System Design
Design Document

Authors:
Nick Khoury
Nathan Moeller
Roderick Ratliff
Analise Rodenberg

Group: 10

Document Revision History

Date	Version	Description
4/10/2016	0.0	Initial draft
5/1/2016	1.0	Final draft

Contents

1 Introduction
1.1 Purpose
1.2 System Overview
1.3 Design Objectives
1.4 References
1.5 Definitions, Acronyms, and Abbreviations
2 Design Overview
2.1 Introduction
2.2 Environment Overview
2.3 System Architecture
2.4 Constraints and Assumptions
3 Interfaces and Data Stores
3.1 System Interfaces
3.1.1 UI Facade
3.1.2 Messaging Transactor Facade
3.1.3 Database Facade
3.2 Data Stores
4 Structural Design
4.1 Class Diagram
4.2 Class Descriptions
4.2.1 Applicant Class
4.2.2 Classes in the University Review System
4.2.3 Classes in the Program Subsystem
5 Dynamic Model
5.1 Scenarios
6 Non-functional requirements
7 Supplementary Documentation

1 Introduction

1.1 Purpose

The purpose of this document is to describe the design and structure of the U-ADMT system. This document includes a class diagram that describes how classes are related to each other, and sequence diagrams that describes the order of events with the system.

1.2 System Overview

The system serves to function as middleware for the application review process for applicants to the University. Using the external database of applications along with an external messaging transactor and UI the system will help to distribute and keep track of the reviews, comments and scores assigned to an application during the review process. It will notify the messaging transactor as each stage of the review process is completed.

There will be four kinds of users in the system who will interact with it via the UI. There will be an Applicant user who can check the status of an application and respond to offers of admittance. College Users make the final decision regarding admittance and manage the capacities of programs and their waitlists. Department Users assign Faculty Users to review applications and then compile those reviews into a score and recommendation for the College Users to use. Finally Faculty Users who create reviews of individual applications.

1.3 Design Objectives

The goal of this design document is to show the functionality of U-ADMT middleware. This functionality includes applicants checking their application status, users reviewing applications, acceptlist and waitlist management, and assigning faculty user to review specific applications. These functions are described in a object oriented class diagram and a few sequence diagrams. The way our objects are created makes it very simple to know what users have certain abilities. We believe this will make the system intuitive and easy to use for our users.

1.4 References

See Group 10 SRS v.2 for further details.

1.5 Definitions, Acronyms, and Abbreviations

Below are new terms used in this document. For terms not in this list, see the SRS section 1.2

Term, Alternate Terms Abbreviation	Definition
Accept List	A list of applications who have been accepted to the program

Pending List	A list of applications who have yet to respond to their offer
Wait List	A list of applications to are waiting to be accepted or rejected from a program
Assignment	A class that links a faculty user to an application. This link indicates that this user must create a review for this application
SRS	Our system requirements specification document

