User/Tech Story Acceptance Test Planning

Below are the stories that we have chosen to implement this iteration. For each story the points weighting (on a scale of 1-5) are shown. Each story will be tested against an Acceptance Criteria. If the function passes the Acceptance Criteria then the story can be considered Completed.

2pts - [TECH STORY] The maze will be a 2D map made of tiles.
Acceptance Criteria:

- The user interface can display a 5x5 maze with three types of tiles Wall, Goal and Clear so that the robot can move on it.

Acceptance Testing:

- *Test name:* Test 2D map made of tiles Wall, Goal and Clear
- *Goal:* To ensure the 2D map has tiles Wall, Goal and Clear
- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal
    c. Navigate to the application folder using the ‘cd’ command
    d. Type in ‘kivy navibot.py’ and press Enter
- *Procedure:*
    a. Switch to the Maze screen by touching the ‘Maze’ navigation button found at the bottom right of the screen.
- *Expected Outcome:*
    a. The 5x5 2D maze made of tiles is displayed at the Maze screen.
    b. Each of the tiles are uniquely distinguishable by color.
- *Actual Outcome:*

1pts - [TECH STORY] There will be three types of tiles: Wall, Goal, and Clear.
Acceptance Criteria:

- The maze can display three types of tiles which are Wall in red, Goal in blue and Clear in green so that different tiles can perform differently.

Acceptance Testing:

- *Test name:* Test there are three types of tiles Wall, Goal and Clear
- *Goal:* To ensure the maze has three different types of tiles Wall, Goal and Clear shown to the user
- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal
    c. Navigate to the application folder using the ‘cd’ command
    d. Type in ‘kivy navibot.py’ and press Enter
- *Procedure:*

    a. Switch to the Maze screen by touching the ‘Maze’ navigation button found at the bottom right of the screen.
- *Expected Outcome:*
    a. The maze shows:
        - “C” in green as Clear tile,
        - “W” in red as Wall tile, and
        - “G” in blue as Goal tile
- *Actual Outcome:*

1pts - [TECH STORY] The Robot can travel into a Clear tile.
Acceptance Criteria:
- The robot can pass a Clear tile.

Acceptance Testing:
- *Test name:* test the robot can pass a Clear tile
- *Goal:* to make sure the robot can travel through the Clear tile
- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal
    c. Navigate to the application folder using the ‘cd’ command
    d. Type in ‘kivy navibot.py’ and press Enter
- *Procedure:*
    a. Locate the “MOVE” program block in the left side of the screen.
    b. Drag and drop the “MOVE” block to the right side of the screen.
    c. Repeat steps a. -> b. one more time.
    d. Switch to the Maze screen by touching the ‘Maze’ navigation button found at the bottom right of the screen.
- *Expected Outcome:* the robot will travel through two Clear tiles
- *Actual Outcome:*

2pts - [TECH STORY] If the Robot enters the Goal tile, the player wins and the game ends.
Acceptance Criteria:
- When the robot reaches the Goal, the maze disappear, and an win message shows.

Acceptance Testing:
- *Test name:* test when the robot enters the Goal tile, the player wins and the game ends
- *Goal:* to make sure when the robot step onto the Goal tile, the program ends
- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal
    c. Navigate to the application folder using the ‘cd’ command
    d. Type in ‘kivy navibot.py’ and press Enter
- *Procedure:*

a. Locate the “MOVE” program block in the left side of the screen.
b. Drag and drop the “MOVE” block to the right side of the screen.
c. Repeat steps a. -> b. one more time.
d. Locate the “TURN_C” program block in the left side of the screen.
e. Drag and drop the “TURN_C” block to the right side of the screen.
f. Locate the “MOVE” program block in the left side of the screen.
g. Drag and drop the “MOVE” block to the right side of the screen.
h. Repeat steps f. -> g. two more times.
i. Locate the “TURN_A” program block in the left side of the screen.
j. Drag and drop the “TURN_A” block to the right side of the screen.
k. Locate the “MOVE” program block in the left side of the screen.
l. Drag and drop the “MOVE” block to the right side of the screen.
m. Switch to the Maze screen by touching the ‘Maze’ navigation button found at the bottom right of the screen.
n. Touch the “RUN” Button in the top left of the screen.

