**College Admission System Project:**

**1. Use Case Diagram**

**Purpose:**

The **Use Case Diagram** visually represents the functional requirements of the college admission system. It highlights the interactions between actors (users) and the system, showing how students and administrators interact with various system functions.

**Actors:**

* **Student**: The primary user who interacts with the system to fill, edit, view, and submit forms, as well as make payments.
* **Admin**: Manages students, forms, and handles administrative tasks.
* **Bank**: An external system that processes payments.

**Key Use Cases:**

* **Login**: Both students and admins must log in to access the system.
* **Fill Form**: Students fill out admission forms with necessary information.
* **Edit Form**: Before submission, students may edit forms if required.
* **Submit Form**: After completing a form, students submit it for review and approval.
* **View Form**: Students can view previously filled forms.
* **Manage Forms**: Admins oversee the forms, ensuring completeness and correctness.
* **Manage Users**: Admins create, update, or delete student accounts.
* **Payment**: After submitting a form, students must make a payment, which involves interaction with the bank.

**Relationships:**

* **<<include>>**: For example, submitForm includes the payment process because submitting a form requires payment.
* **<<extend>>**: The edit Form use case can extend from the view Form use case since students may only edit forms they have viewed.

**Importance:**

* **Use Case Diagrams** help identify system requirements and ensure that every function needed by the student or admin is represented.

**2. Sequence Diagram**

**Purpose:**

A **Sequence Diagram** illustrates the flow of interactions between objects (classes) over time, especially when fulfilling a specific use case, such as a student logging in or submitting a form.

**Key Sequence Diagrams:**

1. **Student Login Sequence**:
   * **Actors**: Student, Authentication System
   * **Flow**:
     1. The student enters login credentials.
     2. The system validates the credentials using the Authentication class.
     3. Upon successful validation, the student is granted access to the system.
2. **Form Submission Sequence**:
   * **Actors**: Student, Form, Payment, Bank
   * **Flow**:
     1. The student fills out the form and submits it.
     2. The system validates the form data using the Form class.
     3. The system initiates a payment request using the Payment class.
     4. The payment is processed by the Bank class, which confirms the transaction.
     5. After successful payment, the form status is updated, and a confirmation is sent to the student.
3. **Admin Managing Forms Sequence**:
   * **Actors**: Admin, Form, Student
   * **Flow**:
     1. The admin logs into the system.
     2. The admin selects a form to review or edit.
     3. The system fetches the form details using the Form class.
     4. The admin updates the form status or edits any necessary details.

**Objects in Sequence Diagrams:**

* **Student**: The user initiating actions such as login, form submission, or payment.
* **Admin**: Manages student forms and users.
* **Form**: Holds admission-related data.
* **Payment**: Manages transactions.
* **Bank**: Processes payments externally.
* **Authentication**: Validates login credentials.

**Message Types:**

* **Synchronous Messages**: Represent calls between objects, such as when a student submits a form, and the system calls the Payment class to process the transaction.
* **Return Messages**: Confirm the success of operations, such as successful login or payment confirmation.

**Importance:**

* **Sequence Diagrams** are crucial for visualizing the detailed, step-by-step interactions that occur during specific use cases. They provide insight into how various system components (objects/classes) communicate to achieve a particular task.