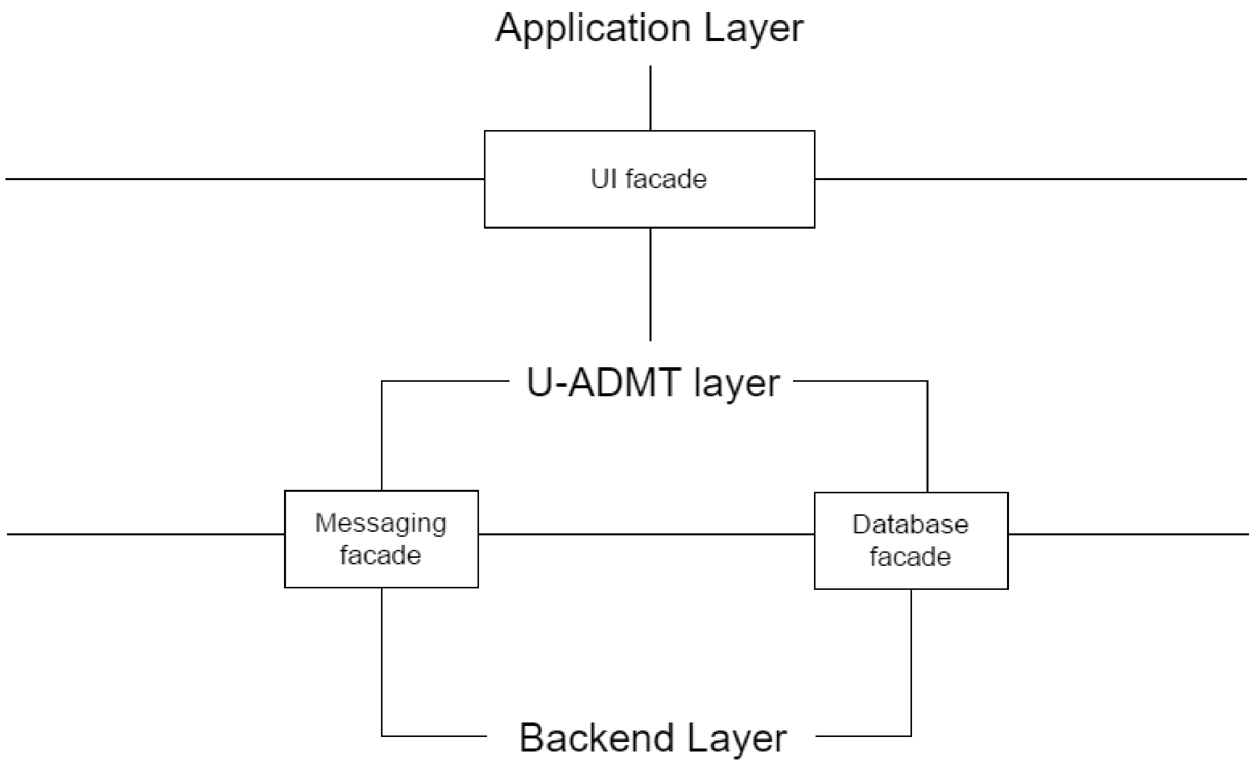
2 Design Overview

2.1 Introduction

Our design approach that we implemented for this assignment is object-oriented design. This was the most intuitive approach for this system because of the objects that exist within the system. Each object has a set of unique attributes and actions. For example, one of the main goals of the system is to review applications. To model this we created a review object that is associated with 1 application. This functionality also allows a UI for the user to easily create a review. This leads to a layered architecture, where our system is the middleware. Our system handles the functionality that the UI requests. At the bottom of the architecture is the database, which stores the data that our middleware needs to complete its requests from the UI layer.

