PROJECT REPORT OF GOOGLE CLASSROOM

Implementation of Google Classroom using OOP Principles

Google Classroom is built on the principles of **Object-Oriented Programming (OOP)**, which allows the system to be modular, scalable, and easily maintainable. OOP helps in organizing the software into reusable and interconnected objects representing real-world entities like users, courses, and assignments.

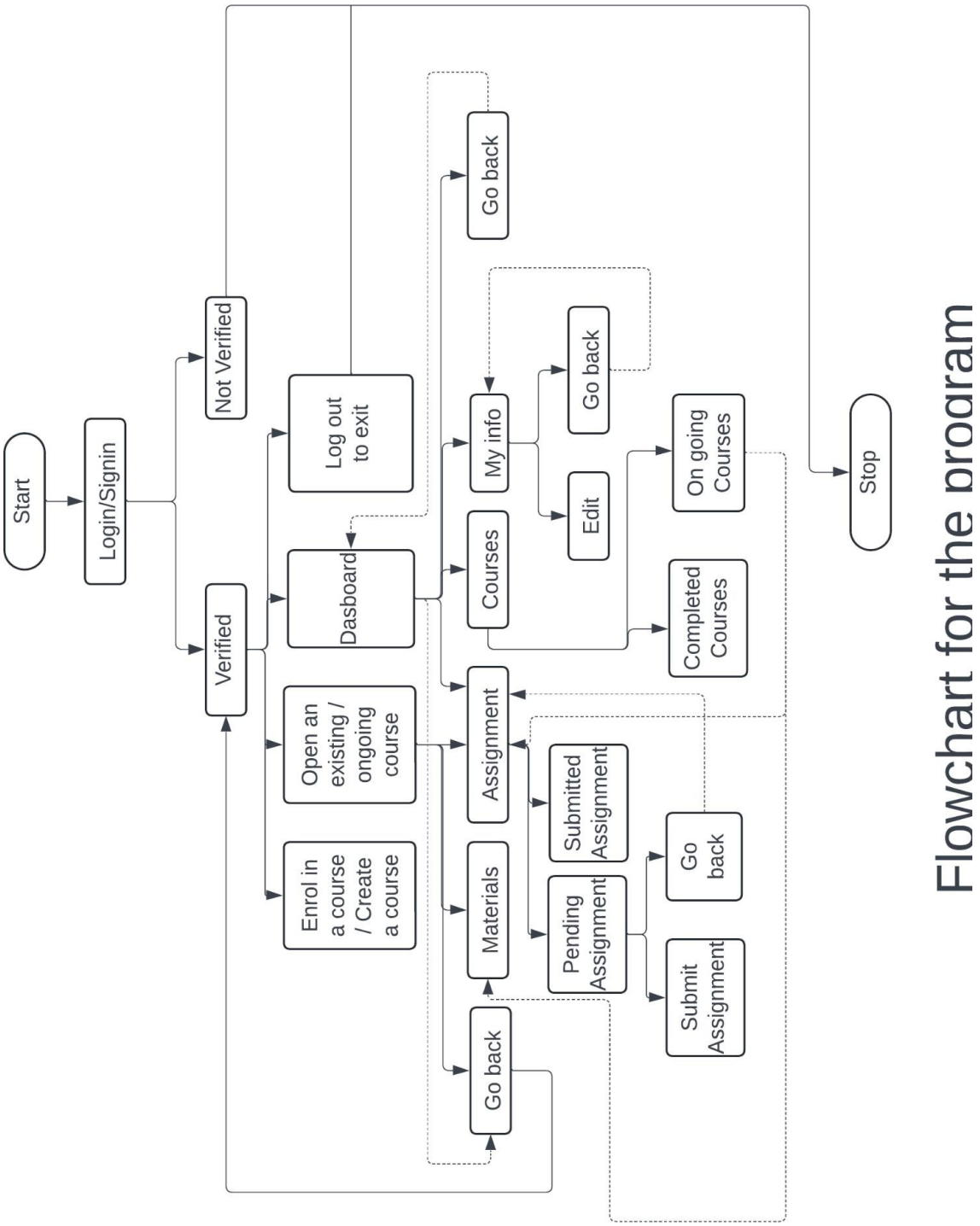
KEY FEATURES OF THE PROJECT INCLUDE:

- **Teacher/Student Registration and login**
- **Assignment management:** Teachers can distribute and collect assignments through a shared interface.
- **Course creation and enrollment:** Teachers create courses, and students can join using the course codes.
- **Real-time communication:** Educator can give announcements to students.
- **Grading system:** Teachers can grade assignments and provide feedback, while students can track their progress.

OOPS CONCEPTS APPLIED:

- **File handling**
- **Inheritance**
- **Exception handling using try and catch**
- **Abstraction**
- **Encapsulation**
- **Operator overloading (<<)**
- **Static function , method overriding**

System Architecture Flowchart of Google Classroom



CLASSES, OBJECTS IN THE PROEJCT

The design of Google Classroom revolves around several key classes and objects that represent real-world entities and abstract processes. Below is an overview of the major classes and their roles in our project :

1. **User Class**
    - **Attributes:** Full `name`, `email`, password, `role` (e.g., Teacher or Student)
    - **Methods:** `view_my_account()`
2. **Teacher Class (inherits from User)**
    - **Attributes:** `mycourses;`
    - **Methods:** `gradeAssignment(), teacherOptions(), postAnnouncement(),view_my_account(),viewSubmissions()`
3. **Student Class (inherits from User)**
    - **Attributes:** `studentId.`
    - **Methods:** `viewCourses(), submitAssignment(), viewGrades(),view_my_account(), studentOptions(), displayAssignmentsForCourse(), submitAssignment()`
4. **Course Class**
    - **Attributes:** string `courseName`, int `courseId`, string `teacher`, `students.`
    - **Methods:** `addAssignment(), courseOptions()`
    - **Description:** The `Course` class manages the structure of a course, tracking enrolled students and assignments.
5. **Assignment Class**
    - **Attributes:** `assignmentId, title, description, dueDate, courseId`

- **Methods:** `viewAssignmentDetails()`.

6. **Announcement Class**

- **Attributes:** `string title, string message, int cId, int announcementId.`
- **Methods:** `displayAnnouncement(),savetoCSV().`
- **Description:** This class allows teachers to post announcements for students, providing updates on course activities.

7. **Submission Class**

- **Attributes:** submissionId, assignmentId, studentId, submissionDate, grade, feedback, filePath
- **Methods:** `addSubmission();`

APPLICATION OF OOP CONCEPTS IN THE PROJECT :

1. **Abstraction**

★ Abstraction simplifies interactions by hiding unnecessary details from users enhancing ease of use and maintainability.

- The submission class has private attributes so that the important information relevant to the user is marked as public.
- Teachers and students interact with announcements through a simple interface, without knowing how messages are sent in the backend.

- Getters and setters are used to control access to class attributes, preventing unauthorized or accidental modifications.

---

2. **Method overriding**
    - Both `Teacher` and `Student` inherit from the `User` class and override the view_my_account`()` method to show personalized profiles. Teachers see courses they manage, while students view their enrolled courses.

---

3. **Inheritance**
    - The teacher and the Student classes are inherited from the ‘User’ class.

4. **File handling :**

    - ❖ Used file handling to store, retrieve students, teachers, announcements, courses, submissions details using **csv** files such as announcements.csv, submissions.csv, courses.csv, students.csv, teachers.csv whenever the data is needed.
