Final Project Guide: Student Management System

Project Overview

As a Python developer at **EduTech Solutions**, you're tasked with building a **Student Management System**. This system allows users to manage student records with secure login functionality, basic CRUD operations, and file-based data storage.

This project provides hands-on experience with:

- User authentication and file handling
- CRUD operations (Create, Read, Update, Delete)
- Menu-driven interfaces
- Data validation and search functionality

System Capabilities

- Secure login system with session logging
- Add, view, search, and update student records.
- Store all data in text files
- Generate basic student reports
- User activity logging

Follow the instructions below to complete the project in **Jupyter Lab**.

Step 1: Create the Project Structure

Launch JupyterLab, create a new Jupyter Notebook, and name it "student_management_system.ipynb".

Step 2: import the Required Modules

Launch JupyterLab, before coding, ensure that the required modules are installed. In a Jupyter Lab cell, run the following import statements:

```
import os
import getpass
import datetime
```

This imports the required modules. These modules allow you to work with date and time, and JSON data.

Step 3: Store Files Path.

```
# ------------------------------
# File paths
# ------------------------------
USER_FILE = "users.txt"
STUDENT_FILE = "students.txt"
LOG_FILE = "activity_log.txt"
```

Step 4: Authentication System

The authentication system secures access to the Student Management System. It loads user credentials from a file, allows new users to be saved, and checks login details against stored records. Successful logins are recorded in an activity log, while invalid attempts are denied. This ensures only authorized users can manage student data.

✓ Log and Load user's functions

```
def log_activity(username, action):
    """Log user actions with timestamp"""
    with open(LOG_FILE, "a") as log:
        log.write(f"{datetime.datetime.now()} - {username} - {action}\n")

def load_users():
    """Load users from file into a dictionary"""
    users = {}
    if os.path.exists(USER_FILE):
        with open(USER_FILE, "r") as f:
            for line in f:
                if "," in line:
                    uname, pwd = line.strip().split(",", 1)
                    users[uname] = pwd
    return users
```

✓ Save User function

```
def save_user(username, password):
    """Save new user to the file"""
    with open(USER_FILE, "a") as f:
        f.write(f"{username},{password}\n")
```

✓ Authenticate User function.

```
def authenticate():
    """Authenticate user login"""
    users = load_users()
    username = input("Enter Username: ")
    password = getpass.getpass("Enter Password: ")

    if username in users and users[username] == password:
        log_activity(username, "Logged in")
        print("\n☑ Login Successful!\n")
        return username
    else:
        print("\n❌ Invalid credentials!\n")
        return None
```

🪜 Step 5: Student Data Management (CRUD)

The student data management functions handle records stored in a file. They allows users to add new students, view all records, search for specific students, update details, and delete entries. This ensures accurate and organized management of student information.

✓ Add Student function.

```
def add_student(username):
    roll = input("Enter Roll Number: ")
    name = input("Enter Name: ")
    grade = input("Enter Grade: ")

    with open(STUDENT_FILE, "a") as f:
        f.write(f"{roll},{name},{grade}\n")

    log_activity(username, f"Added student {roll}")
    print("☑ Student added successfully!")
```

✓ View Student function.

```
def view_students():
    if not os.path.exists(STUDENT_FILE):
        print("No records found.")
        return

    with open(STUDENT_FILE, "r") as f:
        print("\n--- Student Records ---")
        for line in f:
            roll, name, grade = line.strip().split(",")
            print(f"Roll: {roll} | Name: {name} | Grade: {grade}")
```

✓ Search Student function.

```
def search_student():
    roll_no = input("Enter Roll Number to search: ")
    found = False

    if os.path.exists(STUDENT_FILE):
        with open(STUDENT_FILE, "r") as f:
            for line in f:
                # Skip empty or invalid lines
                parts = line.strip().split(",")
                if len(parts) != 3:
                    continue
                roll, name, grade = parts
                if roll == roll_no:
                    print(f"☑ Found: Roll: {roll} | Name: {name} | Grade:
{grade}")
                    found = True
                    break
```