2.2 Environment Overview

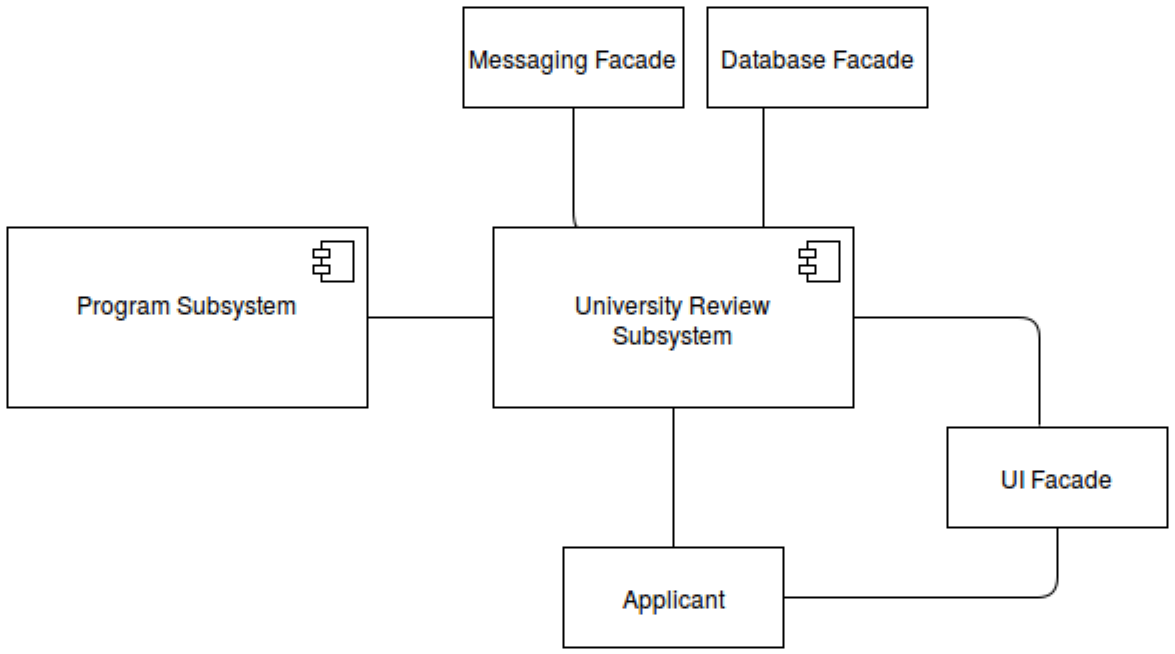
This system is designed to act as middleware between the UI that users interact, the database of applications to the University and the Messaging Transactor. Our system will receive requests for applications from the University Users through the UI and will retrieve the desired application from the database. The UI will interact directly with users and their requests. The UI will be able to call upon our system to enact the requests of the users ranging from requesting an application and its status to submitting reviews of an application. Our system will also keep track of all of the applications in the database. If a user requests an application, our system will retrieve it from the database. Finally, as the review process progresses, the system will notify the Messaging Transactor to send out emails to the appropriate users. All of the above is detailed more in 3.1.



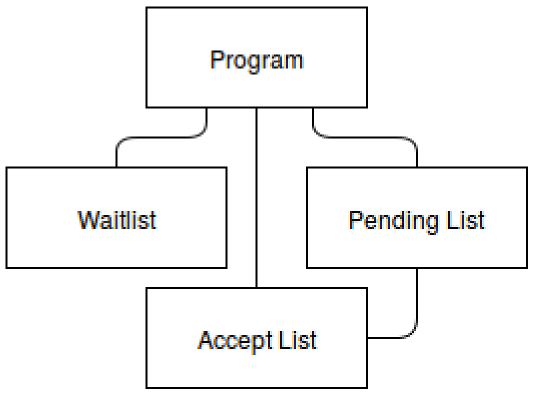
The system will live in a larger environment consisting of 3 major layers. First, the UI layer which handles the user input and interaction within the system. The U-ADMT middleware layer is the focus of this document which handles the functionality of the system. Finally, the backend layer handles the storage and messaging that occurs within the system.

