**Task 4: DRAW SHAPE REPORT**

# a) COVER PAGE

CONTENTS:

[a) **COVER PAGE**](#_a)_COVER_PAGE)

[b) **REQUIREMENTS SPECIFICATION**](#_b)_REQUIREMENTS_SPECIFICATION)

[c) **ALGORITHM DESIGN**](#_c)_ALGORITHM_DESIGN)

[d) **USER INTERFACE DESIGN**](#_d)_USER_INTERFACE)

[e) **PLANNING AND MONITORING PROGRESS**](#_e)_PLANNING_AND)

# b) REQUIREMENTS SPECIFICATION

FUNCTIONAL REQUIREMENTS

1. The system must allow users to scan a QR code containing shape information.
2. The system must parse the QR code to extract the shape type and dimensions.
3. The system must validate the input to ensure:

* Shape type is either a square (S) or a triangle (T).
* Side lengths are between 15 cm and 85 cm.
* Triangle inputs follow the Triangle Inequality Theorem.

1. The system must display an error message if the input is invalid and prompt the user to re-enter a valid QR code.
2. The system must calculate the movement time required for SwiftBot to draw the shape based on predefined speed calibration data.
3. The system must instruct the SwiftBot to move and turn correctly to draw the specified shape.
4. The system must provide real-time feedback on progress (e.g., percentage completion bar).
5. The system must indicate completion by blinking SwiftBot’s green lights.
6. The system must log the drawn shape details, including:

* Shape type and dimensions
* Time taken to complete the drawing
* Largest shape drawn (by area)
* Most frequently drawn shape
* Average drawing time

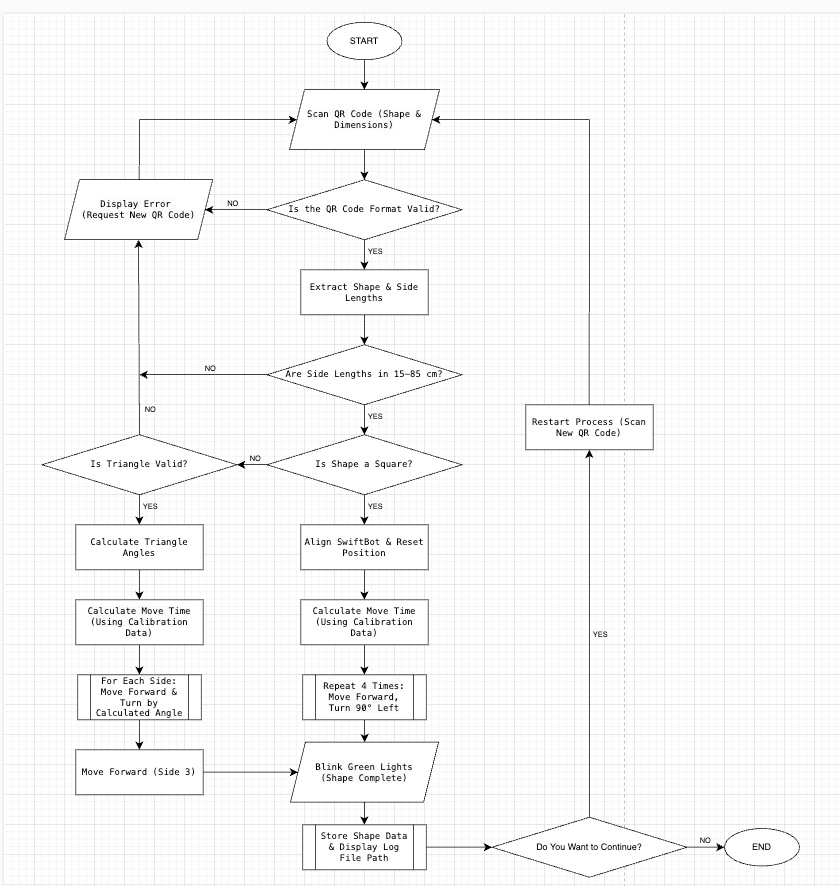
1. The system must prompt the user to either scan a new QR code or exit the program.

