```
import json
import os

class Library:
    DATA_FILE = "library_data.json"

    def __init__(self):
        """
        The constructor initializes the library system by checking
        if the data file exists. If not, it creates an empty file to store book data.
        """
        self.initialize_library()

    def initialize_library(self):
        """
        This method checks if the data file ('library_data.json') exists.
        If not, it creates an empty file so that books can be saved later.
        """
        if not os.path.exists(self.DATA_FILE):
            with open(self.DATA_FILE, 'w') as file:
                json.dump([], file)  # Create an empty list in the file if it doesn't exist

    def load_data(self):
        """
        This method loads the current list of books from the data file.
        If the file doesn't exist or is empty, it returns an empty list.

        Returns:
            list: List of books stored in the data file.
        """
        try:
            with open(self.DATA_FILE, 'r') as file:
                return json.load(file)
        except (json.JSONDecodeError, FileNotFoundError):
            return []  # Return an empty list if the file is missing or can't be read

    def save_data(self, data):
        """
        This method saves the provided list of books to the data file.

        Args:
            data (list): The list of books to save to the file.
        """
        with open(self.DATA_FILE, 'w') as file:
            json.dump(data, file, indent=4)  # Save data with a pretty-printed format

    def add_book(self):
        """
        This method prompts the user to add a new book. It asks for the book's title, author,
        and year of publication. The new book is added to the library with a unique ID.
        """
        print("\n-- Add Book --")
        title = input("Enter book title: ").strip()
        author = input("Enter book author: ").strip()
        year = input("Enter year of publication: ").strip()

        if not year.isdigit():
            print("Year must be a number!")  # Validate that year is a number
            return

        books = self.load_data()
        book_id = len(books) + 1  # Generate a unique ID for the book
        book = {
            "id": book_id,
            "title": title,
            "author": author,
            "year": int(year),
            "status": "available"
        }
        books.append(book)  # Add the new book to the list
        self.save_data(books)  # Save the updated list of books
        print(f"Book '{title}' added successfully!")

    def delete_book(self):
        """
        This method allows the user to delete a book by entering its ID.
        If the book with the specified ID is found, it is removed from the library.
        """
        print("\n-- Delete Book --")
        try:
            book_id = int(input("Enter book ID to delete: ").strip())
        except ValueError:
            print("Invalid ID. Please enter a number.")  # Handle invalid input (non-numeric)
            return

        books = self.load_data()
        for book in books:
            if book["id"] == book_id:
                books.remove(book)  # Remove the book from the list
                self.save_data(books)  # Save the updated list
                print(f"Book with ID {book_id} deleted successfully!")
                return

        print(f"No book found with ID {book_id}.")  # Handle case when ID is not found

    def search_book(self):
        """
        This method lets the user search for books by title, author, or year.
        The user provides a search query, and the program filters the books
        based on the selected criteria (title, author, or year).
        """
        print("\n-- Search Book --")
        criteria = input("Search by (title/author/year): ").strip().lower()
        query = input("Enter search query: ").strip()
        books = self.load_data()
        results = []

        if criteria == "title":
            results = [book for book in books if query.lower() in book["title"].lower()]
        elif criteria == "author":
            results = [book for book in books if query.lower() in book["author"].lower()]
        elif criteria == "year" and query.isdigit():
            results = [book for book in books if book["year"] == int(query)]
        else:
            print("Invalid search criteria.")  # Handle invalid search criteria
            return

        if results:
            print("\nSearch Results:")
            for book in results:
                print(f"ID: {book['id']}, Title: {book['title']}, Author: {book['author']}, Year: 
{book['year']}, Status: {book['status']}")
        else:
            print("No matching books found.")  # If no books match the search

    def display_books(self):
        """
        This method displays a list of all books in the library, showing
        the ID, title, author, year, and status for each book.
        """
        print("\n-- All Books --")
        books = self.load_data()
        if books:
            for book in books:
                print(f"ID: {book['id']}, Title: {book['title']}, Author: {book['author']}, Year: 
{book['year']}, Status: {book['status']}")
        else:
            print("No books in the library.")  # If there are no books in the library

    def change_status(self):
        """
        This method allows the user to change the status of a book.
        The user enters the book's ID and selects either 'available' or 'issued'
        as the new status for the book.
        """
        print("\n-- Change Book Status --")
        try:
            book_id = int(input("Enter book ID: ").strip())
        except ValueError:
            print("Invalid ID. Please enter a number.")  # Handle invalid ID input
            return

        new_status = input("Enter new status ('available' or 'issued'): ").strip().lower()
        if new_status not in ["available", "issued"]:
            print("Invalid status. Please choose 'available' or 'issued'.")  # Validate status input
            return

        books = self.load_data()
        for book in books:
            if book["id"] == book_id:
                book["status"] = new_status  # Update the book's status
                self.save_data(books)  # Save the updated list of books
                print(f"Status of book with ID {book_id} updated to '{new_status}'.")
                return

        print(f"No book found with ID {book_id}.")  # If book ID is not found

    def main_menu(self):
        """
        This method displays the main menu with options for the user to choose
        what action they want to perform. It returns the user's choice.

        Returns:
            str: The user's menu choice.
        """
        print("\nLibrary Management System")
        print("1. Add Book")
        print("2. Delete Book")
        print("3. Search Book")
        print("4. Display All Books")
        print("5. Change Book Status")
        print("6. Exit")

        choice = input("Enter your choice: ").strip()
        return choice

    def run(self):
        """
        This method runs the entire program. It continuously displays the main menu
        and allows the user to perform different actions (add, delete, search, etc.)
        until the user chooses to exit.
        """
        while True:
            choice = self.main_menu()
            if choice == '1':
                self.add_book()
            elif choice == '2':
                self.delete_book()
            elif choice == '3':
                self.search_book()
            elif choice == '4':
                self.display_books()
            elif choice == '5':
                self.change_status()
            elif choice == '6':
                print("Exiting the system. Goodbye!")  # Exit the program
                break
            else:
                print("Invalid choice. Please try again.")  # Handle invalid menu choice

# Main block to run the program
if __name__ == "__main__":
    library = Library()
    library.run()
```