✓ **Update Student function.**

```
def update_student(username):
    roll_no = input("Enter Roll Number to update: ")
    students = []
    updated = False

    if os.path.exists(STUDENT_FILE):
        with open(STUDENT_FILE, "r") as f:
            students = f.readlines()

        with open(STUDENT_FILE, "w") as f:
            for line in students:
                roll, name, grade = line.strip().split(",")
                if roll == roll_no:
                    print(f"Current Data → Name: {name}, Grade: {grade}")
                    new_name = input("Enter new Name: ")
                    new_grade = input("Enter new Grade: ")
                    f.write(f"{roll},{new_name},{new_grade}\n")
                    updated = True
                    log_activity(username, f"Updated student {roll}")
                else:
                    f.write(line)

    if updated:
        print("☑ Student record updated.")
    else:
        print("❌ Student not found.")
```

✓ Delete Student function

```python
def delete_student(username):
    roll_no = input("Enter Roll Number to delete: ")
    students = []
    deleted = False

    if os.path.exists(STUDENT_FILE):
        with open(STUDENT_FILE, "r") as f:
            students = f.readlines()

        with open(STUDENT_FILE, "w") as f:
            for line in students:
                # Skip empty or invalid lines
                parts = line.strip().split(",")
                if len(parts) != 3:
                    f.write(line)  # keep invalid lines unchanged
                    continue

                roll, name, grade = parts
                if roll == roll_no:
                    deleted = True
                    # Log deletion
                    log_activity(username, f"Deleted student {roll}")
                else:
                    f.write(line)

    if deleted:
        print("☑ Student record deleted.")
    else:
        print("❌ Student not found.")
```

📝 Step 6: Report Generation

```python
def generate_report():
    if not os.path.exists(STUDENT_FILE):
        print("No records available.")
        return

    total = 0
    grades = {}
    with open(STUDENT_FILE, "r") as f:
        for line in f:
            roll, name, grade = line.strip().split(",")
            total += 1
            grades[grade] = grades.get(grade, 0) + 1

    print("\n📊 Student Report")
    print(f"Total Students: {total}")
    for g, count in grades.items():
        print(f"Grade {g}: {count} student(s)")
```

🪜 Step 7: Main Program

The function below allow users to remove events, it prompts for the event title to delete. It filters the events list to exclude any event with a matching title (case-insensitive) and updates the list. A confirmation message is displayed regardless of whether the event was found or not.

```
def main():
    print("===== Student Management System =====")

    # Ensure admin user exists
    if not os.path.exists(USER_FILE):
        print("No users found. Create an Admin account.")
        uname = input("Set Admin Username: ")
        pwd = getpass.getpass("Set Admin Password: ")
        save_user(uname, pwd)
        print("☑ Admin created! Please restart the program.")

    username = None
    while not username:
        username = authenticate()

    # Menu
    while True:
        print("\n--- Main Menu ---")
        print("1. Add Student")
        print("2. View All Students")
        print("3. Search Student")
        print("4. Update Student")
        print("5. Delete Student")
        print("6. Generate Report")
        print("7. Logout & Exit")

        choice = input("Enter choice: ")

        if choice == "1":
            add_student(username)
        elif choice == "2":
            view_students()
        elif choice == "3":
            search_student()
        elif choice == "4":
            update_student(username)
        elif choice == "5":
            delete_student(username)
        elif choice == "6":
            generate_report()
        elif choice == "7":
            log_activity(username, "Logged out")
            print("👋 Goodbye!")
            break
        else:
            print("❌ Invalid choice! Please try again.")


if __name__ == "__main__":
    main()
```

🪜 Step 8: Document the Process

Ensure that the JupyterLab notebook includes headings, detailed comments, and documentation. This documentation should provide clear insights into each step of the process. Ensure that anyone reviewing the notebook gains a clear understanding of each step, creating a comprehensive reference for future use.

Step 9: Save and Submit Your JupyterLab File

Upon completing the project, save your JupyterLab file and send the project file to **assignments@ritaafrica.com**.

***Note:** Test all functionalities to ensure they work as expected. Make the project look professional. The deadline for submission and publishing is 11:59 PM GMT on Thursday, Sept 4th, 2025.*

Step 10: Publish the Project on LinkedIn

Share your completed project on LinkedIn to showcase your skills. Include your own comments, tag the RITA Africa Official LinkedIn page, and use the provided hashtags to join the conversation. Here's a suggested structure for your post:

🎓 Student Management System – A Capstone Project from the RITA Africa Python Fundamental Course!

I'm excited to share a Student Management System I built during the RITA Africa bootcamp (tag the official RITA Africa LinkedIn page). This project shows how Python can be applied to build real-world applications that make data management easier and more secure.

