SheetX: Mini-Excel Flask App

2023 – 2027

Submitted by:

Usman Ali Ashraf 2023-CS-106

Supervised by:

Mr. Ali Raza

Course:

CSC-200L Data Structures and Algorithms (L)

Department of Computer Science

University of Engineering and Technology

Lahore, Pakistan

Contents

Introduction

SheetX is a modern, lightweight clone of Microsoft Excel built with **Flask** on the backend and **vanilla JavaScript** on the frontend. It's designed to give us the feel of a spreadsheet app, right inside your browser, with support for importing/exporting data, infinite scrolling, formulas, formatting, search, and even clipboard functionality.

The goal of this project was to create something both **practical** and **fun to use**: a fast, smooth, green-black themed spreadsheet that doesn't rely on heavy frameworks. Everything is managed in memory, so we don't need to set up a database.

How to Run the Project

1. Install Flask:
2. `pip install flask`
3. Start the application:
4. `python app.py`
5. Open your browser and go to: **http://127.0.0.1:5000/**

That's it! You'll be greeted by your very own SheetX spreadsheet.

Project Structure

The folder layout is simple and clean:

```
Mini-Excel-in-Flask/
├── app.py                    # Main Flask server
├── spreadsheet/              # Backend logic
│   ├── __init__.py
│   ├── cell.py               # Cell-level operations
│   └── spreadsheet.py        # Spreadsheet grid logic
├── static/                   # Frontend files
│   ├── styles.css            # Custom styling
│   └── app.js                # Core frontend logic
├── templates/                # HTML templates
│   └── index.html
└── README.md
```

User Interface & Controls

SheetX is designed with a **green-black theme** that feels modern yet comfortable.

Ribbon Controls (Top Bar)

The ribbon at the top provides quick access to all the important features:

- **Formulas Dropdown**: Quick insert of popular formulas (`SUM`, `AVERAGE`, `MAX`, etc.).
- **Functions Dropdown**: Insert advanced functions (`IF`, `CONCATENATE`, `LEFT`, etc.).
- **Font Size Buttons**: Increase (`A+`) or decrease (`A-`) text size in selected cells.

- **Bold / Italic**: Style your cell content with a single click.
- **Import CSV**: Upload and load data from a CSV file.
- **Export CSV**: Save the current sheet to your computer with a timestamped name.
- **Search Bar**: Instantly highlight all cells containing your search term.

Data Structures Used

At the heart of **Mini Excel** lies a simple yet powerful data design: a **2D list of Cell objects**. This setup gives us the flexibility of a real spreadsheet while keeping everything fast and easy to manage.

1. Spreadsheet Grid (2D List)

The main structure is a **two-dimensional list** (`self.grid`) that represents rows and columns:

```
self.grid = [[Cell() for _ in range(cols)] for _ in range(rows)]
```

- Think of it as a table:
    - Each **row** is a list.
    - Each **column** is a position inside that row.
- Access pattern: `grid[row][col]`
- Default size: **20 × 20**, but it dynamically expands when we add more rows or columns.

This 2D grid mirrors how spreadsheets naturally work, making cell lookups super quick.

2. Cell Objects

Every element in the grid is a **Cell** object that stores both the content and its formatting:

```
class Cell:
    def __init__(self, value='', font_size=16, bold=False, italic=False,
bg_color=None, text_color=None):
        self.value = value
        self.font_size = font_size
        self.bold = bold
        self.italic = italic
        self.bg_color = bg_color
        self.text_color = text_color
```

- **value** → The actual content (text, numbers, formulas).
- **font_size, bold, italic** → Styling options.
- **bg_color & text_color** → Custom coloring for each cell.

This design means we're not just storing numbers or strings, we're keeping formatting right alongside the content.

3. Dynamic Growth

The grid isn't fixed. When we scroll further or import a bigger CSV, the app auto-expands:

```
def _ensure_size(self, min_rows, min_cols):
    if min_rows > self.rows:
        self.add_rows(min_rows - self.rows)
    if min_cols > self.cols:
        for row in self.grid:
            row.extend([Cell() for _ in range(min_cols - self.cols)])
        self.cols = min_cols
```

- Adding new rows → Creates fresh lists of Cell objects.
- Adding new columns → Extends each existing row with new cells.