- *Expected Outcome:*
    a. The robot will travel through all the path and reach the Goal tile.
    b. Then the maze disappear and the win message “YOU WIN!” shows.
- *Actual Outcome:*

1pts - [TECH STORY] The Robot cannot pass through a Wall tile.
Acceptance Criteria:

- The robot cannot step into the Wall tile.

Acceptance Testing:

- *Test name:* test the robot cannot step into a Wall tile
- *Goal:* to make sure the robot cannot travel through a Wall tile
- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal
    c. Navigate to the application folder using the ‘cd’ command
    d. Type in ‘kivy navibot.py’ and press Enter
- *Procedure:*
    a. Locate the “MOVE” program block in the left side of the screen.
    b. Drag and drop the “MOVE” block to the right side of the screen.
    c. Repeat steps a. -> b. two more times.
    d. Switch to the Maze screen by touching the ‘Maze’ navigation button found at the bottom right of the screen.
    e. Touch the “RUN” Button in the top left of the screen.
- *Expected Outcome:*
    a. The robot will travel through the two Clear tiles and stop in front of the Wall tile without stepping onto it
- *Actual Outcome:*

3pts - [USER STORY] As a student, I want to move a language block construct using drag and drop so that I can create programs.
Acceptance Criteria:

- The user interface has two screen: Programming screen and Maze screen.
- At the Programming screen, it is separated into two parts: code block area on the left and programming area on the right.
- The user drag one of the code block on the left to the programming area on the right, and then drop the code block. The code block then returns to the original position, and a copy is added to the programming area.

Acceptance Testing:

- *Test name:* The user creates a simple program
- *Goal:* To ensure that the drag and drop functionality works correctly
- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal
    c. Navigate to the application folder using the ‘cd’ command
    d. Type in ‘kivy navibot.py’ and press Enter
- *Procedure:*
    a. Locate the “MOVE” program block in the left side of the screen.
    b. Drag and drop the “MOVE” block to the right side of the screen.
    c. Locate the “TURN_C” program block in the left side of the screen.
    d. Drag and drop the “TURN_C” block to the right side of the screen.
- *Expected Outcome:*
    a. The user created program is shown on the right half of the current screen
    b. It has a “MOVE” block and a “TURN_C” block ordered from top to bottom
    c. The original “MOVE” and “TURN_C” blocks are in their original positions
- *Actual Outcome:*

3pts - [USER STORY] As a student, I want to run the program so that I can see how the Robot will act based on my current program.
Acceptance Criteria:

- When the program starts, the Robot starts at 0,0 (top-left maze corner) and is facing East
- After drag and drop to create the program, switch to the Maze screen, and click Run button on the left. The program is created and the user can see the robot’s movements and its current direction.

Acceptance Testing:

- *Test name:* User creates and runs a simple program
- *Goal:* To ensure that the program executes in order and the robot’s movements are updated correctly
- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal

    c. Navigate to the application folder using the ‘cd’ command
    d. Type in ‘kivy navibot.py’ and press Enter
- *Procedure:*
    a. Locate the “MOVE” program block in the left side of the screen.
    b. Drag and drop the “MOVE” block to the right side of the screen.
    c. Locate the “TURN_C” program block in the left side of the screen.
    d. Drag and drop the “TURN_C” block to the right side of the screen.
    e. Switch to the Maze screen by touching the ‘Maze’ navigation button found at the bottom right of the screen.
    f. Touch the “RUN” Button in the top left of the screen.
- *Expected Outcome:*
    a. The robot has moved forward one tile, and is now facing ‘S’
- *Actual Outcome:*

2pts - [TECH STORY] The programming language can detect distance between the Robot and the nearest Wall in front of it.

Acceptance Criteria:
- When the program starts, the Robot starts at 0,0 (top-left maze corner) and is facing East
- The program will return an integer and show this message to tell the user the distance between the robot and the nearest Wall the robot is facing currently.