2.3 System Architecture

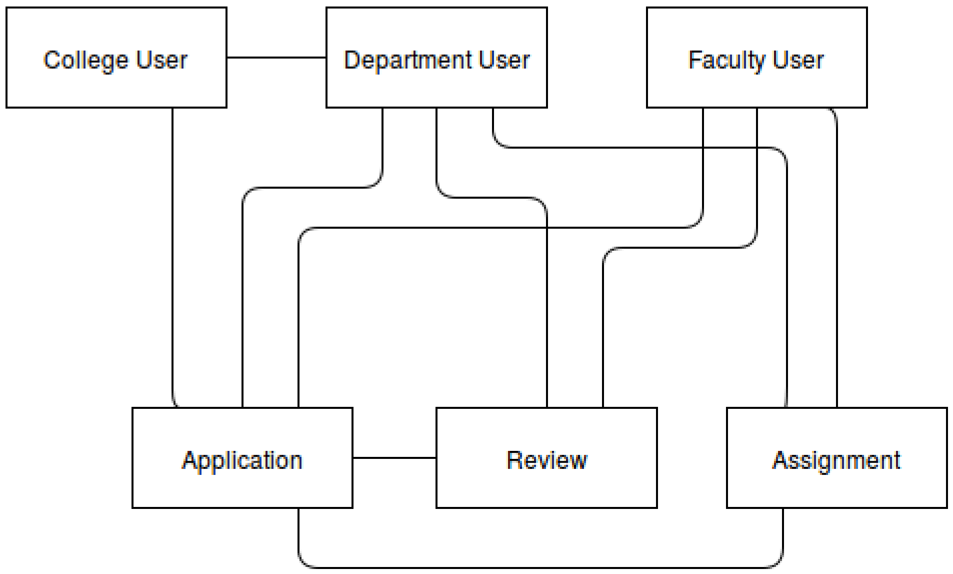
Our system features two major subsystems: the Program Subsystem and the University Review Subsystem. The majority of the work done by the system is handled by the University Review Subsystem. An application is part of the University Review Subsystem and changes to this class results in changes in the Program Subsystem. Also, specific attributes in the Program Subsystem are set by College Users which reside inside the University Review Subsystem. The Program Subsystem links to certain applications in the University Review Subsystem to keep track of which have been admitted to the program and which have been waitlisted.



Above is a diagram showing how the system is comprised of the two subsystems along with the facades for the UI, database and Messaging Transactor and the Applicant Class.



The first of the two subsystems is the Program Subsystem. It represents a single program and department in the University (e.g. CSCI doctorate program) along with the lists of applicants who may be and have been admitted to the program. Each Program Subsystem has a Program Class which keeps track of the waitlist, accept list and pending list as well as the admitted capacity. When an Application in the University Review Subsystem has its official status changed to “accept” or “waitlist”, the Program Subsystem responds by calculating if there is still room on the respective lists and then changing the official status if necessary and putting the application on the correct list.



The other subsystem is the University Review Subsystem. This subsystem does most of the work overall by retrieving applications from the database and allowing University Users to make reviews and score applications. These are stored in a Review object which is linked to an Application. The College Users are able to set the official status of an Application regarding admittance. Once this is set, the messaging facade triggers the Messaging Transactor to email the next user in the chain. Also, the Program Subsystem makes changes if necessary as explained above.

2.4 Constraints and Assumptions

1. Must interact with external DB, UI, and messaging transactor

Our system must work with other external systems not under our control. This includes an external database, external user interface, and an external messaging transactor. To incorporate our system with these external systems we have include a facade to interact with each external system.

3 Interfaces and Data Stores

3.1 System Interfaces

3.1.1 UI facade

This interface is used to control what portions of the middleware the UI can interact with. These interactions include an applicant accepting or rejecting offers, an applicant checking their application status, a faculty/department/college user creating a review, a department user group screening applications, and a college user setting the final decision for an application.

3.1.2 Messaging Transactor facade

This interface is used to control when notifications are sent out to other users of U-ADMT. Messages are sent when a new review is created, a faculty user fails to review an application by the assigned deadline, and when an official status for an application has been set.

3.1.3 Database facade

This interface controls the access to application data. This facade is used when U-ADMT needs application data not pertaining to the middleware.

3.2 Data Stores

An external database stores user reviews and application data. All of the other classes in our system are stored persistently. These include the entire program subsystem, the classes that represent users (faculty, department, college), and the assignment class. The application must store the classes indicated.

4 Structural Design

4.1 Class Diagram

See included file

We designed the system focusing on how different users will interact with the system. Since each type of user has a different set of abilities, we created four different classes which, when instantiated, will represent an individual user of that type. Each of these User classes simulates the actions that actual users of the system will do, such as respond to an offer or review an application.