NON-FUNCTIONAL REQUIREMENTS

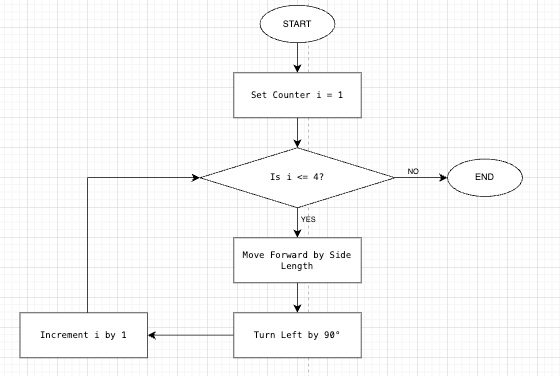
1. **Usability:** The command-line interface must be intuitive and provide clear instructions to the user.
2. **Performance:** The system must process the QR code and begin drawing within 3 seconds of input validation.
3. **Reliability:** The system must correctly interpret input at least 99% of the time.
4. **Portability:** The program must run on different operating systems (Windows, macOS, Linux) that support Java.
5. **Availability:** The program must provide a continuous operation mode until the user chooses to exit.
6. **Maintainability:** The system codebase must be modular, making it easy to add support for additional shapes in the future.
7. **Scalability:** The system must support multiple shapes per QR code input (up to 5 shapes in sequence).
8. **Security:** The system must prevent invalid or malicious QR code inputs from crashing the application.

# c) ALGORITHM DESIGN

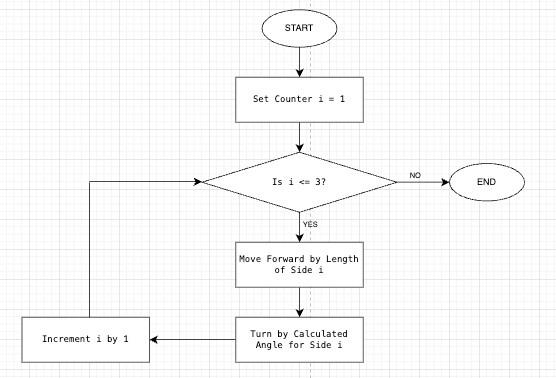
Flowchart: Drawing A Shape (Square & Triangle)



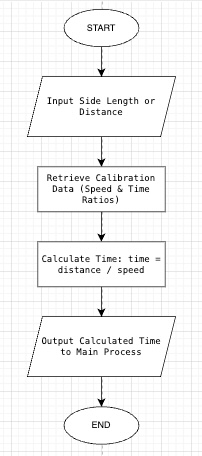
## Flowchart: Repeat 4 Times (Square Drawing)



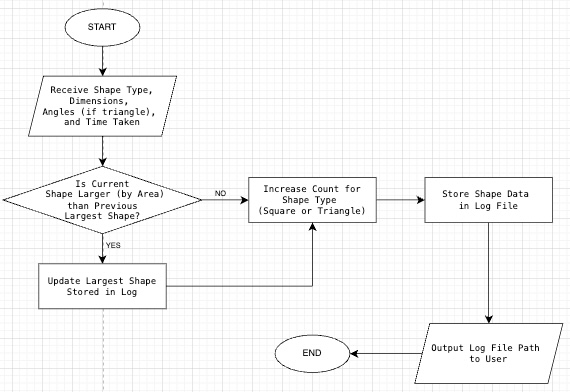
## Flowchart: For Each Side (Triangle Drawing)



## Flowchart: Calculate Move Time (Using Calibration Data)

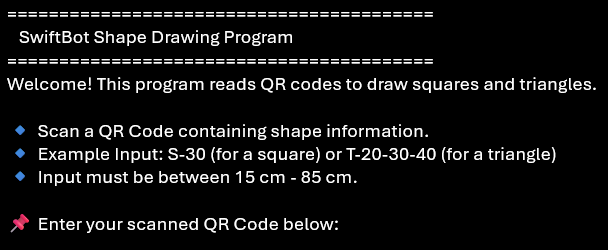


## Flowchart: Store Shape Data & Display Log File



# d) USER INTERFACE DESIGN

1. WELCOME SCREEN



This screen welcomes the user and provides step-by-step instructions on how to scan a QR code to input shape details.

--------------------------------------------------------------------------------------------------------------

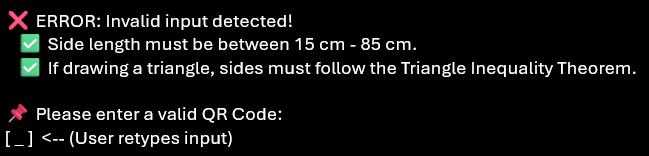
2. QR CODE INPUT PROMPT



This screen allows the user to enter the scanned QR code that contains shape and size information.