Acceptance Testing:
- *Test name:* Detect distance from starting location
- *Goal:* To ensure that the programming language can correctly determine the distance between the robot and the wall in front of it
- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal
    c. Navigate to the application folder using the ‘cd’ command
    d. Type in ‘kivy navibot.py’ and press Enter
- *Procedure:*
    a. Locate the “DETECT” program block in the left side of the screen.
    b. Drag and drop the “DETECT” block to the right side of the screen.
    c. Switch to the Maze screen by touching the ‘Maze’ navigation button found at the bottom right of the screen.
    d. Touch the “RUN” Button in the top left of the screen.
- *Expected Outcome:*
    a. The “Distance” Label in the bottom left of the screen displays ‘2’
- *Actual Outcome:*

Acceptance Testing:
- *Test name:* Detect distance when facing a wall
- *Goal:* To ensure that the programming language can correctly determine the distance between the robot and the wall in front of it

- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal
    c. Navigate to the application folder using the ‘cd’ command
    d. Type in ‘kivy navibot.py’ and press Enter
- *Procedure:*
    a. Locate the “MOVE” program block in the left side of the screen.
    b. Drag and drop the “MOVE” block to the right side of the screen.
    c. Repeat steps a. -> b. once more.
    d. Locate the “DETECT” program block in the left side of the screen.
    e. Drag and drop the “DETECT” block to the right side of the screen.
    f. Switch to the Maze screen by touching the ‘Maze’ navigation button found at the bottom right of the screen.
    g. Touch the “RUN” Button in the top left of the screen.
- *Expected Outcome:*
    a. The “Distance” Label in the bottom left of the screen displays ‘0’
- *Actual Outcome:*

1pts - [TECH STORY] The programming language can make the Robot move one step forward.
Acceptance Criteria:
- When the program starts, the Robot starts at 0,0 (top-left maze corner) and is facing East
- Given the Robot is facing a tile it can move in to (Clear or Goal), then when commanded to move forward the robot will move to the tile it is facing.

Acceptance Testing:
- *Test name:* The robot is commanded to move forward
- *Goal:* To ensure that when commanded to, the robot will move forward one tile (given that the tile is a valid tile)
- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal
    c. Navigate to the application folder using the ‘cd’ command
    d. Type in ‘kivy navibot.py’ and press Enter
- *Procedure:*
    a. Locate the “MOVE” program block in the left side of the screen.
    b. Drag and drop the “MOVE” block to the right side of the screen.
    c. Switch to the Maze screen by touching the ‘Maze’ navigation button found at the bottom right of the screen.
    d. Touch the “RUN” Button in the top left of the screen.
- *Expected Outcome:*
    a. The robot will move forward one tile
- *Actual Outcome:*

2pts - [TECH STORY] The programming language can make the Robot turn 90 degrees clockwise a specific number of times.

Acceptance Criteria:

- IMPORTANT: The "specific number of times" has not been implemented yet. For the moment, a single TURN_C code block turns the robot once.
- When the program starts, the initial direction of the robot is East.
- Each time run TURN_C code block, the robot's direction will change by the sequence north, east, south and west.

Acceptance Testing:

- *Test name:* Test that the sequence for clockwise
- *Goal:* To ensure that the sequence for clockwise turning is correct
- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal
    c. Navigate to the application folder using the 'cd' command
    d. Type in 'kivy navibot.py' and press Enter
- *Procedure:*
    a. Locate the "TURN_C" program block in the left side of the screen.
    b. Drag and drop the "TURN_C" block to the right side of the screen.
    c. Switch to the Maze screen by touching the 'Maze' navigation button found at the bottom right of the screen.
    d. Locate the "Direction" Label along the left side of the screen.
    e. Record the initial Direction value.
    f. Touch the "RUN" Button in the top left of the screen.
    g. Record the new Direction value.
    h. Repeat steps f. -> g. three more times.
- *Expected Outcome:*
    a. The initial Direction value is 'E'
    b. The three subsequent values are 'S', 'W', and 'N' in this order
    c. The final direction value is 'E'
- *Actual Outcome:*

2pts - [TECH STORY] The programming language can make the Robot turn 90 degrees anti-clockwise a specific number of times.

Acceptance Criteria:

- IMPORTANT: The "specific number of times" has not been implemented yet. For the moment, a single TURN_A code block turns the robot once.
- When the program starts, the initial direction of the robot is East.
- Each time run TURN_A code block, the robot's direction will change by the sequence north, west, south and east.

