Faculty of ICT, Mahidol University

Gemini Project Phase 2 (Revision)

By

6488073 Chalisa Sae-ngow

6488089 Pattaravit Suksri

6488100 Jiraruch Tantiyavarong

6488105 Kantinan Yontawil

6488128 Thanapat Nonpassopon

6488134 Jirateep Rudeerudchanawong

Submitted to

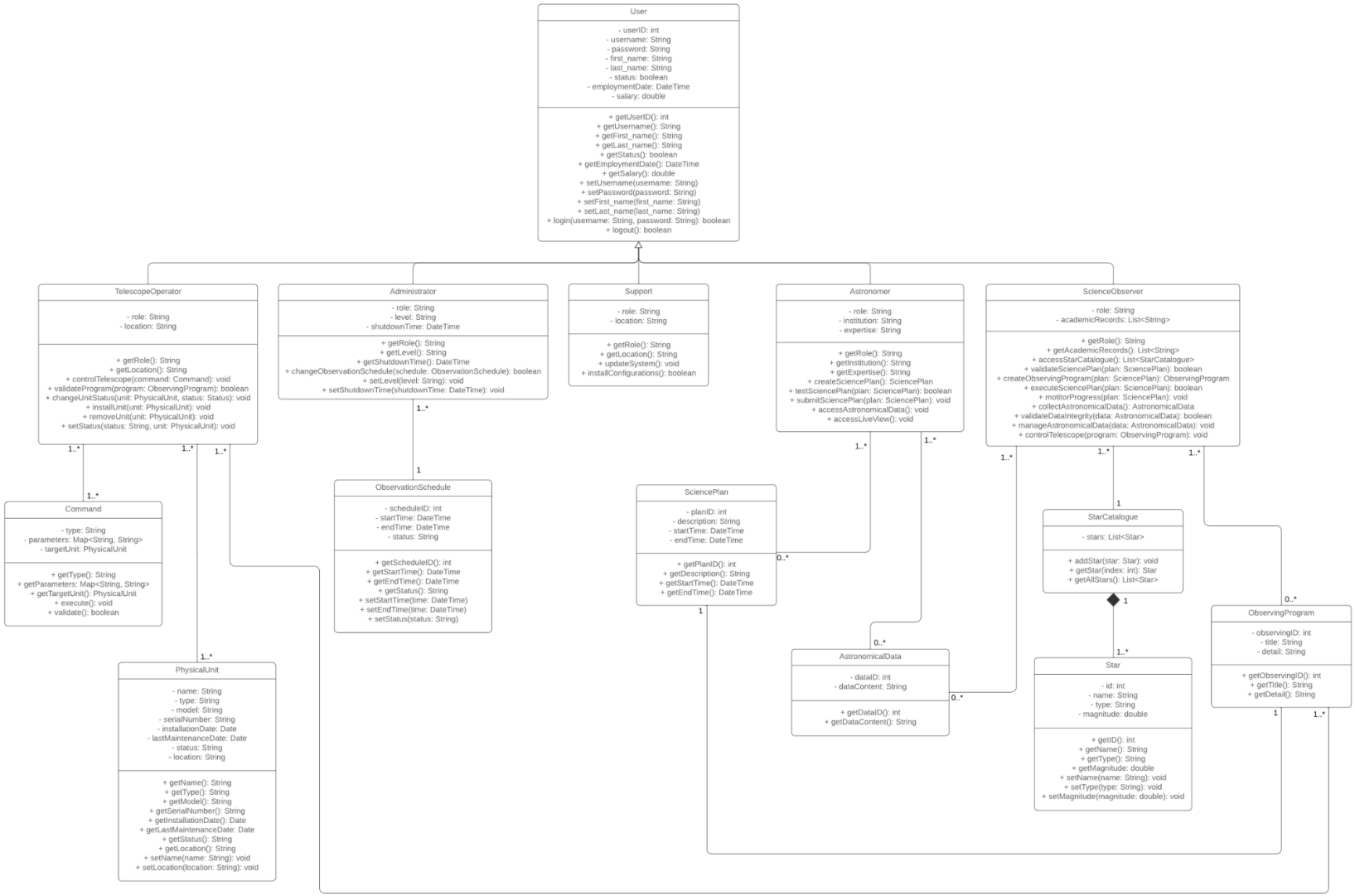
Semester 2/2023

ITCS431 Software Design & Development

Dr. Chaiyong Ragkhitwetsagul / Asst. Prof. Dr. Morakot Choetkiertikul

A report submitted as the fulfillment of the requirements for the assignment

Gemini Class Diagram


User
- userID: int
- username: String
- password: String
- first_name: String
- last_name: String
- status: boolean
- employmentDate: DateTime
- salary: double
+ getUserID(): int
+ getUsername(): String
+ getFirst_name(): String
+ getLast_name(): String
+ getStatus(): boolean
+ getEmploymentDate(): DateTime
+ getSalary(): double
+ setUsername(username: String)
+ setPassword(password: String)
+ setFirst_name(first_name: String)
+ setLast_name(last_name: String)
+ login(username: String, password: String): boolean
+ logout(): boolean
TelescopeOperator
- role: String
- location: String
+ getRole(): String
+ getLocation(): String
+ controlTelescope(command: Command): void
+ validateProgram(program: ObservingProgram): boolean
+ changeUnitStatus(unit: PhysicalUnit, status: Status): void
+ installUnit(unit: PhysicalUnit): void
+ removeUnit(unit: PhysicalUnit): void
+ setStatus(status: String, unit: PhysicalUnit): void
Administrator
- role: String
- level: String
- shutdownTime: DateTime
+ getRole(): String
+ getLevel(): String
+ getShutdownTime(): DateTime
+ changeObservationSchedule(schedule: ObservationSchedule): boolean
+ setLevel(level: String): void
+ setShutdownTime(shutdownTime: DateTime): void
Support
- role: String
- location: String
+ getRole(): String
+ getLocation(): String
+ updateSystem(): void
+ installConfigurations(): boolean
Astronomer
- role: String
- institution: String
- expertise: String
+ getRole(): String
+ getInstitution(): String
+ getExpertise(): String
+ createSciencePlan(): SciencePlan
+ testSciencePlan(plan: SciencePlan): boolean
+ submitSciencePlan(plan: SciencePlan): void
+ accessAstronomicalData(): void
+ accessLiveView(): void
ScienceObserver
- role: String
