time, the column sorts in descending order, with the arrow pointing downwards.

The snapshot below depicts sorting feature by the "Description" and "UnitPrice" columns:


Categories (Ready)

Category Nan	Description ▼
Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
Beverages	Soft drinks, coffees, teas, beers, and ales
Seafood	Seaweed and fish
Meat/Poultry	Prepared meats
Produce	Dried fruit and bean curd
Confections	Desserts, candies, and sweet breads

Products

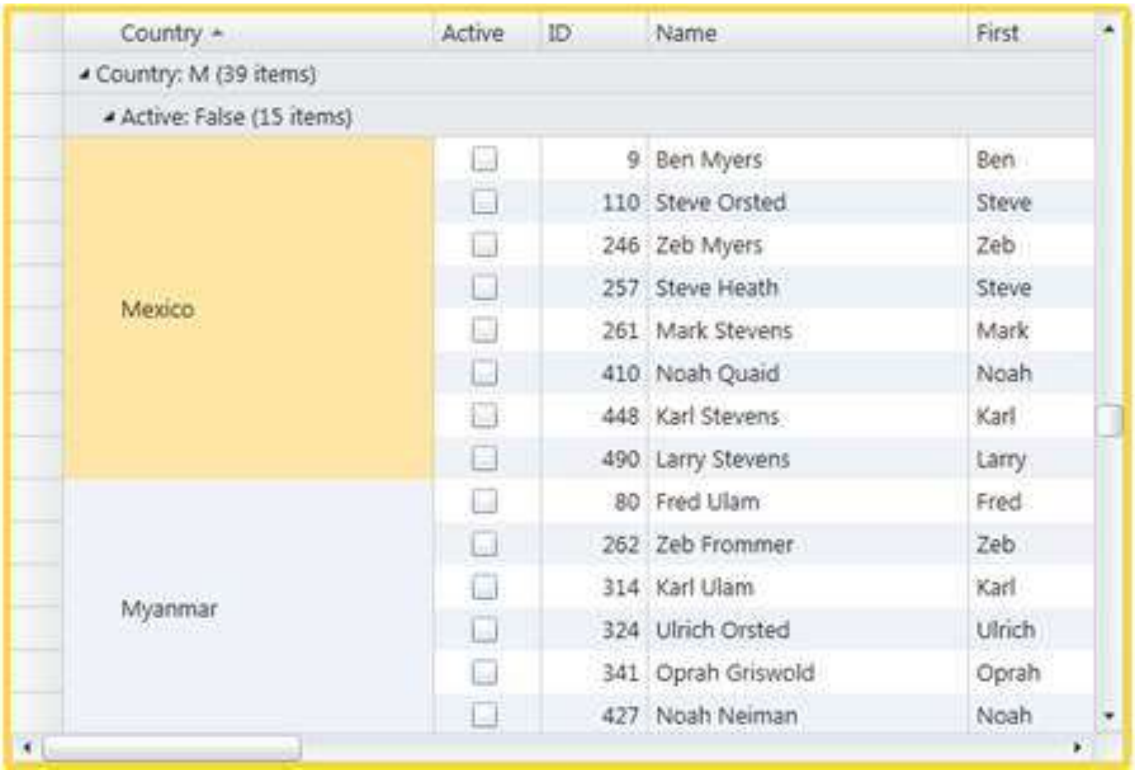
Product Name	Quantity Per Unit	Unit Price ▲	Discontinued
Guaraná Fantástica	12 - 355 ml cans	4.50	☑
Rhönbräu Klosterbi	24 - 0.5 l bottles	7.75	☐
Laughing Lumberja	24 - 12 oz bottles	14.00	☐
Sasquatch Ale	24 - 12 oz bottles	14.00	☐
Outback Lager	24 - 355 ml bottles	15.00	☐
Chai	10 boxes x 20 bags	18.00	☐
Chartreuse verte	750 cc per bottle	18.00	☐
Lakkalikööri	500 ml	18.00	☐
Steeleye Stout	24 - 12 oz bottles	18.00	☐
Chang	24 - 12 oz bottles	19.00	☐
Ipoh Coffee	16 - 500 g tins	46.00	☐
Côte de Blaye	12 - 75 cl bottles	263.50	☐

(1.6 *Flexgrid Sorting*)

The user can also choose to set the AllowSorting property
in FlexGrid class or AllowSorting property in the Column class to 'false' to prevent sorting in the entire FlexGrid control or a desired column within the control.

FlexGrid supports cell merging to allow data to span across multiple rows and columns. The cell merging capability can be used to enhance the appearance of data.
Cell merging can be enabled by setting the **AllowMerging** property of FlexGrid in code. To merge columns, you need to set the AllowMerging property for each column that you want to merge to **true**.
Similarly, to merge rows, set the AllowMerging property to **true** for each row to be merged. You can use **AllowMerging Enum** to specify areas like **Cells**, **ColumnHeaders** etc. of the grid for merging.

The following image shows merged columns in a FlexGrid.



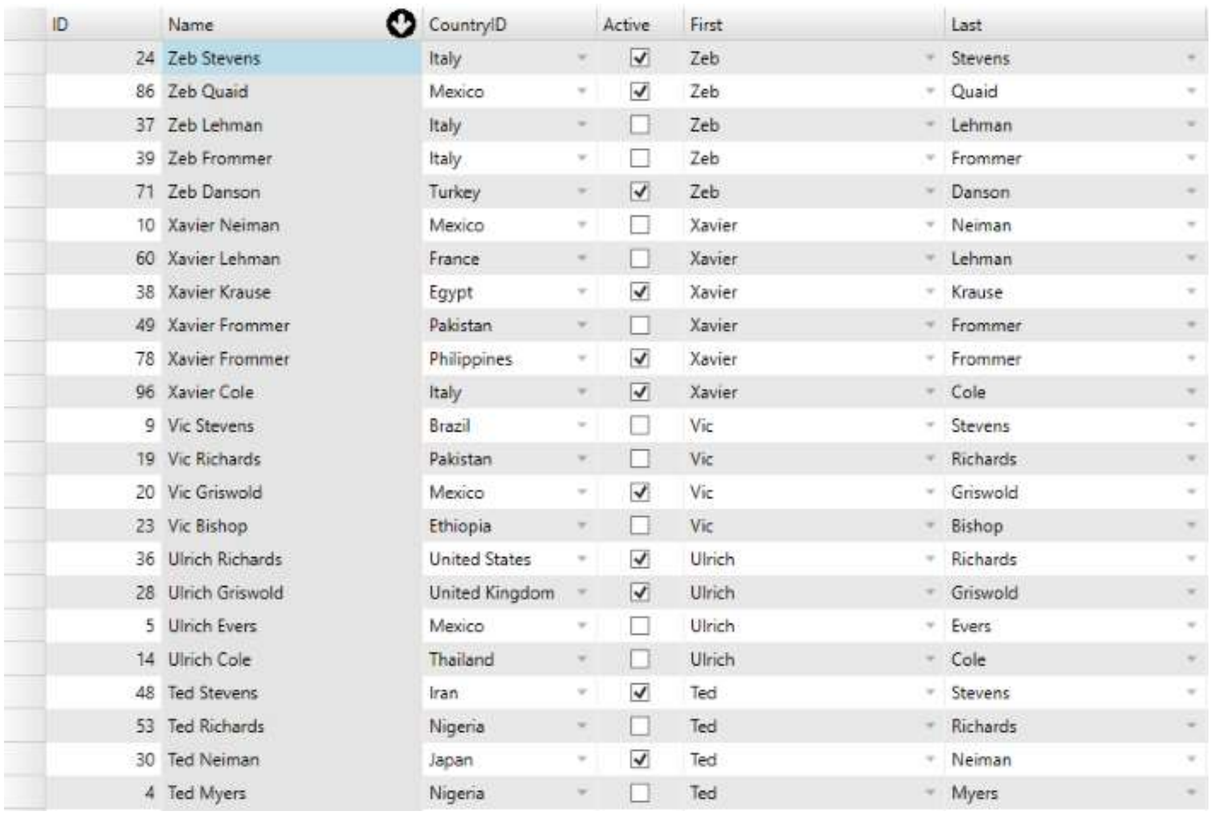
Country ▲	Active	ID	Name	First
◢ Country: M (39 items)				
◢ Active: False (15 items)				
Mexico	☐	9	Ben Myers	Ben
	☐	110	Steve Orsted	Steve
	☐	246	Zeb Myers	Zeb
	☐	257	Steve Heath	Steve
	☐	261	Mark Stevens	Mark
	☐	410	Noah Quaid	Noah
	☐	448	Karl Stevens	Karl
	☐	490	Larry Stevens	Larry
Myanmar	☐	80	Fred Ulam	Fred
	☐	262	Zeb Frommer	Zeb
	☐	314	Karl Ulam	Karl
	☐	324	Ulrich Orsted	Ulrich
	☐	341	Oprah Griswold	Oprah
	☐	427	Noah Neiman	Noah

(1.7 *Flexgrid Merging*)

The code below illustrates merging columns (cells) containing the same **country and name**.

FlexGrid displays various icons during its operations such as sorting, filtering etc. These icons can be changed using various icon templates provided in the FlexGrid control. These icon templates can be accessed through following properties.

You can change the icons set by these templates either to the built-in icons provided by the FlexGrid or to your own custom image, geometric figures, font etc as an icon.
The following image displays a custom image which is set as a sort icon for sorting values in descending order.

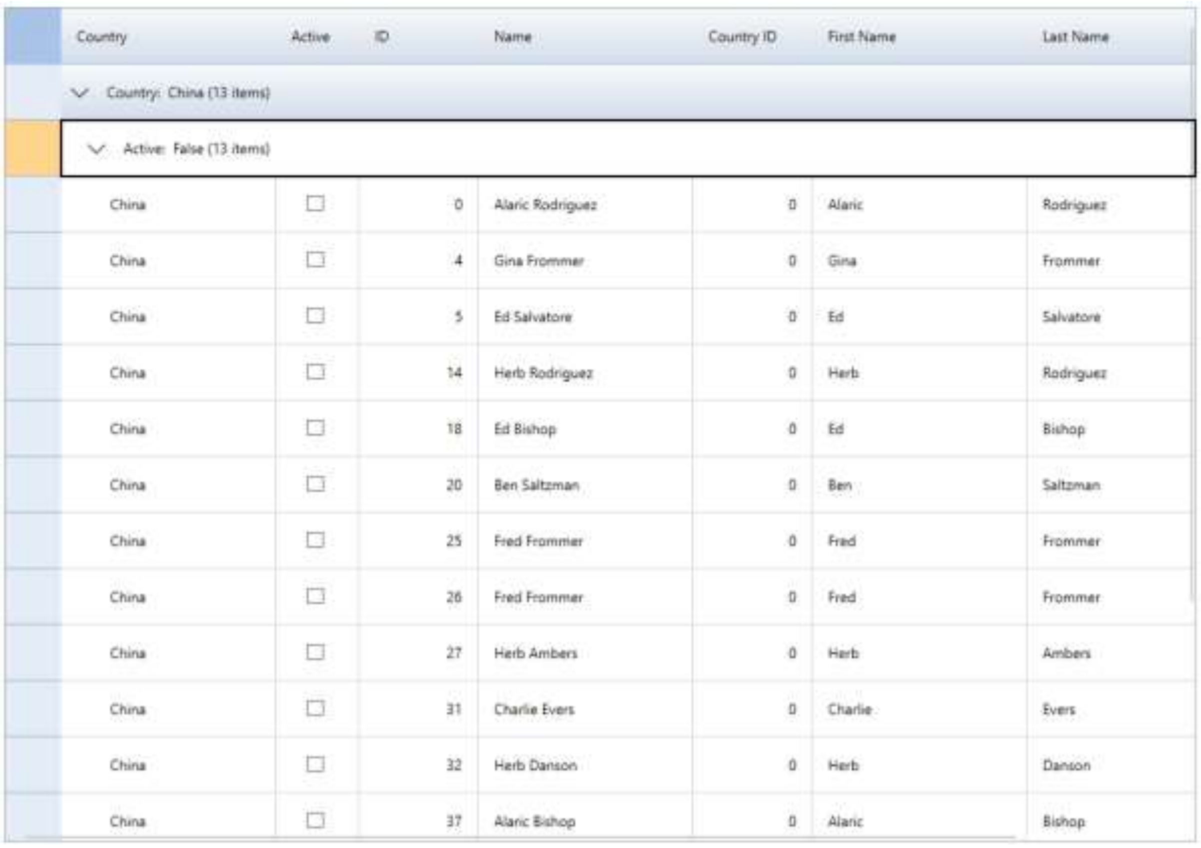


ID	Name	CountryID	Active	First	Last
24	Zeb Stevens	Italy	☑	Zeb	Stevens
86	Zeb Quaid	Mexico	☑	Zeb	Quaid
37	Zeb Lehman	Italy	☐	Zeb	Lehman
39	Zeb Frommer	Italy	☐	Zeb	Frommer
71	Zeb Danson	Turkey	☑	Zeb	Danson
10	Xavier Neiman	Mexico	☐	Xavier	Neiman
60	Xavier Lehman	France	☐	Xavier	Lehman
38	Xavier Krause	Egypt	☑	Xavier	Krause
49	Xavier Frommer	Pakistan	☐	Xavier	Frommer
78	Xavier Frommer	Philippines	☑	Xavier	Frommer
96	Xavier Cole	Italy	☑	Xavier	Cole
9	Vic Stevens	Brazil	☐	Vic	Stevens
19	Vic Richards	Pakistan	☐	Vic	Richards
20	Vic Griswold	Mexico	☑	Vic	Griswold
23	Vic Bishop	Ethiopia	☐	Vic	Bishop
36	Ulrich Richards	United States	☑	Ulrich	Richards
28	Ulrich Griswold	United Kingdom	☑	Ulrich	Griswold
5	Ulrich Evers	Mexico	☐	Ulrich	Evers
14	Ulrich Cole	Thailand	☐	Ulrich	Cole
48	Ted Stevens	Iran	☑	Ted	Stevens
53	Ted Richards	Nigeria	☐	Ted	Richards
30	Ted Neiman	Japan	☑	Ted	Neiman
4	Ted Myers	Nigeria	☐	Ted	Myers

(1.8 *Flexgrid Customization*)

FlexGrid also allows you to change the appearance of the different icons used in the control using the Icon class. The Icon class is an abstract class that provides a series of different objects that can be used for displaying monochromatic icons which can easily be tinted and resized.

