Initial Report of Group BLOCKS7
Block Model Compression Algorithm

a1162576	Karl Asenstorfer
a1806297	Po-Yi Lee
a1804817	Xiaoman Li
a1784375	Yuanpeng Liu
a1782685	Yang Lu
a1797683	Jiaping Qi
a1786785	Hechen Wang
a1784184	Kaiyang Xue
a1811518	Liuyang Yun

Table of Content

Project Vision

The customer, Maptek, regularly sends spatial data encoded as Block Models between their various services and to support their customers. These models can be Gigabytes in size. Compressing these models is an important requirement, since it will reduce their bandwidth and storage requirements, and allow them to serve customers with less performant internet connections. Therefore, by developing an algorithm that compresses this data to a high degree, helps solving this requirement.

Maptek also requires the algorithm to complete the task in a reasonable amount of time. A slow algorithm will add large latency into their systems, which they want to avoid. Therefore speed is also a key requirement. A balance between compression ratio and speed of execution is needed. Various compression techniques will be explored to see which combination will give the best compression at reasonable speeds.

The customer is running a friendly competition for all the other teams in the course, so that we can all share our approaches, and get inspiration for refining our algorithm as it is applied to increasingly more challenging datasets. We aim to perform near the top of both the speed and compression leader-boards, with a heavier weighting towards compression, because our team will be using Python rather than C.

Customer Q&A

Having limited knowledge about block model compression, we just attended the kickoff meeting and listened to the clients' introduction of the project.

However, we had a better understanding of the project by researching related algorithms and asked clients the following questions on the project via Slack:

- What is the final application (or the actual demand) that the customers expect for the Block Model Compression?
- Will the whole project for this semester be improving the block model compression algorithm?
- Are the researched methods such as Pruning, Quantization, Low-rank and approximation, and sparsity, etc. that we are looking for? Or is it better to implement the algorithm from scratch?
- Could you explain what kind of output in the uploaded python file is required for the Maptek system? For example, "print out all lines of the block" or "save all lines of the blocks in a CSV file"?

In the kickoff meeting, some students from other groups had been prepared and asked technical questions. From this experience of the first meeting with the customer, we learned that it would be better to prepare some questions for some potential problems in the future. We should have researched some of the papers about block model compression before the kickoff meeting.

