DegreeWorks System Design Document

CSCE247: Software Engineering

Pookie Bears

TABLE OF CONTENTS

Introduction and Design Overview

System Overview

This design document provides a comprehensive overview of the system's architecture and functionality to record and display all of the classes, data, and attributes of the degreework application. This management system helps keep track of the users' information, and listings of majors and classes and provides search and filtering functionalities for students and advisors to manage their accounts and find relevant courses based on their current major.

References

Requirements Document

Environment Overview

The application will be compiled and executed using Java, HTML, etc. The program will be run on the server environment, processing users' requests, and managing data storage. The client-side execution involves rendering user interfaces and handling user interactions through web browsers.

Data Storage

Our system will actively use array lists containing information stored in JSON files. These files are accessed and updated to match the logged user through JSON file readers when the main method of the system is run.

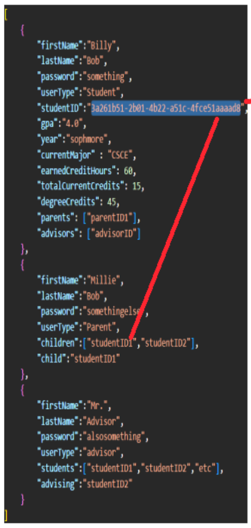
The application relies on three core JSON files:

- Class: Stores class details that do not relate to other branches besides roadmap, like class name, number, and subject code independent of other objects.
- Users: Contains information about all users of the system, such as roles, permissions, and progress data.

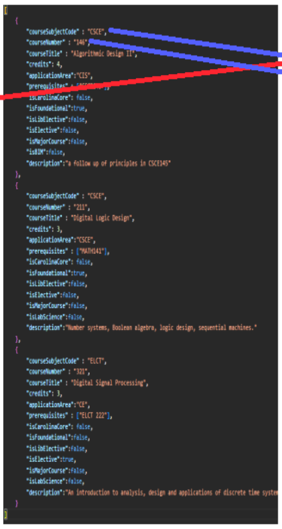
- Roadmap: stores the necessary credit pathways required for each section of the degree, like Carolina Cores and computing fundamentals.

There are also two additional files, a major connector and student courses, that connect information between the above-mentioned files. The student Course Info connects each student's credentials to the course, establishing the roadmap if the student has completed the course, is currently completing it, or has yet to start it. It also stores the completed grade for each one if completed, and establishes if it's transferred from another college or high school. The student connector allows for the possibility of viewing the student's class credentials for other majors than their current one, in case they decide to transfer. The roadmap stores the necessary credits required for each section of the degree, like Carolina Cores and degree fundamentals.

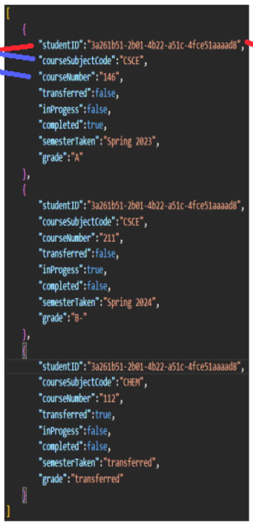
Example JSON Files (From left to right: **Student**, **Course**, **studentCourseInfo**, **studentConnector**, **Roadmap**):



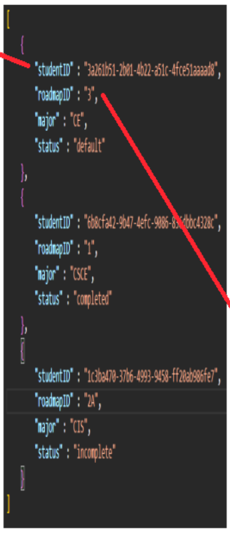
```
[
    {
        "firstName":"Billy",
        "lastName":"Bob",
        "password":"something",
        "userType":"Student",
        "studentID":"3a261b51-2b01-4b22-a51c-4fce51aaaad8",
        "gpa":"4.0",
        "year":"sophmore",
        "currentMajor" : "CSCE",
        "earnedCreditHours": 60,
        "totalCurrentCredits": 15,
        "degreeCredits": 45,
        "parents": ["parentID1"],
        "advisors": ["advisorID"]
    },
    {
        "firstName":"Millie",
        "lastName":"Bob",
        "password":"somethingelse",
        "userType":"Parent",
        "children":["studentID1","studentID2"],
        "child":"studentID1"
    },
    {
        "firstName":"Mr.",
        "lastName":"Advisor",
        "password":"alsosomething",
        "userType":"advisor",
        "students":["studentID1","studentID2","etc"],
        "advising":"studentID2"
    }
]
```



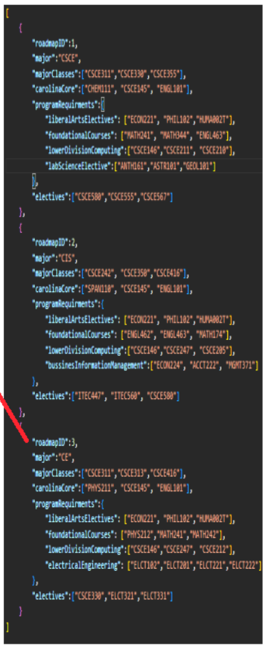
```
[
    {
        "courseSubjectCode" : "CSCE",
        "courseNumber" : "146",
        "courseTitle" : "Algorithmic Design II",
        "credits": 4,
        "applicationArea":"CIS",
        "prerequisites" : [illegible],
        "isCarolinaCore": false,
        "isFoundational":true,
        "isLibElective":false,
        "isElective":false,
        "isMajorCourse":false,
        "isBIM":false,
        "description":"a follow up of principles in CSCE145"
    },
    {
        "courseSubjectCode" : "CSCE",
        "courseNumber" : "211",
        "courseTitle" : "Digital Logic Design",
        "credits": 3,
        "applicationArea":"CSCE",
        "prerequisites" : ["MATH141"],
        "isCarolinaCore": false,
        "isFoundational":true,
        "isLibElective":false,
        "isElective":false,
        "isMajorCourse":false,
        "isLabScience":false,
        "description":"Number systems, Boolean algebra, logic design, sequential machines."
    },
    {
        "courseSubjectCode" : "ELCT",
        "courseNumber" : "321",
        "courseTitle" : "Digital Signal Processing",
        "credits": 3,
        "applicationArea":"CE",
        "prerequisites" : ["ELCT 222"],
        "isCarolinaCore": false,
        "isFoundational":false,
        "isLibElective":false,
        "isElective":true,
        "isMajorCourse":false,
        "isLabScience":false,
        "description":"An introduction to analysis, design and applications of discrete time syste
    }
]
```



```
[
    {
        "studentID":"3a261b51-2b01-4b22-a51c-4fce51aaaad8",
        "courseSubjectCode":"CSCE",
        "courseNumber":"146",
        "transferred":false,
        "inProgess":false,
        "completed":true,
        "semesterTaken":"Spring 2023",
        "grade":"A"
    },
    {
        "studentID":"3a261b51-2b01-4b22-a51c-4fce51aaaad8",
        "courseSubjectCode":"CSCE",
        "courseNumber":"211",
        "transferred":false,
        "inProgess":true,
        "completed":false,
        "semesterTaken":"Spring 2024",
        "grade":"B-"
    },
    {
        "studentID":"3a261b51-2b01-4b22-a51c-4fce51aaaad8",
        "courseSubjectCode":"CHEM",
        "courseNumber":"112",
        "transferred":true,
        "inProgess":false,
        "completed":false,
        "semesterTaken":"transferred",
        "grade":"transferred"
    }
]
```