FlexGrid supports grouping through I**DataCollection** interface. You can create hierarchical views in FlexGrid by defining each level of grouping through the **PropertyGroupDescription** class. Using the PropertyGroupDescription object, you can select the property to group data, and implement **ValueConverter** to determine how to use property value while grouping. You can also disable grouping at the grid level by setting the grid's GroupRowPosition property to **None**.
The following image shows data grouped by country and their active state. Users can click the icons on group headers to collapse or expand the groups, as they would do with a TreeView control.



Country	Active	ID	Name	Country ID	First Name	Last Name
Country: China (13 items)						
Active: False (13 items)						
China	☐	0	Alaric Rodriguez	0	Alaric	Rodriguez
China	☐	4	Gina Frommer	0	Gina	Frommer
China	☐	5	Ed Salvatore	0	Ed	Salvatore
China	☐	14	Herb Rodriguez	0	Herb	Rodriguez
China	☐	18	Ed Bishop	0	Ed	Bishop
China	☐	20	Ben Saltzman	0	Ben	Saltzman
China	☐	25	Fred Frommer	0	Fred	Frommer
China	☐	26	Fred Frommer	0	Fred	Frommer
China	☐	27	Herb Ambers	0	Herb	Ambers
China	☐	31	Charlie Evers	0	Charlie	Evers
China	☐	32	Herb Danson	0	Herb	Danson
China	☐	37	Alaric Bishop	0	Alaric	Bishop

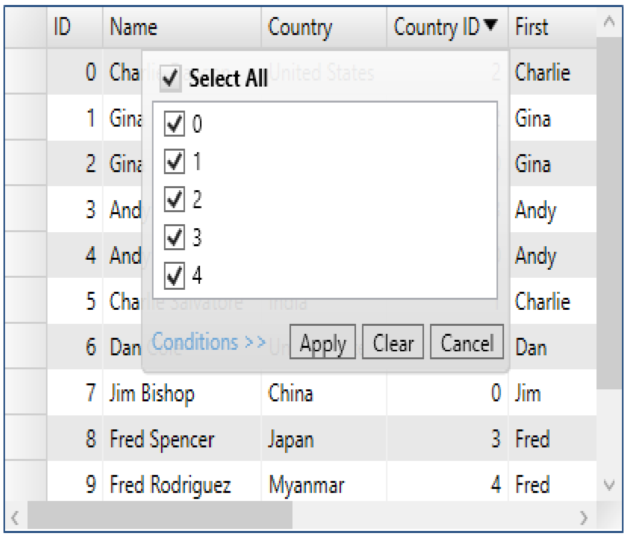
(1.9 *Flexgrid Expand/Collapse Row*)

The following code example illustrates data grouping by country and active state through DataCollection.

FlexGrid provides Excel-like filtering feature to filter data through drop down icons in column headers. This functionality is available in FlexGrid through a separate control called FlexGridFilter, which is implemented through **FlexGridFilter**. Once the FlexGridFilter control is added, the grid displays a drop-down icon on hovering the column headers. The drop-down icon shows an editor that allows users to specify filters on columns. Users may choose between the two types of filters:

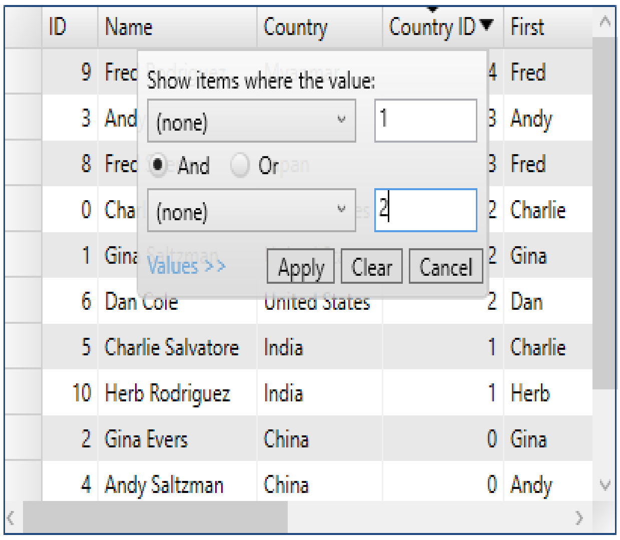
- **Value filter**: This filter lets you filter specific values in the column.
- **Condition filter**: This filter lets you specify conditions composed of an operator (greater than, less than, etc.) and a parameter. The conditions can be combined using an **AND** or an **OR** operator.

The images below show the filters displayed on clicking the drop-down icon.



ID	Name	Country	Country ID	First
0	Cha			Charlie
1	Gina			Gina
2	Gina			Gina
3	And			Andy
4	And			Andy
5	Cha			Charlie
6	Dan			Dan
7	Jim Bishop	China	0	Jim
8	Fred Spencer	Japan	3	Fred
9	Fred Rodriguez	Myanmar	4	Fred

Value Filter



ID	Name	Country	Country ID	First
9	Fred		4	Fred
3	And		3	Andy
8	Fred		3	Fred
0	Cha		2	Charlie
1	Gina		2	Gina
6	Dan Cole	United States	2	Dan
5	Charlie Salvatore	India	1	Charlie
10	Herb Rodriguez	India	1	Herb
2	Gina Evers	China	0	Gina
4	Andy Saltzman	China	0	Andy

Show items where the value:
(none) 1
● And ○ Or
(none) 2

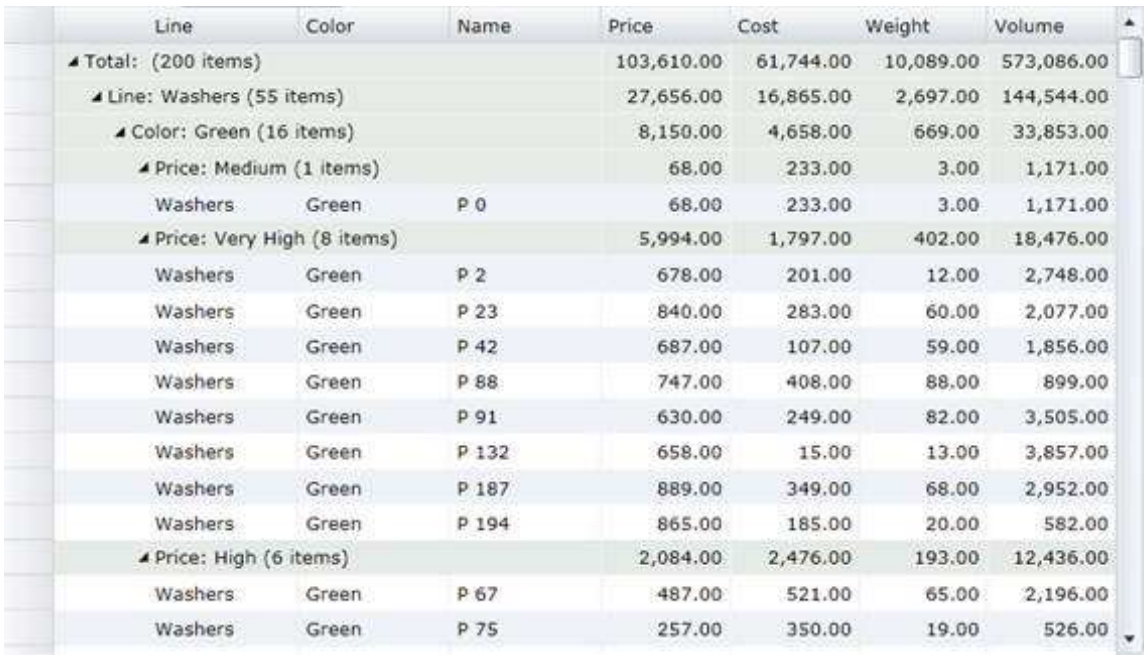
Conditional Filter

(*1.10 Flexgrid Filtering*)

FlexGrid lets you compute and display aggregate value for each group created after grouping data. The control shows groups in a collapsible outline format and automatically displays the number of items in each group. However, you can go one step further and display aggregate values for every grouped column. This feature enables users in drawing out more insights from random data.

In the following section learn how to implement Data Aggregation in FlexGrid .

In the below example, a company's sales data grouped by country or product category can be more useful if the aggregate sales can be indicated against each country and product category in the grid itself. The Column class provides the **GroupAggregate** property that can be set to **Aggregate** to automatically calculate and display aggregates. The aggregates calculated by setting the GroupAggregate property automatically recalculate the aggregate values when the data changes.
The following image shows a FlexGrid with aggregate values displayed in the columns.

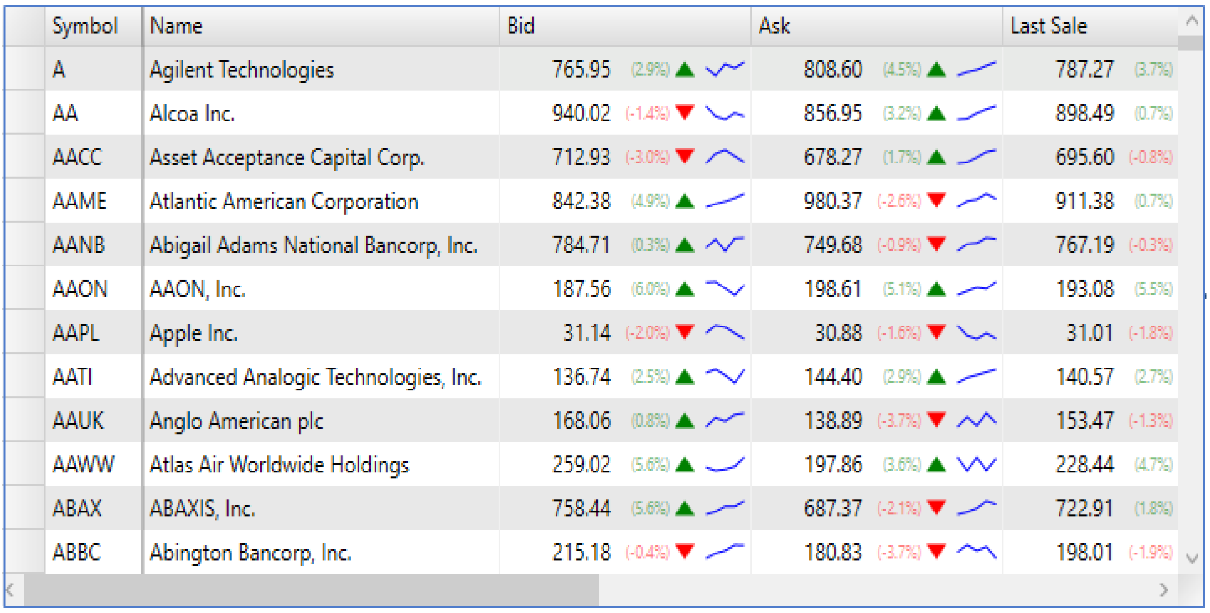


Line	Color	Name	Price	Cost	Weight	Volume
◢ Total: (200 items)			103,610.00	61,744.00	10,089.00	573,086.00
◢ Line: Washers (55 items)			27,656.00	16,865.00	2,697.00	144,544.00
◢ Color: Green (16 items)			8,150.00	4,658.00	669.00	33,853.00
◢ Price: Medium (1 items)			68.00	233.00	3.00	1,171.00
Washers	Green	P 0	68.00	233.00	3.00	1,171.00
◢ Price: Very High (8 items)			5,994.00	1,797.00	402.00	18,476.00
Washers	Green	P 2	678.00	201.00	12.00	2,748.00
Washers	Green	P 23	840.00	283.00	60.00	2,077.00
Washers	Green	P 42	687.00	107.00	59.00	1,856.00
Washers	Green	P 88	747.00	408.00	88.00	899.00
Washers	Green	P 91	630.00	249.00	82.00	3,505.00
Washers	Green	P 132	658.00	15.00	13.00	3,857.00
Washers	Green	P 187	889.00	349.00	68.00	2,952.00
Washers	Green	P 194	865.00	185.00	20.00	582.00
◢ Price: High (6 items)			2,084.00	2,476.00	193.00	12,436.00
Washers	Green	P 67	487.00	521.00	65.00	2,196.00
Washers	Green	P 75	257.00	350.00	19.00	526.00

(*1.11 Aggregation***)**

FlexGrid comes with the CellFactory class to create every cell that appears on the grid. You can create custom cells by creating a class in your project that implements the ICellFactory interface and assign it to the CellFactory property of FlexGrid.
Custom ICellFactory classes can be highly specialized and application-specific, or can be very generic and reusable. In general, custom ICellFactory classes are simpler than custom columns since they deal directly with cells. Implementing custom **CellFactory** classes is fairly easy because you can inherit from the default CellFactory class included with the FlexGrid class. The default CellFactory class was designed to be extensible, so you can let it handle all the details of cell creation and customize only what you need.

The following image shows custom cells created through CellFactory in FlexGrid.



