RMIT UNIVERSITY

Further Programming
COSC2391
Assignment 2: Graphic Design Application

Assessment Type	**Individual assignment; no group work.** Submit online via Canvas → Assignments → Assignment 2. Marks are awarded for meeting requirements as closely as possible according to assignment specifications and the supplied rubric. Clarifications or updates may be made via announcements.
Due Date	**Week 13, Friday 3rd June 11:59pm.** Late submission penalty will be applied unless special consideration has been granted. **There will be 3 milestones: week 8 in-lab milestone check, week 11 in-lab milestone check and week 14 post submission final interview.** You will describe the key concepts adopted in your code, demonstrate code execution, and explain your code in the milestones.
Marks	50 marks out of 100 for assignment 2. 3 milestones (5 marks for week 8 in-lab milestone checking, 5 marks for week 11 in-lab milestone checking, and 6 marks for week 14 post submission final interview) will add up to 16 marks in addition to the 50 marks.

1. Overview

NOTE: Carefully read this document. In addition, regularly follow the Canvas assignment discussion board for assignment related clarifications and discussion.

In this assignment, you are required to work individually to develop a Graphic Design Application, named Smart Canvas, to create posters, cards, and other visual content. You will practice your knowledge of object-oriented design principles, design patterns, Java Collections Framework, generics, exceptions, graphical user interfaces, serialization and deserialization, and unit testing. You will use Java SE 8 or later and JavaFX.

Smart Canvas is a desktop-based application for creating graphic design. It has a similar concept to Canva, a web-based graphic design platform to enable users to create social media graphics, presentations, posters, documents, and other visual content. You can have a look at Canva website (https://www.canva.com/en_au/) and choose “play with Canva” option to see how a Canva looks like.

Task specification

Basic functional requirements are listed below (mandatory).

- The application can have many users.
- Each user can create a profile, with a unique username, password, the first name and the last name, and the profile photo. If there is no photo, a default profile photo is provided by the application.
- Once the username and password are created, the user can log in.
- Each user has an empty board after login. The board should display user’s profile photo, first name, and last name.
- Each user can perform following actions:
    - Edit the profile (i.e., change the first name or the last name, select a new profile photo).
    - Create a new canvas with dimensions.
    - Zoom in/out the canvas.
    - Change the background color of the canvas.
    - Clear the canvas.
    - Export the canvas as an image.

  - Add an element to the canvas. Elements include text, image, circle, and rectangle.
  - Move an element.
  - Rotate an element and show the angle for rotation when rotating.
  - Resize an element and show element size when resizing.
  - Delete an element.
  - For a text element, change the text content, font size, font type (e.g., Calibri), text color, font style (i.e., bold, italic), text alignment (left, centre, and right), and border color, border width and background color of the text box.
  - For an image element, change another image.
  - For a circle/rectangle element, change border color, border width and shape color.
  - Log out (and go back to login page).
  - Check the version of the application (i.e., About page).
- The profile data should persist between logins for the same user. Note a password should be hashed and stored as the hash value in the database rather than a plaintext. You do not need to store the canvas.

Advanced functional requirements are listed below (optional).

- The application provides a premium membership (assume the premium user needs to pay additional fee to unlock the premium feature). A premium user has access to 3 templates to choose from. A regular user can subscribe as a premium user to unlock the premium feature and a premium user can unsubscribe from the premium membership at any time.

Please check the Appendix A for a sample interaction with the Smart Canvas.

Disclaimer: *the specification in this assignment is intended to represent a simplified version of a system in real life and thus is not meant to be a 100% accurate simulation of any real system or service that is being used commercially.*

2. Assessment criteria

This assessment will determine your ability to implement Object-Oriented Java code according to the specifications shown below. In addition to functional correctness (i.e., getting your code to work) you will also be assessed on code quality. Specifically:

- You are required to demonstrate your understanding of object-oriented programming principles, including encapsulation, abstraction, inheritance, and polymorphism.
- Your object-oriented design should provide high cohesion and low coupling following well established design principles and patterns.
- You are required to use well-tested Java collections.
- You are required to validate all user inputs and catch and handle all exceptions.
- You are required to demonstrate that you have done adequate unit testing using the JUnit framework.
- Your interaction design must include appropriate JavaFX elements to make the system easy-to-navigate and consistent and respond clearly to every user action.
- Your application needs to have a persistent state that must be stored in a file or a database.
- You are required to write properly documented code.
- You should analyse your design choice in building this desktop-based application in a separate one-page report.

