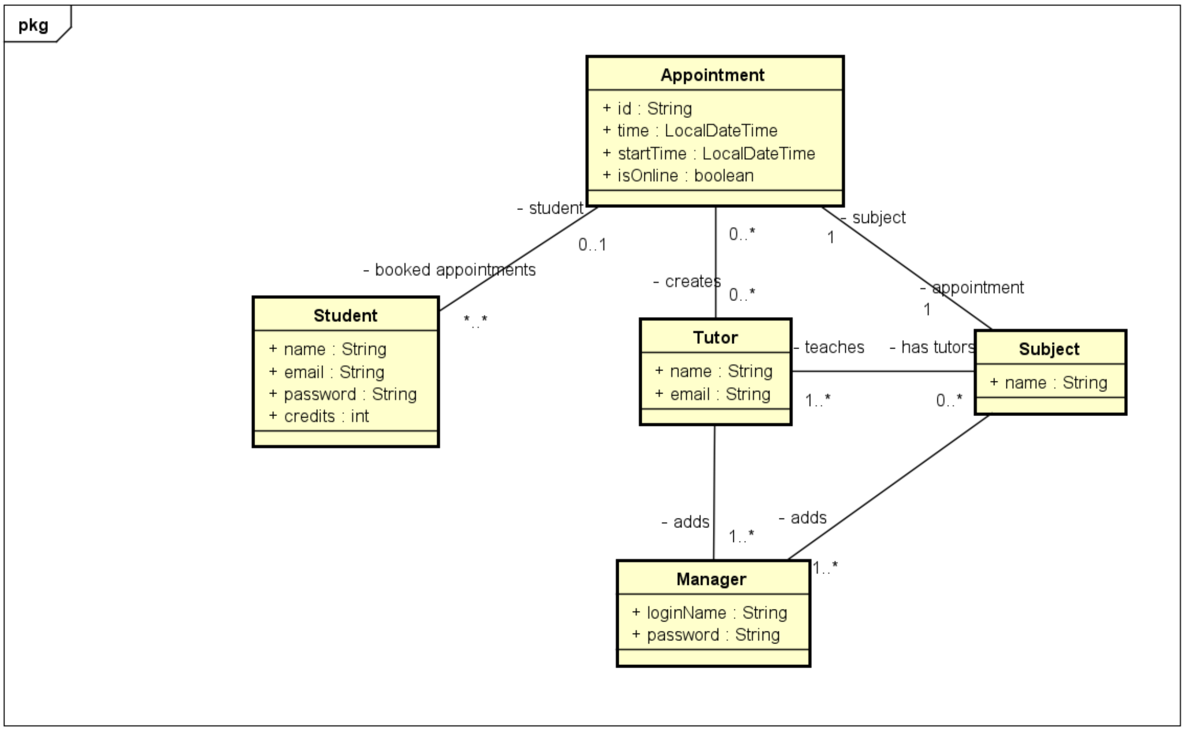

pkg
Appointment
+ id : String
+ time : LocalDateTime
+ startTime : LocalDateTime
+ isOnline : boolean
- student
0..1
- booked appointments
..
Student
+ name : String
+ email : String
+ password : String
+ credits : int
0..*
- creates
0..*
Tutor
+ name : String
+ email : String
- subject
1
- appointment
1
- teaches
- has tutors
1..*
0..*
Subject
+ name : String
- adds
1..*
- adds
1..*
Manager
+ loginName : String
+ password : String

Part 1: Domain Class Model (DCM)

Class Descriptions

Manager

- Attributes:
  - loginName: String
  - password: String
- Responsibilities:
  - Add new subjects to the system
  - Add new tutors
  - Manage system login

Tutor

- Attributes:
  - name: String
  - email: String
- Responsibilities:
  - Sign up for subjects
  - Create appointments
  - Set new password

Student

- Attributes:
  - name: String
  - email: String
  - password: String
  - credits: int
- Responsibilities:
  - Sign up for the system
  - Buy credits
  - Book and cancel appointments