Symbol	Name	Bid				Ask				Last Sale	
A	Agilent Technologies	765.95	(2.9%)	▲		808.60	(4.5%)	▲		787.27	(3.7%)
AA	Alcoa Inc.	940.02	(-1.4%)	▼		856.95	(3.2%)	▲		898.49	(0.7%)
AACC	Asset Acceptance Capital Corp.	712.93	(-3.0%)	▼		678.27	(1.7%)	▲		695.60	(-0.8%)
AAME	Atlantic American Corporation	842.38	(4.9%)	▲		980.37	(-2.6%)	▼		911.38	(0.7%)
AANB	Abigail Adams National Bancorp, Inc.	784.71	(0.3%)	▲		749.68	(-0.9%)	▼		767.19	(-0.3%)
AAON	AAON, Inc.	187.56	(6.0%)	▲		198.61	(5.1%)	▲		193.08	(5.5%)
AAPL	Apple Inc.	31.14	(-2.0%)	▼		30.88	(-1.6%)	▼		31.01	(-1.8%)
AATI	Advanced Analogic Technologies, Inc.	136.74	(2.5%)	▲		144.40	(2.9%)	▲		140.57	(2.7%)
AAUK	Anglo American plc	168.06	(0.8%)	▲		138.89	(-3.7%)	▼		153.47	(-1.3%)
AAWW	Atlas Air Worldwide Holdings	259.02	(5.6%)	▲		197.86	(3.6%)	▲		228.44	(4.7%)
ABAX	ABAXIS, Inc.	758.44	(5.6%)	▲		687.37	(-2.1%)	▼		722.91	(1.8%)
ABBC	Abington Bancorp, Inc.	215.18	(-0.4%)	▼		180.83	(-3.7%)	▼		198.01	(-1.9%)

(*1.12 CellFactory*)

When using custom cells, it is important to understand that grid cells are transient. Cells are constantly created and destroyed as the user scrolls, sorts, or selects ranges on the grid. This process is known as virtualization and is quite common in applications. Without virtualization, a grid would typically have to create several thousand visual elements at the same time, which would impact its performance.

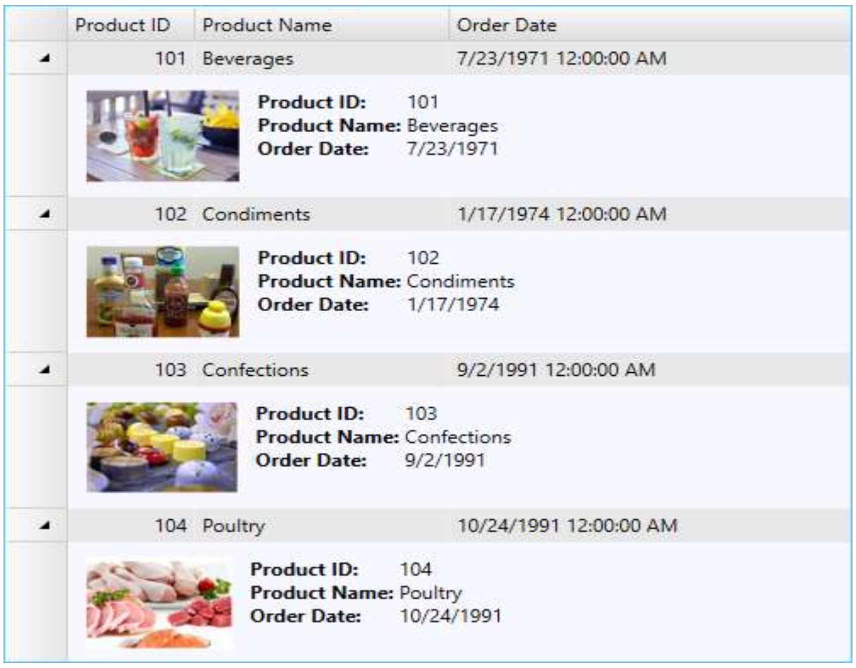
If you want more control over the printing process, use the GetPageImages method to automatically break up the grid into images that can be rendered onto individual pages. Each image is a 100% accurate representation of a portion of the grid, including styles, custom elements, repeating row and column headers on every page, and so on.
The **GetPageImages** method also allows callers to scale the images so the entire grid renders in actual size, scales to fit onto a single page, or scales to the width of a single page.
Once you have obtained the page images, you can use the printing support to render them into documents with complete flexibility. For example, you can create documents that contain several grids, charts, and other types of content. You can also customize headers and footers, add letterheads, and so on.
The following sections demonstrate how an application can render the **FlexGrid** using the **GetPageImages** onto a print document in either platform.

Printing a FlexGrid

Printing documents in requires the following steps:

1. Create a **PrintDialog** object.
2. If the dialog box's **ShowDialog** method returns true, then:
3. Create a **Paginator** object that will provide the document content.
4. Call the dialog's **Print** method.

Row details template is a data panel that can be added to each row for displaying details. FlexGrid provides the flexibility to show information about each row through a template. You can embed text, UI elements, and data-bound controls, such as InputPanel, in the row details template. For each row, you can insert a data template to present its summary and show/provide details in other controls, such as text box, without affecting the dimensions of the grid. You can also use this template to create hierarchical grids displaying grouped data. In this example, we use row details template to display product-related information



Product ID	Product Name	Order Date
101	Beverages	7/23/1971 12:00:00 AM
102	Condiments	1/17/1974 12:00:00 AM
103	Confections	9/2/1991 12:00:00 AM
104	Poultry	10/24/1991 12:00:00 AM

Product ID: 101
Product Name: Beverages
Order Date: 7/23/1971

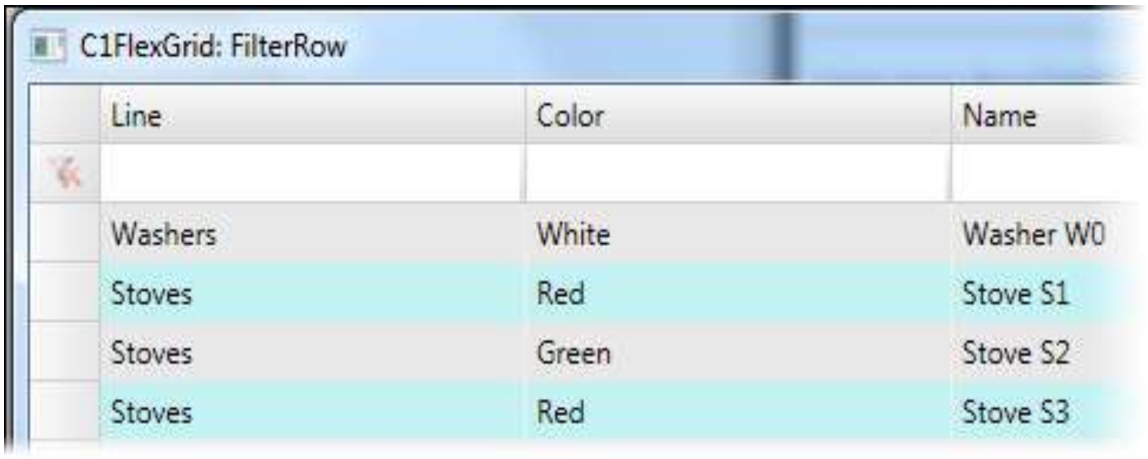
Product ID: 102
Product Name: Condiments
Order Date: 1/17/1974

Product ID: 103
Product Name: Confections
Order Date: 9/2/1991

Product ID: 104
Product Name: Poultry
Order Date: 10/24/1991

(*1.13 Detail Row*)

Each key piece of the control's style is surfaced as a simple color property. This leads to a **Gauge** has **PointerFill** and **PointerStroke** properties,whereasa **DataGrid** has **Selecte dBrush** and **MouseOverBrush** for rows.
Let's say you have a control on your form that does not support ClearStyle. You can take the XAML resource created by ClearStyle and use it to help mold other controls on your form to match (such as grabbing exact colors). Or let's say you'd like to override part of a style set with ClearStyle (such as your own custom scrollbar). This is also possible because ClearStyle can be extended and you can override the style where desired.
ClearStyle is intended to be a solution to quick and easy style modification but you're still free to do it the old fashioned way with WIJMO's controls to get the exact style needed. ClearStyle does not interfere with those less common situations where a full custom design is required.
You can completely change the appearance of the FlexGrid control by setting one or more properties, For example, if you set the **AlternatingRowBackground** property to "#FFC3F2F2", the FlexGrid control appears similar to the following:



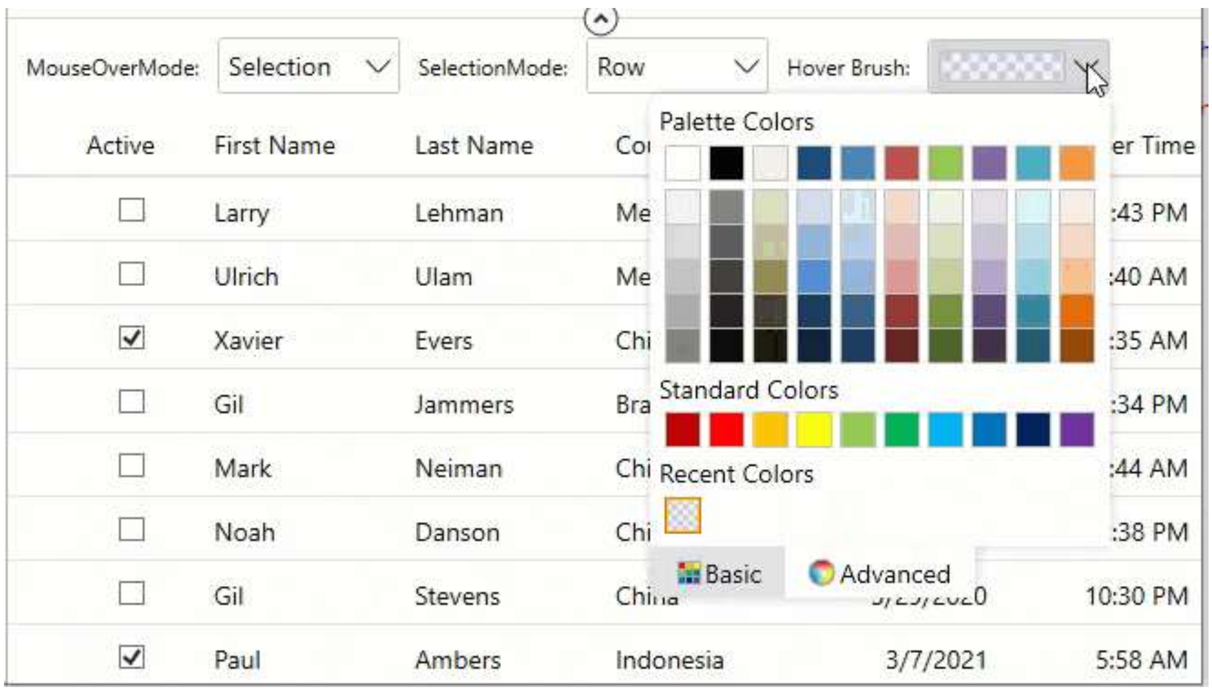
C1FlexGrid: FilterRow

	Line	Color	Name
	Washers	White	Washer W0
	Stoves	Red	Stove S1
	Stoves	Green	Stove S2
	Stoves	Red	Stove S3

(*1.14 Alternate Row Background color*)

Hover styles are applied to cells when the mouse hovers over the grid. It provides visual cues so that the user can apply styling to a cell, row, column or to a specific cell according to the GridSelectionMode enumeration when they are hovered on prior to selection or editing.

The hover styles can make the FlexGrid appear more "alive" and interactive.



MouseOverMode: Selection SelectionMode: Row Hover Brush:

Active	First Name	Last Name	Co		er Time
☐	Larry	Lehman	Me		:43 PM
☐	Ulrich	Ulam	Me		:40 AM
☑	Xavier	Evers	Chi		:35 AM
☐	Gil	Jammers	Bra		:34 PM
☐	Mark	Neiman	Chi		:44 AM
☐	Noah	Danson	Chi		:38 PM
☐	Gil	Stevens	China		10:30 PM
☑	Paul	Ambers	Indonesia	3/7/2021	5:58 AM

(*1.15 Selection Mode*)

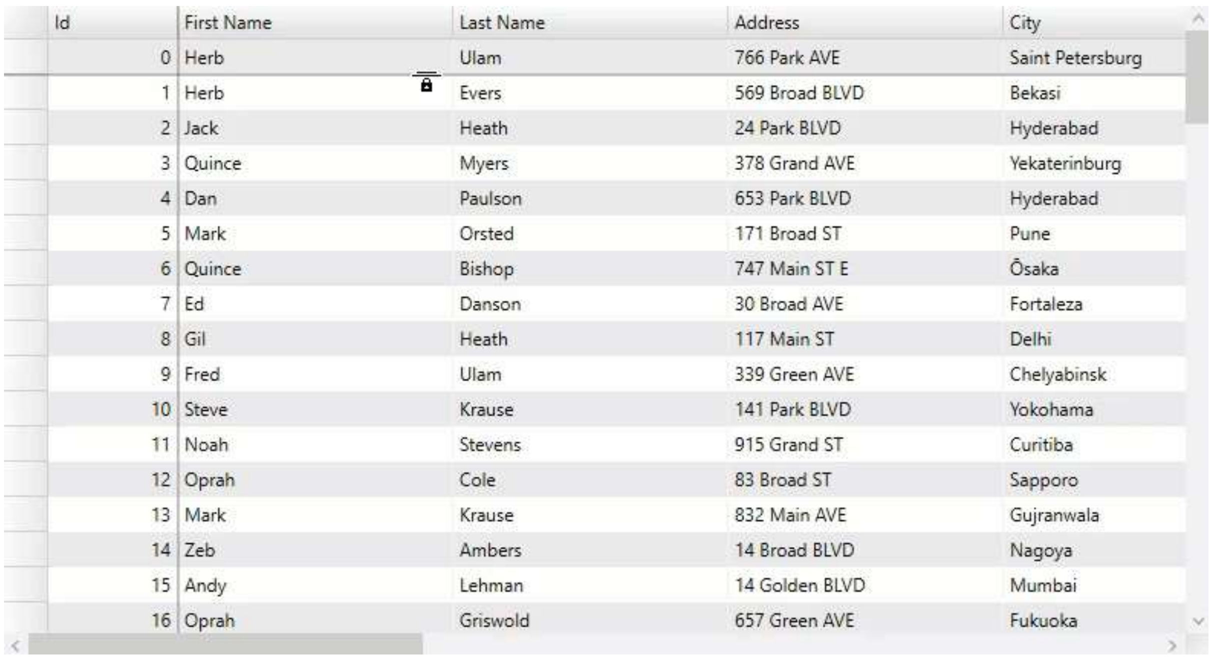
The **MouseOverMode** property and **MouseOverBrush** property in **GridBase** class. The MouseOverMode property helps to set how the mouse hovers over the grid, while the MouseOverBrush property sets a brush to paint the background of the selected cells. Also, **SelectionMode** property is equally important as it helps to set how the cells or rows are selected in the grid, since styling after all is applied based on cell hovering before selection.
Moreover, the SelectionMode property calls the **GridSelectionMode** enumeration, the MouseOverMode property calls the **GridMouseOverMode** enumeration and the MouseOverBrush property calls the **Brush** class.