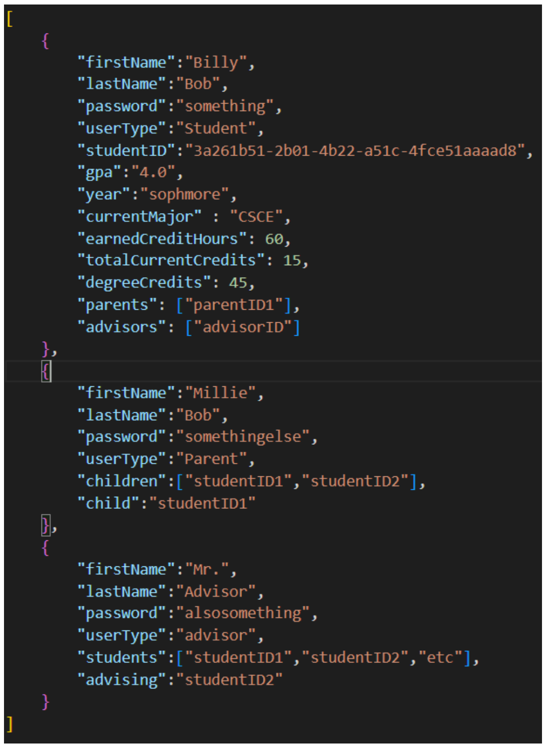


```
[
    {
        "studentID" : "3a261b51-2b01-4b22-a51c-4fce51aaaad8",
        "roadmapID" : "3",
        "major" : "CE",
        "status" : "default"
    },
    {
        "studentID" : "6b8cfa42-9b47-4efc-9086-83[illegible]dbbc4328c",
        "roadmapID" : "1",
        "major" : "CSCE",
        "status" : "completed"
    },
    {
        "studentID" : "1c3ba470-37b6-4993-9458-ff20ab986fe7",
        "roadmapID" : "2A",
        "major" : "CIS",
        "status" : "incomplete"
    }
]
```



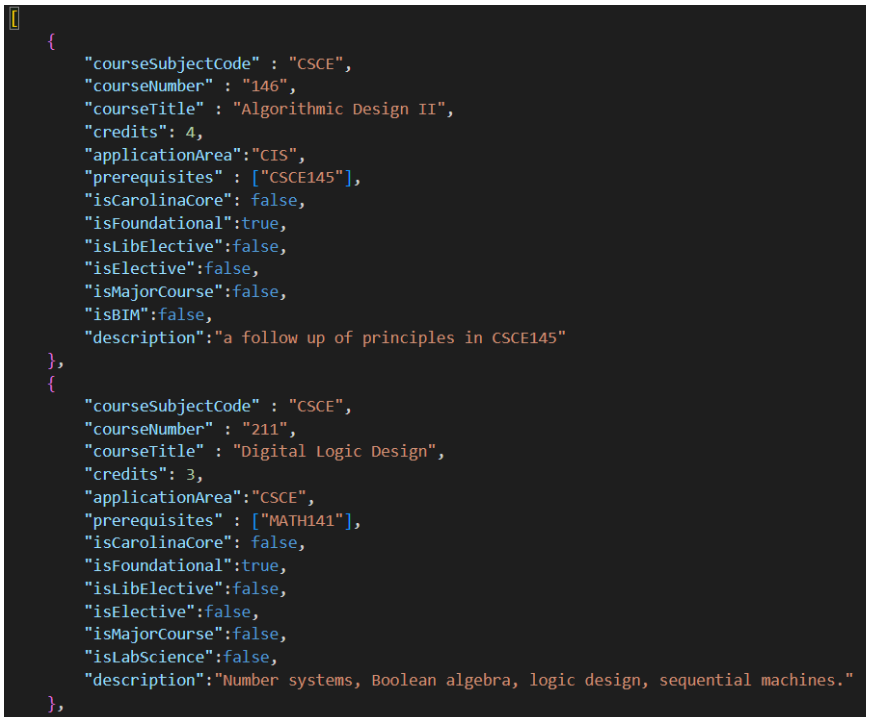
```
[
    {
        "roadmapID":1,
        "major":"CSCE",
        "majorClasses":["CSCE311","CSCE330","CSCE355"],
        "carolinaCore":["CHEM111", "CSCE145", "ENGL101"],
        "programRequirments":{
            "liberalArtsElectives": ["ECON221", "PHIL102","HUMA002T"],
            "foundationalCourses": ["MATH241", "MATH344", "ENGL463"],
            "lowerDivisionComputing":["CSCE146","CSCE211", "CSCE210"],
            "labScienceElective":["ANTH161","ASTR101","GEOL101"]
        },
        "electives":["CSCE580","CSCE555","CSCE567"]
    },
    {
        "roadmapID":2,
        "major":"CIS",
        "majorClasses":["CSCE242", "CSCE350","CSCE416"],
        "carolinaCore":["SPAN110", "CSCE145", "ENGL101"],
        "programRequirments":{
            "liberalArtsElectives": ["ECON221", "PHIL102","HUMA002T"],
            "foundationalCourses": ["ENGL462", "ENGL463", "MATH174"],
            "lowerDivisionComputing":["CSCE146","CSCE247", "CSCE205"],
            "bussinesInformationManagement":["ECON224", "ACCT222", "MGMT371"]
        },
        "electives":["ITEC447", "ITEC560", "CSCE580"]
    },
    {
        "roadmapID":3,
        "major":"CE",
        "majorClasses":["CSCE311","CSCE313","CSCE416"],
        "carolinaCore":["PHYS211", "CSCE145", "ENGL101"],
        "programRequirments":{
            "liberalArtsElectives": ["ECON221", "PHIL102","HUMA002T"],
            "foundationalCourses": ["PHYS212","MATH241","MATH242"],
            "lowerDivisionComputing":["CSCE146","CSCE247", "CSCE212"],
            "electricalEngineering": ["ELCT102","ELCT201","ELCT221","ELCT222"]
        },
        "electives":["CSCE330","ELCT321","ELCT331"]
    }
]
```

users.json:



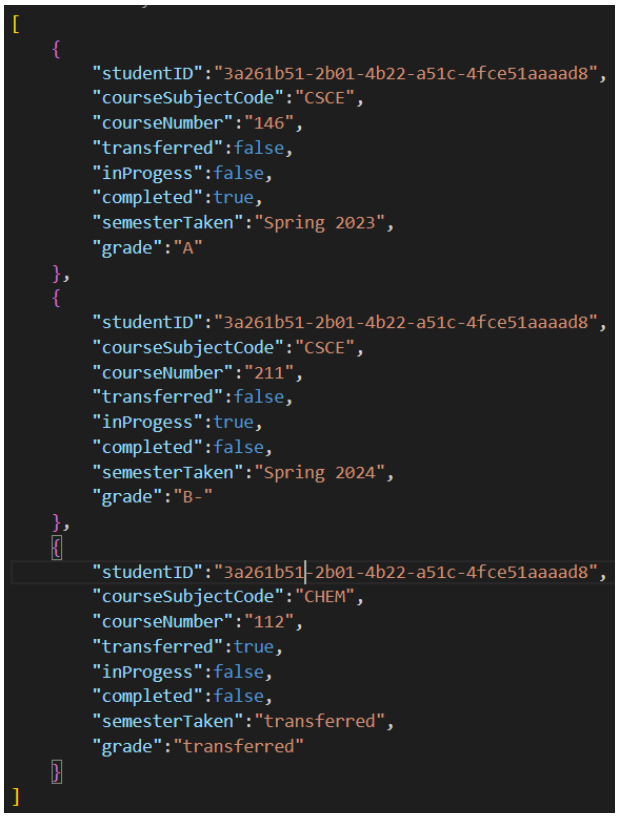
```
[
    {
        "firstName":"Billy",
        "lastName":"Bob",
        "password":"something",
        "userType":"Student",
        "studentID":"3a261b51-2b01-4b22-a51c-4fce51aaaad8",
        "gpa":"4.0",
        "year":"sophmore",
        "currentMajor" : "CSCE",
        "earnedCreditHours": 60,
        "totalCurrentCredits": 15,
        "degreeCredits": 45,
        "parents": ["parentID1"],
        "advisors": ["advisorID"]
    },
    {
        "firstName":"Millie",
        "lastName":"Bob",
        "password":"somethingelse",
        "userType":"Parent",
        "children":["studentID1","studentID2"],
        "child":"studentID1"
    },
    {
        "firstName":"Mr.",
        "lastName":"Advisor",
        "password":"alsosomething",
        "userType":"advisor",
        "students":["studentID1","studentID2","etc"],
        "advising":"studentID2"
    }
]
```

course.json:

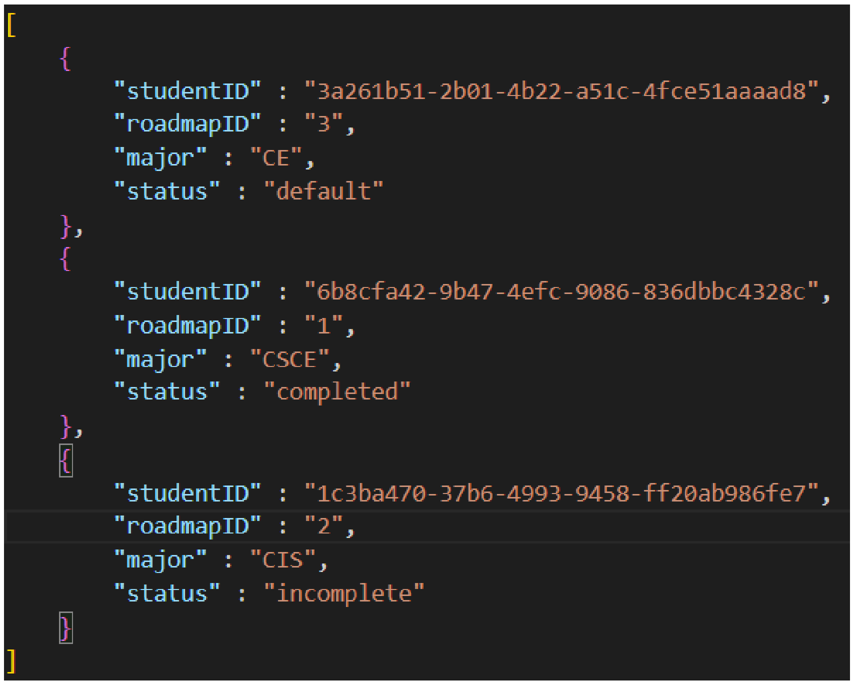


```
[
    {
        "courseSubjectCode" : "CSCE",
        "courseNumber" : "146",
        "courseTitle" : "Algorithmic Design II",
        "credits": 4,
        "applicationArea":"CIS",
        "prerequisites" : ["CSCE145"],
        "isCarolinaCore": false,
        "isFoundational":true,
        "isLibElective":false,
        "isElective":false,
        "isMajorCourse":false,
        "isBIM":false,
        "description":"a follow up of principles in CSCE145"
    },
    {
        "courseSubjectCode" : "CSCE",
        "courseNumber" : "211",
        "courseTitle" : "Digital Logic Design",
        "credits": 3,
        "applicationArea":"CSCE",
        "prerequisites" : ["MATH141"],
        "isCarolinaCore": false,
        "isFoundational":true,
        "isLibElective":false,
        "isElective":false,
        "isMajorCourse":false,
        "isLabScience":false,
        "description":"Number systems, Boolean algebra, logic design, sequential machines."
    },
```

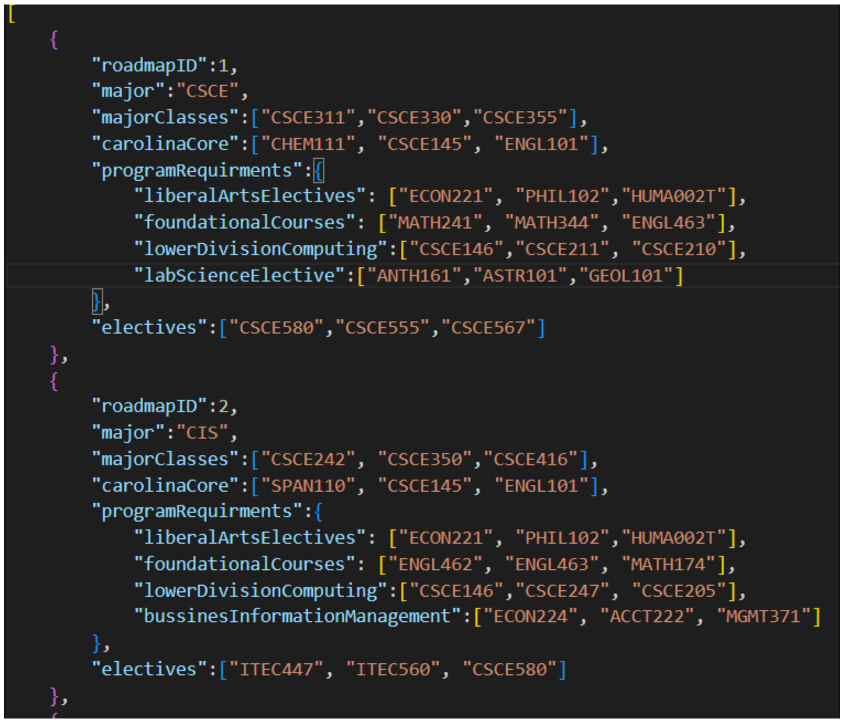
studentCourses.json:


```
[
    {
        "studentID":"3a261b51-2b01-4b22-a51c-4fce51aaaad8",
        "courseSubjectCode":"CSCE",
        "courseNumber":"146",
        "transferred":false,
        "inProgess":false,
        "completed":true,
        "semesterTaken":"Spring 2023",
        "grade":"A"
    },
    {
        "studentID":"3a261b51-2b01-4b22-a51c-4fce51aaaad8",
        "courseSubjectCode":"CSCE",
        "courseNumber":"211",
        "transferred":false,
        "inProgess":true,
        "completed":false,
        "semesterTaken":"Spring 2024",
        "grade":"B-"
    },
    {
        "studentID":"3a261b51-2b01-4b22-a51c-4fce51aaaad8",
        "courseSubjectCode":"CHEM",
        "courseNumber":"112",
        "transferred":true,
        "inProgess":false,
        "completed":false,
        "semesterTaken":"transferred",
        "grade":"transferred"
    }
]
```

majorConnector.json:


```
[
    {
        "studentID" : "3a261b51-2b01-4b22-a51c-4fce51aaaad8",
        "roadmapID" : "3",
        "major" : "CE",
        "status" : "default"
    },
    {
        "studentID" : "6b8cfa42-9b47-4efc-9086-836dbbc4328c",
        "roadmapID" : "1",
        "major" : "CSCE",
        "status" : "completed"
    },
    {
        "studentID" : "1c3ba470-37b6-4993-9458-ff20ab986fe7",
        "roadmapID" : "2",
        "major" : "CIS",
        "status" : "incomplete"
    }
]
```