Users

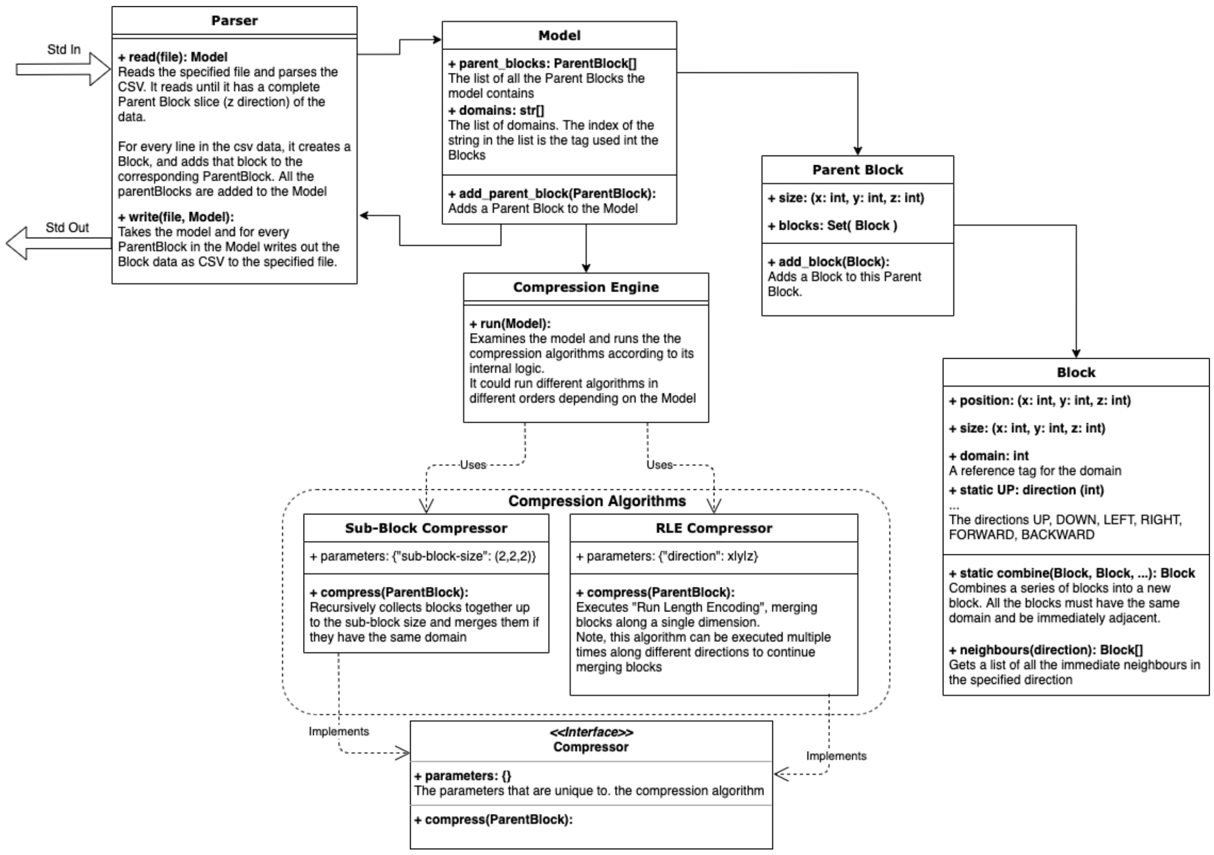
MAPTEK, a company that provides 3D modeling, spatial analysis and design technology to the global mining industry, is our main user. They offer commercial products sold to exploration and production geologists all over the world. It generates huge block model files using machine learning algorithms they run on AWS machines using their input data. Their clients typically then want to download these block model results for local high-performance 3D visualisation in their favourite geology software. By compressing and sending down a coalesced block model (i.e. directly applying the algorithms of this challenge) instead of the raw uncompressed uniform block model that comes out of the machine learning algorithm, they can save their clients a lot of waiting time and bandwidth.

And also there are some potential users in the following:

Geologists: They are interested in predicting and understanding the physical way that the Earth works by feeding in drilling or other sampling data and obtaining domain or grade models. Therefore, the efficient algorithms can help save their time and even the bandwidth if the model processing time can be reduced and the compression rate remains well.

Reservoir Engineers: Using geological models in a more efficient way allows reservoir engineers to identify which recovery options offer the safest and most economic, efficient, and effective development plan for a particular reservoir.

Software Architecture


Parser
+ read(file): Model
Reads the specified file and parses the CSV. It reads until it has a complete Parent Block slice (z direction) of the data.
For every line in the csv data, it creates a Block, and adds that block to the corresponding ParentBlock. All the parentBlocks are added to the Model
+ write(file, Model):
Takes the model and for every ParentBlock in the Model writes out the Block data as CSV to the specified file.
Std In
Std Out
Model
+ parent_blocks: ParentBlock[]
The list of all the Parent Blocks the model contains
+ domains: str[]
The list of domains. The index of the string in the list is the tag used int the Blocks
+ add_parent_block(ParentBlock):
Adds a Parent Block to the Model
Parent Block
+ size: (x: int, y: int, z: int)
+ blocks: Set(Block)
+ add_block(Block):
Adds a Block to this Parent Block.
Compression Engine
+ run(Model):
Examines the model and runs the the compression algorithms according to its internal logic.
It could run different algorithms in different orders depending on the Model
Uses
Uses
Compression Algorithms
Sub-Block Compressor
+ parameters: {"sub-block-size": (2,2,2)}
+ compress(ParentBlock):
Recursively collects blocks together up to the sub-block size and merges them if they have the same domain
RLE Compressor
+ parameters: {"direction": x|y|z}
+ compress(ParentBlock):
Executes "Run Length Encoding", merging blocks along a single dimension.
Note, this algorithm can be executed multiple times along different directions to continue merging blocks
Block
+ position: (x: int, y: int, z: int)
+ size: (x: int, y: int, z: int)
+ domain: int
A reference tag for the domain
+ static UP: direction (int)
...
The directions UP, DOWN, LEFT, RIGHT, FORWARD, BACKWARD
+ static combine(Block, Block, ...): Block
Combines a series of blocks into a new block. All the blocks must have the same domain and be immediately adjacent.
+ neighbours(direction): Block[]
Gets a list of all the immediate neighbours in the specified direction
Implements
Implements
<<Interface>>
Compressor
+ parameters: {}
The parameters that are unique to. the compression algorithm
+ compress(ParentBlock):

