**Milestone 2 Scrum Report**

All students are expected to attend the scrum meetings and to participate. Failure to do so will result in greatly reduced grades.

**GROUP**: 6

**Members Present**:

|  |  |
| --- | --- |
| 1. Chia-Yu Chien | 4. Fang Lin |
| 2. Hansol Nam | 5. |
| 3. Wai Bong Yung | 6. |

Milestone 2 Tasks

Some of the software for the project has already been written for you and is available on Blackboard. You must use this in your project and every team should add it to the source code for their repository. Anything in the main function is simply for demonstration purposes and can be replaced. The software you are being given has not been tested and you will need to test it.

You need to study the problem and the code provided for you and then:

* Add any new data structures you will require This will require a thorough analysis of the problem and the existing software. This should be done by creating a new header file in the directory where the rest of the source code has been placed. You do not want to go back and modify it later if you can avoid it as it will slow the project.
* Create a test plan for the project by replacing the text in the supplied test plan template with your test plan.

**Deliverables due 4 days after your lab day:**

* An analysis of the problem (no written artifacts produced).
* A series of data structures created as header files and **stored in the repository**.
* A test plan stored in the repository.
* Completed scrum report including reflection questions answered.

**Rubric**

|  |  |  |
| --- | --- | --- |
| **Individual** | Group participation (includes GitHub commits and Jira usage) | 80% |
| Teamwork | 20% |
| **Group** | Data structures (complete, correct, and well-designed, updated in the project, and added to the repository) | 25% |
| Test plan (complete, well-written) | 25% |
| Git usage (used properly with good structure) | 10% |
| Jira usage (creates issues, tracks progress) | 20% |
| Scrum report & reflections | 20% |
| **Deadline** | 20% deduction for each day you are late |  |

**Scrum Report**

**Summary of Tasks Completed or Delayed in the last week:**

Here you can list all the tasks completed in the last week along with any tasks which could not be completed with a reason why they could not be completed.

|  |  |  |
| --- | --- | --- |
| **Member** | **Tasks Completed** | **Tasks Delayed/Blocked** |
| Chia-Yu Chien | data structure, reflection review edit, test plan review edit | N/A |
| Hansol Nam | Test plan, test strategy, update Jira |  |
| Wai Bong Yung | Test plan test strategy, commit on GitHub |  |
| Fang Lin | data structure, reflection edit, test plan review edit |  |
|  |  |  |
|  |  |  |
|  |  |  |

For every task delayed or blocked, describe the reason for the delay or block, how it impacts the project and the proposed solution or workaround.

|  |  |
| --- | --- |
| **Delayed or Blocked Task** | NA |
| **Reason for delay or block** |  |
| **Impact on Project** |  |
| **Solution or work-around** |  |
|  |  |
| **Delayed or Blocked Task** |  |
| **Reason for delay or block** |  |
| **Impact on Project** |  |
| **Solution or work-around** |  |

**Summary of Meeting:**

A summary of the main points discussed in the meeting and the outcomes of the discussions.

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
| Assigning Work | **Fang initiate reflection and rest of members review it.**  **Data Structure to be done by Chia-Yu, Fang.**  **Test Plan to be done by Wai, Hansol.**  **Crosschecking to be done after.** | **Set deadline for every works.** |
| Meetings Date Arrangement | **Decide the following meetings and the purpose for every meetings** | **One meeting after lecture; One meeting at the Sunday morning.** |
| Setup OneDrive |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Summary of Decisions Made:**

This will include major architecture and design decisions, testing decisions, prioritization of tasks, dealing with problems encountered and other major outcomes from the meeting.

|  |  |
| --- | --- |
| Decision | Rationale |
| Schedule meetings | To keep members updated and following project schedule |
| Assigning work | Assigned work fair amount on each member |
| Choosing method to share work | Uploaded files on one drive to share and edit for all members |
|  |  |
|  |  |
|  |  |
|  |  |