--------------------------------------------------------------------------------------------------------------

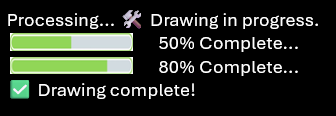
3. VALIDATION ERROR SCREEN



If the input is invalid, this screen alerts the user and explains the issue, ensuring they enter correct values.

--------------------------------------------------------------------------------------------------------------

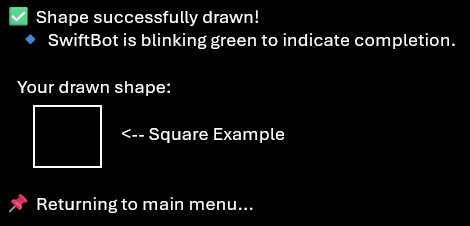
4. PROCESSING SCREEN



This screen informs the user that the SwiftBot is actively drawing the shape and provides a visual progress indicator.

--------------------------------------------------------------------------------------------------------------

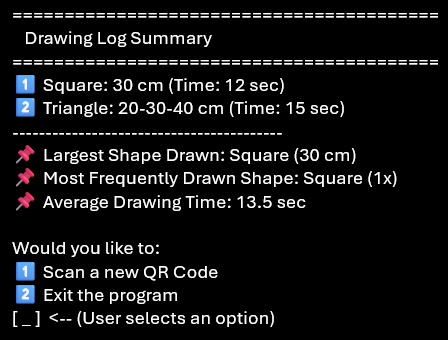
5. DRAWING COMPLETION SCREEN



This screen confirms that the drawing is complete and provides an ASCII representation of the drawn shape.

--------------------------------------------------------------------------------------------------------------

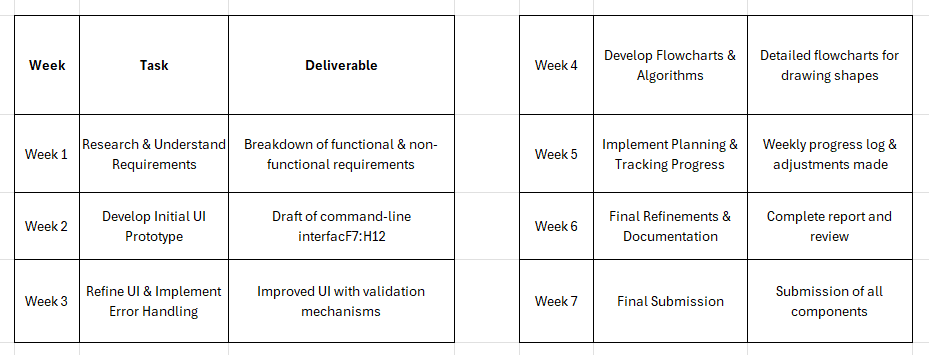
6. LOG SUMMARY & EXIT PROMPT



This screen provides a summary of the shapes drawn, logs their details, and allows the user to choose whether to scan another QR code or exit.

# e) PLANNING AND MONITORING PROGRESS

PLAN & TASK BREAKDOWN



## Challenges Faced & Solutions Implemented

* **Understanding Requirements**: Initially, breaking down the task was challenging due to complexity. **Solution:** I referred to lecture notes and discussed with my tutor to clarify expectations.
* Flowchart **Complexity**: Developing **detailed flowcharts** while ensuring clarity was difficult. **Solution:** I incrementally refined them based on feedback from tutor meetings.
* **UI Prototype Adjustments**: Ensuring the command-line UI was user-friendly required revisions. **Solution:** Iterative testing and feedback loops helped refine the design.
* **Time Management**: Juggling this assessment with other coursework was challenging. **Solution:** I followed my Gantt chart strictly and allocated buffer time for unexpected delays.

Adjustments to the Initial Plan

* **Expanded UI Prototype Development**: Initially planned for **one week but** took **two weeks** due to iterative improvements based on tutor feedback.
* **More Detailed Flowcharts**: Due to tutor suggestions, **additional refinements** were made to ensure logical flow in algorithms.
* **Improved Documentation**: Initially focused on technical implementation but later expanded documentation for better clarity and completeness.