Subject

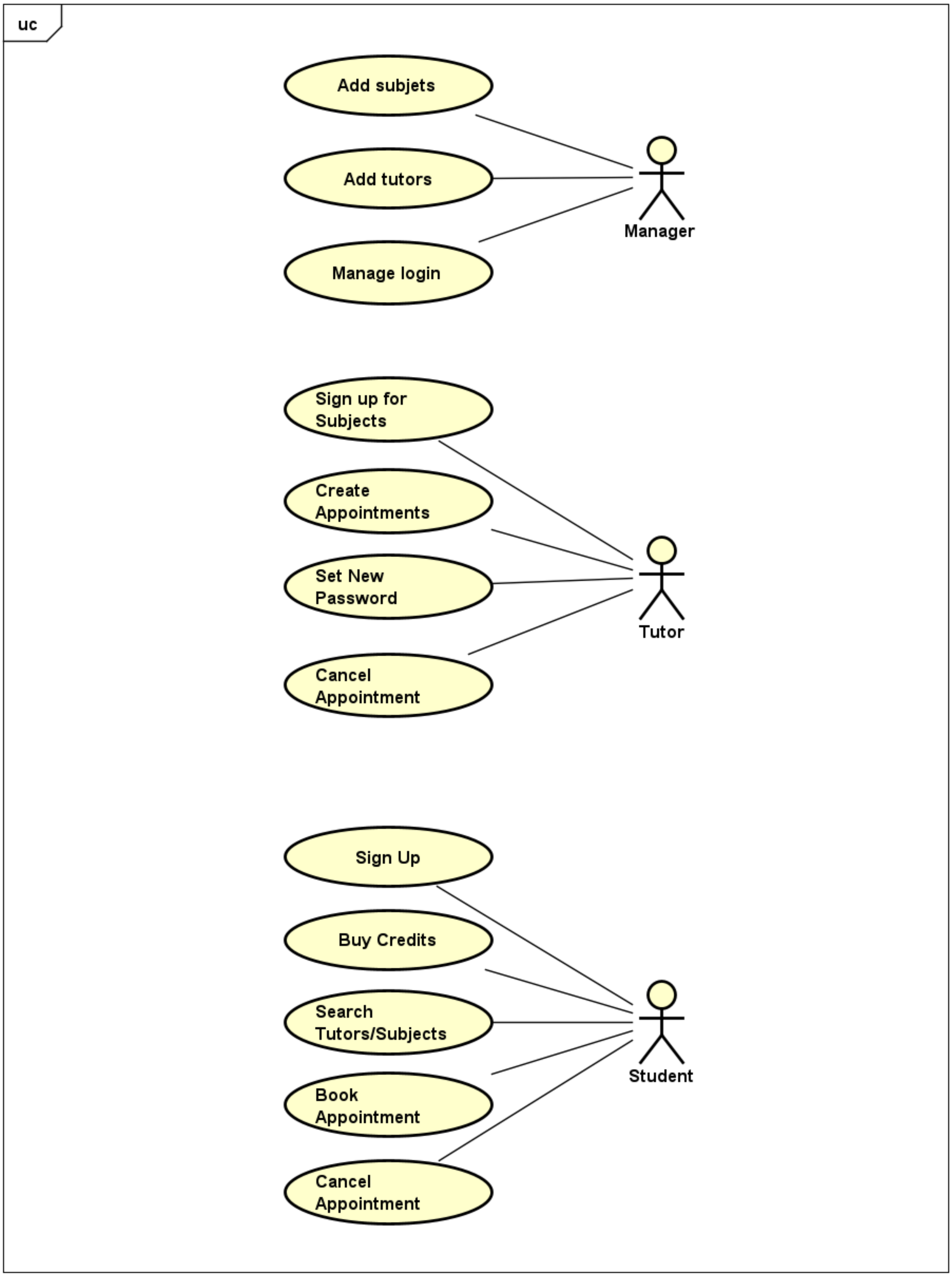
- Attributes:
  - name: String
- Responsibilities:
  - Represent teachable subjects in the system

Appointment

- Attributes:
  - id: String
  - time: LocalDateTime
  - startTime: LocalDateTime
  - isOnline: boolean
- Responsibilities:
  - Represent tutoring sessions
  - Link students, tutors, and subjects

Diagram Explanation

The Domain Class Model shows the key entities in the tutoring system and their relationships. Associations demonstrate how different classes interact, such as a student booking appointments or a tutor creating appointments for specific subjects.


uc
Add subjets
Add tutors
Manage login
Manager
Sign up for Subjects
Create Appointments
Set New Password
Cancel Appointment
Tutor
Sign Up
Buy Credits
Search Tutors/Subjects
Book Appointment
Cancel Appointment
Student

Part 2: Use Case Diagram
Manager Use Cases

1. Add Subjects
   - Scenario: Manager enters a new subject name
   - Success: Subject added if it doesn't exist
   - Error: System notifies if subject already exists
2. Add Tutors
   - Scenario: Manager provides tutor's name and email
   - Success: Tutor added with temporary password
   - Error: System notifies if tutor already exists
3. Manage Login
   - Scenario: Manager logs into the system
   - Success: Access to system functions
   - Error: Access denied if credentials are incorrect