This ensures the spreadsheet always has room for new data.

4. Why This Works So Well

- **O(1) access**: Jump straight to any cell like `grid[5][3]`.
- **Natural spreadsheet mapping**: Rows & columns just like Excel.
- **Easy serialization**: Convert grid → JSON → frontend without hassle.
- **Lightweight but flexible**: Perfect balance between speed and feature support.

Searching Algorithm

The project uses a **Linear Search** to find values in the spreadsheet.

- **Process**: Loops through all table cells (<td>) and checks if the cell's text contains the user's search term (case-insensitive). All matches are highlighted, and the first match is focused and scrolled into view.
- **Complexity**:
    - **Time**: O (n × m)
    - **Space**: O(k)
- **Why Linear Search?**
    - Data is unsorted and stored in DOM elements
    - Substring matching is required instead of exact matching
    - Easy to implement and update in real time

Code Implementation

```
highlightSearchMatches(term) {

  // Clear previous highlights

  this.tbody.querySelectorAll('td.search-match')

    .forEach(td => td.classList.remove('search-match'));

  this.searchMatches = [];

  this.currentMatchIdx = 0;
```

```
 if (!term) return;
 const lowerTerm = term.toLowerCase();
 const tds = this.tbody.querySelectorAll('td');
 for (const td of tds) {
   if (td.textContent.toLowerCase().includes(lowerTerm)) {
     td.classList.add('search-match');
     this.searchMatches.push(td);
   }
}
```

Core Features

1. Infinite Scroll

Instead of loading the entire sheet at once, SheetX dynamically loads rows as we scroll. This keeps the interface smooth even with thousands of rows.

- Scroll down → new rows are rendered.
- Prevents memory overload.
- Seamless experience without page reloads.

2. Import & Export CSV

SheetX makes it super easy to bring in your data or take it with you.

- **Import**: Select a `.csv` file, and the sheet instantly populates.
- **Export**: Download the current sheet as a `.csv` file.
- **Auto-Naming**: Exported files include a timestamp, e.g. `SP_20250728_153000.csv`.

3. Cell Formatting

We can style your data just like in Excel:

- **Bold / Italic** → Toggle text emphasis.
- **Font Size** → Increase/decrease size for readability.
- **Text & Background Colours** → Highlight important cells (paintbrush and text color options).
- **Multi-Selection Support** → Drag or Ctrl+Click multiple cells to format at once.

4. Formulas & Functions

SheetX supports both **direct formulas** and **Excel-like functions**.

Basic Math Formulas:

- `=A1+A2`
- `=A1-B2`, `=A1*B2`, `=A1/B2`

Range Functions:

- `=SUM(A1:A10)`
- `=AVERAGE(A1:A10)`

- `=MIN(A1:A10)`
- `=MAX(A1:A10)`
- `=COUNT(A1:A10)`

Advanced Functions:

- `=IF(A1>10, "Yes", "No")`
- `=CONCATENATE(A1, B1, "text")`
- `=LEFT(A1, 3)`
- `=RIGHT(A1, 4)`
- `=LEN(A1)`
- `=ROUND(A1, 2)`

Condition Syntax for IF:
Supports operators like =, >, <, >=, <=.

5. Clipboard Support

SheetX supports copy-paste like a real spreadsheet:

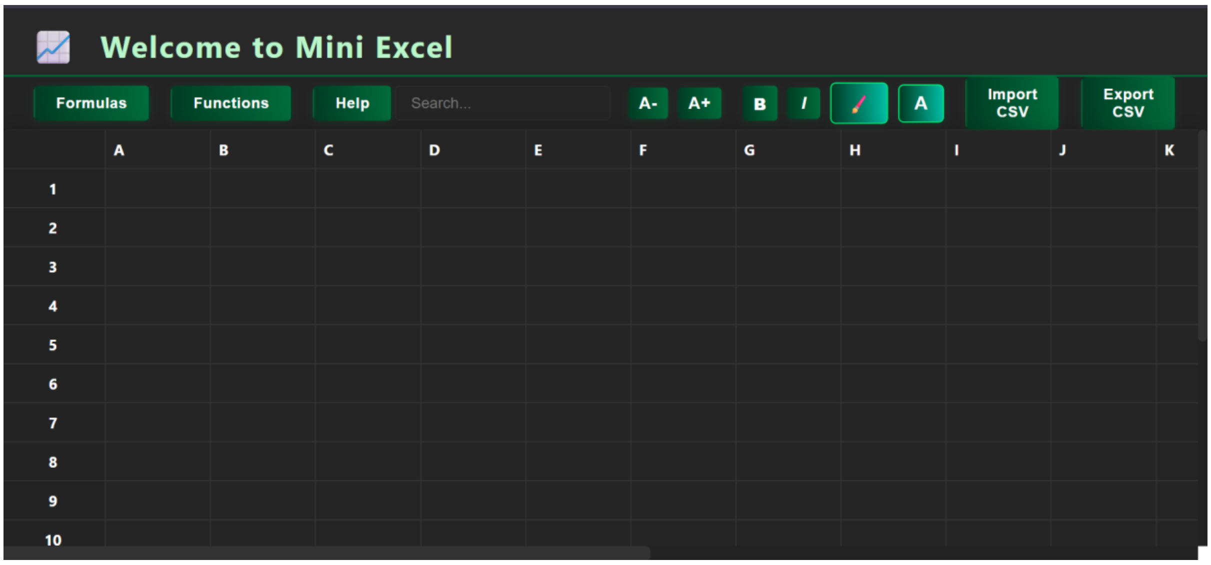
- **Ctrl + C** → Copy selected cells.
- **Ctrl + V** → Paste data into cells.
- Multi-cell paste supported: paste tabular data directly from Excel/Google Sheets.

6. Search Functionality

Finding data in large sheets is easy:

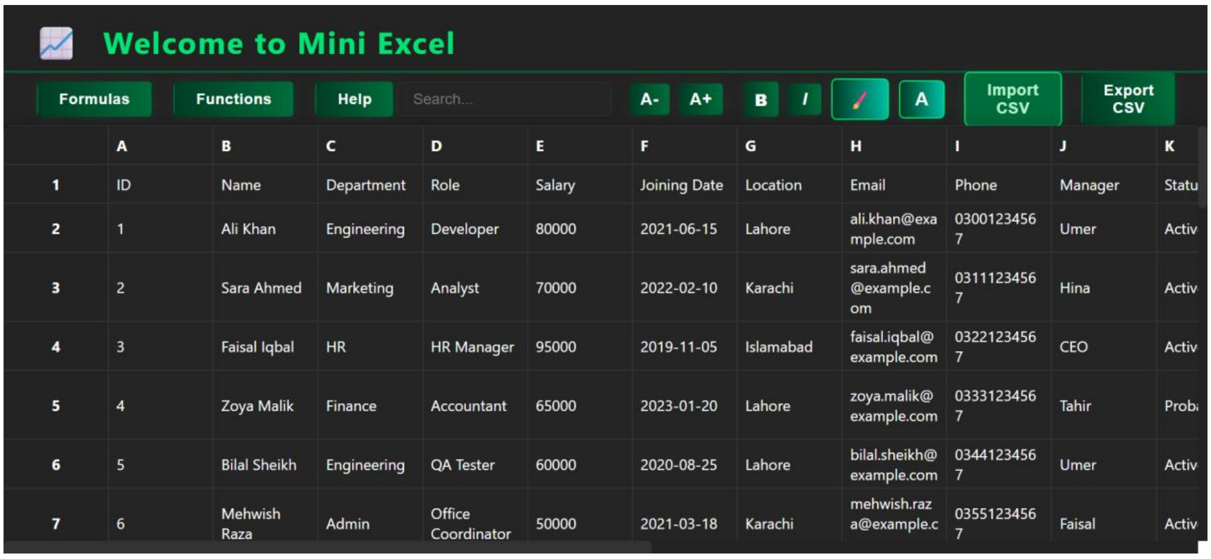
- Type a term into the **search bar** and hit Enter.
- All visible matching cells are highlighted in **yellow**.
- The first match automatically scrolls into view.
- Press Escape to clear highlights.

Wireframes
Here are some visuals from SheetX.