To facilitate the use of these classes, we made use of the facade design pattern. This ensures the represented user class will only have access to the appropriate subset of the system that pertains to them. The facade only provides access to methods in the User classes and therefore forces the UI team to make actions through them.

4.2 Class Descriptions

4.2.1 Class: Applicant

- Purpose: To model the attributes and actions that an applicant has within the system
- Constraints: None

Attribute Descriptions

1 .Attribute: name
Type: String
Description: The name of the applicant
Constraints: None

2. Attribute: type
Type: String
Description: The type of the applicant (undergraduate, graduate, professional)

Method Descriptions

1. Method: acceptOffer(application)
Return type: boolean
Parameters: An application
Return values: success or failure
Pre-condition: The application must have an officialStatus of accepted
Post-condition: The applications offerStatus is changed to accepted, and all other application's offerStatus (associated with this applicant) is changed to rejected
Attributes read/used: application.offerStatus,
Methods called: Application.setOfferStatus(), Applicant.rejectOffer() (if needed)
Processing logic: Sets application.offerStatus to "accept" through application.setOfferStatus("accept") and finds all of the other applications associated to the given Applicant, if there are any, and calls rejectOffer(other_application).

2. Method: rejectOffer(application)
Return type: boolean
Parameters: An application
Return values: success or failure
Pre-condition: The application must have an officalStatus of accepted
Post-condition: The applications offerStatus is changed to rejected
Attributes read/used: application.offerStatus
Methods called: Application.setOfferStatus()
Processing logic: Calls application.setOfferStatus("rejected") to set the offer status to "rejected".

3. Method: checkStatus(application)
Return type: String
Parameters: An application
Return values: accepted, rejected, or waitlist
Pre-condition: none
Post-condition: no change
Attributes read: none
Methods called: Application.getOfficialStatus()
Processing logic: returns the official status of a given application
4. Method: getType()
Return value: string
Parameters: none
Return value: Applicant.type
Pre-Condition: none
Post-Condition: no change
Attributes read/used: Applicant.type
Processing logic: return Applicant.type

4.2.2 Classes in the University Review Subsystem

4.2.2.1 Class: University user

- Purpose: This class is an abstract class that defines the attributes and actions that every user within the university has .
- Constraints: This class is not instantiable.

Attribute Descriptions

1. Attribute: name
Type: string
Description: The name of the faculty user within the university
Constraints: None

2. Attribute: email
Type: string
Description: The email of the faculty user within the university
Constraints: None

Method Descriptions

1. Method: filterApplications(filterCondition)
Return type: list
Parameters: filterCondition
Return value: list of applications
Pre-Condition: none
Post-Condition: the list of applications are filtered by the given condition
Attributes read/used: none
Methods called: none

Processing Logic: returns a list of applications filtered by the given filter condition

2. Method: reviewApplication(application)
Return type: void
Parameters: an application object
Return value: void
Pre-Condition: none
Post-Condition: a review object is created associated to the application
Attributes read/used: none
Methods called: none
Processing logic: creates a review object associated with an application

3. Method: addToReview(Review review, string field, string value)
Return type: boolean
Parameters: review - the review of interest to be modified; field - the name of the field within the application to be modified; value - the value to be put in.
Return value: success or failure
Pre-Condition: review exists
Post-Condition: review.field has been modified to contain value.
Attributes read/used: review.field
Methods called: Review.addComment(), Review.setScore(), Review.setRecStatus(), Review.SubmitReview()
Processing logic: read in field and determine which attribute of interest it is. Call the corresponding set method within review with the possibly type changed value.

4. Method: getFromReview(Review review, string field)
Return type: string
Parameter: review - the review of interest; field - the attribute within the review to be returned.
Pre-Condition: review already exists
Post-Condition: Review.field is returned
Attributes read/used: Review.field
Methods called: Review.getComment(), Review.getScore(), Review.getRecStatus(), Review.getCompleted()
Processing logic: Read in the string field and call the corresponding get method in review (review.getField()) and return that value.