This software architecture embraces abstraction and modularity. The concept of a Model holds all the information about the data that needs to be compressed. From there the data is divided into the largest indivisible collection of blocks that can be processed, the Parent Blocks. They in turn contain the individual blocks. The Compression Engine has access to a collection of different modular algorithms, which can be applied differently to different Parent Blocks. The compression algorithms are modular, but all implement that same interface and access the data through the Block abstraction. This allows each to be developed separately but composed together. This will allow a set of building blocks to be developed, which can be applied in different ways through experimentation. This approach also opens the door for parallel processing, both for different ParentBlocks, and potentially different compression algorithms applied to different Parent Blocks.

Decoupling the input and output from the data-model would allow easy changes to how data is fed into the system and produced afterward. Also, visualisation systems could be easily added to the pipeline without changing any of the compression code.

Tech Stack and Standards

We intend to develop this software in Python 3. Most of the software could be counted as “backend”, since it will be used to compress data without direct interaction from a user. We intend for the final product to only depend on standard library classes, reducing dependencies and maximising portability. For the “front-end” a visualisation tool may be used to see how the algorithm operates. This will require external dependencies (e.g. matplotlib). However these visualisations will be used internally by the team to help evaluate the performance of the algorithm.

We may at some point re-implement the system in C. However this is not a priority. A C implementation will likely be faster, but there is a large time/effort overhead with developing in C. A good compression algorithm can be re-implemented to run faster, but a fast algorithm can’t be easily made to compress better. Once we have consistent good compression, we may re-evaluate implementing the system in C. A more likely candidate would be to use multiprocessing in Python to achieve speed increases.

Development Tools:

No IDE was mandated. Any editor the development team felt comfortable with can be used. We believe the team will be more productive with the tools they are already familiar with. This includes VS Code and Atom editors. Both of these have plugins to assist in Python Development.

Python Virtual Environments (venv module) will be used to organise all development dependencies. This is built into the Python standard library and easily works across all operating systems.

The flake8 and mccabe static code analysis packages will be used. These tools check (“lint”) the code for unused variables etc., code complexity, and ensure the code and comments are written to the Python community standard PEP8. These can be integrated nicely into editors and can give interactive feedback.

Code Standard:

All pull requests for code must be submitted on our Github repository, tested and reviewed by 2 other team members (or 1 in the case of documentation/admin).

Static code analysis must pass for the branch to be merged. The Static Analysis ensures the code is written to PEP8 style standards (what all python code should aim for), and for other issues such as unused variables, and cyclomatic complexity. PEP8 is especially important for the developers because it mandates a set of conventions for things such as class/function names, spacing, and comment style. This allows all the developers to be able to read every developer’s code in the same way.

Communication Tools:

Slack for general discussions, technical questions, and keeping in contact with the PO. Slack was set up by the PO and keeping all team communication in the same place simplifies communicating, and makes it easier to find previous discussion points. Slack also allows the team to communicate asynchronously, especially since the team is working across multiple time-zones.