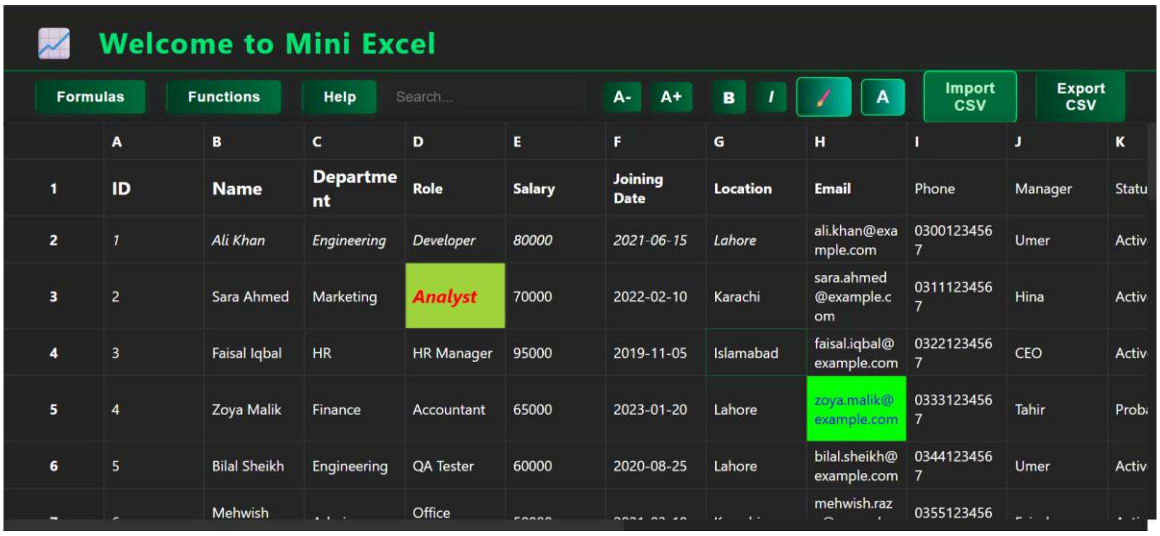
	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											

Figure 1: HomePage of SheetX



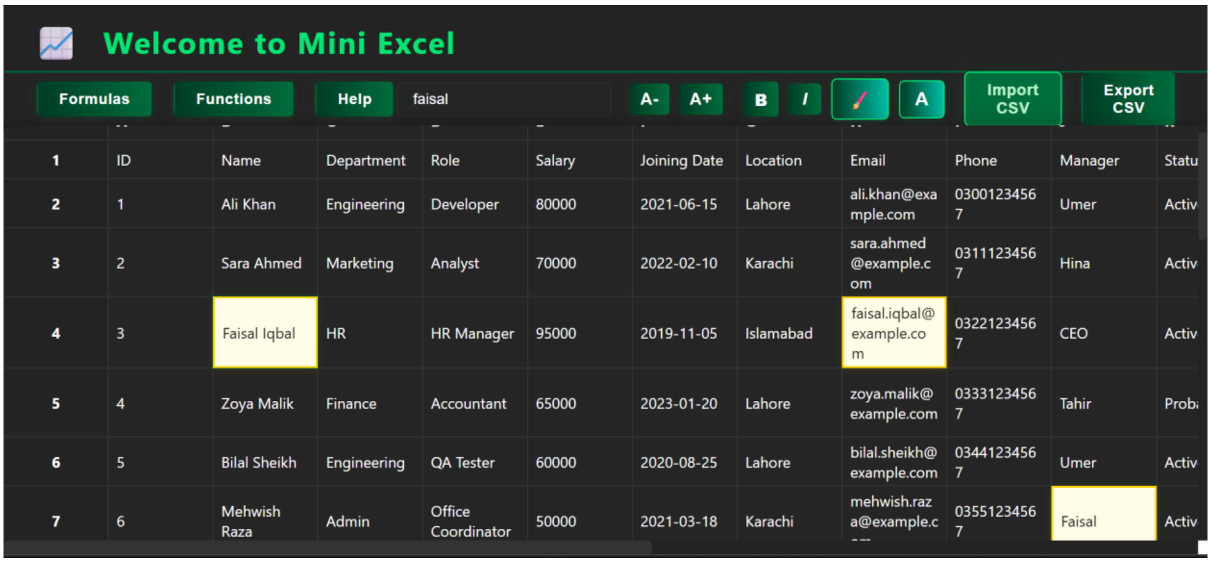
	A	B	C	D	E	F	G	H	I	J	K
1	ID	Name	Department	Role	Salary	Joining Date	Location	Email	Phone	Manager	Statu
2	1	Ali Khan	Engineering	Developer	80000	2021-06-15	Lahore	ali.khan@example.com	03001234567	Umer	Activ
3	2	Sara Ahmed	Marketing	Analyst	70000	2022-02-10	Karachi	sara.ahmed@example.com	03111234567	Hina	Activ
4	3	Faisal Iqbal	HR	HR Manager	95000	2019-11-05	Islamabad	faisal.iqbal@example.com	03221234567	CEO	Activ
5	4	Zoya Malik	Finance	Accountant	65000	2023-01-20	Lahore	zoya.malik@example.com	03331234567	Tahir	Prob
6	5	Bilal Sheikh	Engineering	QA Tester	60000	2020-08-25	Lahore	bilal.sheikh@example.com	03441234567	Umer	Activ
7	6	Mehwish Raza	Admin	Office Coordinator	50000	2021-03-18	Karachi	mehwish.raza@example.c	03551234567	Faisal	Activ