4.2.2.1.1 Subclass: Faculty user

- Purpose: An instance of this class is made to model the specific attributes and actions that a faculty user has.
- Constraints: none

Method Descriptions

1. Method: getAssignments()

Return type: list
Parameters: none
Return value: a list of assignment objects that pertain to this faculty user
Pre-Condition: none
Post-Condition: no change
Attributes read/used: none
Methods called: Assignment.getApplication()
Processing logic: returns the list of assignments that the faculty user has.

4.2.2.1.2 Subclass: Department User

- Purpose: An instance of this class is made to model the specific attributes and actions that a department user has.
- Constraints: none

Attribute Descriptions

1. Attribute: reviewDeadline
 Type: date
 Description: The date that this user must group screen all applications by
 Constraints: none

Method Descriptions

1. Method: createAssignment(FacultyUser, Application, date)
Return type: boolean
Parameters: a faculty user, application, a datae
Return value: success or failure
Pre-Condition: none
Post-Condition: no change
Attributes read/used: none
Methods called: none
Processing Logic: creates an assignment object that assigns a faculty user to review an application

4.2.2.1.3 Subclass: College User

- Purpose: An instance of this class is made to model the specific attributes and actions that a college user has.
- Constraints: none

Method Descriptions

1. Method: setReviewDeadline(departmentUser, deadline)
Return type: boolean
Parameters: a department user, deadline
Return value: success or failure
Pre-Condition: none
Post-Condition: The review deadline for a department user has been set
Attributes read/used: none
Methods called: none

Processing Logic: sets the department user attribute “reviewDeadline” to the given deadline.

2. Method: setProgramCapacity(program, capacity)
Return type: boolean
Parameters: a program object, capacity integer
Return value: success or failure
Pre-Condition: none
Post-Condition: The capacity for the program has been updated
Attributes read/used: none
Methods called: Program.setCapacity()
Processing Logic: sets the program’s capacity to the given integer

3. Method: setWaitlistCapacity(waitlist, capacity)
Return type: boolean
Parameters: a waitlist object, capacity integer
Return value: success or failure
Pre-Condition: none
Post-Condition: The capacity for the waitlist has been updated
Attributes read/used: none
Methods called: Waitlist.setCapacity()
Processing Logic: sets the program’s capacity to the given integer

4. Method: setFinalDecision(application, status)
Return type: boolean
Parameters: an application, status to set to
Return value: success or failure
Pre-Condition: none
Post-Condition: The offerStatus attribute is updated
Attributes read/used: none
Methods called: Application.setOfficialStatus()
Processing Logic: sets the application’s status to the given status

5. Method: setScoreThreshold(Program program, double score)
Return type: boolean
Parameters: program - the program of interest; score - the score to be set to
Return value: success or failure
Pre-Condition: none
Post-Condition: Program.scoreThreshold is now score
Attributes read/used: Program.scoreThreshold
Methods called: Program.setScoreThreshold()
Processing logic: Call program.setScoreThreshold passing in score.

4.2.2.2 Class: Assignment

- Purpose: An instance of this class links an application to a faculty member. This describes which faculty member must review which applications.
- Constraints: none

Attribute Descriptions

1. Attribute: deadline
Type: date
Description: The date when a review must be completed by
Constraints: None

Method Descriptions

1. Method: getApplication()
Return type: Application
Parameters: none
Return value: an application object
Pre-Condition: none
Post-Condition: none
Attributes read/used: none
Methods called: none
Processing Logic: returns the application that the assignment is associated with

4.2.2.3 Class: Application

- Purpose: Keeps track of the status and information in an application submitted to the University.
- Constraints: There will be one instance of this class for each application submitted to the University.