Zoom for interactive group calls. This software was already known to work and can be operated by all team members. It also allows screen sharing to allow the team to all see the same thing. This is especially important since most of the team are working remotely.

Group Meetings and Team Member Roles:

Our team meeting will be held every Wednesday and Sunday from 8:00 pm to 8:30 pm. The upcoming meeting will be held virtually on Zoom.

The upcoming meeting schedule:

Week	Sprint	Meeting Date/ Due Date	Starting Time/ Due time	Note
2	0	5 Aug (Thursday)	20:30	Team meeting
		8 Aug (Sunday)	20:20	Team meeting
3	1	11 Aug (Wednesday)	20:00	Team meeting
		15 Aug (Sunday)	20:00/ 23:59	Team meeting/ Initial Report, Snapshot 1.1
4		18 Aug (Wednesday)	20:00	Team meeting
		22 Aug (Sunday)	20:00/ 23:59	Team meeting/ Snapshot 1.2
5	2	25 Aug (Wednesday)	20:00	Sprint Retrospective 1 meeting
		29 Aug (Sunday)	20:00/ 23:59	Team meeting/ Snapshot 2.1, Sprint Retrospective 1
6		1 Sep (Wednesday)	20:00	Team meeting
		5 Sep (Sunday)	20:00/ 23:59	Team meeting/ Snapshot 2.2
7	3	8 Sep (Wednesday)	20:00	Sprint Retrospective 2 meeting
		12 Sep(Sunday)	20:00/ 23:59	Team meeting/ Snapshot 3.1, Sprint Retrospective 2
8		15 Sep (Wednesday)	20:00	Team meeting
		19 Sep (Sunday)	20:00/ 23:59	Team meeting/ Snapshot 3.2
Break				To be discussed.
Break				To be discussed.

9	4	6 Oct (Wednesday)	20:00	Sprint Retrospective 3 meeting
		10 Oct (Sunday)	20:00/ 23:59	Team meeting/ Snapshot 4.1, Sprint Retrospective 3
10		13 Oct (Wednesday)	20:00	Team meeting
		17 Oct (Sunday)	20:00/ 23:59	Team meeting/ Snapshot 4.2
11	5	20 Oct (Wednesday)	20:00	Sprint Retrospective 4 meeting
		24 Oct (Sunday)	20:00/ 23:59	Team meeting/ Snapshot 5.1, Sprint Retrospective 4
12		27 Oct (Wednesday)	20:00	Team meeting
		31 Oct (Sunday)	20:00/ 23:59	Team meeting/ Snapshot 5.2
13	-	-	-	Final Report Final Presentation

Four Sprint Retrospective meetings will take place on Wednesday of the first week after one sprint is finished. The meeting dates are 25th Aug (Wednesday), 8th Sep (Wednesday), 6th Oct (Wednesday) and 20th Oct (Wednesday).

The customers have set up an additional feedback channel with every team in Slack, where we can always ask questions about the user story and technical problems we met.