3. Learning Outcomes

This assessment is relevant to the following Learning Outcomes:

- CLO1: explain the purpose of OO design and apply the following OO concepts in Java code: inheritance, polymorphism, abstract classes, interfaces and generics.
- CLO2: describe and document diagrammatically the OO design of the Java Collection Framework (JCF) and apply this framework in Java code.

- CLO3: describe and document diagrammatically the OO design of the JavaFX APIs and apply these APIs to create graphical user interface (GUI) code.
- CLO4: demonstrate proficiency using an integrated development environment such as Eclipse for project management, coding and debugging.
- CLO5: describe and document diagrammatically common OO design patterns such as Model View Controller (MVC), Singleton, Facade and apply in Java code.

4. Assessment details

In this assignment, you will incrementally build a graphic design application, named Smart Canvas. The assignment is structured into 3 milestones which must be demonstrated to your lab tutor (adding up to 16 marks in addition to the 50 marks for the assignment).

Part A (mandatory)

You are required to work on the backend part of the application (excluding data storage, which will be covered in Part C) regarding the basic functions mentioned in the task specification in Section 1 - Overview. You can start this part from week 6 and work on it until the due date.

- You will incorporate object-oriented concepts to design and implement classes (including abstraction, encapsulation, inheritance, and polymorphism).
- You will need to choose appropriate data structures to store relevant information (e.g., a list of users). You MUST use Java Collections Framework.
- You are required to write unit tests for all classes that include functions in the backend part (excluding data storage) using JUnit framework. Your unit tests should have a good coverage of the typical use cases of your classes and test all reasonable corner cases. You do not need to write unit tests for getters and setters.
- You are required to catch and handle all exceptions in your program. Avoid throwing an unspecific Exception; throw a specific exception. You can create custom exception classes if necessary.
- You will need to document the code properly by following Code Conventions for the Java Programming Language by Oracle: https://www.oracle.com/java/technologies/javase/codeconventions-introduction.html
- You will adhere to SOLID design principles (see materials in week 6).

Part B (mandatory)

You are required to work on the frontend part of the application and connect the frontend with the backend regarding the basic functions mentioned in the task specification in Section 1 - Overview. You can start this part from week 7 and work on it until the due date.

- You MUST use JavaFX.
- Your program should include appropriate JavaFX components to make the application easy-to-navigate. For example, each window should have a window title; menu options are selected from a menu bar.
- Your program should respond clearly to every user action. For example, if the user types a wrong password, the login window should display a clear error message and ask the user to type again.
- Your program should provide a visual consistency. Avoid using different styles and labels for similar elements on different windows of the application.
- You are required to follow MVC design pattern to connect the frontend with the backend (see materials in week 8).
- You can use other design patterns such as Singleton if necessary.

Part C (mandatory)

You can start this part from week 9 and work on it until the due date.

- Your program should save the user profile data and allow the application to be restarted in the same state it was in at the end of the previous execution. You must store the data in a database (using JDBC) when the application closes and restore the data when the application starts again. Note the application may crash during the execution (make sure you save the data whenever you change any user profile data).
- You are required to write a one-page document (minimum 300 words; maximum 500 words) to analyse and justify your design choice in building this desktop-based application. Your analysis and justification will need to cover the following questions:
    - How did you apply MVC design pattern to build this application?

- How does your code adhere to SOLID design principles?
- What other design patterns does your code follow? Why did you choose these design patterns?