Key Features:
✅ Secure login with file-based authentication
✅ Add, view, search, update, and delete student records
✅ Generate basic reports on student data
✅ Store data in text files for persistence
✅ Log user activity for better tracking.

Skills Demonstrated:
- 🔹 File handling for data storage and retrieval
- 🔹 Dictionaries, loops, and conditionals for program logic
- 🔹 Functions for modular code organization
- 🔹 Authentication and session logging for security
- 🔹 Menu-driven interfaces for user-friendly navigation.

This project gave me hands-on experience in building structured applications while applying Python fundamentals to solve practical problems. 🚀

Check out the project and let me know your thoughts! 💬
#RiseInTechAfrica #RITAAfricaBootcamp #Python #CapstoneProject #EventPlanner #PythonBeginners

Check out the project and let me know your thoughts!

#RiseInTechAfrica #RITAAfricaBootcamp #RITAAfrica #Python

How to Share:

1. Take a screenshot of your JupyterLab notebook showing the project in action.
2. Record a short screen recording demonstrating key features of your project.
3. Upload the screenshot or screen recording to your LinkedIn post.

Instructions to Publish on GitHub:

Before You Begin:

Make sure you have a GitHub account and have installed Git on your computer.

Option 1:

1. Go to https://github.com and log in.
2. Click the "+" icon in the top-right and choose **"New repository."**
3. Name your repository something like **student-management-system**.
4. Click **"Create repository."**
5. Once the repo is created, click **"Add file" → "Upload files."**
6. Upload your project files: `main.py`, `students.txt`, `users.txt`, `activity_log.txt`, and any other related files.
7. Write a short commit message like **"Initial upload of Student Management System project"** and click **"Commit changes."**

Option 2: Upload Using Git (Optional for advanced users) If you are comfortable with Git on the terminal:

- cd path/to/student_management_system
- git init
- git add.
- git commit -m "Initial commit: Add Student Management System project."
- git remote add origin https://github.com/yourusername/student-management-system.git
- git branch -M main
- git push -u origin main

💭 **What to Include in Your Repository:**

- student_management_system.ipynb – Your main notebook
- README.md – A brief summary of your project (what it does, how to run it, skills learned)

- student.txt – (Optional) Sample result file

✅ **Example README Content:**

🎓 Student Management System

A simple Python-based **Student Management System** developed as a capstone project for the **RITA Africa Python Fundamentals Course**.
This application allows secure login, student record management (CRUD), report generation, and file-based data persistence.

🚀 Features

- 🔑 **Authentication System** – Secure login with user accounts stored in `users.txt`
- 📁 **Student Data Management** – Add, view, search, update, and delete student records
- 📊 **Report Generation** – Generate basic reports on total students and grade distribution
- 📝 **Activity Logging** – Tracks user logins, logouts, and actions in `activity_log.txt`
- 💾 **File-Based Storage** – Data is stored in text files (`students.txt`, `users.txt`) for persistence

---

🛠️ Technologies Used

- **Python 3.x**
- **File Handling (TXT)**
- **Datetime module** (for logging timestamps)
- **Getpass module** (for hidden password entry)

📂 Project Structure

```
student_management_system/
├── main.py              # Main program (menu-driven interface)
├── users.txt            # Stores usernames and passwords
├── students.txt         # Stores student records
└── activity_log.txt     # Stores user activity logs
```

⚙️ How to Run

1. Clone this repository:
2. git clone https://github.com/yourusername/student-management-system.git
3. cd student-management-system
4. Run the program:
5. python main.py
6. If running for the first time, create an **Admin account** when prompted.

📖 Usage

- **Login** with a valid username and password.
- Use the menu to:
    - Add a new student
    - View all students
    - Search for a student by roll number
    - Update student details
    - Delete student records
    - Generate a student report
- **Logout** : when done (session activity will be logged).

📊 Sample Report Output

📊 Student Report

Total Students: 4

Grade A: 2 student(s)

Grade B: 1 student(s)

Grade C: 1 student(s)

🙌 Acknowledgements

This project was created as part of the **RITA Africa Python Fundamentals Bootcamp**.
Special thanks to .....

Feel free to add your own comments and insights to the post, and don't forget to tag the RITA Africa Official LinkedIn page. Be sure to use the hashtags provided to join the conversation and share your achievement with the community!