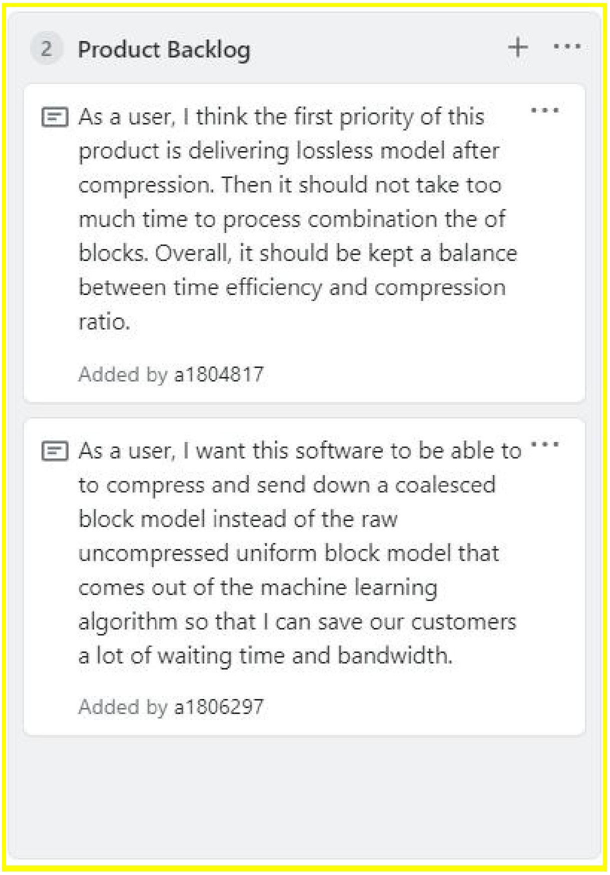
Sprint 1: Po-Yi Lee	a1806297
Sprint 2: Yuanpeng Liu	a1784375
Sprint 3: Karl Asenstorfer	a1162576
Sprint 4: Xiaoman Li	a1804817
Sprint 5: Kaiyang Xue	a1784184

Snapshot Week 2 of Group BLOCKS7PG

(Snapshot 0)

Product Backlog and Task Board:

Product Backlog:



2 Product Backlog

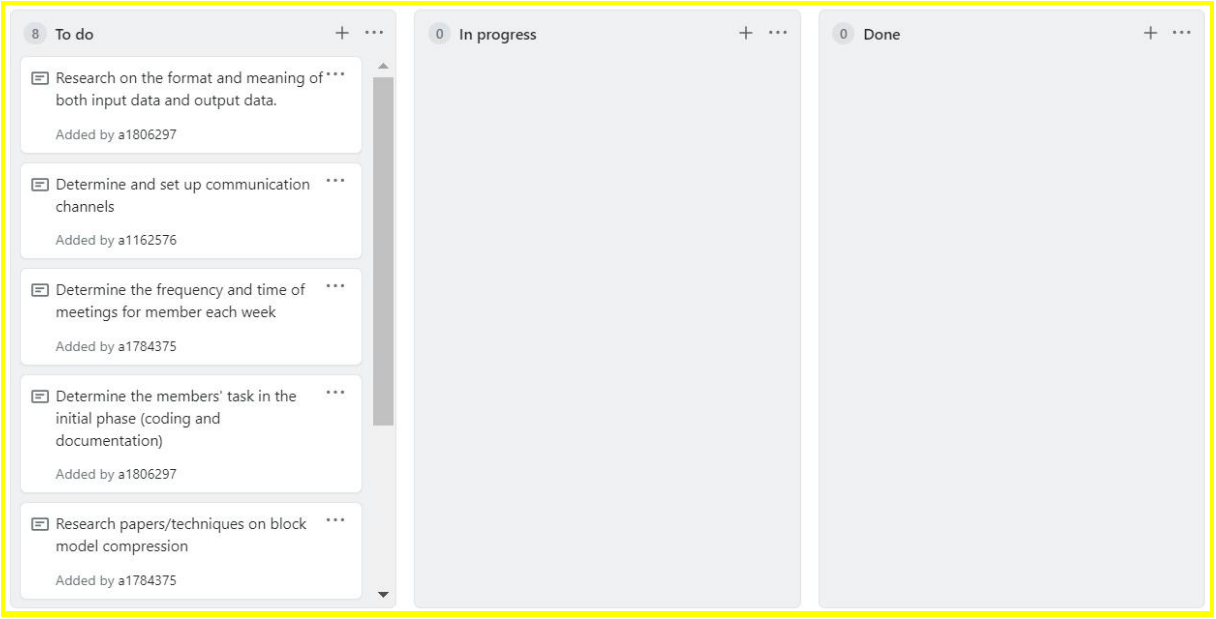
As a user, I think the first priority of this product is delivering lossless model after compression. Then it should not take too much time to process combination the of blocks. Overall, it should be kept a balance between time efficiency and compression ratio.