- academicRecords: List<String>
+ getRole(): String
+ getAcademicRecords(): List<String>
+ accessStarCatalogue(): List<StarCatalogue>
+ validateSciencePlan(plan: SciencePlan): boolean
+ createObservingProgram(plan: SciencePlan): ObservingProgram
+ executeSciencePlan(plan: SciencePlan): boolean
+ motitorProgress(plan: SciencePlan): void
+ collectAstronomicalData(): AstronomicalData
+ validateDataIntegrity(data: AstronomicalData): boolean
+ manageAstronomicalData(data: AstronomicalData): void
+ controlTelescope(program: ObservingProgram): void
Command
- type: String
- parameters: Map<String, String>
- targetUnit: PhysicalUnit
+ getType(): String
+ getParameters: Map<String, String>
+ getTargetUnit(): PhysicalUnit
+ execute(): void
+ validate(): boolean
ObservationSchedule
- scheduleID: int
- startTime: DateTime
- endTime: DateTime
- status: String
+ getScheduleID(): int
+ getStartTime(): DateTime
+ getEndTime(): DateTime
+ getStatus(): String
+ setStartTime(time: DateTime)
+ setEndTime(time: DateTime)
+ setStatus(status: String)
SciencePlan
- planID: int
- description: String
- startTime: DateTime
- endTime: DateTime
+ getPlanID(): int
+ getDescription(): String
+ getStartTime(): DateTime
+ getEndTime(): DateTime
StarCatalogue
- stars: List<Star>
+ addStar(star: Star): void
+ getStar(index: int): Star
+ getAllStars(): List<Star>
PhysicalUnit
- name: String
- type: String
- model: String
- serialNumber: String
- installationDate: Date
- lastMaintenanceDate: Date
- status: String
- location: String
+ getName(): String
+ getType(): String
+ getModel(): String
+ getSerialNumber(): String
+ getInstallationDate(): Date
+ getLastMaintenanceDate: Date
+ getStatus(): String
+ getLocation(): String
+ setName(name: String): void
+ setLocation(location: String): void
AstronomicalData
- dataID: int
- dataContent: String
+ getDataID(): int
+ getDataContent(): String
Star
- id: int
- name: String
- type: String
- magnitude: double
+ getID(): int
+ getName(): String
+ getType(): String
+ getMagnitude: double
+ setName(name: String): void
+ setType(type: String): void
+ setMagnitude(magnitude: double): void
ObservingProgram
- observingID: int
- title: String
- detail: String
+ getObservingID(): int
+ getTitle(): String
+ getDetail(): String
1..*
1
0..*