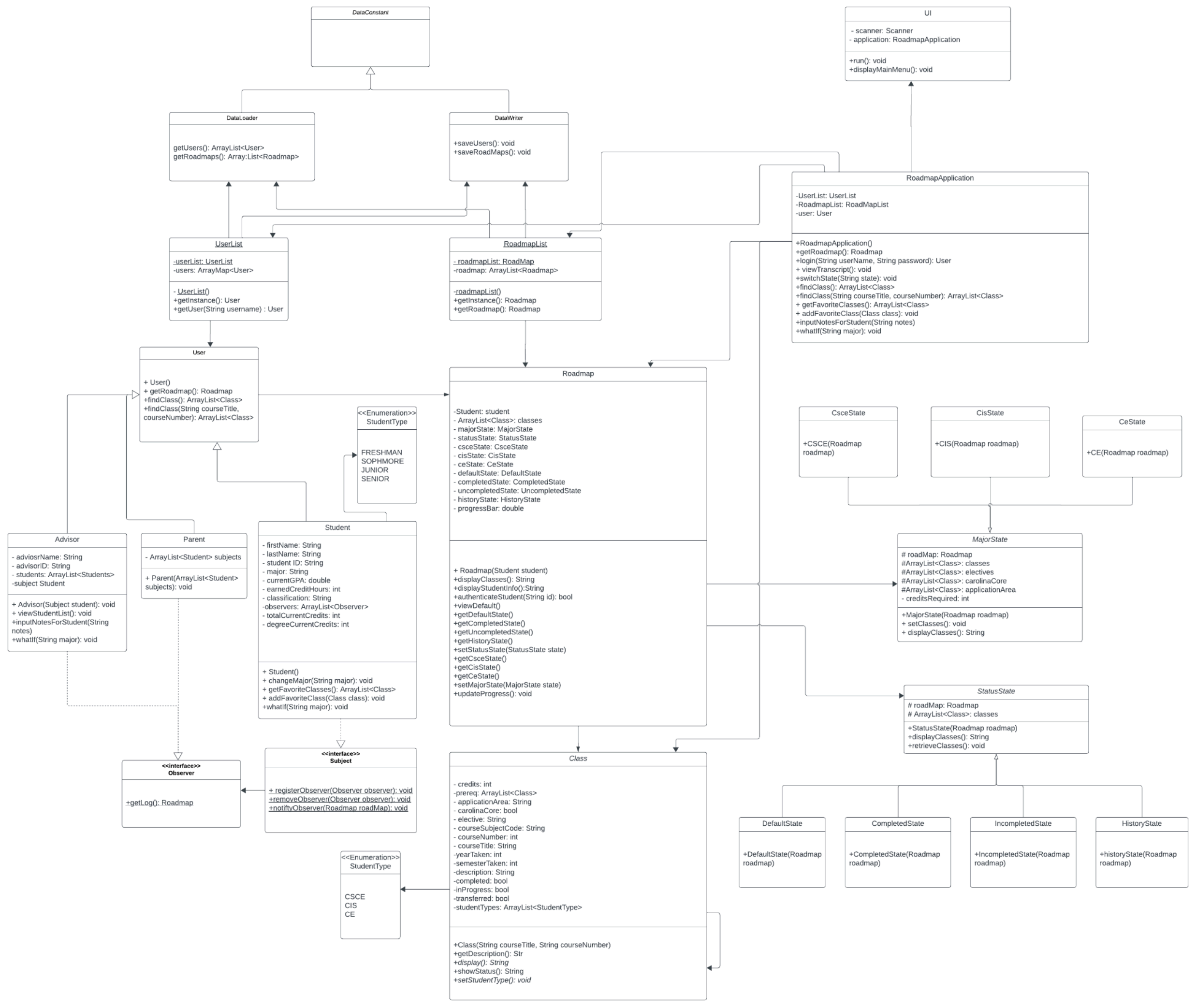
roadmap.json:



```
[
    {
        "roadmapID":1,
        "major":"CSCE",
        "majorClasses":["CSCE311","CSCE330","CSCE355"],
        "carolinaCore":["CHEM111", "CSCE145", "ENGL101"],
        "programRequirments":{
            "liberalArtsElectives": ["ECON221", "PHIL102","HUMA002T"],
            "foundationalCourses": ["MATH241", "MATH344", "ENGL463"],
            "lowerDivisionComputing":["CSCE146","CSCE211", "CSCE210"],
            "labScienceElective":["ANTH161","ASTR101","GEOL101"]
        },
        "electives":["CSCE580","CSCE555","CSCE567"]
    },
    {
        "roadmapID":2,
        "major":"CIS",
        "majorClasses":["CSCE242", "CSCE350","CSCE416"],
        "carolinaCore":["SPAN110", "CSCE145", "ENGL101"],
        "programRequirments":{
            "liberalArtsElectives": ["ECON221", "PHIL102","HUMA002T"],
            "foundationalCourses": ["ENGL462", "ENGL463", "MATH174"],
            "lowerDivisionComputing":["CSCE146","CSCE247", "CSCE205"],
            "bussinesInformationManagement":["ECON224", "ACCT222", "MGMT371"]
        },
        "electives":["ITEC447", "ITEC560", "CSCE580"]
    },
```