This feature in **FlexGrid** allows you to freeze the rows and columns using mouse the **AllowFreezing** Enum to **Columns** to freeze only columns, **Rows** to freeze only rows, or **Both** to freeze both columns and rows. Conversely, to disable freezing, set the AllowFreezing Enum to None, which is the default setting. This Enum can be set either in the designer or in code.

Implementation

Locate the **AllowFreezing** Enum in the Properties window and set it to Rows, Columns, Both or None.
Now, you can manually use the mouse drag to adjust the number of frozen columns and rows as needed.

Feature Illustration
When the mouse pointer becomes the lock rows or the lock columns icon, click and drag the mouse over the rows or columns to freeze in a sequence. Setting the AllowFreezing Enum to **Both** allows both rows and columns to be frozen at the same time, as seen in the following GIF.



Id	First Name	Last Name	Address	City
0	Herb	Ulam	766 Park AVE	Saint Petersburg
1	Herb	Evers	569 Broad BLVD	Bekasi
2	Jack	Heath	24 Park BLVD	Hyderabad
3	Quince	Myers	378 Grand AVE	Yekaterinburg
4	Dan	Paulson	653 Park BLVD	Hyderabad
5	Mark	Orsted	171 Broad ST	Pune
6	Quince	Bishop	747 Main ST E	Ōsaka
7	Ed	Danson	30 Broad AVE	Fortaleza
8	Gil	Heath	117 Main ST	Delhi
9	Fred	Ulam	339 Green AVE	Chelyabinsk
10	Steve	Krause	141 Park BLVD	Yokohama
11	Noah	Stevens	915 Grand ST	Curitiba
12	Oprah	Cole	83 Broad ST	Sapporo
13	Mark	Krause	832 Main AVE	Gujranwala
14	Zeb	Ambers	14 Broad BLVD	Nagoya
15	Andy	Lehman	14 Golden BLVD	Mumbai
16	Oprah	Griswold	657 Green AVE	Fukuoka

(1.16 Freezing)