Part D (optional)

You may optionally implement the advanced functions mentioned in the task specification in Section 1 - Overview. The completion of part D will provide bonus marks (i.e., 5 marks). You can start this part from week 7 if you are planning to obtain some bonus marks. Templates can be hard coded and stored in memory.

5. Submission

You will submit all the relevant material as listed below via Canvas.

- You must submit a zip file of your project.
- You can submit your assignment as many times as you would like prior to the due date. The latest submission will be graded.
- You must not submit compiled files (*.class files) and any IDE related files (e.g., .settings folder generated by Eclipse). Please check your submission carefully, make sure all java files have been included.

After the due date, you will have 5 business days to submit your assignment as a late submission. Late submissions will incur a penalty of 10% of the full assignment 2 marks per day. After these five days, Canvas will be closed, and you will lose ALL assignment 2 marks.

6. Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods,
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity

7. Marking Guidelines

- GUI (10/50)
- Functionality (20/50)
- Unit testing (2/50)
- Exception handling (3/50)
- Object oriented design and choice of data structures (5/50)
- Source code quality (5/15)
- One-page report for justification of design choice (5/50)
- Milestone checks (16/16)

GUI: The application is easy-to-navigate and consistent and responds clearly to every user action. **Note marks for GUI are checked during the post-submission interview. If you do not attend the final interview, you will be awarded ZERO mark for GUI.**

Functionality: All basic functions required by specification are implemented correctly. **Note marks for Functionality are checked during the post-submission interview. If you do not attend the final interview, you will be awarded ZERO mark for Functionality.**

Unit testing: Unit tests have a good coverage of the typical use cases of classes and testing all reasonable corner cases.

Exception handling: All exceptions caught and handled; using specific exception classes; create at least 1 custom exception class.

Object oriented design and choice of data structures: Appropriate choice of data structures and class designs.

Source code quality: Adequately documented and properly indented code; appropriate class/method/variable names.

Justification of design choice: Analyse and justify your design choice in building this desktop-based application by covering the questions in Part C of the assignment specification.

Milestone checks:

- In-lab milestone check in week 8 (5 marks): Clearly explain the class design; complete a prototype of the backend part of the application (excluding data storage); complete the GUI design of the login and sign-up (see Figures 1-2 in Appendix).
- In-lab milestone check in week 11 (5 marks): Clearly explain how the code follows MVC; complete required functions and demo via code execution, including user login and sign-up, change user profile, create a new canvas, add elements (text, image, circle, and rectangle), and clear the canvas.
- Post submission interview in week 14 (6 marks): Clearly perform the demo of all functions implemented (note GUI and functionality are marked during the demo); clearly explain the code and answer all questions.

8. Appendix A

The following screenshots demonstrate a sample interaction with the Smart Canvas. You do not have to follow this precisely and are encouraged to come up with your own designs. These illustrate the required functionality.

Figure 1. Login window.