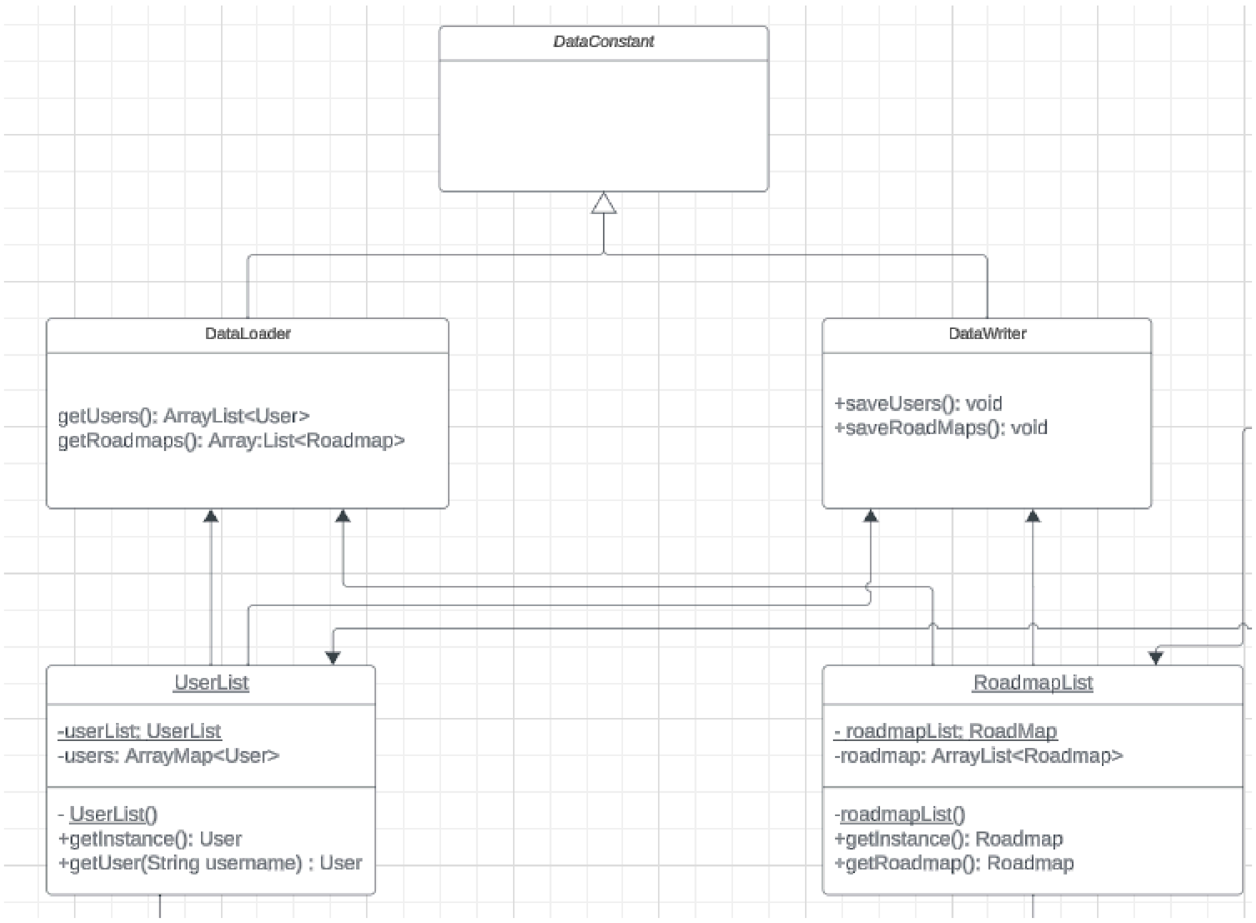
Class Diagram - Structural Design

UML Overview



DataConstant
DataLoader
getUsers(): ArrayList<User>
getRoadmaps(): Array:List<Roadmap>
DataWriter
+saveUsers(): void
+saveRoadMaps(): void
UI
- scanner: Scanner
- application: RoadmapApplication
+run(): void
+displayMainMenu(): void
UserList
-userList: UserList
-users: ArrayMap<User>
- UserList()
+getInstance(): User
+getUser(String username) : User
RoadmapList
- roadmapList: RoadMap
-roadmap: ArrayList<Roadmap>
-roadmapList()
+getInstance(): Roadmap
+getRoadmap(): Roadmap
RoadmapApplication
-UserList: UserList
-RoadmapList: RoadMapList
-user: User
+RoadmapApplication()
+getRoadmap(): Roadmap
+login(String userName, String password): User
+ viewTranscript(): void
+switchState(String state): void
+findClass(): ArrayList<Class>
+findClass(String courseTitle, courseNumber): ArrayList<Class>
+ getFavoriteClasses(): ArrayList<Class>
+ addFavoriteClass(Class class): void
+inputNotesForStudent(String notes)
+whatIf(String major): void
User
+ User()
+ getRoadmap(): Roadmap
+findClass(): ArrayList<Class>
+findClass(String courseTitle, courseNumber): ArrayList<Class>
<<Enumeration>> StudentType
FRESHMAN
SOPHMORE
JUNIOR
SENIOR
Roadmap
-Student: student
- ArrayList<Class>: classes
- majorState: MajorState
- statusState: StatusState
- csceState: CsceState
- cisState: CisState
- ceState: CeState
- defaultState: DefaultState
- completedState: CompletedState
- uncompletedState: UncompletedState
- historyState: HistoryState
- progressBar: double
+ Roadmap(Student student)
+displayClasses(): String
+displayStudentInfo():String
+authenticateStudent(String id): bool
+viewDefault()
+getDefaultState()
+getCompletedState()
+getUncompletedState()
+getHistoryState()
+setStatusState(StatusState state)
+getCsceState()
+getCisState()
+getCeState()
+setMajorState(MajorState state)
+updateProgress(): void
CsceState
+CSCE(Roadmap roadmap)
CisState
+CIS(Roadmap roadmap)
CeState
+CE(Roadmap roadmap)
MajorState
# roadMap: Roadmap
#ArrayList<Class>: classes
#ArrayList<Class>: electives
#ArrayList<Class>: carolinaCore
#ArrayList<Class>: applicationArea
- creditsRequired: int
+MajorState(Roadmap roadmap)
+ setClasses(): void
+ displayClasses(): String
Advisor
- adviosrName: String
- advisorID: String
- students: ArrayList<Students>
-subject Student
+ Advisor(Subject student): void
+ viewStudentList(): void
+inputNotesForStudent(String notes)
+whatIf(String major): void
Parent
- ArrayList<Student> subjects
+ Parent(ArrayList<Student> subjects): void
Student
- firstName: String
- lastName: String
- student ID: String
- major: String
- currentGPA: double
- earnedCreditHours: int
- classification: String
-observers: ArrayList<Observer>
- totalCurrentCredits: int
- degreeCurrentCredits: int
+ Student()
+ changeMajor(String major): void
+ getFavoriteClasses(): ArrayList<Class>
+ addFavoriteClass(Class class): void
+whatIf(String major): void
StatusState
# roadMap: Roadmap
# ArrayList<Class>: classes
+StatusState(Roadmap roadmap)
+displayClasses(): String
+retrieveClasses(): void
<<interface>> Observer
+getLog(): Roadmap
<<interface>> Subject
+ registerObserver(Observer observer): void
+removeObserver(Observer observer): void
+notiftyObserver(Roadmap roadMap): void
Class
- credits: int
-prereq: ArrayList<Class>
- applicationArea: String
- carolinaCore: bool
- elective: String
- courseSubjectCode: String
- courseNumber: int
- courseTitle: String
-yearTaken: int
-semesterTaken: int
-description: String
-completed: bool
-inProgress: bool
-transferred: bool
-studentTypes: ArrayList<StudentType>
+Class(String courseTitle, String courseNumber)
+getDescription(): Str
+display(): String
+showStatus(): String
+setStudentType(): void
<<Enumeration>> StudentType
CSCE
CIS
CE
DefaultState
+DefaultState(Roadmap roadmap)
CompletedState
+CompletedState(Roadmap roadmap)
IncompletedState
+IncompletedState(Roadmap roadmap)
HistoryState
+historyState(Roadmap roadmap)

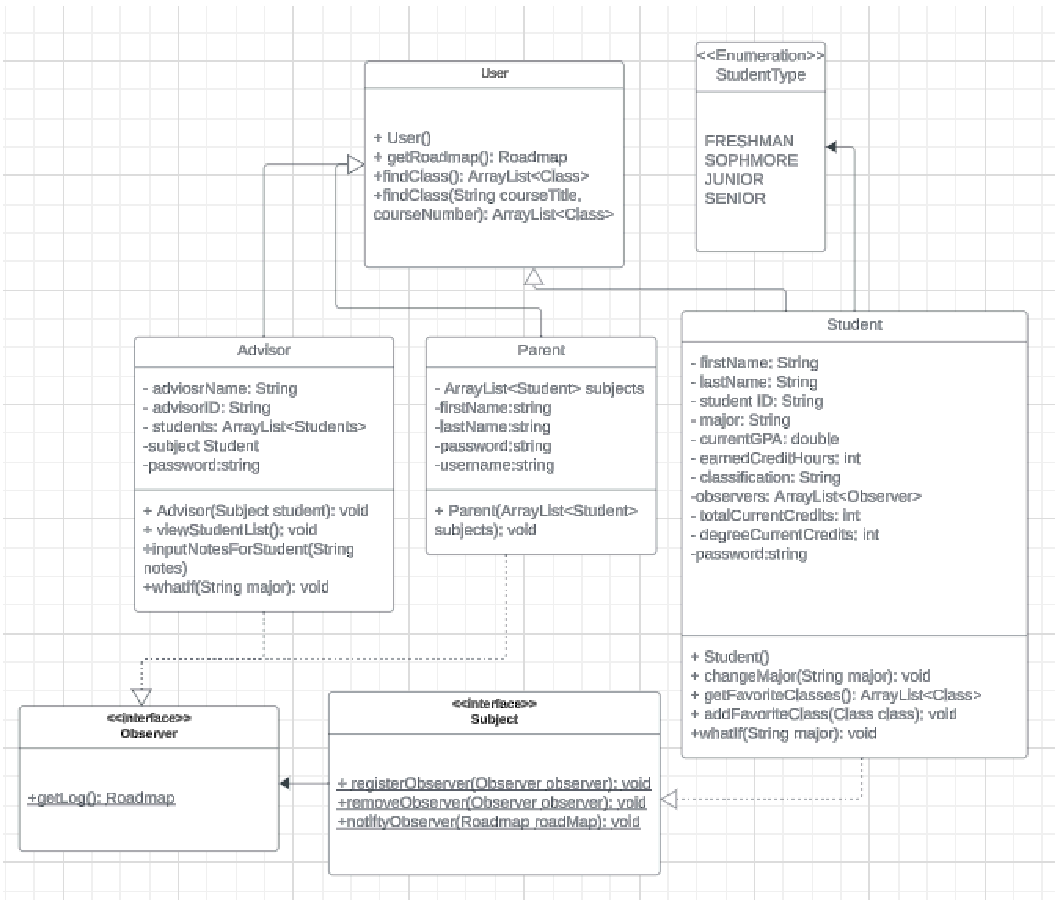
DataConstant / Data Loader / Data Writer / UserList / RoadmapList



Description:

This is the core component of the data management for our application. The DataConstant stores static data and acts as a central vessel for our fixed values and information. The DataLoader and DataWriter take the data from DataConstant. They read and load the data and allow us to write data across our application. The UserList and RoadmapList collect user and roadmap objects and their data respectfully associated with them. These lists are used to manage, manipulate, and access data throughout our application.

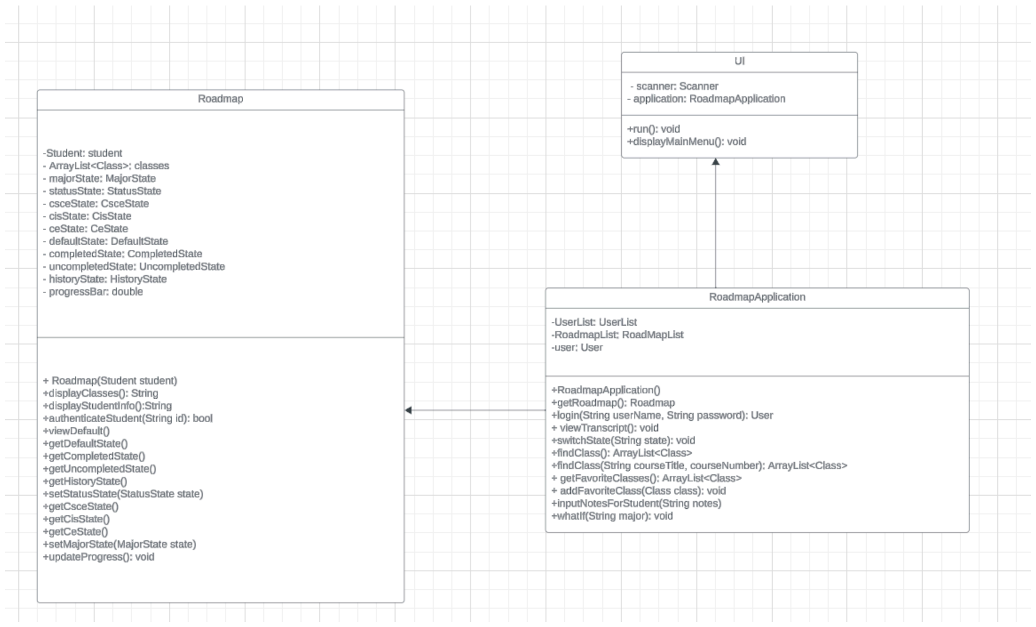
<u>User / Student / Advisor / Parent / Interface Subject / Interface Observer</u>