**Tasks Attempted During Meeting:**

Each member is assumed to participate in the scrum meeting and contribute to the completion of the scrum report and reflections. Since the scrum meeting will not take more than 20-30 minutes, there is lots of time left to undertake some of the actual work tasks. In the table below, each member should list what they did to complete the scrum report, the reflections, and 1-4 other tasks they completed during the class period. If a task cannot be completed, the student should indicate why this was not possible.

|  |  |  |  |
| --- | --- | --- | --- |
| Member | Task Attempted | Time Spent | Complete? |
| Fang | **Assigning work** | **1hr** | **Yes** |
| Hansol | **Assigning work, Setup OneDrive** | **1hr** | **Yes** |
| Wai | **Assigning work** | **1hr** | **Yes** |
| Chia-Yu | **Assigning work** | **1hr** | **Yes** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Scrum Tasks Selected for Next Week**:

The tasks each member has selected to pursue for this class or the next week.

|  |  |
| --- | --- |
| Group Member | Task Description |
| Fang | Data Structure, Review Test Plan |
| Hansol | Test Plan, Review Data Structure, test strategy |
| Wai | Test Plan, Review Data Structure, test strategy |
| Chia-Yu | Data Structure, Review Test Plan |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

This is where you should highlight the major accomplishments of the class.

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Setup timeframe for MS2 | **Accelerate work process** |
| Update Jira | **Keep all members updated** |
| Meeting scheduled | **Review works and share their thoughts** |
|  |  |
|  |  |
|  |  |
|  |  |

**Things That Went Well in This Meeting:**

Here you can highlight things which worked well. This indicates that the way you worked on these items is working and should be continued.

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Success |
| Task assignment | **Everyone knows what should do for m2** |
| Meeting arrangement | **The scheduled meeting works for everyone’s schedule.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Things That Did NOT Go Well in This Meeting:**

This is where you can list things which did not go well in the class. You should analyze why this happened and suggest how you can improve it next time. This will lead to the goal of *continuous process improvement*.

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Problem and How to do Better |
| N/A |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflection Questions:**

Answer the following questions using your own words. Make sure that each answer comprises a minimum of 100 words.

* In this milestone you were asked to design the data structure for the project. Print the data structure below then explain each item.

const int WEIGHT\_MAX = 2500;

const int VOLUME\_MAX = 100;

Define the maximum weight capacity for each truck and maximum column limit.

const int validBoxSizes[3] = {1,3,5};

Define the valid sizes of the package can be shipped

const char LEFT\_MOST\_ROW = 'A';

const char RIGHT\_MOST\_ROW = 'Y';

const int TOP\_MOST\_COL = 1;

const int DOWN\_MOST\_COL = 25;

Define the limit constraints of the map

struct Shipment {

int m\_weight;

int m\_boxSize;

struct Point m\_dest;

};

This is the basic information for a shipment: weigth of box, size of box and destination point of shipment.

struct Truck {

int m\_id;

struct Route route;

int m\_weight\_capacity;

int m\_volume\_capacity;

};

This is the basic information of a truck. truck identification number , truck predefined route, and the weight and volume the truck right now have for the box.

/\*\*

\* Function: isValidWeight

\* - Check if the field `m\_weight` of the given shipment is valid.

\*

\* @param shipment - Shipment struct containing the destination coordinates (row and column) of the shipment.

\* @returns - integer, return true if valide, otherwise return false

\*/

int isValidWeight(struct Shipment \*shipement);

Check shipment weight valid.

/\*\*

\* Function: isValidBoxSize

\* - Check if the field `m\_boxSize` of the given shipment is valid.

\*

\* @param shipment - Shipment struct containing the destination coordinates (row and column) of the shipment.

\* @returns - integer, return true if valide, otherwise return false

\*/

int isValidBoxSize(struct Shipment \*shipement);

Check package column valid

/\*\*

\* Function: isValidDest

\* - Check if the field `m\_dest` of the given shipment is valid.

