Networking Ninja's Team Strategy (In Progress):

Weekly Group Check-ins:

- **Agenda**:
  - Review last week's progress.
  - Discuss any blockers or issues faced.
  - Plan for the upcoming week.
- **Participants**: All team members.
- **Outcome**: Updated sprint backlog, clear next steps.

Weekly TA Meetings:

- **Agenda**:
  - Present sprint progress.
  - Validate approach and receive feedback.
  - Plan for the next sprint.
- **Participants**: All team members and TA.
- **Outcome**: TA's feedback incorporated, strategic adjustments made.

In Class Stand-ups:

- **Format**: 10-minute meeting.
- **Questions**:
  - What did you do yesterday?
  - What will you do today?
  - Are there any blockers?
- **Outcome**: Team synchronization, early identification of issues.

Pair Programming Sessions:

- **Format**: Scheduled based on tasks.
- **Purpose**:
  - Cross-functional collaboration (Client-Server integration).
  - Knowledge sharing and improved code quality.
- **Outcome**: Completed features with collaborative effort.

Individual Work Sessions:

- **Format**: Allocated based on personal schedules.
- **Purpose**:
  - Focused work on specific tasks.
  - Learning and upskilling in required technologies.
- **Outcome**: Individual progress on assigned tasks, skill enhancement.

Our Changes to the Agile Methodology (In Progress)

Pair Programming with Rotating Pairs:

- **Why**: Enhances code quality, facilitates knowledge sharing, and reduces the "bus factor" by ensuring multiple team members are familiar with various parts of the codebase.
- **How**: Each week, we rotate pairs among team members to work on different parts of the project. This rotation ensures that everyone gets exposure to both Client and Server tasks, promoting a well-rounded skill set across the team.

Focus Blocks for Learning:

- **Why**: Allocating dedicated time for learning new technologies and experimenting with innovative solutions ensures continuous skill development and encourages creative problem-solving.
- **How**: Each team member sets aside specific time blocks each week to focus solely on learning and experimenting with new technologies relevant to the project.

Integrated Sprint Reviews and Retrospectives:

- **Why**: Combining sprint reviews and retrospectives allows for a holistic evaluation of the sprint, addressing both the progress made and the process improvements needed.
- **How**: At the end of each sprint, we conduct a combined meeting where we review the completed work and discuss what went well, what didn’t, and how we can improve.

Sprint 1

Our goal for sprint1 is to address the majority of MVP requirements. This allows us to adhere to scrum methodologies by being able to declare the project “done” at any time moving forward.

General Goals

- Spreadsheet UI implementation
- Spreadsheet ability implementation
- Spreadsheet functionality implementation
- Server implementation (REST API)

General Pair Assignments:
Client: Shaan, Shubham
Server: Brandon, Emily

Specific Requirements Being Implemented:

#	Task	Status	Micro-Tasks	Tag
1	Spreadsheet user interface of cells labeled alphabetically by column and numbered by rows	Done	• Create a cell visual UI • Create a table of those cells and label them	Client
1a	Spreadsheet front end GUI	Done		Client
1b	Spreadsheet cell functionality that allows users to select and edit the contents of the cell.	Done	• Cell must be clickable • Cell must have a text field to edit	Client
2	Ability to create, open, edit, and delete spreadsheets	Done	• Creating a tab bar • Create a X to delete a sheet • Create + in the tab bar to add sheets • Make tabs	Client

			clickable and have individual sheets	
2a	Front end file taskbar that implements create, edit, and delete	Done	●	Client
3	SQL Database implementation with spreadsheet data on the Server	Done		Server

Sprint 2

Our goal for sprint2 is to address the majority of Desirables requirements. This allows us to add additional features to the already existing application giving our users further flexibility and control over permissions.

General Goals

- Spreadsheet Login and permissions UI implementation
- Spreadsheet permissions(view, edit)
- Spreadsheet functionality Login
- Spreadsheet Function interpretation and execution of operation

General Pair Assignments:
Client: Shaan, Emily
Server: Brandon, Shubham

Specific Requirements Being Implemented:

#	Task	Status	Description	Worked on by:
3c	Multiple users can edit the same spreadsheet	HIGH	This is supposed to be a collaborative app so multiple users should be able to work on the same sheet	Server
4	User profiles	HIGH	User profiles and logins will be used to secure data on the database	Client