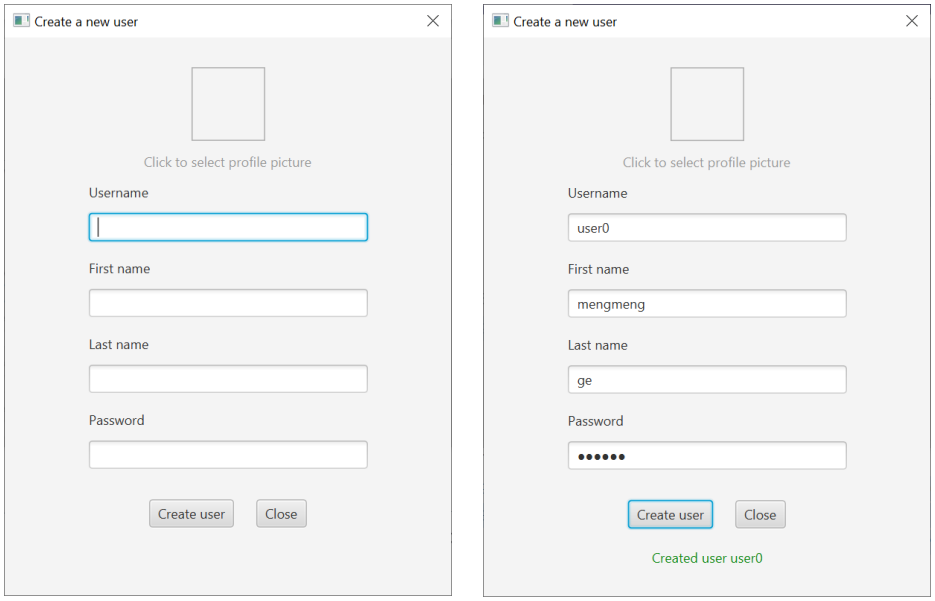

Figure 2. a) Profile creation window with empty information. b) Profile creation window after the user fills in the information and clicks “Create user” button.

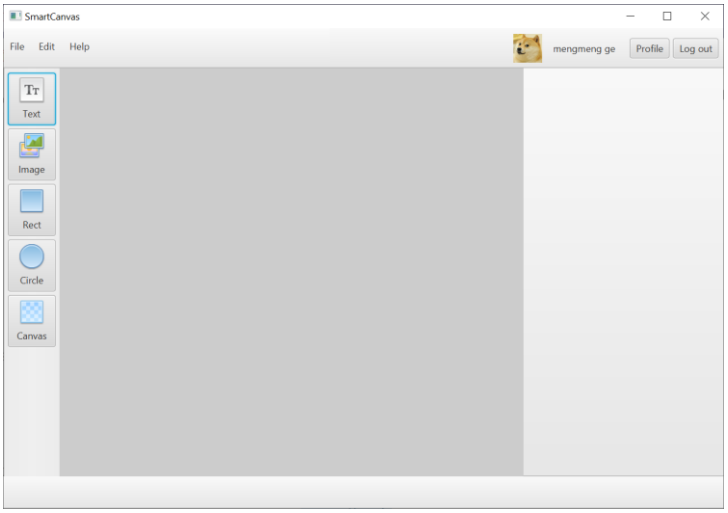

Figure 3. Empty board after the user logs in.

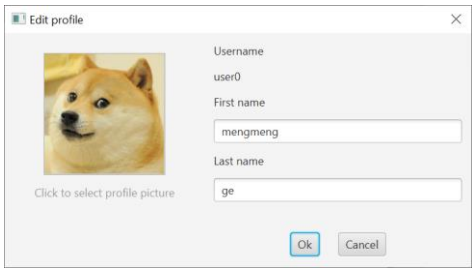

Figure 4. Profile editing window.

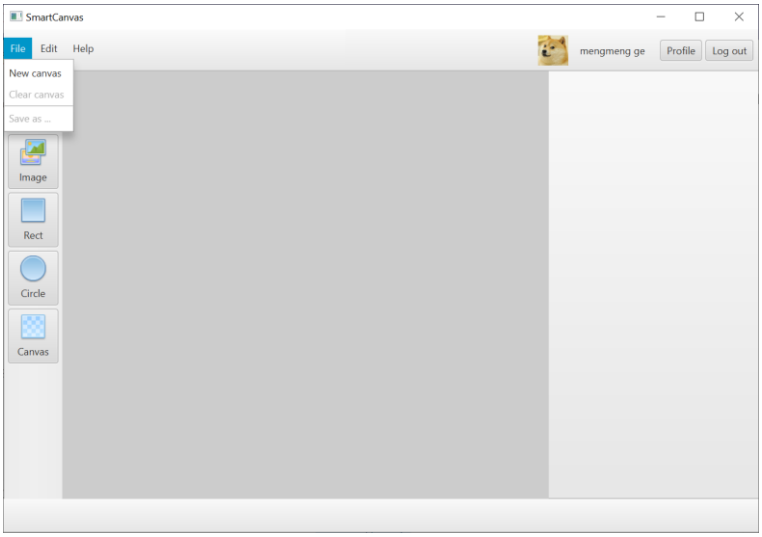

Figure 5. Menu for canvas.