User
+ User()
+ getRoadmap(): Roadmap
+findClass(): ArrayList<Class>
+findClass(String courseTitle, courseNumber): ArrayList<Class>
<<Enumeration>>
StudentType
FRESHMAN
SOPHMORE
JUNIOR
SENIOR
Advisor
- adviosrName: String
- advisorID: String
- students: ArrayList<Students>
-subject Student
-password:string
+ Advisor(Subject student): void
+ viewStudentList(): void
+inputNotesForStudent(String notes)
+whatIf(String major): void
Parent
- ArrayList<Student> subjects
-firstName:string
-lastName:string
-password:string
-username:string
+ Parent(ArrayList<Student> subjects): void
Student
- firstName: String
- lastName: String
- student ID: String
- major: String
- currentGPA: double
- earnedCreditHours: int
- classification: String
-observers: ArrayList<Observer>
- totalCurrentCredits: int
- degreeCurrentCredits: int
-password:string
+ Student()
+ changeMajor(String major): void
+ getFavoriteClasses(): ArrayList<Class>
+ addFavoriteClass(Class class): void
+whatIf(String major): void
<<interface>>
Observer
+getLog(): Roadmap
<<interface>>
Subject
+ registerObserver(Observer observer): void
+removeObserver(Observer observer): void
+notifityObserver(Roadmap roadMap): void

Description: This flow component of our application represents various user roles and interaction mechanisms within the whole system. The User is our core user class that defines user information and basic functionalities like getting the roadmap and profile management. It serves as the foundation for specific roles. The Student, Advisor, and Parent all inherit the base User class. The Student class adds attributes and operations specific to students, such as enrolled courses, grades, and progress tracking. The Parent and Advisor classes add observer-specific operations, such as reviewing student profiles. The Interface Subject manages observers and notifies them about changes. The Interface Object defines the contract for objects interested in receiving updates from subjects. Overall, this portion of the application applies aggregation-styled flow for our User and the specific different user types

Roadmap / RoadmapApplication / UI

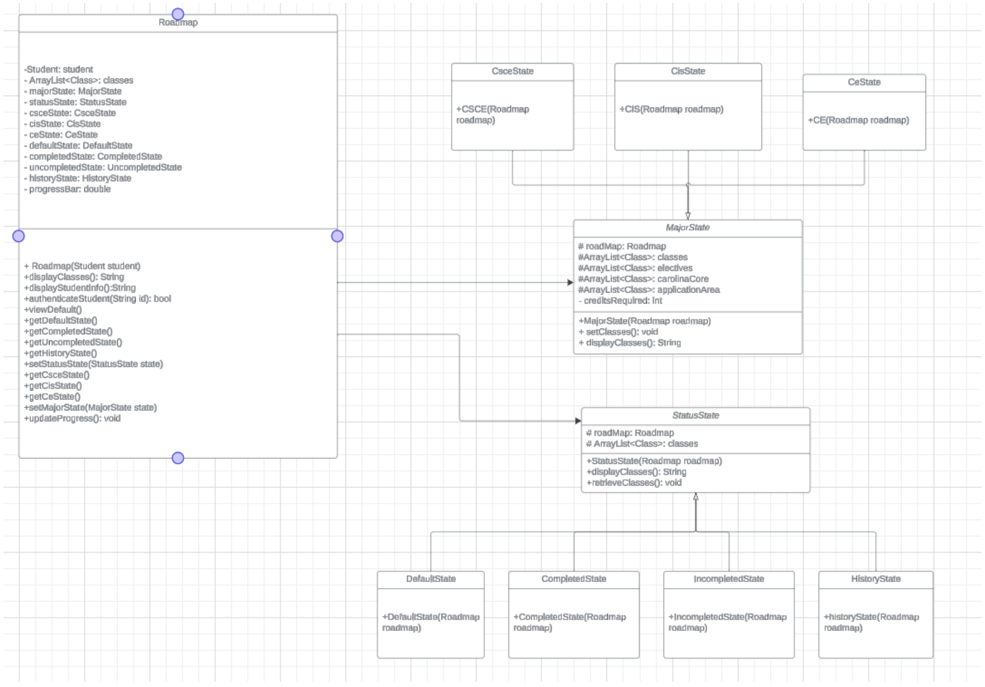


UI
- scanner: Scanner
- application: RoadmapApplication
+run(): void
+displayMainMenu(): void
Roadmap
-Student: student
- ArrayList<Class>: classes
- majorState: MajorState
- statusState: StatusState
- csceState: CsceState
- cisState: CisState
- ceState: CeState
- defaultState: DefaultState
- completedState: CompletedState
- uncompletedState: UncompletedState
- historyState: HistoryState
- progressBar: double
+ Roadmap(Student student)
+displayClasses(): String
+displayStudentInfo():String
+authenticateStudent(String id): bool
+viewDefault()
+getDefaultState()
+getCompletedState()
+getUncompletedState()
+getHistoryState()
+setStatusState(StatusState state)
+getCsceState()
+getCisState()
+getCeState()
+setMajorState(MajorState state)
+updateProgress(): void
RoadmapApplication
-UserList: UserList
-RoadmapList: RoadMapList
-user: User
+RoadmapApplication()
+getRoadmap(): Roadmap
+login(String userName, String password): User
+ viewTranscript(): void
+switchState(String state): void
+findClass(): ArrayList<Class>
+findClass(String courseTitle, courseNumber): ArrayList<Class>
+ getFavoriteClasses(): ArrayList<Class>
+ addFavoriteClass(Class class): void
+inputNotesForStudent(String notes)
+whatIf(String major): void

Description:

This component of the system creates the user interface that determines each sequence of actions for the backend. After the user logs into the system or creates new login information, the user interface displays the options for the different uses of the roadmap, allowing the user to access different forms of information retaining to the roadmap, or add new information for other users to view.

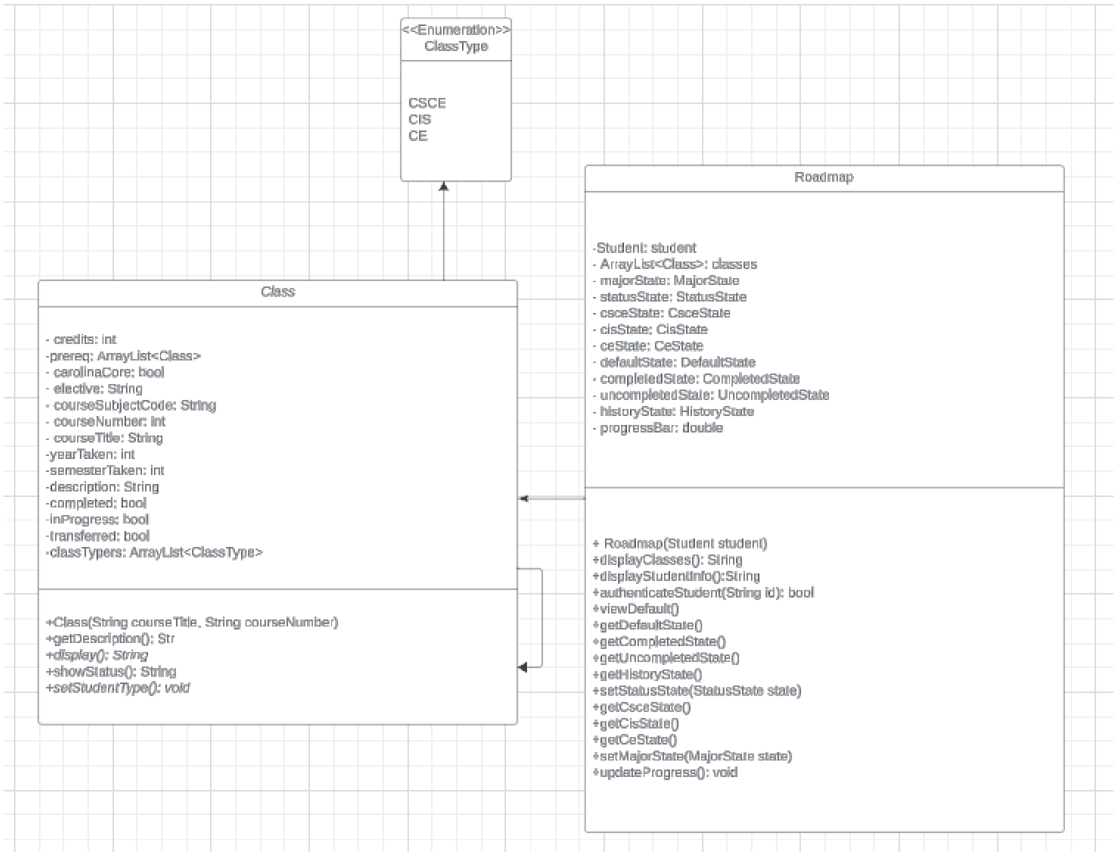
<u>Roadmap / Major States (CSCE, CIS, CE) / Status States (Default, Completed, Incomplete, History)</u>