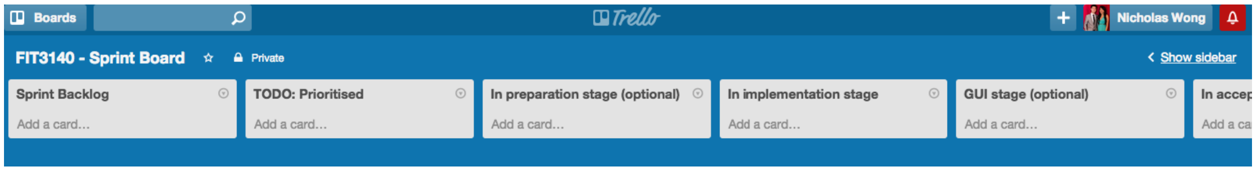
Acceptance Testing:

- *Test name:* Test that the sequence for anti-clockwise

- *Goal:* To ensure that the sequence for anti-clockwise turning is correct
- *Setup:*
    a. Ensure Kivy is installed (see kivy.org for instructions)
    b. Open a Terminal
    c. Navigate to the application folder using the ‘cd’ command
    d. Type in ‘kivy navibot.py’ and press Enter
- *Procedure:*
    a. Locate the “TURN_A” program block in the left side of the screen.
    b. Drag and drop the “TURN_A” block to the right side of the screen.
    c. Switch to the Maze screen by touching the ‘Maze’ navigation button found at the bottom right of the screen.
    d. Locate the “Direction” Label along the left side of the screen.
    e. Record the initial Direction value.
    f. Touch the “RUN” Button in the top left of the screen.
    g. Record the new Direction value.
    h. Repeat steps f. -> g. three more times.
- *Expected Outcome:*
    a. The initial Direction value is ‘E’
    b. The three subsequent values are ‘N’, ‘W’, and ‘S’ in this order
    c. The final direction value is ‘E’
- *Actual Outcome:*

Backlog Status at End of Sprint

All stories were implemented (part 1)



Boards

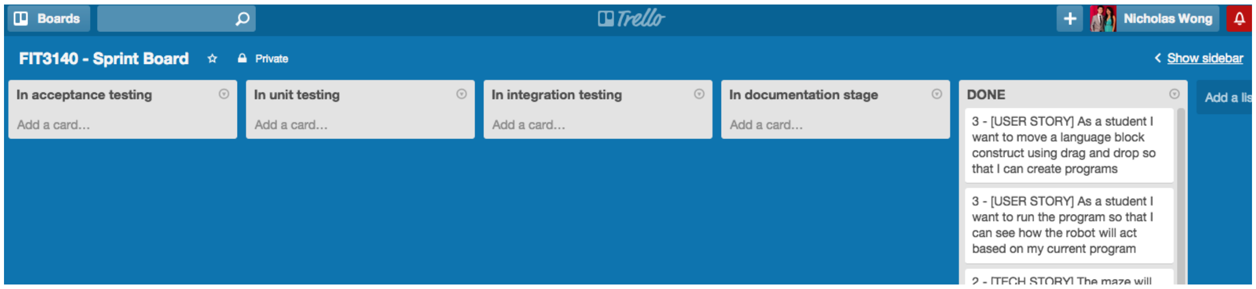
Nicholas Wong

FIT3140 - Sprint Board ☆ Private

Show sidebar

- Sprint Backlog — Add a card…
- TODO: Prioritised — Add a card…
- In preparation stage (optional) — Add a card…
- In implementation stage — Add a card…
- GUI stage (optional) — Add a card…
- In accep — Add a ca

All stories were implemented (part 2)