CHAPTER 2

Coding

//order.js

```
import '@grapecity/wijmo.styles/wijmo.css';
import './styles.css';
//
import * as input from '@grapecity/wijmo.input';
import * as wjNav from '@grapecity/wijmo.nav';
//
import { DateRange } from './DateRange';
import { ServerRequest } from './ServerRequest';
import { data } from './config.json';
import './report';
import './dashboard';
//
let salesOrderTabPanel;
let salesDate;
let salesTime;
let salesLocationList;
let multiSelectMenu;
let inputNumberControls = new Array();
let inputUnits = new Array();
let totalUnits = 0;
```

```
document.readyState === 'complete' ? initOrder() : window.onload =
initOrder;

function initOrder() {
    setDateControl();
    setTimeControl();
    setLocationList();
    setMultiSelectMenuCheckboxes();
    createInputNumberControl();
    setTabPanel();
}

//This function is to set properties of date control
function setDateControl() {
    let date = new DateRange(30, 'current');
    salesDate = new input.InputDate('#input-date', {
        format: 'MMM dd, yyyy',
        min: date.minDays(),
        max: date.maxDate()
    });
}

//This function is to properties property of time control
function setTimeControl() {
    salesTime = ne
```

```
 input.InputTime('#input-time', {
        format: 'h:mm tt',
        min: data.timing.startTime,
        max: data.timing.endTime,
        isEditable: true
    });
}

//This function is to set properties location list box
function setLocationList() {
    salesLocationList = new input.ComboBox('#combo-box', {
        displayMemberPath: 'location',
        selectedValuePath: 'id',
        itemsSource: data.locationData,
        placeholder: "location",
        isAnimated: true,
        isRequired: false,
        isEditable: false,
        selectedIndex: -1,
        onSelectedIndexChanged: () => {
            multiSelectMenu.isDisabled = false;
        }
    });
}

//This function is to set properties of multi select menu checkboxes
```

```
function setMultiSelectMenuCheckboxes() {
    multiSelectMenu = new input.MultiSelectListBox('#multi-select', {
        itemsSource: getCheckableData(),
        displayMemberPath: 'items.item',
        checkedMemberPath: 'checked',
        isDisabled: true,
        showFilterInput: true,
        onCheckedItemsChanged: showMenuInputControls
    });
}

//This function is to create menu item input controls
function createInputNumberControl() {
    let inputNumbersDiv = document.getElementById('input-numbers-div');
    let template = '<div class="input-number" id={name}>' +
        '<img for={id} src={source} width="40px" height="40px">' +
        '<label for={id}>{name} 1 Unit Price : {unitPrice} </label><br>' +
        '<input id={id}>' +
        '</div>';
    for (let i = 0; i < data.menuData.length; i++) {
        let id = 'input-number' + data.menuData[i].id;
        let source = data.menuData[i].image;
        let step = 1;
        let html = wijmo.format(template, {
            id: id,
            source: source,
            name: data.menuData[i].item,
            unitPrice: data.menuData[i].unitPrice,
            step: step,
        });

        inputNumbersDiv.appendChild(wijmo.createElement(html));
        inputUnits[i] = new input.InputNumber('#' + id, {
            format: 'd',
            step: step,
            min: 1,
            valueChanged: orderDetails
        });
    }
}

//This function is to show/hide menu item input controls
function showMenuInputControls() {
    inputNumberControls = document.getElementsByClassName('input-
number');
    checkedMenuItems = this.checkedItems;
    totalItems = checkedMenuItems.length;
    for (let counter = 0; counter < inputNumberControls.length; counter++) {
        let itemFound = false;
        for (let subcounter = 0; subcounter < checkedMenuItems.length;
subcounter++) {
```

```
            if (checkedMenuItems[subcounter].items.item ==
inputNumberControls[counter].id) {

document.getElementById(checkedMenuItems[subcounter].items.item).styl
e.display = 'block';
                itemFound = true;
                inputUnits[counter].value = 1;
                break;
            }
        }
        if (itemFound != true) {
            inputNumberControls[counter].style.display = 'none';
            inputUnits[counter].value = 0;
        }
    }
    showOrderDetails();
    if (totalItems == 0)
        document.getElementById("order-details").style.display = 'none';
    else
        document.getElementById("order-details").style.display = 'block';
}


//This function is to calculate total units and total price of the items
function orderDetails() {
    let tempTotalUnits = 0;
    let tempTotalPrice = 0;
    for (let counter = 0; counter < inputUnits.length; counter++) {
        tempTotalPrice += inputUnits[counter].value *
data.menuData[counter].unitPrice;
        tempTotalUnits += inputUnits[counter].value;
    }
    totalUnits = tempTotalUnits;
    totalPrice = tempTotalPrice;
    showOrderDetails();
}

//This function is to show order details
function showOrderDetails() {
    let orderDetailsDiv = document.getElementById("order-details");
    orderDetailsDiv.innerHTML = '<hr><p> Items : ' + totalItems + ',
Number of Units :' + totalUnits + '</p><h1>Total Price : ' + totalPrice +
'</h1><hr>';
}

//Event on form to submit form data
document.querySelector('#submit-btn').addEventListener('click',
submitFormData);

//This function is to send form data to server
function submitFormData() {
    if (checkedMenuItems.length != 0) {
```

```
//
let count = 0;
let products = new Array();
for (let i = 0; i < inputUnits.length; i++) {
    if (inputUnits[i].value != 0) {
        checkedMenuItems[count].items.unit = inputUnits[i].value;
        count++;
    }
}
//
for (let i = 0; i < checkedMenuItems.length; i++) {
    const { image, ...product } = checkedMenuItems[i].items;
    products.push(product);

}
//
let formData = {
    "salesDate": salesDate.text,
    "salesTime": salesTime.text,
    "salesLocation": salesLocationList.text,
    "items": products,
    "unit": totalUnits,
    "totalSalesPrice": totalPrice
};
//
ServerRequest.postRequest('http://localhost:4040', formData);
```

```
        //
    } else {
        alert('Please Select Location and Menu Items to place the order');
    }
}
// This function is to create a data source with items and an extra checkable
member
function getCheckableData() {
    return data.menuData.map(items => ({ items: items, checked: false }));
}

//This function is to set tab panel
function setTabPanel() {
    salesOrderTabPanel = new wjNav.TabPanel('#sales-order-tabPanel');
    salesOrderTabPanel.tabs.deferUpdate(function () {
        let headerspane = document.getElementsByClassName('tab-headers');
        for (let counter = 0; counter < data.headers.length; counter++) {
            let tabHeader = document.createElement('a');
            tabHeader.textContent = data.headers[counter].displayName;
            salesOrderTabPanel.tabs.push(new wjNav.Tab(tabHeader,
headerspane[counter]));
        }
    });
    salesOrderTabPanel.selectedIndex = 0;
}
```

```
//DateRange.js
class DateRange {
    constructor(min, max) {
        if (typeof (min) === 'string') {
            this.dateMin = min;
        } else if (typeof (min) === 'number') {
            this.daysMin = min;
        } else if (typeof (max) === 'string') {
            this.dateMax = max;
        } else {
            this.daysMax = max;
        }
    }

    //This function is to get prior date from current date using days
    minDays() {
        let date = new Date();
        let min = date;
        min.setDate(date.getDate() - this.daysMin);
        return min;

    }

    //This function is to get future date from current date using days
    maxDays() {
        let date = new Date();
        let max = date;
        max.setDate(date.getDate() + this.daysMax);
        return max;
    }

    //This function is to get prior date from current date using date
    minDate() {
        let date = new Date(this.dateMin);
        return this.min !== 'current' ? date : new Date();
    }

    //This function is to get prior date from current date using date
    maxDate() {
        let date = new Date();
        return this.maxDate !== 'current' ? date : new Date();
    }
}
//
export { DateRange };

//RecordsMsg.js
class ErrorMessage {
    constructor(contentDiv, errorDiv, errorMessage) {
        this.contentDiv = contentDiv;
        this.errorDiv = errorDiv;
        this.errorMessage = errorMessage;
    }

    // This function is to create message
    createMessage(length) {
        if (length == 0) {
            document.getElementById(this.contentDiv).style.display = 'none';
            let errorMsgHeading = document.createElement('h1');
            errorMsgHeading.textContent = this.errorMessage;
            errorMsgHeading.setAttribute('id', 'no-data');
            let reportDiv = document.getElementById(this.errorDiv);
            reportDiv.appendChild(errorMsgHeading);
        }
    }
}
//
export { ErrorMessage };

//ChartsData.js

class OrdersData {
    //This  function is to get Sales By Product Data
    static getSalesByProductData(ordersData) {
        let salesByProducts = [];
        ordersData.forEach(element => {
            element.items.forEach(subElement => {
                var findedIndex;
                //
                let product = salesByProducts.find((item, index) => {
                    findedIndex = index;
                    return item.item === subElement.item;
                });
                //
                if (product != null) {
                    salesByProducts[findedIndex].unit += subElement.unit;
                } else {
                    salesByProducts.push({ "item": subElement.item, "unit":
subElement.unit })
                }
                //
            });
        });
        return salesByProducts;
    }

    //This function is to get Sales By Location Data
    static getSalesByLocationData(ordersData) {
        let salesByLocation = [];
        ordersData.forEach(element => {
            var findedIndex;
            let product = salesByLocation.find((item, index) => {
                findedIndex = index
                return item.salesLocation === element.salesLocation
            })
            if (product != null) {
                salesByLocation[findedIndex].unit += element.unit
            } else {
                salesByLocation.push({ "salesLocation": element.salesLocation,
"unit": element.unit })
            }
        });
        return salesByLocation;
    }

    //This function is to get monthly sales data
    static getMonthlySalesData(ordersData){
        let salesByMonth = [];
        ordersData.forEach(element => {
            var findedMonthIndex;
            let product = salesByMonth.find((item, index) => {
                findedMonthIndex = index
                return item.salesDate === Intl.DateTimeFormat('en-US', {
month: 'short'}).format(Date.parse(element.salesDate))
            })
            if (product != null) {
                element.items.forEach(e1=>{
                    let unitCount = e1.item in salesByMonth[findedMonthIndex];
                    if(unitCount){
                        salesByMonth[findedMonthIndex][`${e1.item}`] += e1.unit;
                    }else{
                        salesByMonth[findedMonthIndex][`${e1.item}`] = e1.unit;
                    }
                })
            } else {
                let itemUnits = {};
                element.items.forEach(e2=>{
                    itemUnits[`${e2.item}`] = e2.unit;
                })
                itemUnits.salesDate = Intl.DateTimeFormat('en-US', { month:
'short'}).format(Date.parse(element.salesDate));
                salesByMonth.push(itemUnits);
            }
        });
        return salesByMonth;
    }
}
//
export { OrdersData };

//ServerRequest.js
class ServerRequest {
    //This function is to make post request to the server
    static postRequest(url, data) {
        let http = new XMLHttpRequest();
        http.open('POST', url, true);
```

```
        http.setRequestHeader('Content-type', 'application/json');

        http.onreadystatechange = function () {
            if (http.readyState == 4 && http.status == 200) {
                alert(http.response);
            }
        }
        http.send(JSON.stringify(data));
    }

    //This function is to make get request to the server using wijmo
httpRequest
    static wijmoGetRequest(fileName, callback) {
        wijmo.httpRequest(fileName, {
            success: (xhr) => {
                let response = JSON.parse(xhr.response);
                callback(response);
            }
        });
    }
}

export { ServerRequest }

//config.json
{
  "data": {
   "menuData": [
     {
      "id": "item001",
      "item": "Chocolate",
      "unitPrice": 50.00,
      "image":"https://th.bing.com/th/id/OIP.8rEV6_7PowIPNOcd80-
euAHaIQ?pid=Api&rs=1"
     },
     {
      "id": "item002",
      "item": "Maggie",
      "unitPrice": 12.00,

"image":"https://th.bing.com/th/id/OIP.HKbgA39fVsiPmoO5dAoshQHaIq
?w=136&h=180&c=7&o=5&pid=1.7"
     },
     {
      "id": "item003",
      "item": "IceCream",
      "unitPrice": 35.00,

"image":"https://th.bing.com/th/id/OIP.5pYpHck87rJDhXINIIRAkwHaHa?
w=200&h=200&c=7&o=5&pid=1.7"
     },
     {
        "id": "item004",
        "item": "Blanket",
        "unitPrice": 60.00,

"image":"https://th.bing.com/th?q=Fleece+Throw+Blankets&w=120&h=12
0&c=1&rs=1&qlt=90&cb=1&pid=InlineBlock&mkt=en-
IN&adlt=moderate&t=1&mw=247"
      }
    ],
    "locationData": [
      {
        "id": "loc001",
        "location": "Noida"
      },
      {
        "id": "loc002",
        "location": "Ghaziabad"
      },
      {
        "id": "loc003",
        "location": "Delhi"
      }
    ],
    "timing": {
      "startTime": "09:00",
      "endTime": "21:00"
    },
    "headers": [
      {
        "id": "001",
        "name": "Order",
        "displayName": "Order"
      },
      {
        "id": "002",
        "name": "Reports",
        "displayName": "Reports"
      },
      {
        "id": "003",
        "name": "Dashboard",
        "displayName": "Dashboard"
      }
    ]
  }
}

//dashboard.js
import '@grapecity/wijmo.styles/wijmo.css';
import './styles.css';
import * as core from '@grapecity/wijmo';
//
```

```
import * as chart from '@grapecity/wijmo.chart';
import * as animation from '@grapecity/wijmo.chart.animation';
import { ServerRequest } from './ServerRequest';
import { ErrorMessage } from './RecordsMsg';
import { OrdersData } from './chartsData';
//
document.readyState === 'complete' ? initDashboard() : window.onload =
initDashboard;
//
function initDashboard() {
    ServerRequest.wijmoGetRequest('orders.json', dashboardContent);
}

//This function is to set Dashboard Content
function dashboardContent(response) {
    if (response.length != 0) {

createProductSalesContributionLineChart(OrdersData.getMonthlySalesDat
a(response));

createSalesByProductChart(OrdersData.getSalesByProductData(response))
;

createSalesByLocationChart(OrdersData.getSalesByLocationData(response
));
```

```
createProductSalesContributionChart(OrdersData.getSalesByProductData(r
esponse));
    }
    else {
        let msg = new ErrorMessage('dashboard-content', 'dashboard-tab',
'Records not Available');
        msg.createMessage(response.length);
    }
}

//This function is to create Sales by Product chart
function createSalesByProductChart(data) {
    let salesByProductChart = new chart.FlexChart('#sales-by-product-chart',
{
        header: 'Sales By Product',
        legend: {
            position: chart.Position.Bottom
        },
        itemsSource: data,
        bindingX: 'item',
        axisX: {
            labelAngle: -45
        },
        series: [
            {
                binding: 'unit',
                name: 'Product'
            }
        ],
        axisY: {
            title: 'Products'
        },
        palette: [
            'rgba(0,204,163,1)', 'rgba(0,0,0,1)']
    });
    //
    new animation.ChartAnimation(salesByProductChart);
}

//This function to create Sales By Location chart
function createSalesByLocationChart(data) {
    let salesByLocationChart = new chart.FlexChart('#sales-by-location-
chart', {
        header: 'Sales By Location',
        chartType: chart.ChartType.Bar,
        legend: {
            position: chart.Position.Bottom
        },
        bindingX: 'salesLocation',
        series: [{
            binding: 'unit',
            name: 'Total Sales'
        }],
        axisY: {
            title: "Products"
        },
        itemsSource: data,
        palette: [
            'rgba(70,107,176,1)', 'rgba(200,180,34,1)', 'rgba(20,136,110,1)',
'rgba(181,72,54,1)',
            'rgba(110,89,68,1)', 'rgba(139,56,114,1)', 'rgba(115,178,43,1)',
'rgba(184,115,32,1)',
            'rgba(20,20,20,1)'
        ]
    });
    //
    new animation.ChartAnimation(salesByLocationChart);
}


//This function is to create product sales contribution
function createProductSalesContributionChart(data) {
    let sum = data.map(c => c.unit).reduce((sum, cur) => sum + cur);
    let productSalesContributionChart = new chart.FlexPie('#product-sales-
contribution-chart', {
        header: 'Product Sales Contribution %',
        bindingName: 'item',
        binding: 'unit',
        legend: {
            position: chart.Position.Bottom
        },
        dataLabel: {
            position: 4,
            content: (ht) => {
                return `${ht.name} ${core.Globalize.format(ht.value / sum,
'p2')}`;
            }
        },
        itemsSource: data,
        palette: ['rgba(156,136,217,1)', 'rgba(163,215,103,1)',
'rgba(142,195,192,1)', 'rgba(233,195,169,1)',
            'rgba(145,171,54,1)', 'rgba(212,204,192,1)', 'rgba(97,187,216,1)',
'rgba(226,215,111,1)', 'rgba(128,113,90,1)']
    });
    new animation.ChartAnimation(productSalesContributionChart);

}

//This function is to create Product Sales Contribution Line Chart
function createProductSalesContributionLineChart(data)
{
    let linechart = new chart.FlexChart('#monthly-product-sales-
contribution-chart', {
```

```
header: 'Monthly Sales',
legend: {
   position: chart.Position.Bottom
},
chartType: chart.ChartType.Line,
bindingX: 'salesDate',
series: [ {
      binding: 'Chocolate',
      name: 'Chocolate'
   },
   {
      binding: 'Maggie',
      name: 'Maggie'
   },
   {
      binding: 'IceCream',
      name: 'IceCream'
   },
   {
      binding: 'Blanket',
      name: 'Blanket'
   }
],
axisY: {
   title: 'Total Sales'
},
        itemsSource: data,
        palette: ['rgba(42,159,214,1)', 'rgba(119,179,0,1)', 'rgba(153,51,204,1)',
'rgba(255,136,0,1)', 'rgba(204,0,0,1)', 'rgba(0,204,163,1)',
'rgba(61,109,204,1)', 'rgba(82,82,82,1)', 'rgba(0,0,0,1)']
    });
    //
    new animation.ChartAnimation(linechart);
}

//report.js
import * as wjcwijmo from '@grapecity/wijmo.grid';
import * as input from '@grapecity/wijmo.input';
import * as wjgrouppanel from '@grapecity/wijmo.grid.grouppanel';
import * as wjGridFilter from '@grapecity/wijmo.grid.filter';
import { FlexGridSearch } from '@grapecity/wijmo.grid.search';
//
import { DateRange } from './DateRange';
import { ServerRequest } from './ServerRequest';
import { ErrorMessage } from './RecordsMsg';
//
let reportGrid;
let view = new wijmo.CollectionView();;
let dateRangeControl;
//
document.readyState === 'complete' ? initReport() : window.onload =
initReport;
```

```
//
function initReport() {
    ServerRequest.wijmoGetRequest('orders.json', reportContent);
}

//This function is to set Report Tab Content
function reportContent(response) {
    if (response.length != 0) {
        view.sourceCollection = response;
        setSalesReportGrid();
        setGroupPanel();
        setFlexGridSearch();
        setPager();
        setDateRangeControl();
    }
    else {
        let msg = new ErrorMessage('report-content', 'report-tab', 'Records not
Available');
        msg.createMessage(response.length);
    }

}

//This function is to set report grid properties
function setSalesReportGrid() {
    reportGrid = new wjcwijmo.FlexGrid('#sales-order-grid', {
        autoGenerateColumns: false,
        childItemsPath:'items',
        isReadOnly: true,
        columns: [
            { binding: 'salesOrderId', header: 'Sales Order ID', visible: false,
minWidth: 0 },
            { binding: 'salesDate', header: 'Sales Date', align: 'center' },
            { binding: 'salesTime', header: 'Sales Time', width: 110, align:
'center' },
            { binding: 'salesLocation', header: 'Sales Location', width: 135 },
            { binding: 'item', header: 'Products', width:120},
            { binding: 'unitPrice', header:'UnitPrice'},
            { binding: 'unit', header: 'Sales Units', width:125},
            { binding: 'totalSalesPrice', header: 'Sales Price', width: 125 },
        ],
        itemsSource: view,
        itemFormatter: (panel, r, c, cell) => {
            if (panel.cellType == wjcwijmo.CellType.Cell) {
                let col = panel.columns[c];
                switch (col.binding) {
                    case 'salesTime':
                        //changeCellBackGroundColor(panel, r, c, cell);
                        break;
                }
            }
        }
```

```
    });
    //
    reportGrid.autoSizeColumns();
    reportGrid.collapseGroupsToLevel(0);
    new wjGridFilter.FlexGridFilter(reportGrid);

}

function columnGroupCollapsedChanged(){

}

//This function is to set Group Panel
function setGroupPanel() {
    new wjgrouppanel.GroupPanel('#theGroupPanel', {
        placeholder: 'Drag columns here to create groups',
        grid: reportGrid
    });

}

//This function is to create Search control
function setFlexGridSearch() {
    new FlexGridSearch('#search-sales-records', {
        placeholder: 'search',
        grid: reportGrid,
    });
}

//This function is to create pagination
function setPager() {
    view.pageSize = 50;
    new input.CollectionViewNavigator('#pager', {
        byPage: true,
        headerFormat: 'Page {currentPage:n0} of {pageCount:n0}',
        cv: view

    });
}

//This function is to set DateRange control.
function setDateRangeControl() {
    let date = new DateRange('1/1/2010', 'current');
    dateRangeControl = new input.InputDateRange('#date-range-filter', {
        monthCount: 1,
        format: 'MMM dd, yyyy',
        min: date.minDate(),
        max: date.maxDate()
    });
}

//This function is to change cell background
```

```
function changeCellBackGroundColor(panel, r, c, cell) {
    let time = panel.getCellData(r, c);
    if (time.includes('PM')) {
        let hour = parseInt(time.split(':'));
        let minutes = parseInt(time.split(':')[1].split(' '));
        if ((hour == 9 && minutes > 0) || (hour > 9 && minutes >= 0)) {
            cell.style.background = 'red';
        }
    }
}

// This event is to filter data with date range
document.querySelector('#date-filter-btn').addEventListener('click',
filterGridData);
function filterGridData() {
    reportGrid.collectionView.filter = function (item) {
        let dataDate = new Date(item.salesDate);
        return (dataDate >= dateRangeControl.value && dataDate <=
dateRangeControl.rangeEnd);

    }
}
//styles.css
#main-div{
    margin-left: 10%;
    margin-right: 30%;
    width : 40%
}

#date-div{
    float:left;
    width : 35%;
}

#time-div{
        float:right;
        width : 60%;
}

#location-div{
    float: left;
    margin-top: 3%;
    width : 100%;
}

#menu-div{
    float:left;
    margin-top: 2%;
    width:70%
}

#input-numbers-div{
    margin-top: 3%;
    float : right;
    width : 50%;
    margin-bottom: 5%;
}

.input-number{
    display: none;
    margin-bottom: 5px;
}
#order-details{
    margin-left:50%;
}

#submit-btn{
    align-self: center;
    margin-top: 3%;
    margin-left: 25%;
    background-color:#1976d2;
    color: white;
    border: 0;
    border-radius: 5px;
    height: 30px;
    width: 100px;
    font-size: medium;
    font-family: Verdana;
```

```
}

.wj-tabheader{
   margin-right: 50px;
   font-size: 20px;
}

.wj-tabpane {
   padding:10px;
  }
  .wj-tabpanes{
     width:115%
  }
  .wj-tabpanel>div>.wj-tabheaders>.wj-tabheader:after {
   position: absolute;
   background:#1976d2;
   height: 5px;
}

.wj-tabpanel>div>.wj-tabheaders>.wj-tabheader.wj-state-active {
   color: black;
   text-transform: uppercase;
}

.wj-tabpanel>div>.wj-tabpanes {
   overflow: auto;
   border-top: 1px solid #1976d2;
}

.wj-tabpanel>div>.wj-tabheaders>.wj-tabheader {
   position: relative;
   display: inline-block;
   text-align: center;
   padding: 8px 12px;
   color: inherit;
   font-weight: 700;
   text-transform: capitalize;
}

#filter-div{
   min-width: fit-content;
}



#no-data{
   color:red;
   text-align:center;
   margin-top:50%;
}
```

```
#report-content{
    min-width: min-content;;
}

#date-filter-btn{
    margin-top: 3%;
    margin-left:10px;
    background-color:#1976d2;
    color: white;
    border: 26;
    border-radius: 5px;
    height: 28px;
    font-size: medium;
    font-family: Verdana;
}

#pager{
    margin-left : 25%
}

#search-sales-records{
    margin-top:3%;
    float:right



}
```

```
#sales-by-location-chart{
   width:49%;
   float:right
}
#sales-by-product-chart {
   width:49%;
   float:left
}

#product-sales-contribution-chart{
   float:left;
   width:49%
}
#monthly-product-sales-contribution-chart{
   width:49%;
   float:right
}
//apiserver.js
const express = require('express');
const app = express();
const port = 4040;
const bodyparser = require('body-parser');
const cors = require('cors');
const fs = require('fs');
//
app.use(cors());
```

```
//
app.use(bodyparser.urlencoded({ extended: true }));
app.use(bodyparser.json());
//
app.post('/', (req, res) => {
    res.send("Successful");
    readWriteJsonFile(req.body);
});

//This function is to read/write data from orders.json file
function readWriteJsonFile(formData) {
    fs.readFile('./orders.json', 'utf-8', function (err, data) {
        if (err) throw err

        var fileData = JSON.parse(data);
        let salesOrderId = generateOrderId();
        formData.salesOrderId = salesOrderId;
        fileData.push(formData);

        fs.writeFile('./orders.json', JSON.stringify(fileData), 'utf-8', function
(err) {
            if (err) throw err
            console.log('Data Written Successfully!');
        })
    })
}
```

```
app.listen(port, () => {
    console.log(`server listening at port ${port}`);
});


//This function is to generate unique order ID's
function generateOrderId() {
    let now = Date.now().toString();
    now += now + Math.floor(Math.random() * 10);
    return [now.slice(0, 4), now.slice(4, 10), now.slice(10, 14)].join('-');
}
//index.html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Sales Order Application</title>
    <script src="./node_modules/systemjs/dist/system.js"></script>
    <script src="./systemjs.config.js"></script>
    <script>
        System.import('./src/order');
    </script>
</head>

<body>
    <form>
        <div id="main-div">
            <div id="sales-order-tabPanel"></div>
            <div class="tab-headers">
                <div id="date-div">
                    <label for="input-date">Date</label></br>
                    <div id="input-date"></div>
                </div>
                <div id="timeDiv">
                    <label for="input-time">Time</label></br>
                    <div id="input-time"></div>
                </div>
                <div id="location-div">
                    <label for="combo-box">Select Location</label></br>
                    <div id="combo-box"></div>
                    <hr>
                </div>
                <div id="menu-div">
                    <div id="multi-select"></div>
                    <div id="input-numbers-
div"></div></br></br></br></br></br>
                    <div id='order-details'></div>
                </div>
                <button type='submit' id="submit-btn">Submit</button>
            </div>
    </form>
    <div class="tab-headers">
        <div id='report-tab'>
            <div id='report-content'>
        <div id="filter-div">
            <input id='date-range-filter'>
            <button type="button" id='date-filter-btn'>Go</button>
            <div id="search-sales-records"></div>
        </div></br></br></br>
                <div id="theGroupPanel"></div>
                <div id="sales-order-grid"></div>
                <div id="pager"></div>
            </div>
        </div>
    </div>
    <div class="tab-headers">
        <div id="dashboard-tab">
            <div id ="dashboard-content">
        <div id="sales-by-product-chart"></div>
        <div id="sales-by-location-chart"></div>
        <div id="product-sales-contribution-chart"></div>
        <div id="monthly-product-sales-contribution-chart"></div>
    </div>
```

```
    </div>
    </div>
    </div>
</body>

</html>
//package.json

{
  "name": "salesorderapplicationwijmocontrol",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "lite-server"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@grapecity/wijmo.purejs.all": "^5.20203.766",
    "body-parser": "^1.19.0",
    "cors": "^2.8.5",
    "express": "^4.17.1",
    "file-system": "^2.2.2",
    "systemjs": "^0.19.42",
    "systemjs-plugin-babel": "0.0.25",
    "systemjs-plugin-css": "^0.1.37",
    "systemjs-plugin-json": "^0.2.2"
  },
  "devDependencies": {
    "lite-server": "^2.6.1"
  }
}

//systemjs.config.json

(function (global) {
    System.config({
        transpiler: 'plugin-babel',
        babelOptions: {
            es2015: true
        },
        meta: {
            '*.css': { loader: 'css' },
            '*.json': { loader: 'plugin-json' }
        },
        paths: {
            // paths serve as alias
            'npm:': 'node_modules/'
        },
        // map tells the System loader where to look for things
        map: {
'@grapecity/wijmo': 'npm:@grapecity/wijmo/index.js',
'@grapecity/wijmo.input': 'npm:@grapecity/wijmo.input/index.js',
'@grapecity/wijmo.grid': 'npm:@grapecity/wijmo.grid/index.js',
'@grapecity/wijmo.styles': 'npm:@grapecity/wijmo.styles',
'@grapecity/wijmo.nav': 'npm:@grapecity/wijmo.nav/index.js',
'@grapecity/wijmo.grid.grouppanel':
'npm:@grapecity/wijmo.grid.grouppanel/index.js',
'@grapecity/wijmo.grid.filter': 'npm:@grapecity/wijmo.grid.filter/index.js',
'@grapecity/wijmo.grid.search':
'npm:@grapecity/wijmo.grid.search/index.js',
'@grapecity/wijmo.chart': 'npm:@grapecity/wijmo.chart/index.js',
'@grapecity/wijmo.chart.animation':
'npm:@grapecity/wijmo.chart.animation/index.js',
'css': 'npm:systemjs-plugin-css/css.js',
'plugin-babel': 'npm:systemjs-plugin-babel/plugin-babel.js',
'systemjs-babel-build': 'npm:systemjs-plugin-babel/systemjs-babel-
browser.js',
          'plugin-json': 'npm:systemjs-plugin-json/json.js'
        },
        // packages tells the System loader how to load when no filename
and/or no extension
        packages: {
          src: {
            defaultExtension: 'js'
          },
            "node_modules": {
                defaultExtension: 'js'
            },
        }
    });
})(this);
```

```
//orders.json
[{"salesDate":"Jan 31, 2021","salesTime":"9:00
AM","salesLocation":"Ghaziabad","items":[{"id":"item001","item":"Choco
late","unitPrice":50,"unit":2},{"id":"item002","item":"Maggie","unitPrice":
12,"unit":3},{"id":"item003","item":"IceCream","unitPrice":35,"unit":4}],"
unit":9,"totalSalesPrice":276,"salesOrderId":"1612-041531-
3071"},{"salesDate":"Jan 31, 2021","salesTime":"10:45
AM","salesLocation":"Delhi","items":[{"id":"item002","item":"Maggie","u
nitPrice":12,"unit":4},{"id":"item003","item":"IceCream","unitPrice":35,"u
nit":2}],"unit":6,"totalSalesPrice":118,"salesOrderId":"1612-151825-
7891"},{"salesDate":"Feb 01, 2021","salesTime":"9:00
AM","salesLocation":"Ghaziabad","items":[{"id":"item002","item":"Maggi
e","unitPrice":12,"unit":2},{"id":"item003","item":"IceCream","unitPrice":
35,"unit":3},{"id":"item004","item":"Blanket","unitPrice":60,"unit":4}],"u
nit":9,"totalSalesPrice":369,"salesOrderId":"1612-151846-
5771"},{"salesDate":"Feb 01, 2021","salesTime":"9:00
AM","salesLocation":"Ghaziabad","items":[{"id":"item001","item":"Choco
late","unitPrice":50,"unit":3},{"id":"item002","item":"Maggie","unitPrice":
12,"unit":4},{"id":"item003","item":"IceCream","unitPrice":35,"unit":2}],"
```