Roadmap
-Student: student
- ArrayList<Class>: classes
- majorState: MajorState
- statusState: StatusState
- csceState: CsceState
- cisState: CisState
- ceState: CeState
- defaultState: DefaultState
- completedState: CompletedState
- uncompletedState: UncompletedState
- historyState: HistoryState
- progressBar: double
+ Roadmap(Student student)
+displayClasses(): String
+displayStudentInfo():String
+authenticateStudent(String id): bool
+viewDefault()
+getDefaultState()
+getCompletedState()
+getUncompletedState()
+getHistoryState()
+setStatusState(StatusState state)
+getCsceState()
+getCisState()
+getCeState()
+setMajorState(MajorState state)
+updateProgress(): void
CsceState
+CSCE(Roadmap roadmap)
CisState
+CIS(Roadmap roadmap)
CeState
+CE(Roadmap roadmap)
MajorState
# roadMap: Roadmap
#ArrayList<Class>: classes
#ArrayList<Class>: electives
#ArrayList<Class>: carolinaCore
#ArrayList<Class>: applicationArea
- creditsRequired: int
+MajorState(Roadmap roadmap)
+ setClasses(): void
+ displayClasses(): String
StatusState
# roadMap: Roadmap
# ArrayList<Class>: classes
+StatusState(Roadmap roadmap)
+displayClasses(): String
+retrieveClasses(): void
DefaultState
+DefaultState(Roadmap roadmap)
CompletedState
+CompletedState(Roadmap roadmap)
IncompletedState
+IncompletedState(Roadmap roadmap)
HistoryState
+historyState(Roadmap roadmap)

Description:

The state design pattern is an essential functionality of the road map's usage. Having two states of the roadmap allows the user to view each major's list of required classes alongside the completion status of each major. The major state determines which JSON object needs to be accessed for display, while the status state determines what information from that object is shown to the user. The roadmap can switch between CSCE, CIS, and CE, alongside the default status of all classes, only the completed classes, only the incomplete classes, or the history of the previous roadmap.

Roadmap/Class



Description:

This is the functional component of the relationship between class and roadmap. The roadmap stores array lists of class objects for quick access and display of required sections of the degree. It essentially acts as a separate storage/organizational function for the classes based on the current state of the roadmap display. Each class also has a relationship with itself, as some classes require other classes to be taken. Having an array list of classes within each class eases the workload of the roadmap class.

Class Descriptions

Classes

DataConstant:

Purpose: Stores fixed data values that are used throughout the application. These values are typically configuration settings, constants, or other data that don't change frequently.

Placement in UML: The data values are placed outside the main class structure, potentially near the system configuration section. This represents the start of the data flow.

Data Loader:

Purpose: Responsible for loading data from external sources into the application. This can involve reading from databases, files, APIs, or other systems.

Placement in UML: Placement is associated with the specific data source it interacts with. In this case, DataConstant.

Data Writer:

Purpose: Writing data from the application to external destinations. This can involve saving data to databases, files, or other systems.

Placement in UML: Placement is associated with the specific data source it interacts with. In this case, DataConstant.

UserList:

Purpose: Represents a collection of user objects or their associated data. This list can be used to manage, manipulate, and access all types of users and their information.

Placement in UML: Used specifically for every type of user's functionalities, it is placed near the User class and located within the data access layer.

Attributes: Contain all of the user IDs, usernames, roles, permissions, progress data, and other relevant user details.

RoadmapList:

Purpose: Represents a collection of roadmap objects or their associated data. This list can be used to manage, manipulate, and access roadmap information, which typically outlines learning paths, course sequences, and prerequisites.

Placement in UML: Similar to UserList, it is placed near the Roadmap class and is located within the data access layer.

Attributes: Contain all of the course IDs, names, descriptions, dependencies, prerequisites, completion statuses, and other roadmap-related data.

User:

Purpose: Represents a general user of the system. This class captures core user information and provides basic functionalities related to user management.

Placement in UML: Sits at the heart of the system, within the domain module of all the user types, and connects to the UserList object.

Attributes: Includes data like user ID, username, password, name, role (e.g., student, advisor, parent), and additional profile information.

Student:

Purpose: This class represents an individual enrolled in the school program. Inherit from User, adding student-specific attributes (e.g., major, enrolled courses, grades, GPA, credit hours) and operations (e.g., registering for courses, viewing grades, accessing learning materials).

Placement in UML: Because of the inheritance of the User class, it is composed within the User class as a type of User. The Student User also connects with a StudentType Enumeration that correlates with the student classification (freshmen, sophomore, junior, senior).

Attribute: Alongside all the general user attributes, the Student class also includes data like firstName, lastName, studentID, major, currentGPA, earnedCreditHours, classification, and other student-specific data.

Advisor:

*Purpose***:** This class represents someone who guides and supports students in their academic journey and decision-making. Inherit directly from User, extending it with advisor-specific attributes (e.g., expertise areas, availability) and operations (e.g., scheduling appointments, managing student profiles).

Placement in UML: Because of the inheritance of the User class, it is composed within the User class as a type of User.

Attribute: Alongside all the general user attributes, the Advisor class also includes data like advisorName, advisorID, an array of advised students, the student objects themselves, and other advisor-specific data.

Parent:

Purpose: This class represents a parent/guardian of a student user. Similar to Advisor, it can view student(s) users. Inherit directly from User, extending it with the ability to view and observe student(s) users.

Placement in UML: Because of the inheritance of the User class, it is composed within the User class as a type of User.

Attribute: Separate class associated with the User class through a relationship attribute (e.g., parentID linked to a User object).

Roadmap:

Purpose: Represents a sequence and advancement of a Student User for their scholarly journey. Contains all data involving a student and their progression in academics.

Placement in UML: Resides within a module dedicated to learning management or course progression. Could also be part of a domain model representing educational structures.

Attributes: Contain data for a student's roadmap like a list of classes, majorState, statusState, defaultState, completedState, uncompletedState, historyState, and progression status.

RoadmapApplication:

Purpose: This application manages and presents the roadmaps to the users. It facilitates access to information, progress tracking, and potentially additional features related to educational pathways. It also helps with initializing the application, loading user data, accessing and manipulating roadmaps, managing user interactions through the UI, and potentially integrating with external systems (e.g., course enrollment, learning platforms).

Placement in UML: It is situated at the top of the class diagram to represent the overall application in a high-level view.

Attributes: Hold a global application configuration, user authentication data, and references to core components like the roadmap data source and UI interface.

UI:

Purpose: Represents the user interface elements and interactions through which users access and interact with the RoadmapApplication. This includes the visual components and layout, as well as user input handling.

Placement in UML: Is further defined in separate diagrams specifically focusing on the UI design. In the main system diagram, it is shown as a single class subsystem that interacts with the RoadmapApplication.

Attributes: Hold references to specific user interface elements like buttons, lists, course displays, and progress indicators.

MajorState:

Purpose: Represents the different specialization states within the application (CSCE, CIS, and CE). These states define variations in the roadmap based on the chosen major.