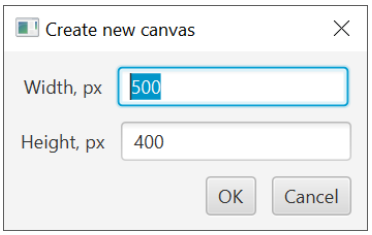

Figure 6. Create a new canvas with dimensions.

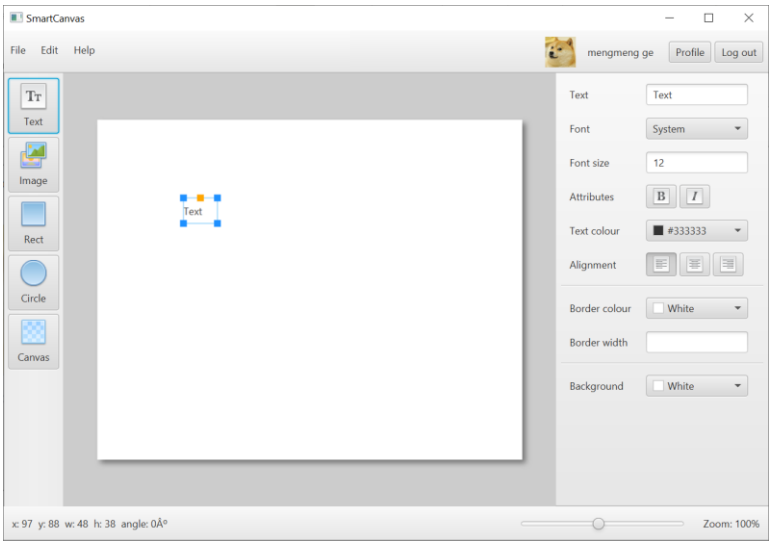

Figure 7. Add a text element; when selecting the text, options for editing the text features are shown.

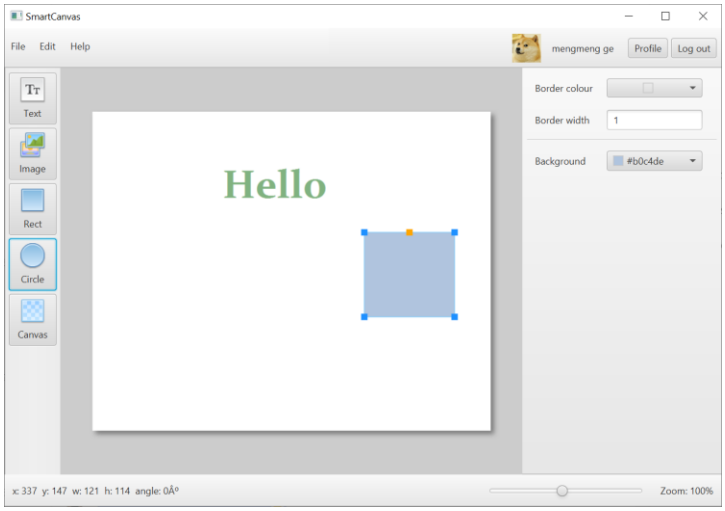

Figure 8. Add a rectangle element; when selecting the rectangle, options for editing its features are shown.

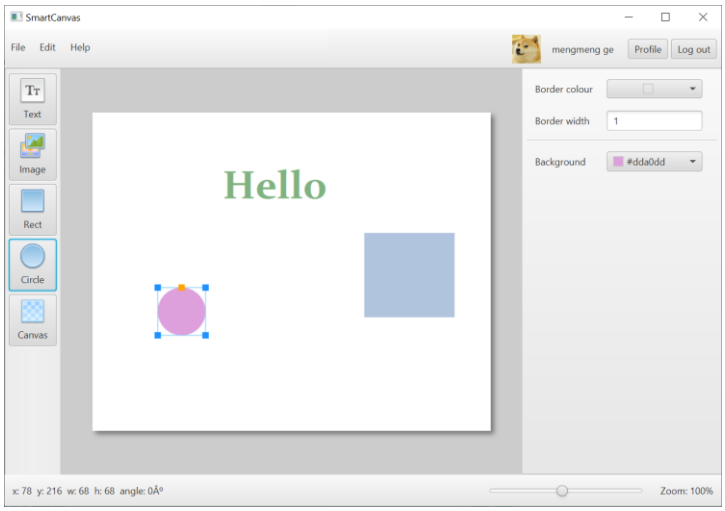

Figure 9. Add a circle element; when selecting the circle, options for editing its features are shown.