Figure 2: Imported csv file



	A	B	C	D	E	F	G	H	I	J	K
1	**ID**	**Name**	**Department**	**Role**	**Salary**	**Joining Date**	**Location**	**Email**	Phone	Manager	Statu
2	*1*	*Ali Khan*	*Engineering*	*Developer*	*80000*	*2021-06-15*	*Lahore*	ali.khan@example.com	03001234567	Umer	Activ
3	2	Sara Ahmed	Marketing	***Analyst***	70000	2022-02-10	Karachi	sara.ahmed@example.com	03111234567	Hina	Activ
4	3	Faisal Iqbal	HR	HR Manager	95000	2019-11-05	Islamabad	faisal.iqbal@example.com	03221234567	CEO	Activ
5	4	Zoya Malik	Finance	Accountant	65000	2023-01-20	Lahore	zoya.malik@example.com	03331234567	Tahir	Prob
6	5	Bilal Sheikh	Engineering	QA Tester	60000	2020-08-25	Lahore	bilal.sheikh@example.com	03441234567	Umer	Activ
		Mehwish		Office				mehwish.raz	0355123456		

Figure 3: Text and Cell Formatting



1	ID	Name	Department	Role	Salary	Joining Date	Location	Email	Phone	Manager	Statu
2	1	Ali Khan	Engineering	Developer	80000	2021-06-15	Lahore	ali.khan@example.com	03001234567	Umer	Activ
3	2	Sara Ahmed	Marketing	Analyst	70000	2022-02-10	Karachi	sara.ahmed@example.com	03111234567	Hina	Activ
4	3	Faisal Iqbal	HR	HR Manager	95000	2019-11-05	Islamabad	faisal.iqbal@example.com	03221234567	CEO	Activ
5	4	Zoya Malik	Finance	Accountant	65000	2023-01-20	Lahore	zoya.malik@example.com	03331234567	Tahir	Prob
6	5	Bilal Sheikh	Engineering	QA Tester	60000	2020-08-25	Lahore	bilal.sheikh@example.com	03441234567	Umer	Activ
7	6	Mehwish Raza	Admin	Office Coordinator	50000	2021-03-18	Karachi	mehwish.raza@example.c	03551234567	Faisal	Activ

Figure 4: Search feature

Technical Details

Backend

- Powered by **Flask**.
- Uses an **in-memory 2D grid** (no database required).
- Handles cell updates, formula evaluation, and CSV import/export.

Frontend

- Built in **vanilla JavaScript** for speed.
- Handles rendering, selection, formatting, clipboard, and scrolling.
- Infinite scroll ensures efficient row rendering.

Data Persistence

- Currently **session-only** (data resets on server restart).
- Import/Export provides manual save/load functionality.

Future Improvements

Some planned upgrades include:

- Persistent storage (Database or file-based).
- Insert/Delete rows and columns.
- Freeze panes & merge cells.
- Full-sheet backend search.
- User authentication for personal sheets.