## ID:1 - Use case description

| **Use Case Name:** Access astronomical data | **ID**: 1 | **Importance Level:** Medium |
|---|---|---|
| **Primary Actor:** Science observer | | **Use Case Type:** Essential |

**Stakeholders and Interests:**
Science observer - wants to collect and manage astronomical data
Astronomer - wants to access collected astronomical data

**Brief Description:**
This use case describes the ability of astronomers and science observers to access collected astronomical data from the system.

**Trigger:** The astronomer wants to know and access astronomical data for analysis.

**Type**: External

**Relationships:**
Association: Astronomer
Include: -
Extend: -
Generalization: User

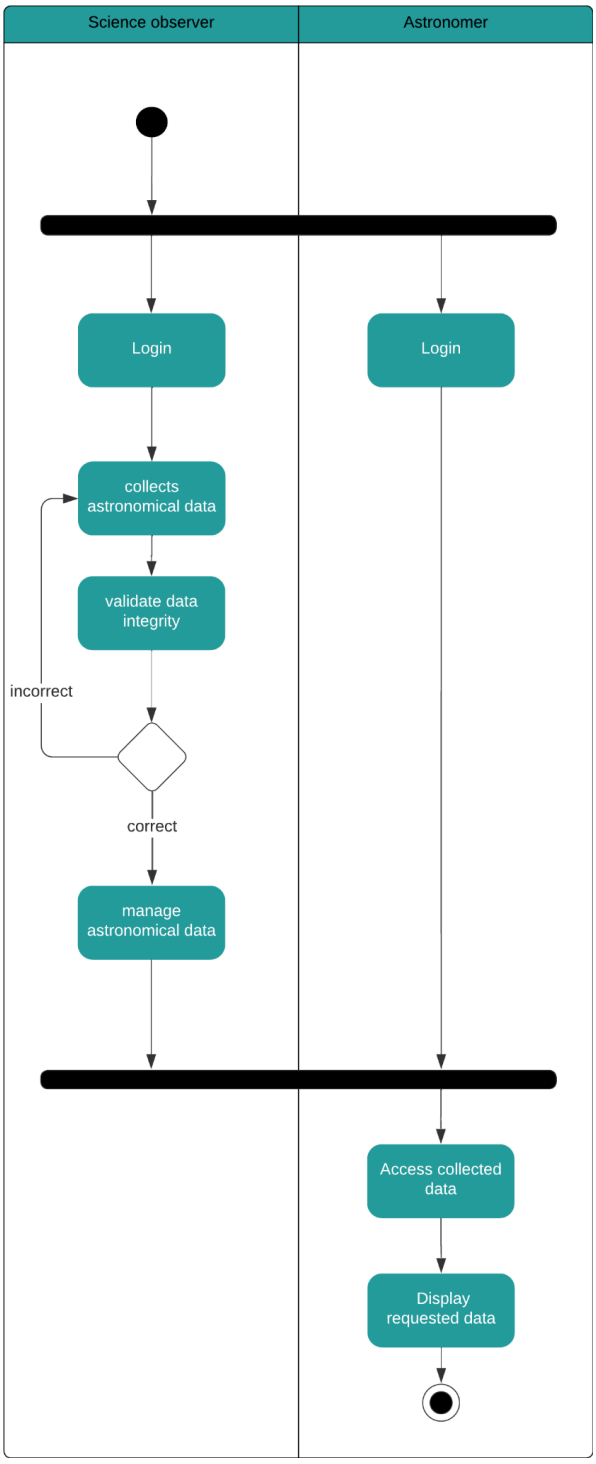
**Normal Flow of Events:**

1. The astronomer or science observer logs into the system.
2. The science observer has access to the star catalog.
3. The science observer collects astronomical data.
4. The science observer validates the integrity of the collected astronomical data.
5. The astronomer accesses collected astronomical data.