unit":9,"totalSalesPrice":268,"salesOrderId":"1612-154427-1231"},{"salesDate":"Feb 01, 2021","salesTime":"9:00 AM","salesLocation":"Noida","items":[{"id":"item001","item":"Chocolate","unitPrice":50,"unit":4},{"id":"item002","item":"Maggie","unitPrice":12,"unit":4},{"id":"item003","item":"IceCream","unitPrice":35,"unit":4}],"unit":12,"totalSalesPrice":388,"salesOrderId":"1612-154437-1641"},{"salesDate":"Feb 01, 2021","salesTime":"9:00 AM","salesLocation":"Ghaziabad","items":[{"id":"item002","item":"Maggie","unitPrice":12,"unit":1},{"id":"item003","item":"IceCream","unitPrice":35,"unit":1},{"id":"item004","item":"Blanket","unitPrice":60,"unit":1}],"unit":3,"totalSalesPrice":107,"salesOrderId":"1612-154443-8751"},{"salesDate":"Feb 01, 2021","salesTime":"9:00 AM","salesLocation":"Noida","items":[{"id":"item002","item":"Maggie","unitPrice":12,"unit":1},{"id":"item003","item":"IceCream","unitPrice":35,"unit":1}],"unit":2,"totalSalesPrice":47,"salesOrderId":"1612-154449-7711"},{"salesDate":"Feb 03, 2021","salesTime":"10:00 AM","salesLocation":"Delhi","items":[{"id":"item002","item":"Maggie","unitPrice":12,"unit":7},{"id":"item003","item":"IceCream","unitPrice":35,"unit":5}],"unit":12,"totalSalesPrice":259,"salesOrderId":"1612-438459-3111"},{"salesDate":"Feb 04, 2021","salesTime":"10:30 AM","salesLocation":"Delhi","items":[{"id":"item001","item":"Chocolate","unitPrice":50,"unit":1},{"id":"item002","item":"Maggie","unitPrice":12,"unit":1},{"id":"item003","item":"IceCream","unitPrice":35,"unit":1}],"unit":3,"totalSalesPrice":97,"salesOrderId":"1612-438468-1351"},{"salesDate":"Feb 01, 2021","salesTime":"10:30 AM","salesLocation":"Noida","items":[{"id":"item001","item":"Chocolate"

,"unitPrice":50,"unit":4},{"id":"item002","item":"Maggie","unitPrice":12,"
unit":1},{"id":"item003","item":"IceCream","unitPrice":35,"unit":5},{"id":
"item004","item":"Blanket","unitPrice":60,"unit":4}],"unit":14,"totalSalesP
rice":627,"salesOrderId":"1612-438483-6471"},{"salesDate":"Feb 01,
2021","salesTime":"10:30
AM","salesLocation":"Ghaziabad","items":[{"id":"item001","item":"Choco
late","unitPrice":50,"unit":7},{"id":"item002","item":"Maggie","unitPrice":
12,"unit":3}],"unit":10,"totalSalesPrice":386,"salesOrderId":"1612-438496-
4781"},{"salesDate":"Feb 04, 2021","salesTime":"9:00
AM","salesLocation":"Noida","items":[{"id":"item001","item":"Chocolate"
,"unitPrice":50,"unit":5},{"id":"item002","item":"Maggie","unitPrice":12,"
unit":6},{"id":"item003","item":"IceCream","unitPrice":35,"unit":6},{"id":
"item004","item":"Blanket","unitPrice":60,"unit":8}],"unit":25,"totalSalesP
rice":1012,"salesOrderId":"1612-438546-6991"},{"salesDate":"Feb 04,
2021","salesTime":"9:00
AM","salesLocation":"Delhi","items":[{"id":"item003","item":"IceCream",
"unitPrice":35,"unit":1},{"id":"item004","item":"Blanket","unitPrice":60,"
unit":1}],"unit":2,"totalSalesPrice":95,"salesOrderId":"1612-438863-
3071"},{"salesDate":"Jan 12, 2021","salesTime":"9:30
AM","salesLocation":"Noida","items":[{"id":"item001","item":"Chocolate"
,"unitPrice":50,"unit":4},{"id":"item002","item":"Maggie","unitPrice":12,"
unit":5},{"id":"item003","item":"IceCream","unitPrice":35,"unit":6},{"id":
"item004","item":"Blanket","unitPrice":60,"unit":7}],"unit":22,"totalSalesP
rice":890,"salesOrderId":"1612-843124-5571"},{"salesDate":"Feb 09,
2021","salesTime":"9:00
AM","salesLocation":"Noida","items":[{"id":"item001","item":"Chocolate"

,"unitPrice":50,"unit":3},{"id":"item002","item":"Maggie","unitPrice":12,"unit":4},{"id":"item003","item":"IceCream","unitPrice":35,"unit":4},{"id":"item004","item":"Blanket","unitPrice":60,"unit":2}],"unit":13,"totalSalesPrice":458,"salesOrderId":"1612-844042-4461"},{"salesDate":"Feb 09, 2021","salesTime":"9:00 AM","salesLocation":"Noida","items":[{"id":"item001","item":"Chocolate","unitPrice":50,"unit":1},{"id":"item002","item":"Maggie","unitPrice":12,"unit":1},{"id":"item003","item":"IceCream","unitPrice":35,"unit":1},{"id":"item004","item":"Blanket","unitPrice":60,"unit":1}],"unit":4,"totalSalesPrice":157,"salesOrderId":"1612-844054-6111"},{"salesDate":"Feb 09, 2021","salesTime":"9:00 AM","salesLocation":"Ghaziabad","items":[{"id":"item002","item":"Maggie","unitPrice":12,"unit":4},{"id":"item003","item":"IceCream","unitPrice":35,"unit":10},{"id":"item004","item":"Blanket","unitPrice":60,"unit":8}],"unit":22,"totalSalesPrice":878,"salesOrderId":"1612-844071-4911"},{"salesDate":"Jan 27, 2021","salesTime":"10:30 AM","salesLocation":"Delhi","items":[{"id":"item001","item":"Chocolate","unitPrice":50,"unit":6},{"id":"item002","item":"Maggie","unitPrice":12,"unit":8},{"id":"item003","item":"IceCream","unitPrice":35,"unit":7},{"id":"item004","item":"Blanket","unitPrice":60,"unit":6}],"unit":27,"totalSalesPrice":1001,"salesOrderId":"1612-844101-1461"}

CHAPTER 3

PROJECT FLOW DIAGRAM

3.1 Data Flow Diagram

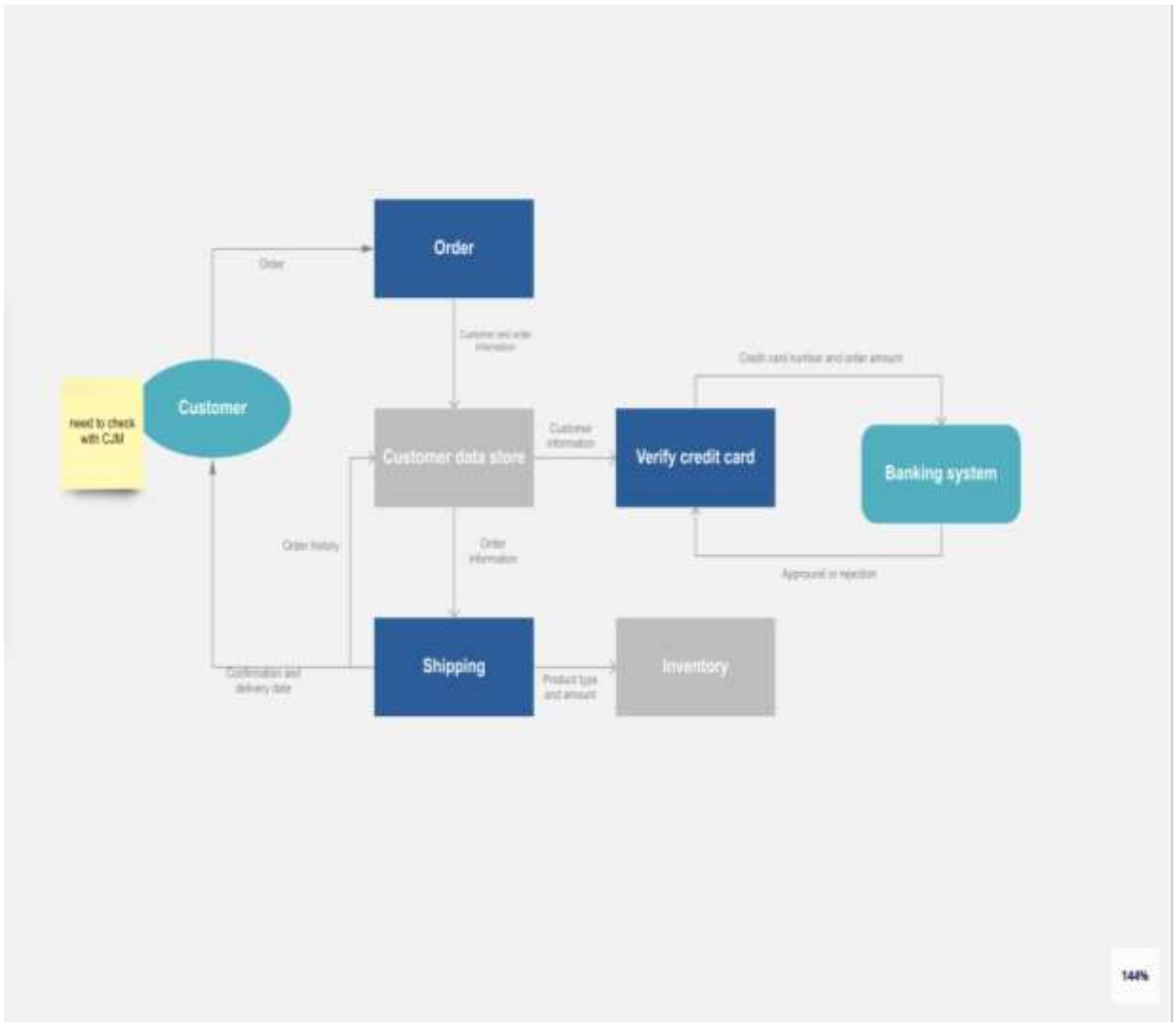
A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model.