4a	Login UI	FAIR	Interactive login functionality for user	Client
4b	Registration UI	FAIR	Register functionality for use	Client
4c	Logout Button	FAIR	To Logout of the application	Client
4d	User password hashed in database	VERY HIGH	Hashing user passwords protects their profile's security	Server
5	Ability for spreadsheet to support and distinguish between different cell types (numerical, alphabetical, boolean, or mixed)	HIGH	For functions support the spreadsheet must be able to know what functions can be applied given the type of the cell	Server
5a	Support for basic numerical functions like addition, subtraction, multiplication, division (+, - , *, / , <, >, =, <>, &, |, :)	HIGH	Mathematical functions allow for a range of user cases like balancing a cost sheet or calculating taxes.	Client
5b	Support for additional boolean operations: IF, SUM, MIN, MAX, NOT, AND, OR	HIGH	Boolean logic allows for more dynamic interactions between cells	Client
6	Copy, Cut, and Paste functionality for single cells	MEDIUM	Quickens the workflow of a user that has to move information from cells to others	Client
7	Ability to add users as editor or viewer status on a spreadsheet	FAIR	User privileges can allow for collaboration even when one user doesn't want the other to potentially manipulate the spreadsheet	Client/Server
8	Connect front-end to back-end			

We are dropping this as this is not vital for our application at this time:

4b	Registration UI	FAIR	Register functionality for use	Client

7	Ability to add users as editor or viewer status on a spreadsheet	FAIR	User privileges can allow for collaboration even when one user doesn't want the other to potentially manipulate the spreadsheet	Client/Server

Backlog that is being moved to sprint 3:

3c	Multiple users can edit the same spreadsheet	HIGH	This is supposed to be a collaborative app so multiple users should be able to work on the same sheet	Server
4c	Logout Button	FAIR	To Logout of the application	Client
4d	User password hashed in database	VERY HIGH	Hashing user passwords protects their profile's security	Server
5b	Support for additional boolean operations: IF, SUM, MIN, MAX, NOT, AND, OR	HIGH	Boolean logic allows for more dynamic interactions between cells	Client
8	Connect front-end to back-end	HIGH		Client/Server

Sprint 3:

Our goal for sprint 3 is to continue to address the majority of Desirables requirements. This allows us to add additional features to the already existing application giving our users further flexibility and control over permissions.

General Goals

- Spreadsheet Login and permissions UI implementation
- Spreadsheet permissions(view, edit)
- Spreadsheet functionality Login
- Spreadsheet Function interpretation and execution of operation

General Pair Assignments:
Client: Shaan, Emily
Server: Brandon, Shubham

3c	Multiple users can edit the same spreadsheet	HIGH	This is supposed to be a collaborative app so multiple users should be able to work on the same sheet	Server
4c	Logout Button	FAIR	To Logout of the application	Client
4d	User password hashed in database	VERY HIGH	Hashing user passwords protects their profile's security	Server
5b	Support for additional boolean operations: IF, SUM, MIN, MAX, NOT, AND, OR	HIGH	Boolean logic allows for more dynamic interactions between cells	Client
8	Connect front-end to back-end	HIGH		Client/Server

Part of sprint 3 is dedicated to refactored our code, and getting our test coverage up, have multiple demos with the TA and correct our understanding of the implementation.

Sprint 4:

Our goal for Sprint 4 is to finish Desirables requirements and complete updates to the Specification. This allows us to add additional features to the already existing application giving our users further flexibility and control over permissions

1	Specification update #1	HIGH	Allow for floating point number and positive and negative values	Client/Server
2	Specification update #2	HIGH	The initial id for sheets is “0”.	Client/Server
3	Specification update #3	HIGH	Add the COPY function: = COPY($A1, "$B1") What this is intended to do is: 1. Get the value of the cell indicated by the first REF, e.g. $A1, and copy it in the REF contained in the String that contains the second argument. 2. Return the copied value 3. This function is only recomputed if $A1 changes.	Client/Server
4	Specification update #4	HIGH	=DEGUB(SUM($A1:$B4)) The DEBUG function should print the value returned by its argument on the standard output and return that value as well.	Client/Server
56	Specification update #5	HIGH	The testing department request that you provide a way to start your client with the following command line arguments myclient --url="https://..." --name="team44" --password="foofum" --publisher="alice" --sheet="S15"	Client/Server

			These first three arguments have the obvious meaning. The last two should do the following: tell you client to register to the server, and open the sheet “S15” from the publisher “alice”. That sheet should be opened in your UI. The testing department thanks you for your cooperation and will make sure to buy the whole team a round of gingerale after the next company standup	
6	Customer note #1	HIGH	Automatic save,	Client/Server
7	Customer note #2	HIGH	Formulas should calculate concurrently when a value that is referenced is updated.	Client
8	Customer note #3	HIGH	One source of truth, the owner of a sheet is the only one that has a source of truth, there cannot be more then one source of truth (data does not needed to be synced across two clients of a owner they should be synced across shared users)	Client/Server
9	Customer note #4	HIGH	Efficient cell computation (should be recursive not hard calculate the whole sheet)	Client
10	Customer note #5	HIGH	Most recent updates is what is reflected on the sheet not any changes that are made in a backlog	Client/Server
11	Support for alphabetical operations: LOWER, UPPER, CONCAT	FAIR	Alphabetical operations allow for formatting cells properly	Client/Server
12	Add/Delete Rows and Cols	FAIR		Client/Server