Placement in UML: Can be modeled as a class or as separate classes inheriting from a base MajorState class. As a result of the inheritance mechanism and the ability to switch between one of the options, the system connects with the three states: CSCE, CIS, and CE.

Attributes: Typically limited to the name or identifier of the major (e.g., ComputerScience, ComputerEngineering, ComputerInformationSystems).

Status State:

*Purpose***:** Represents the different states a student's course can have within a roadmap, indicating the user's progress for that course.

Attributes: Includes names or identifiers for the states (e.g., Default, Completed, Incomplete, History).

Class:

*Purpose***:** Represents an individual learning unit within a program. It provides detailed information about the course content, instructors, schedules, and expectations.

Placement in UML: Resides within a module dedicated to course management or learning content delivery.

Attributes: Includes information like course ID, name, description, instructors, schedule, learning materials, prerequisites, and potentially associated assessments.

Interfaces

Interface Subject:

*Purpose***:** Defines the contract for any object that manages a list of observers and needs to notify them about changes in its state. This interface specifies methods for managing observers and notifying them. The subject has an aggregated relationship with the Observer, holding a list of registered observers.

Placement in UML: It exists near classes implementing the Observer pattern or within a dedicated design patterns section.

Interface Observer:

Purpose: Defines the contract for any object that needs to be notified about changes in another object. In this case, the Subject Interface will update the Student class and notify the other User types. This interface specifies the method(s) used to receive notifications. Observer implements the Observer interface, signifying its ability to receive updates.

Placement in UML: Similarly to Interface Subject, it exists within a dedicated design patterns section or near classes implementing the Observer pattern.

Enumerators

Enumeration StudentType:

*Purpose***:** Manages the variations of student styles and assigns the classification for the student (freshman, sophomore, junior, senior).

Placement in UML: The StudentType enumeration connects directly to the Student class object.

Enumeration ClassType:

Purpose: Manages the variations of student styles and assigns the classification for the student (freshman, sophomore, junior, senior).

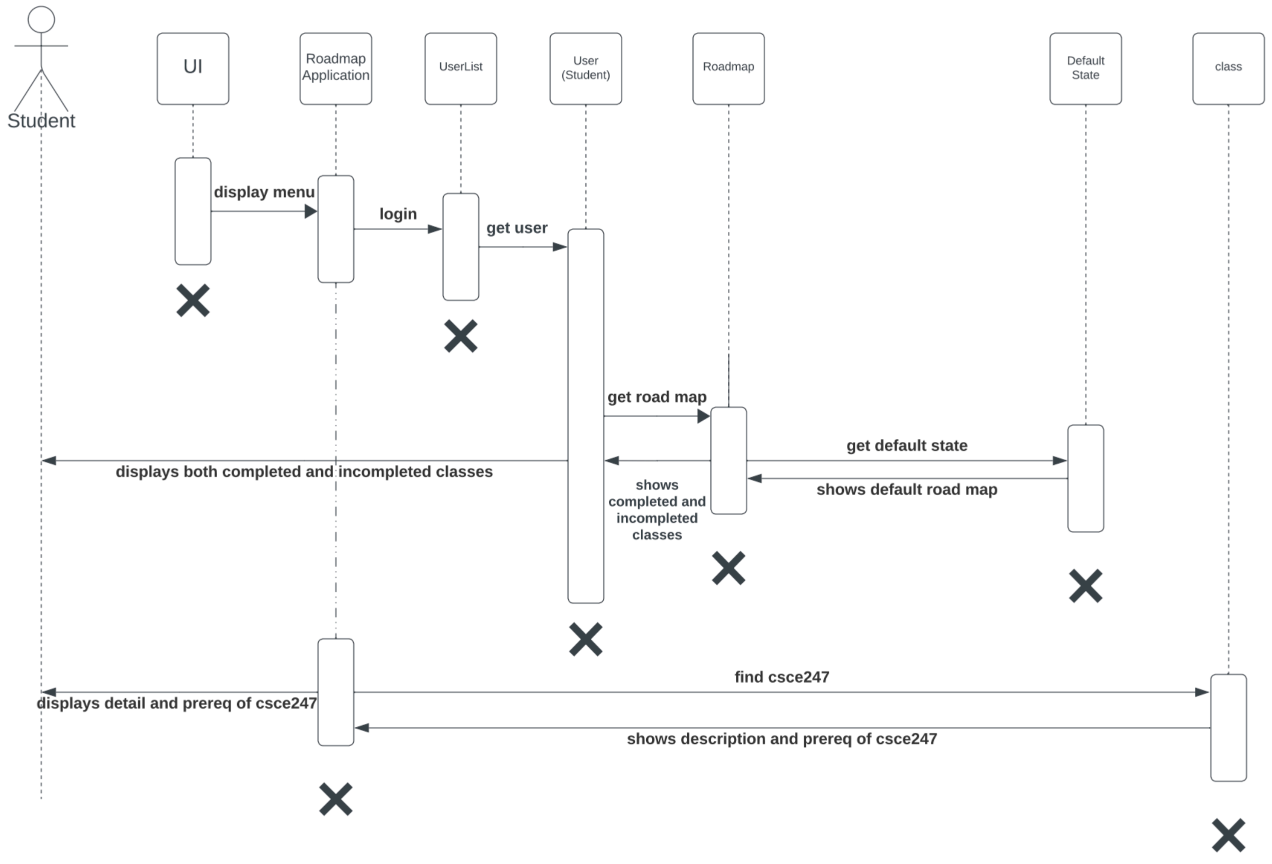
Placement in UML: The StudentType enumeration connects directly to the Student class object.

Sequence Diagram - Dynamic Model

Scenario 1

- *Scenario name:* User is a student
- *Scenario description*: A student logs into the system. They see their major map, listing all the courses they have completed. They see that they have not completed CSCE 247. They look up its details, and its prerequisites

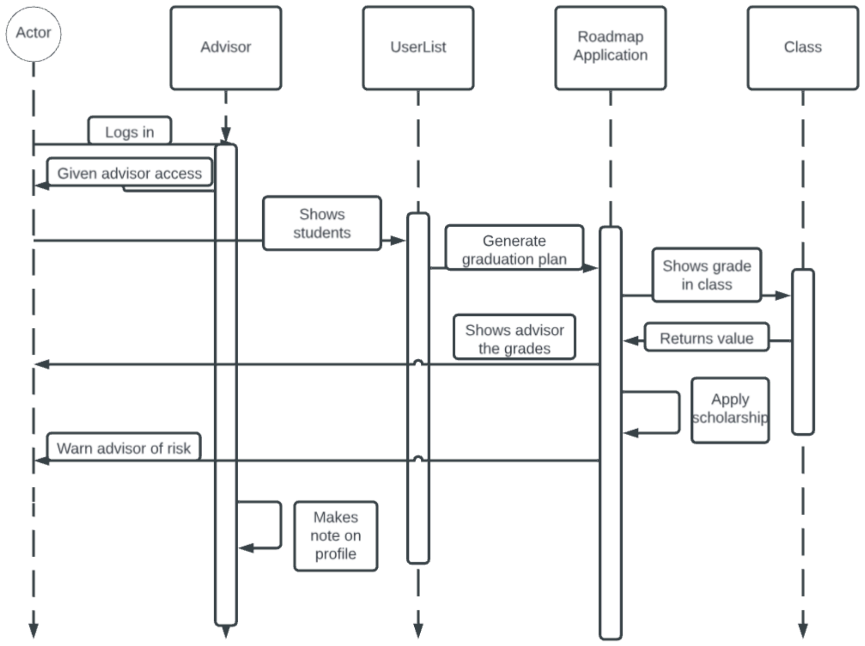
Sequence Diagram:



Scenario 2

- *Scenario name:* User is an advisor
- *Scenario Description:* An advisor logs into the system. They look up their list of advisees and pick a certain advisee. They generate his 8-semester plan and notice that he is at risk of losing his scholarship. They make a note on his account

Sequence Diagram:


Actor
Advisor
UserList
Roadmap Application
Class
Logs in
Given advisor access
Shows students
Generate graduation plan
Shows grade in class
Returns value
Shows advisor the grades
Apply scholarship
Warn advisor of risk
Makes note on profile