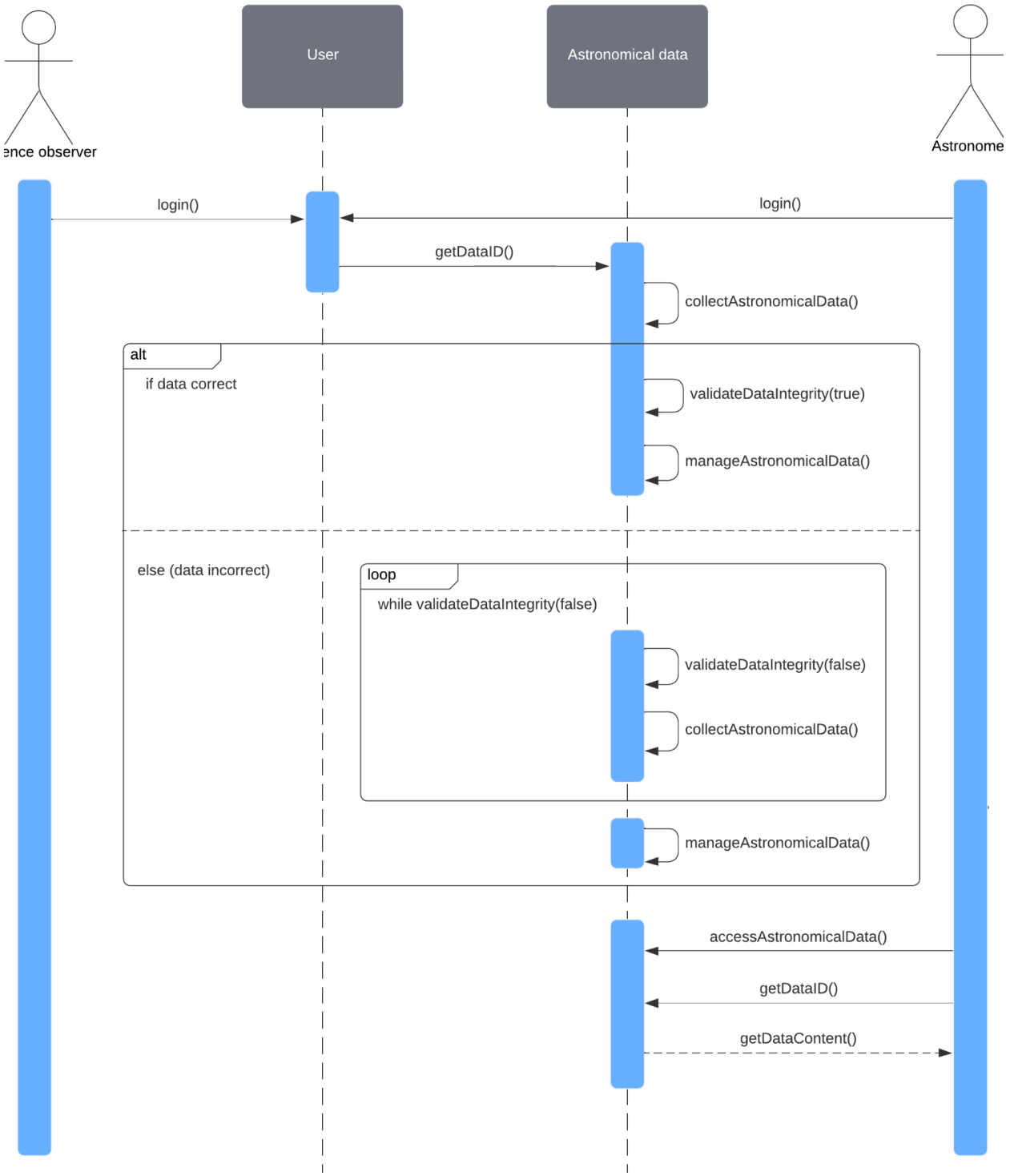
**Subflows:**
S–1: Delete the astronomical data.
1. The science observer wants to delete astronomical data.
2. That astronomical data has been deleted from the system.