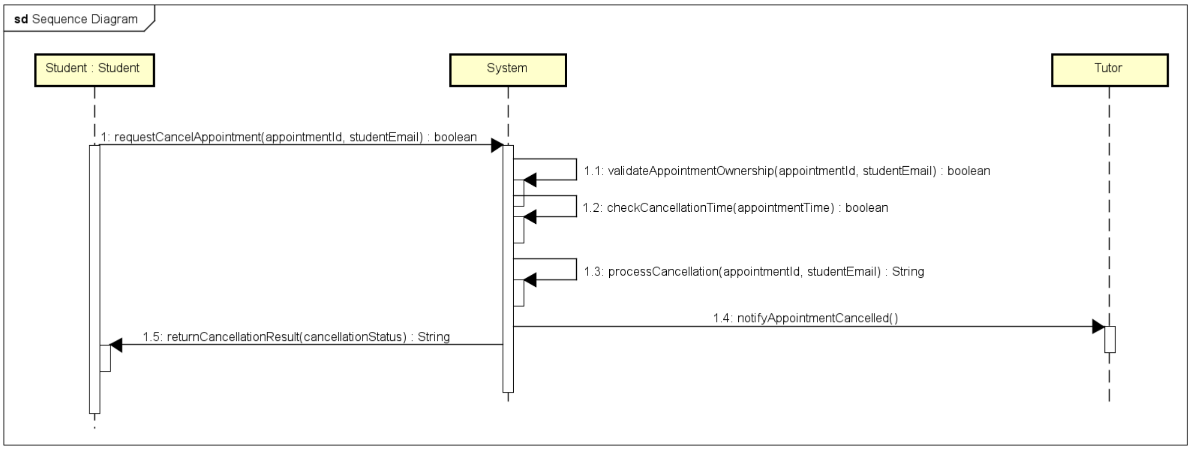
Tutor Use Cases

1. Sign up for Subjects
   - Scenario: Tutor selects subjects to teach
   - Success: Subjects linked to tutor's profile
   - Error: Cannot add non-existent subjects
2. Create Appointments
   - Scenario: Tutor sets available time and session type
   - Success: Appointment created
   - Error: Conflicting appointments or exceeding in-person limits
3. Set New Password
   - Scenario: Tutor changes account password
   - Success: Password updated
   - Error: Incorrect old password or invalid new password

Student Use Cases

1. Sign Up
   - Scenario: Student creates a new account
   - Success: Account created with provided details
   - Error: Email already in use
2. Buy Credits
   - Scenario: Student purchases tutoring credits
   - Success: Credits added to account
   - Error: Payment issues
3. Search Tutors/Subjects
   - Scenario: Student searches for tutoring options
   - Success: List of available appointments
   - Error: No matching tutors or subjects found
4. Book Appointment
   - Scenario: Student selects and books an appointment
   - Success: Appointment booked, credits deducted
   - Error: Insufficient credits or already booked appointment
5. Cancel Appointment
   - Scenario: Student cancels a booked appointment
   - Success: Appointment cancelled
     - 24 hours: Credit returned
     - <24 hours: No credit returned
   - Error: Appointment not found or already cancelled

-


sd Sequence Diagram
Student : Student
System
Tutor
1: requestCancelAppointment(appointmentId, studentEmail) : boolean
1.1: validateAppointmentOwnership(appointmentId, studentEmail) : boolean
1.2: checkCancellationTime(appointmentTime) : boolean
1.3: processCancellation(appointmentId, studentEmail) : String
1.4: notifyAppointmentCancelled()
1.5: returnCancellationResult(cancellationStatus) : String

Part 3: Sequence Diagram

Student Cancels Appointment Workflow

The Sequence Diagram illustrates the process of a student cancelling an appointment:

1. Student requests appointment cancellation
2. System validates appointment ownership
3. System checks cancellation time
4. System processes cancellation
   - If >24 hours: Credit returned
   - If <24 hours: No credit returned
5. System notifies tutor with appointment cancellation.
6. System returns cancellation result to student

Implementation Details

Key Design Considerations

- Implemented high-level abstraction of tutoring system
- Focused on core functionality
- Demonstrated basic object interactions
- Implemented appointment cancellation logic

Constraints and Assumptions

- Campus appointments limited to 8am-6pm
- Online appointments can be scheduled anytime
- Credit system for appointment booking
- Unique email addresses for users

Conclusion

The design provides a flexible and extensible framework for a tutoring appointment system, capturing key interactions between managers, tutors, and students.