Added by a1804817

As a user, I want this software to be able to to compress and send down a coalesced block model instead of the raw uncompressed uniform block model that comes out of the machine learning algorithm so that I can save our customers a lot of waiting time and bandwidth.

Added by a1806297

Task Board:



8 To do

Research on the format and meaning of both input data and output data.
Added by a1806297

Determine and set up communication channels
Added by a1162576

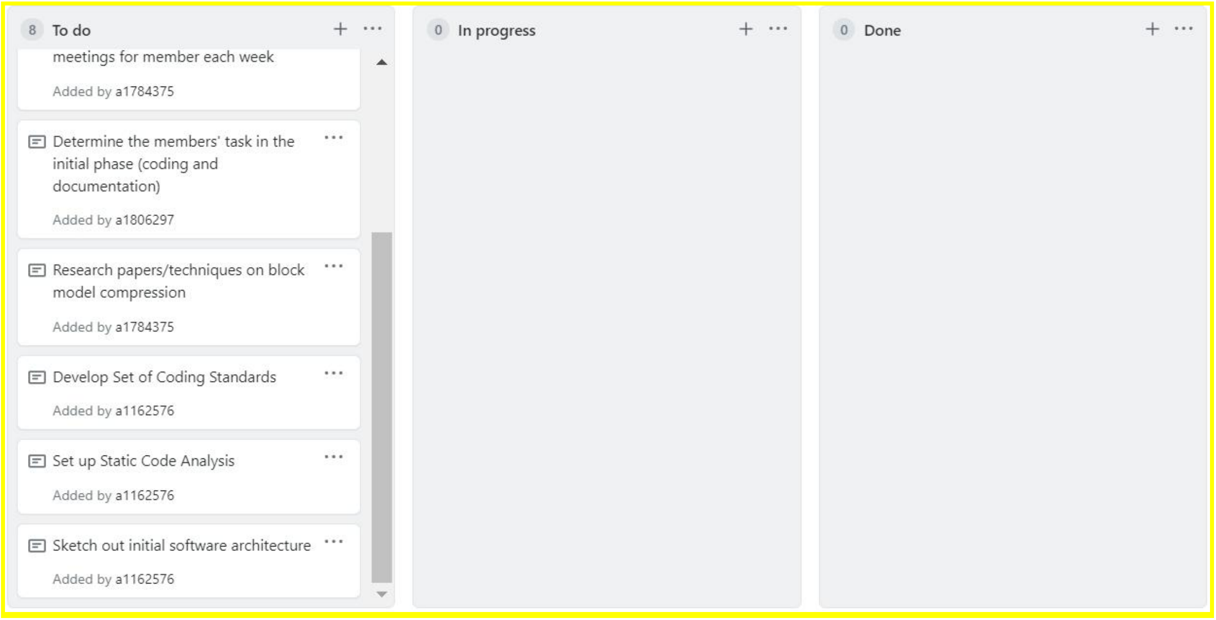
Determine the frequency and time of meetings for member each week
Added by a1784375

Determine the members' task in the initial phase (coding and documentation)
Added by a1806297

Research papers/techniques on block model compression
Added by a1784375

0 In progress

0 Done



8 To do

meetings for member each week
Added by a1784375

Determine the members' task in the initial phase (coding and documentation)
Added by a1806297