**Alternate/Exceptional Flow:**
- The system will notify the user if the requested data is unavailable in the system's database or archives.
- If the system encounters technical issues while retrieving the data, it will notify the user.

## aID:1 - Activity Diagram



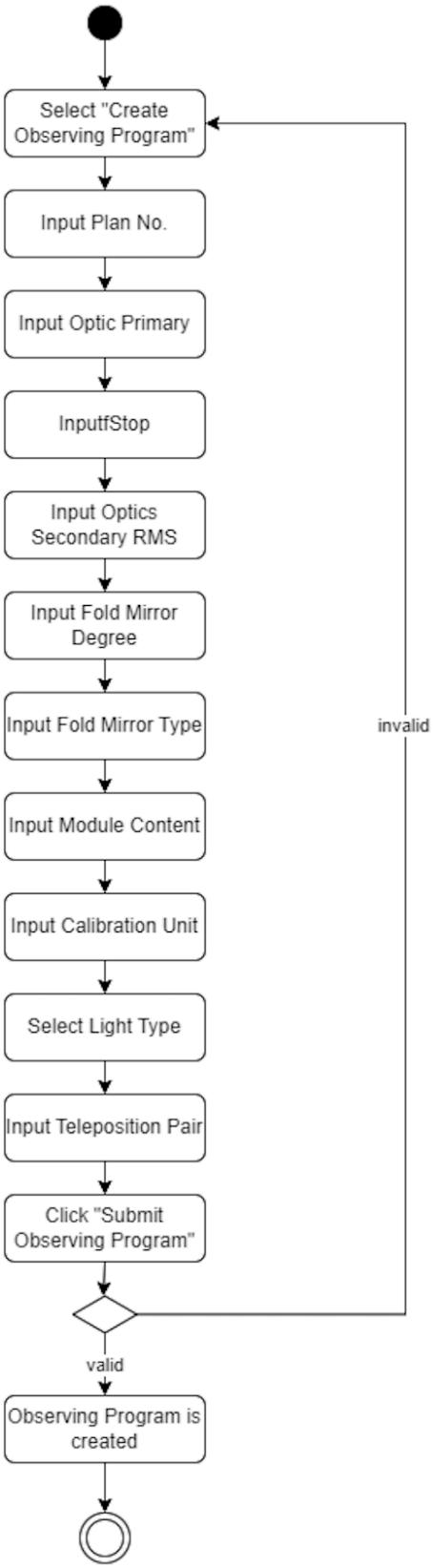
## ID:1 - Sequence Diagram



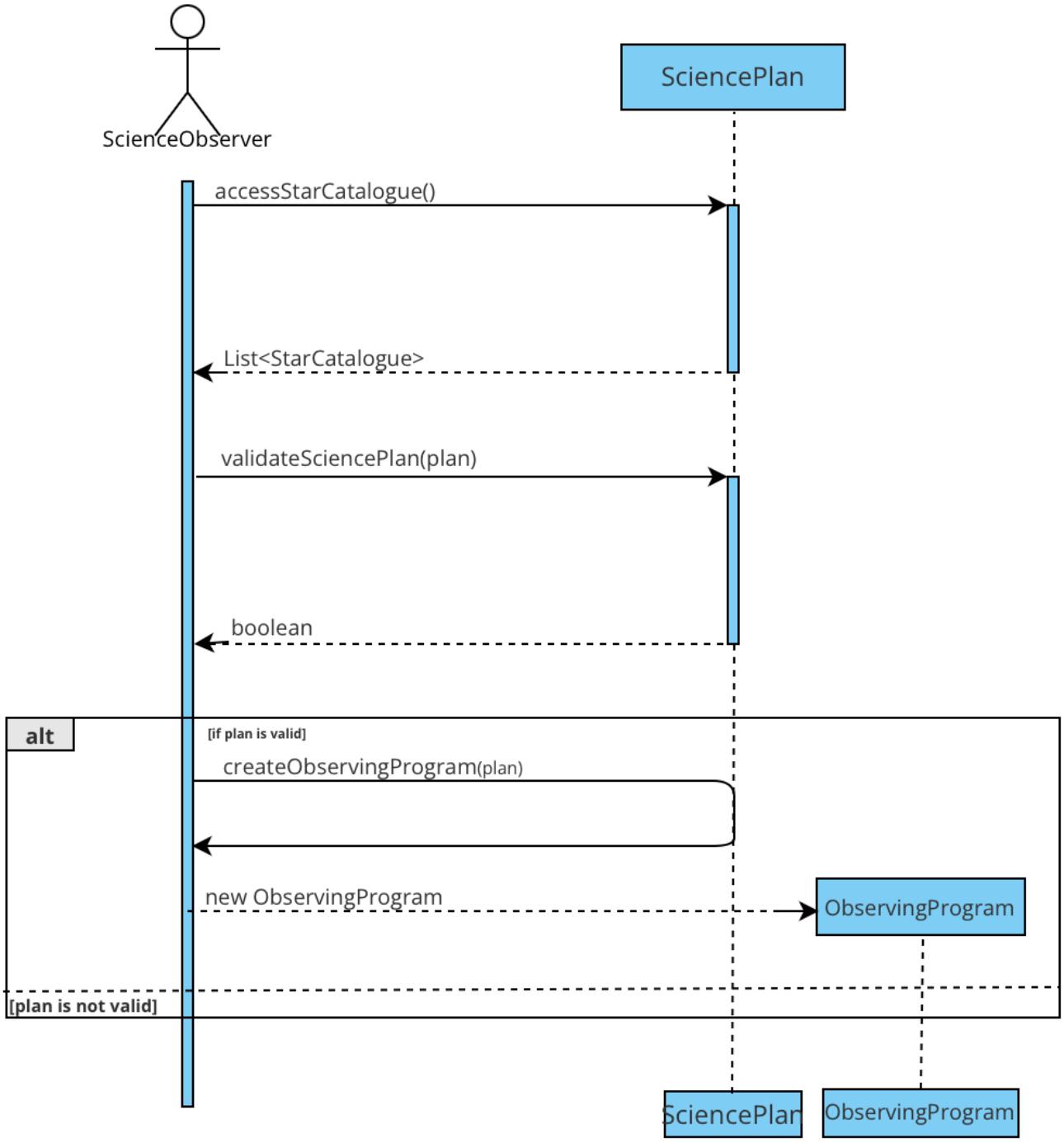
## ID:2 - Use case description

| | | |
|---|---|---|
| **Use Case Name**: Create an observing program | **ID**: 2 | **Importance Level:** High |
| **Primary Actor:** Science observer | | **Use Case Type:** Essential |
| **Brief Description**: The Science Observer creates an observing program based on the Science Plan | | |
| **Trigger:** The astronomer wants to conduct an observing program<br><br>**Type**: External | | |
| **Preconditions:**<br>1. The Science Observer is authenticated by logging in to his/her account<br>2. The Science Plan is created, tested via the virtual telescope, and submitted to the system by the Astronomer<br>3. The Science Plan is validated by the Science Observer | | |
| **Normal Flow of Events:**<br>1. The Science Observer selects the “Create Observing Program” option from the menu<br>2. The system displays a create observing program page, which contains several fields to input and a button to submit the program.<br>3. The Science Observer inputs the details of the observing program including plan no., optic primary, f-Stop, optics secondary RMS, and fold mirror degree.<br>4. The Science Observer selects the fold mirror type option.<br>5. The Science Observer inputs the module content and calibration unit.<br>6. The Science Observer selects the light type of option.<br>7. The Science Observer inputs the teleportation pair.<br>8. The Science Observer clicks the submit button to create the observing program.<br>9. The system displays the state of the observing program (valid/invalid). If it’s invalid, return to step 3. | | |
| **Postcondition:**<br>1. The details and configurations of the observing program are stored in the observing program datastore | | |