\*

\* @param shipment - Shipment struct containing the destination coordinates (row and column) of the shipment.

\* @returns - integer, return true if valide, otherwise return false

\*/

int isValidDest(struct Shipment \*shipement);

Check the destination of a shipment is valid.

/\*\*

\* Function: limitingFactorWithShipment

\* - Calculate the limiting factor of a truck with a extra shipment

\*

\* @param truck - a truck to be calcualted with a extra shipment

\* @param withShipment - the extra shipment

\* @returns - double, return a limiting factor in percentage

\*/

double limitingFactorWithShipment(struct Truck\* truck, struct Shipment \*withShipment);

Calculate the limiting factor of a truck with an extra shipment.

/\*\*

\* Function: findTruckForShipment

\* - Finds the best truck for a shipment. It considers both the load on the truck,

\* the size and weight of the shipment, and the route of the truck to try to place

\* it on a truck which goes closest to the destination. If there is no truck that

\* can deliver the shipment, it returns -1.

\*

\* @param map - the map of the delivery area with buildings on it.

\* @param trcnks - an array of trucks including the route for each of the trucks.

\* @param numTrucks - the number of trucks in the array of trucks.

\* @param shipment - a data structure containing the size and weight of the shipment.

\* @param diverted - a data structure representing the diverted route to the destination (passed by parameter)

\* @returns - integer, representing the index of the truck in the trucks array on which

\* the shipment should be placed. If no truck can take the shipment, then -1 is returned.

\*/

int findTruckForShipment(

struct Map map,

struct Truck trucks[],

int numTrucks,

struct Shipment shipment,

struct Route \*diverted

);

To find the best option truck for the given shipment. Here are the pseudo-code for FindTruckForShipment:

1. Use `IsTruckCanShip` to check each truck's capacity, and find out which truck is available.

2. foreach available truck

3. iterate all points in the truck's route to the shipment destination point and calculate the Euclidean distance to find the cloest point.

4. compare each truck's cloest pint to the destination's distance to find the shortest one.

5. if the distance is the same, use percentage to find the proper truck.

6. if there is no truck can delivery, return null, save for tmr shipping.

7. Use the cloest point and `shortestPath` to find the deverted route

8. Return deverted route and truck index

/\*\*

\* Function: IsTruckCanShip

\* - Test if a trunck can hold a shipment

\*

\* @param truck - a truck to test if it can hold anther given shipment

\* @param shipment - a shipment to add to the truck

\* @returns - integer, representing the result of checking, -1 indicates the shipment

\* cannot be added to the truck, otherwise it can.

\*/

int isTruckCanShip(struct Truck \*truck, struct Shipment \*shipment);

Check a truck can ship a shipment.

* Describe the process you used to analyze and understand the existing software code.  
    
  1.Firstly go through the entire code for the overall view of what the code does. In this program, we need to decide which truck the package will be placed in.

2. Identify the data structures. in this code, we have the Map, Point, and Route. And we add some data structures to hold the information of packages and trucks.

3. Read all the comments in the codes to have a better understanding. Comment is a good way to go through the codes.

4. Find out the constraints. For example, this map is 25X25 map, if the destination is out of the map, the package is not valid.

5.Algorithms analyze: we have an overall algorithm, just follow the rules. And in the shortest path algorithm, we use the Euclidean to calculate distance from the point in the truck path to the destination point.

6. Testing: do a test plan and implementation the test plan.

* What aspects did you consider when creating the test plan? What were the milestones you identified in the test plan?  
  In software development life cycle, we have to this following stage, planning, analysis, design, implementation, testing and integration, and maintenance. We have some aspects to be considered:

1. create test case for each function.

2. create test case for the integration.

3. create the test plan for the edge limit.

For example, perhaps two trucks have the capacity to deliver one package, and both has the same shortest distance from the destination point. We implement a function to calculate the limiting factor for each truck. To ensure this function works correctly, we can adopt multiple unit test to ensure it can pass the scenarios, for which the chosen truck has the best limiting factor.