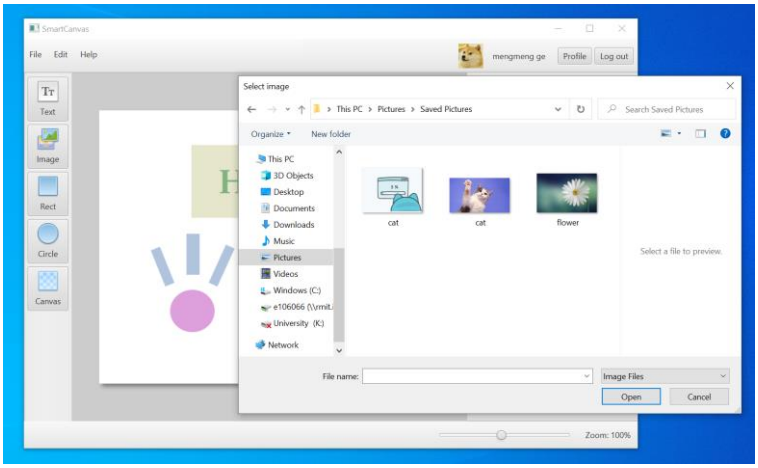

Figure 10. Add an image element (choose from the local machine).

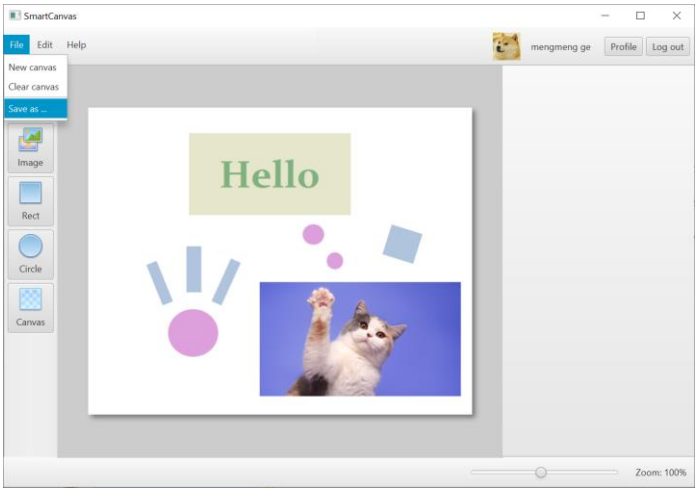

Figure 11. Export the canvas as an image.

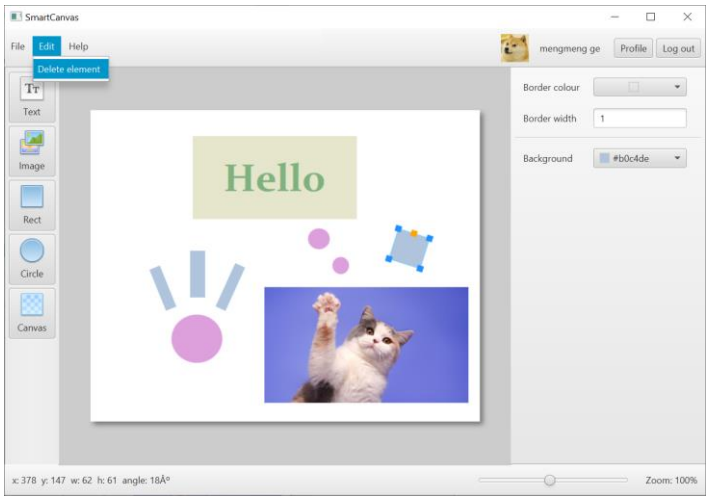

Figure 12. Delete an selected element.