## ID:2 - Activity Diagram



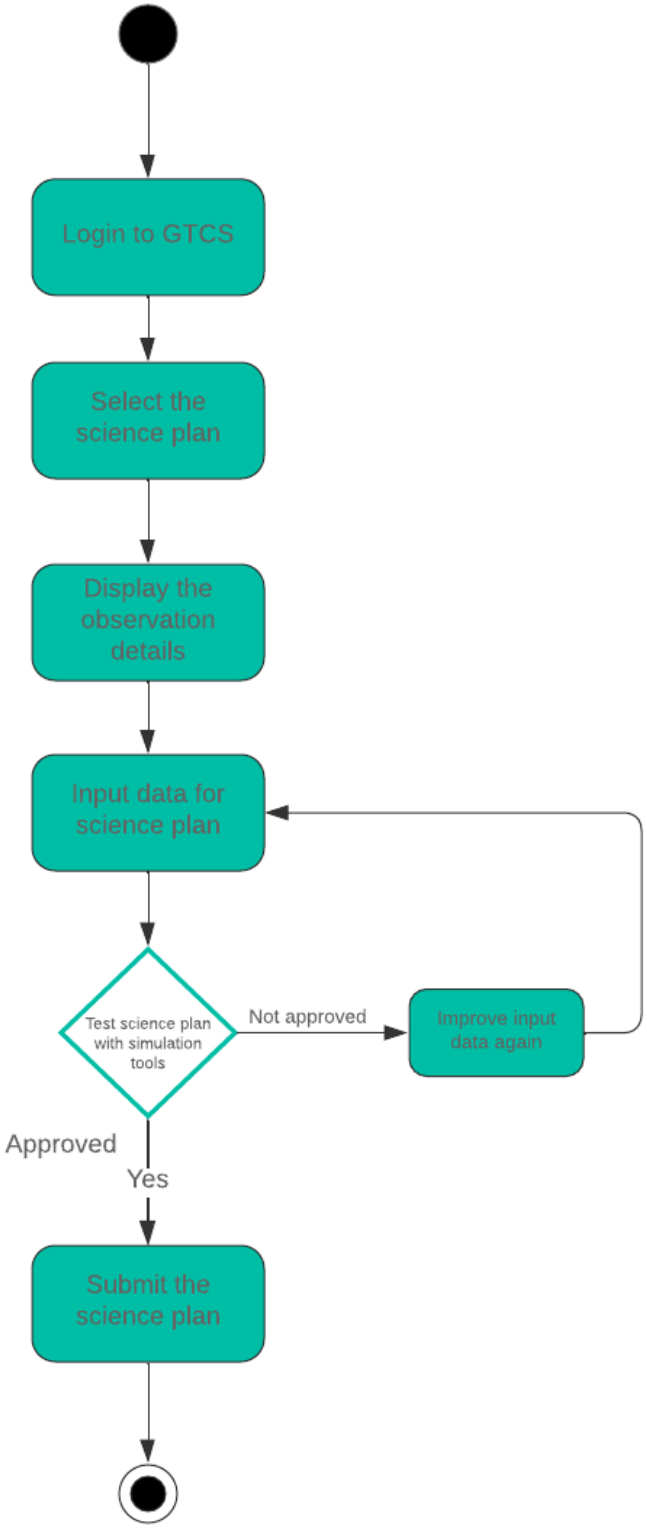
**ID:2 - Sequence Diagram**



## ID:5 - Use case description

| | | |
|---|---|---|
| **Use Case Name:**Create Science plan | **ID**: 5 | **Importance Level**: Hight |
| **Primary Actor:** Astronomer | | **Use Case Type:** Functional |
| **Stakeholders and Interests:**<br>Science observer:Science observation and the collect of data<br>Astronomer:Scientific observation | | |
| **Brief Description**: This use case allows astronomers to create a plan for Science observation. | | |
| **Trigger:** An astronomer plans to execute an observation.<br>Type: initial processing | | |
| **Relationships:**<br>Association:Science observer,Astronomer<br>Include:Test a science plan<br>Extend: for real time adjustment<br>Generalization: - | | |
| **Normal Flow of Events:**<br>1. Astronomers log in the Gemini Telescope Control system(GTCS).<br>2. Astronomers select the science plan.<br>3. The system shows detail about observation times, required instruments.<br>4. Astronomers input data such as<br>5. Astronomers submit scientific plans. | | |
| **Subflows:**<br>Astronomers test the plan in GTCS's simulation capabilities to verify the viability of the science planning. | | |
| **Alternate/Exceptional Flow:**<br>If the science plan is not approved during the validation process, the astronomer will resubmit the plan. | | |

## ID:5 - Activity Diagram



## ID:5 - Sequence Diagram