A picture is worth a thousand words. A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both.

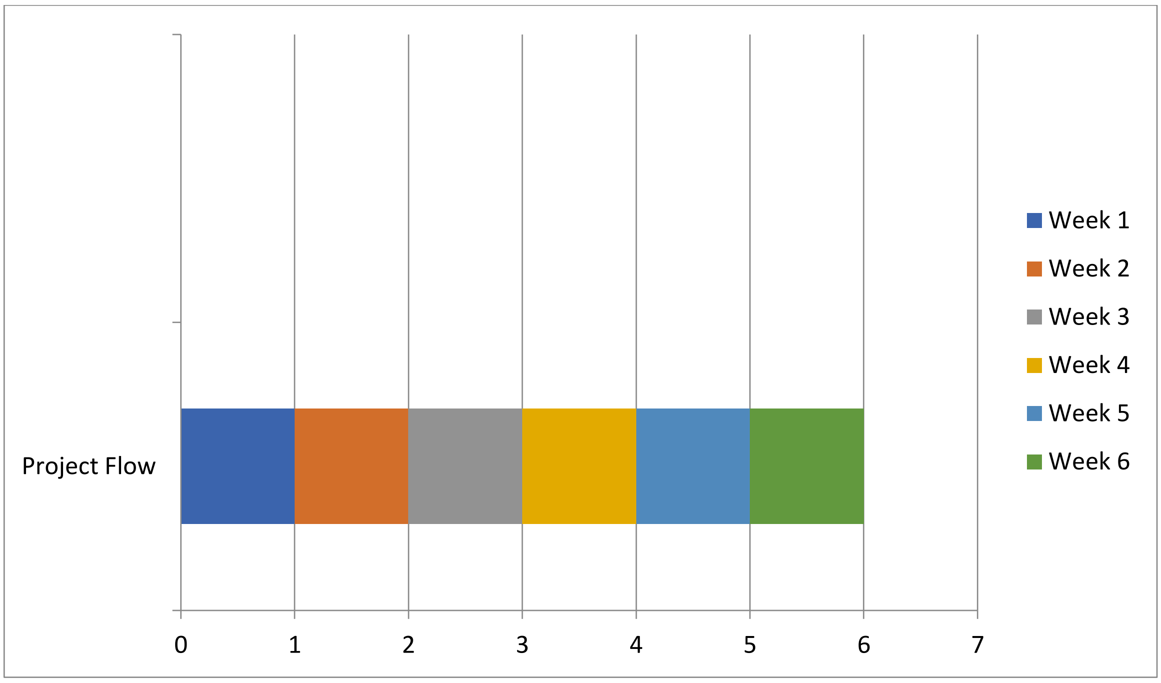
It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

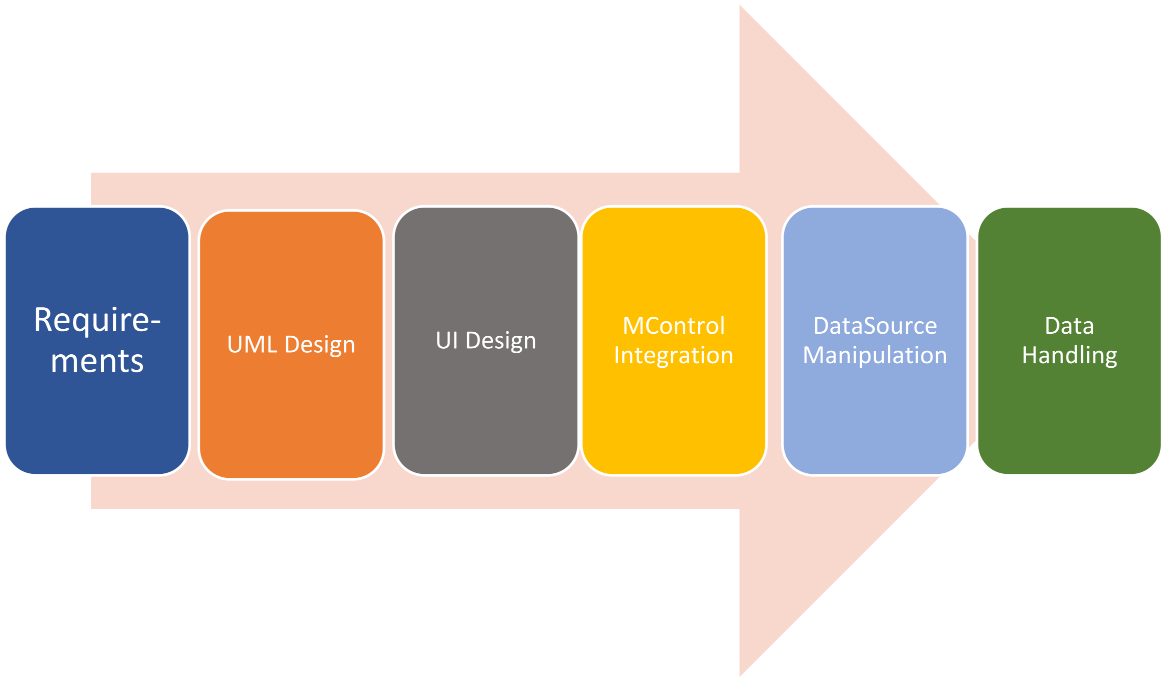
It is usually beginning with a context diagram as level 0 of the DFD diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower-level functions decomposed from the major functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to levels 3, 4 and so on is possible but anything beyond level 3 is not very common. Please bear in mind that the level of detail for decomposing a particular function depending on the complexity that function.



(*3.1 Data Flow Diagram)*

3.2 Gantt Chart





(*3.2 Project time estimation and phases*)

CHAPTER 4

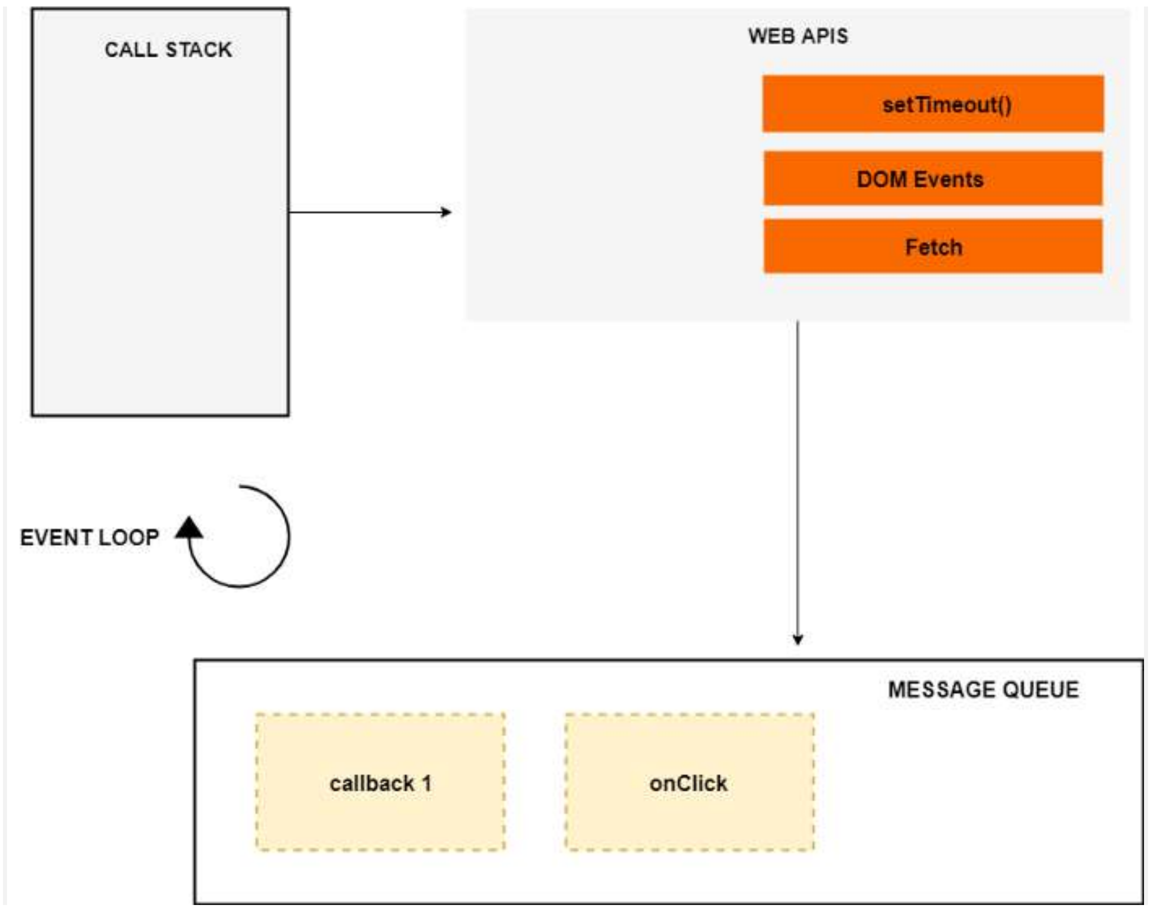
LITERATURE REVIEW

4.1 ASYNCHRONOUS JAVASCRIPT

We can use asynchronous callbacks to make our code non-blocking. For example:

```
const networkRequest = () => {
  setTimeout(() => {
    console.log('Async Code');
  }, 2000);
};console.log('Hello World');networkRequest();
```

Here I have used setTimeout method to simulate the network request. Please keep in mind that the setTimeout is not a part of the JavaScript engine, it’s a part of something known as web APIs (in browsers) and C/C++ APIs (in node.js).To understand how this code is executed we have to understand a few more concepts such event loop and the callback queue (also known as task queue or the message queue).[1]



(4.1 *An Overview of JavaScript Runtime Environment)*

The event loop, Web APIs and the message queue/task queue are not part of the JavaScript engine, it's a part of browser's JavaScript runtime environment or Nodejs JavaScript runtime environment (in case of Nodejs). In Nodejs, the web APIs are replaced by the C/C++ APIs.

4.2 FLEXGRID FEATURES

Which is a great datagrid starts with a fundamentally sound grid. FlexGrid includes only the essential vital features and offers everything else through an extensibility model. Basic Excel-like features like sorting, grouping, and editing are built-in, while the bells and whistles are optional.

The result is a small, fast, familiar, flexible grid that includes:

- Declarative markup
- Cell templates

- Data binding[2]

4.3 NODEJS

Nodejs is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Node.js = Runtime Environment + JavaScript Library

4.3.1 Features of Node.js

Following are some of the important features that make Node.js the first choice of software architects.

Asynchronous and Event Driven − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.

Very Fast − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.

Single Threaded but Highly Scalable − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.

No Buffering − Node.js applications never buffer any data. These applications simply output the data in chunks.[3]

4.4 EXPRESSJS

ExpressJS is a web application framework that provides you with a simple API to build websites, web apps and back ends. With ExpressJS, you need not worry about low level protocols, processes, etc.
Express provides a minimal interface to build our applications. It provides us the tools that are required to build our app. It is flexible as there are numerous modules available on npm, which can be directly plugged into Express.
Express was developed by TJHolowaychuk and is maintained by the Node.js foundation and numerous open source contributors.

Unlike its competitors like Rails and Django, which have an opinionated way of building applications, Express has no "best way" to do something. It is very flexible and pluggable.[4]

4.5 NPM

is a package manager for Node.js with hundreds of thousands of packages. Although it does create some of your directory structure/organization, this is not the main purpose.

The main goal, as you touched upon, is automated dependency and package management. This means that you can specify all of your project's dependencies inside your package.json file, then any time you (or anyone else) needs to get started with your project they can just run npm install and immediately have all of the dependencies installed. On top of this, it is also possible to specify what **versions** your project depends upon to prevent updates from breaking your project.