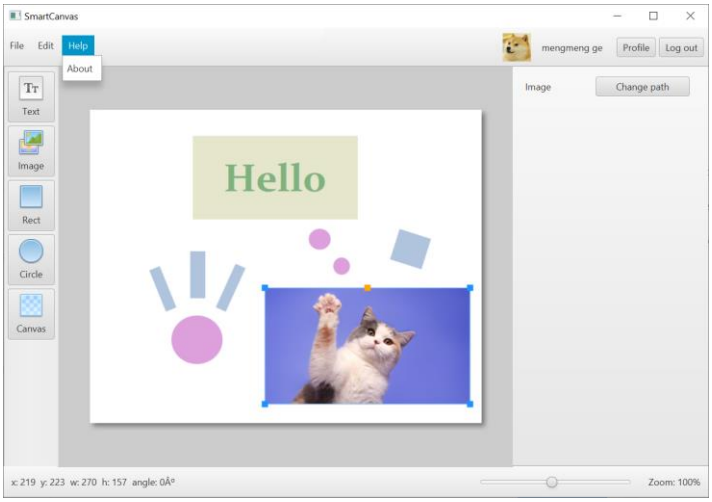

Figure 13. Check About page.

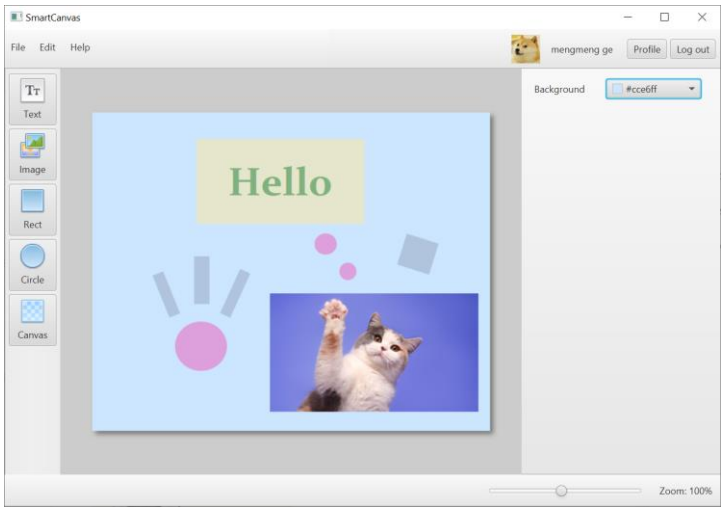

Figure 14. Change the background color of the canvas.

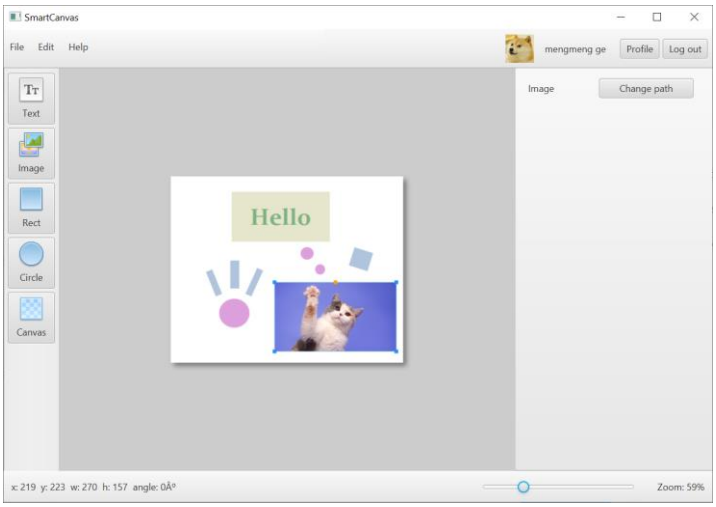

Figure 15. Zoom out the canvas.