Boards

Nicholas Wong

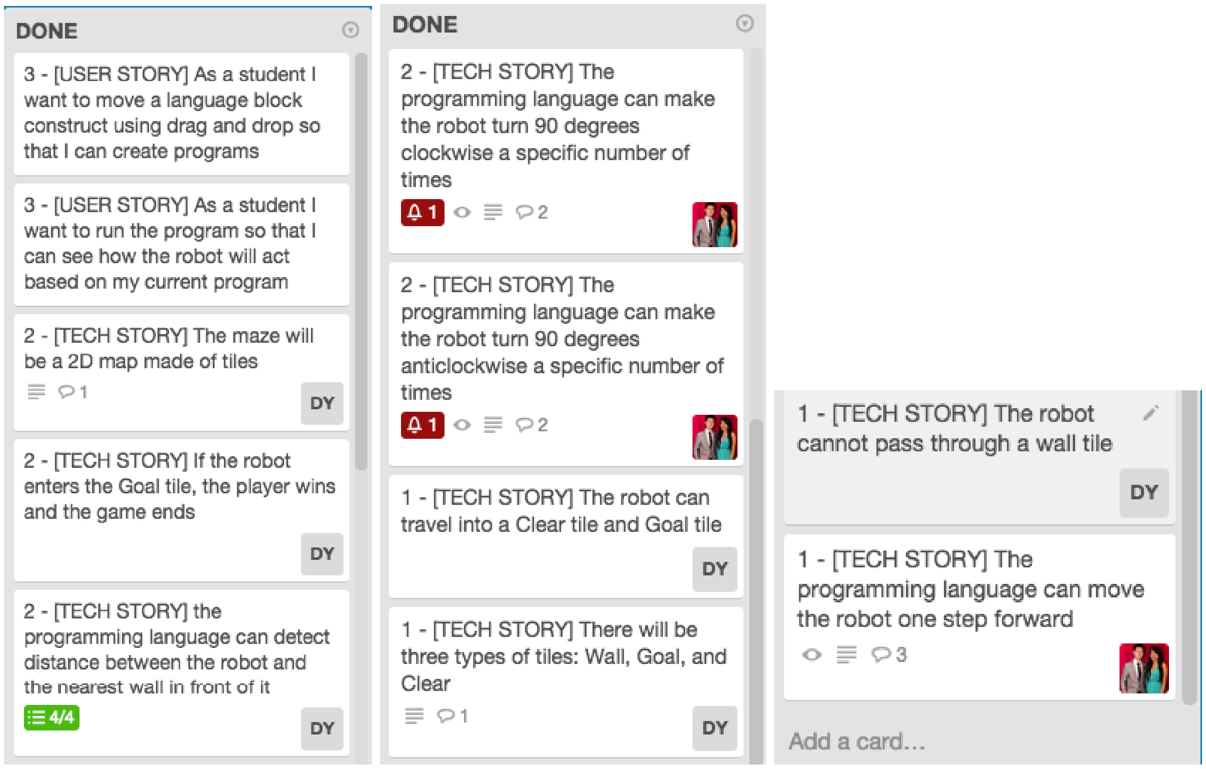
FIT3140 - Sprint Board ☆ Private

Show sidebar

- In acceptance testing — Add a card…
- In unit testing — Add a card…
- In integration testing — Add a card…
- In documentation stage — Add a card…
- DONE
  - 3 - [USER STORY] As a student I want to move a language block construct using drag and drop so that I can create programs
  - 3 - [USER STORY] As a student I want to run the program so that I can see how the robot will act based on my current program
  - 2 - [TECH STORY] The maze will

Add a lis

The "DONE" stories



DONE

- 3 - [USER STORY] As a student I want to move a language block construct using drag and drop so that I can create programs
- 3 - [USER STORY] As a student I want to run the program so that I can see how the robot will act based on my current program
- 2 - [TECH STORY] The maze will be a 2D map made of tiles — 1 — DY
- 2 - [TECH STORY] If the robot enters the Goal tile, the player wins and the game ends — DY
- 2 - [TECH STORY] the programming language can detect distance between the robot and the nearest wall in front of it — 4/4 — DY

DONE

- 2 - [TECH STORY] The programming language can make the robot turn 90 degrees clockwise a specific number of times — 1 — 2
- 2 - [TECH STORY] The programming language can make the robot turn 90 degrees anticlockwise a specific number of times — 1 — 2
- 1 - [TECH STORY] The robot can travel into a Clear tile and Goal tile — DY
- 1 - [TECH STORY] There will be three types of tiles: Wall, Goal, and Clear — 1 — DY

- 1 - [TECH STORY] The robot cannot pass through a wall tile — DY
- 1 - [TECH STORY] The programming language can move the robot one step forward — 3

Add a card…