It is definitely possible to manually download your libraries, copy them into the correct directories, and use them that way. However, as your project (and list of dependencies) grows, this will quickly become time-consuming and messy. It also makes collaborating and sharing your project that much more difficult.[5]

4.7 JAVASCRIPT OBJECTS

There are eight data types in JavaScript. Seven of them are called “primitive”, because their values contain only a single thing (be it a string or a number or whatever). In contrast, objects are used to store keyed collections of various data and more complex entities. In JavaScript, objects penetrate almost every aspect of the language. So we must understand them first before going in-depth anywhere else.
An object can be created with figure brackets {…} with an optional list of *properties*. A property is a “key: value” pair, where key is a string (also called a “property name”), and value can be anything.[6]

4.8 AJAX

Ajax stands for Asynchronous Javascript And Xml and it is used for asynchronous server requests. Ajax is just a means of loading data from the server and selectively updating parts of a web page without reloading the whole page.
Basically, what Ajax does is make use of the browser's built-in XMLHttpRequest (XHR) object to send and receive information to and from a web server asynchronously, in the background, without blocking the page or interfering with the user's experience.
Ajax has become so popular that you hardly find an application that doesn't use Ajax to some extent. The example of some large-scale Ajax-driven online applications are: Gmail, Google Maps, Google Docs, YouTube, Facebook, Flickr, and so many other applications.[7]

4.9 JAVASCRIPT CLASSES

Classes in JavaScript are a special syntax for its prototypical inheritance model that is a comparable inheritance in class-based object oriented languages. Classes are just special functions added to ES6 that are meant to mimic the class keyword from these other languages. In JavaScript, we can have class declarations and class expressions, because they are just functions. So like all other functions, there are function declarations and function expressions.[8]

4.10 SYSTEMJS MODULE LOADER

SystemJS is a hookable , standards-based module loader which is used to load es5 modules. It provides a workflow where code written for production workflows of native ES modules in browsers (like Rollup code-splitting builds), can be transpiled to the System.register module format to work in older browsers that don't support native modules, running almost-native module speeds while supporting top-level await, dynamic import, circular references and live bindings, import.meta.url, module types, import maps, integrity and Content Security Policy with compatibility in older browsers back to IE11.[9]

CHAPTER 5

FEASIBILITY STUDY

As the name implies, a feasibility analysis is used to determine the viability of an idea, such as ensuring a project is legally and **technically feasible** as well as economically justifiable. It tells us whether a project is worth the investment—in some cases, a project may not be doable. There can be many reasons for this, including requiring too many resources, which not only prevents those resources from performing other tasks but also may cost more than an organization would earn back by taking on a project that isn't profitable.

A well-designed study should offer a historical background of the business or project, such as a description of the product or service, accounting statements, details of operations and management, marketing research and policies, financial data, legal requirements, and tax obligations. Generally, such studies precede technical development and project implementation.

Types of Feasibility Study

A feasibility analysis evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are four types of feasibility study—separate areas that a feasibility study examines, described below.

5.1 Economical Feasibility

System is economical feasible and can be easily implement with minimum hardware and software resources as this is a cloud based application platform is provided by cloud provider only there is a need of Internet connection and a Browser application. It is very important for designer to first analyze the system economically and determines that project is economical feasible or not. Costs and benefits of the proposed computer system must always be considered together, because they are interrelated and often interdependent. Although the systems analyst is trying to propose a system that fulfills various information requirements, decisions to continue with the proposed system will than 10 percent?" The systems analyst should realize, however, that he or she cannot rely on what-if analysis for everything if the proposal is to be credible, meaningful, and valuable.

The systems analyst has many forecasting models available. The main condition for choosing a model is the availability of historical data. If they are unavailable, the analyst must turn to one of the judgment methods: estimates from the sales force, surveys to estimate customer demand, Delphi studies (a consensus forecast developed independently by a group of experts through a series of iterations), creating scenarios, or drawing historical analogies.

If historical data are available, the next differentiation between classes of techniques involves whether the forecast is conditional or unconditional. Conditional implies that there is an association among variables in the model or that such a causal relationship exists. Common methods in this group include correlation, regression, leading indicators, econometrics, and input/output models.

Unconditional forecasting means the analyst isn't required to find or identify any causal relationships. Consequently, systems analysts find that these methods are low-cost, easy-to-implement alternatives. Included in this group are graphical judgment, moving averages, and analysis of time-series data. Because these methods are simple, reliable, and cost effective, the remainder of the section focuses on them.

- **Estimation of Trends**

Trends can be estimated in a number of different ways. One way to estimate trends is to use a moving average. This method is useful because some seasonal, cyclical, or random patterns may be smoothed, leaving the trend pattern. The principle behind moving averages is to calculate the arithmetic mean of data from a fixed number of periods; a three-month moving average is simply the average of the last three months. For example, the average sales for January, February, and March is used to predict the sales for April. Then the average sales for February, March, and April are used to predict the sales for May, and so on.

When the results are graphed, it is easily noticeable that the widely fluctuating data are smoothed. The moving average method is useful for its smoothing ability, but at the same time it has many disadvantages. Moving averages are more strongly affected by extreme values than by using graphical judgment or estimating using other methods such as least squares. The analyst should learn forecasting well, as it often provides information valuable in justifying the entire project.

- **Tangible Costs**

The concepts of tangible and intangible costs present a conceptual parallel to the tangible and intangible benefits discussed already. Tangible costs are those that can be accurately projected by the systems analyst and the business's accounting personnel.

Included in tangible costs are the cost of equipment such as computers and terminals, the cost of resources, the cost of systems analysts' time, the cost of programmers' time, and otheremployees.
salaries. These costs are usually well established or can be discovered quite easily, and are the costs that will require a cash outlay of the business.

- **Intangible Costs**

Intangible costs are difficult to estimate and may not be known. They include losing a competitive edge, losing the reputation for being first with an innovation or the leader in a field, declining company image due to increased customer dissatisfaction, and ineffective decision making due to untimely or inaccessible information. As you can imagine, it is next to impossible to project a dollar amount for intangible costs accurately. To aid decision makers who want to weigh the proposed system and all its implications, you must include intangible costs even though they are not quantifiable.

5.2 Technical Feasibility

It is the study of the function performance and constraints that may affect the ability to achieve an acceptable system. The project development requires designer to have technical knowledge of salesforce.com for both application development and database system. Technical feasibility is one of the most important criteria for selecting material for digitisation. The physical characteristics of source material and the project goals for capturing, presenting and storing the digital surrogates dictate the technical requirements. Libraries must evaluate those requirements for each project and determine whether they can be met with the resources available. If the existing staff, hardware and software resources cannot meet the requirements, then the project will need funding to upgrade equipment or hire an outside conversion agency. If these resources are not available, or if the technology does not exist to meet the requirements, then it is not technically feasible to digitise that material.
Considerations for technical feasibility include:

- **Image capture*:*** Image capture requires equipment, such as a scanner or a digital camera. Different types of material require different equipment, and different equipment produces images of differing quality. When selecting materials for digitising, technical questions that need to be addressed include: does the original source material require high resolution to capture? Are there any oversized items in the collection? Are there any bound volumes in the collection? What critical features of the source material must be captured in the digital product? In what condition are the source materials? Will they be damaged by the digitisation process?

- **Presentation***:* Presentation refers to how the digitised materials will be displayed online. Consider the following questions to determine the technical feasibility of presenting the digitised material:
  Will the materials display well digitally?

  How will users use the digital versions?

  How will users navigate within and among digital collections?

  Do the institutionally supported platforms and networked environment have the capability for accessing the images and delivering them with reasonable speed to the target audience?

  Do the images need to be restricted to a specified community?

  Do the images need special display features such as zooming, panning and page turning?

- **Description***:* Some archival and special collections have been catalogued for public use and contain detailed finding aids with descriptions about each item and the collection as a whole. Other collections may not have been reviewed and documented in detail and do not have much information on individual items. Those collections will require more time, human resources and significant additional expense to research the materials, check the accuracy of the information obtained, and write appropriate descriptions to aid in discovery and use of the digital items. Typewritten documents, like the Drew Pearson columns described above, can have reasonably accurate OCR applied to them to replace, for some uses, the detailed descriptions required for discovery of hand-written or picture materials. The selection criteria should clearly state whether the items and collections that do not contain descriptions should be considered for digitisation.

- **Human resources**: When selecting materials for digitisation, the library should consider whether it has the staff and skill sets to support the digitisation, metadata entry, user interface design, programming and search engine configuration that is required for the project to implement the desired functionality. For large collaborative projects, dedicated staff are usually required from each partner. Digital collections also require long-term maintenance, which needs to be considered and planned for. If a project does not have the necessary staff and skills in-house, but funding is available, outsourcing may be a good choice.

5.3 Behavioural Feasibility

In the application domain our system works as an application. There are simple form to fill and service requires no ambiguous entries, all the behavioural entries are simple and GUI based. The system design is very user friendly, interactive. The application should be used by Administrator. People are inherently resistant to change, and computers have been known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely to have toward the development of a computerized system. [t is common knowledge that computer installations have something to do with turnover, transfers, retraining, and changes in employee job status. Therefore, it is understandable that the introduction of a candidate system requires special effort to educate, sell, and train.

In our safe deposit example, three employees are more than 50 years old and have been with the bank over 14 years, four years of which have been in safe deposit. The remaining two employees are in their early thirties. They joined safe deposit about two years before the study. Based on data gathered from extensive interviews, the younger employees want the programmable aspects of safe deposit (essentially billing) put on a computer. Two of the three older employees have voiced resistance to the idea. Their view is that billing is no problem. The main emphasis is customer service-personal contacts with customers. The decision in this case was to go ahead and pursue the project.

5.4 Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

The operational feasibility assessment focuses on the degree to which the proposed development project fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture and existing business processes.

To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational behaviours are to be realised. A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases

An example of an operational feasibility study, or the fourth type, analyzes the inside operations on how a deemed process will work, be implemented, and how to deal with change resistance and acceptance.

Operational feasibility studies are generally utilized to answer the following questions:

- **Process** – How do the end-users feel about a new process that may be implemented?
- **In-House Strategies** – How will the work environment be affected? How much will it change?
- **Adapt & Review** – Once change resistance is overcome, explain how the new process will be implemented along with a review process to monitor the process change.

If an operational feasibility study must answer the six items above, how is it used in the real world? A good example might be if a company has determined that it needs to totally redesign the workspace environment.

After analyzing the technical, economic, and scheduling feasibility studies, next would come the operational analysis. In order to determine if the redesign of the workspace environment would work, an example of an operational feasibility study would follow this path based on six elements:

- **Process** – Input and analysis from everyone the new redesign will affect along with a data matrix on ideas and suggestions from the original plans.
- **Evaluation** – Determinations from the process suggestions; will the redesign benefit everyone? Who is left behind? Who feels threatened?
- **Implementation** – Identify resources both inside and out that will work on the redesign. How will the redesign construction interfere with current work?
- **Resistance** – What areas and individuals will be most resistant? Develop a change resistance plan.
- **Strategies** – How will the organization deal with the changed workspace environment? Do new processes or structures need to be reviewed or implemented in order for the redesign to be effective?
- **Adapt & Review** – How much time does the organization need to adapt to the new redesign? How will it be reviewed and monitored? What will happen if through a monitoring process, additional changes must be made?

The most important part of operational feasibility study is input—from everyone, especially when it affects how or what an organization does as far as processes. If the process were to build a new sports arena for a client, then a study determining how the arena will operate in a way that is conducive to its inhabitants, parking, human flow, accessibility and other elements is a good example of an operational feasibility study.

Create a sample operational feasibility study if you plan to change something inside the company that will affect how the organization runs or when a client asks you to explore a new product or process that will affect elements within their own organization.

CHAPTER 6

TESTING

6.1 BLACK BOX TESTING:-

The technique of testing without having any knowledge of the interior workings of the application is called blackbox testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a blackbox test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

6.2 WHITE BOX TESTING:-

Whitebox testing is the detailed investigation of internal logic and structure of the code. Whitebox testing is also called glass testing or openbox testing. In order to perform whitebox testing on an application, a tester needs to know the internal workings of the code.

6.3 GREY BOX TESTING:-

Greybox testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

6.4 UNIT TESTING:-

Unit Testing contains the testing of each unit of Recruitment Application. We have tested each interface by input values and check whether it is working properly working or not we also tested database connectivity. We have entered value in interface and check that the values are properly goes to corresponding tuples or not.

6.5 INTEGRATION TESTING:-

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom-up integration testing and Topdown integration testing.

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team.

6.6 ACCEPTANCE TESTING:-

This is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client’s requirement. The QA team will have a set of pre-written scenarios and test cases that will be used to test the application. In System Testing we have tested entire Recruitment Application. We have run all programs as a single system and inputs various test cases and analyse that all are going correctly or not. In system testing we have tested various test cases. According to which, Application showed the corresponding error message

CHAPTER 7

Conclusion

The package was designed in such a way that future modifications can be done easily.
Automation of the entire system improves the efficiency.
It provides a friendly graphical user interface which proves to be better when compared to the existing system.
It gives appropriate access to the authorized users depending on their permissions.
A software project means a lot of experience. We learned a lot through this project. This project has sharpened our concept game engine, animation, and the software-hardware interface. We learned a lot about different documentation. Now I have much wider knowledge of the features Java offers and put into practice various object-oriented methods that learnt last semester.

REFERENCES

[1] John Resig, *Pro Javascript Techniques*, 2007

[2] Martin Rinehart, *Javascript Object Oriented Programming*, 2015

[3] Shane Hudson, *Javascript Creativity*, 2014

[4] Terry McNavage, *Javascript for Absolute Begginers*, 2010

[5] Fabio Nelli, Beginning Javascript Charts

[6] Adam Freeman, *Pro Javascript for Web Apps*, 2012

[7] Azat Mardan, *Full Stack JavaScript*, 2015

[8] Sammie Bae, *Javascript Data Structures and Algorithms*, 2019

[9] Russ Ferguson, Keith Cirkel, *Javascript Recipes*, 2017

[10] Jeanine Meyer, *HTML5 and Javascript Projects*, 2011

[11] Russs Ferguson, Christian Heilmann, *Beginning Javascript with DOM Scripting and AJAX*, 2013

[12] Anto Aravinth, Shrikanth Machiraju, *Beginning Functional Javascript*, 2018,

[13] David R. Brooks, *An Introduction to HTML and JavaScript*, 2007

[14] Laurence Svekis, *Modern Javascript Fundamentals*, 2020

[15] Mikael Olsson, *Javascript Quick Syntax Reference*, 2015