Research papers/techniques on block model compression
Added by a1784375

Develop Set of Coding Standards
Added by a1162576

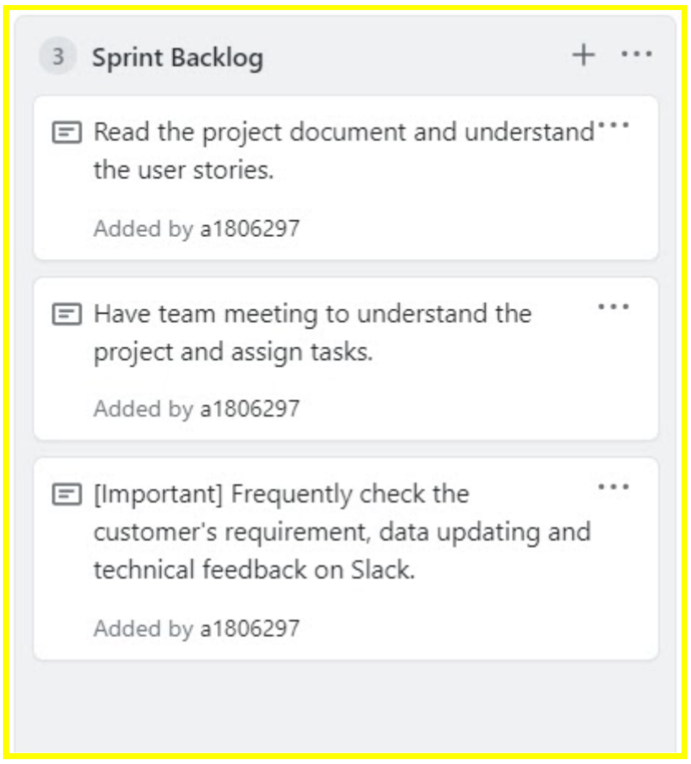
Set up Static Code Analysis
Added by a1162576

Sketch out initial software architecture
Added by a1162576

0 In progress

0 Done

Sprint Backlog and User Stories:



3 Sprint Backlog

Read the project document and understand the user stories.

Added by a1806297

Have team meeting to understand the project and assign tasks.

Added by a1806297

[Important] Frequently check the customer's requirement, data updating and technical feedback on Slack.

Added by a1806297

In the kickoff meeting, the customer shared a project document that described their requirements, including finding the best algorithms for the block model compression and competing with the other teams. To achieve these requirements, we needed to carefully **read the input and output data requirements, the type of uploaded files, the constraints of the project, and so on**.

Before the first sprint, we plan to **have our first meeting to let our team understand the details of the project, the Scrum process and the User Stories**. We also plan to assign tasks to group members to implement the project more efficiently and smoothly.

Moreover, the Scrum Master asks the team to **frequently check the updated information on Slack**, where the customer will update the user stories and any technical feedback. It can help the team design and implement the programs.

Definition of Done:

In the current phase:

- The code we develop is required to take standard input (strings of the form "x, y, z, x_size, y_size, z_size, 'domain'") and produce the result on standard output described in the project documentation.
- Either a .exe file or a Python script must be submitted to a verification service: MAPTEK TITAN.
- According to the user stories, we can submit our code once it improves the compression rate and processing speed, no matter how good they are.
- The datasets of input block models we implement must be comma-separated values (CSV) where each line encodes a block as a string of the form "x, y, z, x_size, y_size, z_size, 'domain'" and the code we develop is required to output a stream of the same format.
- The algorithm we develop must process a block model in slices of no more than parent block thickness at a time, rather than loading the entire input stream into memory first.
- All pull requests of code must be submitted on our Github repository, tested, and reviewed by two other team members (or one in the case of documentation/admin).
- The branches must pass the static code analysis before being merged. The Static Analysis ensures that the codes meet the PEP8 style standards and other issues such as unused variables and cyclomatic complexity.

Summary of Changes:

After receiving the project's requirements and user stories from the customer, we listed the tasks in the "to do" part on Github Projects. The tasks include setting up the communication tools, scheduling our regular weekly meetings, collecting the team member's ideas, deciding the Scrum Master for every sprint, assigning the tasks, etc.

We will develop our team's Definition of Done (DOD) to set up the standard of code we write, ordering and emerging a list of user stories on the Product Backlog to complete the product vision. In the Sprint Backlog, we will set up the goal including reading the project documentation and writing the code from scratch according to the customer's requirement.