Attribute Descriptions:

1. Attribute: officialScore
Type: double
Description: Is the score assigned to the application by the Department User for official scoring purposes.
Constraints: Must be a score between 0 and 100 and will be truncated to four significant digits. Shall be calculated as follows for each applicant type:

Undergraduate: GPA (25%), Extra Curricular activities (20%), recommendations (15%), ACT/SAT (25%), essays (15%)
Graduate: GPA (30%), Relevant work experience (20%), recommendations (20%), GRE (15%), essays (15%)
Professional: Research experiences(25%), recommendations(25%), GPA(25%), essays(25%).

2. Attribute: recommendedStatus
Type: string

Description: Is the status the Department Users recommend for the College Users to take.
Constraints: Must be “accept”, “reject” or “waitlist”. Must be set by Department Users.

3. Attribute: officialStatus
Type: string
Description: Is the status that the College User sets for admittance.
Constraints: Must be “accept”, “reject” or “waitlist”. Must be set by College Users. Can only be set to “accept” or “waitlist” if the corresponding lists are below capacity.

4. Attribute: offerStatus
Type: string
Description: Is the response to an offer of admittance for an application.
Constraints: Must be “accepted”, “pending”, “rejected”. To be set by the Applicant User the application corresponds to. Can only be set to “accepted” if the Applicant has not accepted another offer. Can only be set if the officialStatus is “accept”.

5. Attribute: deadline
Type: date
Description: The deadline for the applicant to respond to their offer
Constraints: This value is only valid when the officialStatus is “accept”. If officialStatus is “waitlist” or “reject”, this value is NULL.

Method Descriptions:

1. Method: setScore(boolean switch, double deptScore)
Return type: boolean
Parameters: switch - a boolean to switch whether or not to augment the score provided by faculty; deptScore - the score to be assigned to the application
Return value: Success or failure
Pre-Condition: none
Post-Condition: Application.officialScore is set to the appropriate score.
Attributes read/used: Application.officialScore, Applicant.type, Review.score
Methods called: Applicant.getType(), Review.getScore()
Processing logic: If switch is true, calculate the score according to the formulas above in the attribute score definition, calling Application.getType() to decide the formula. For the faculty score section, use deptScore.

If switch is false, call Applicant.getType() to pick which formula to use and then in place of the faculty score section call Review.getScore() on each of the reviews associated to the application and use the average of these scores.

2. Method: setRecStatus(string status)
Return type: boolean
Parameters: status - the status of “accept”, “reject” or “waitlist” that is recommended for the College User to use.
Return Value: Success or failure
Pre-Condition: none
Post-Condition: Application.recommendedStatus is set to status.
Attributes read/used: Application.recommendedStatus
Methods called: none
Processing logic: set Application.recommended Status to status.

3. Method: setOfficialStatus(string status)
Return Type: boolean
Parameters: status - the desired official status for the application
Return Value: Success or failure
Pre-Condition: none
Post-Condition: Application.officialStatus is set to the appropriate status of “accept”, “reject” or “waitlist” depending on whether or not the Program of interest is at capacity. The corresponding list (AcceptList, Waitlist or PendingList) is updated accordingly.
Attributes read/used: Program.capacity, AcceptList.numMembers, PendingList.numMembers, Waitlist.numMembers, Waitlist.capacity.
Methods called: Program.getCapacity(), Acceptlist.getNumMembers(), PendingList.getNumMembers(), Waitlist.addMember(), Waitlist.getCapacity(), Waitlist.getNumMembers(), Application.setOfferStatus()
Processing logic: If status is “accept” and AcceptList.numMembers + PendingList.numMembers is less than Program.capacity, set Application.officialStatus to “accept” and Application.setOfferStatus(“pending”).

If status is “accept” and AcceptList.numMembers + PendingList.numMembers is equal to Program.capacity and Waitlist.numMembers is less than Waitlist.capacity, set Application.officialStatus to “waitlist” and call Waitlist.addMember(application).

If status is “accept” and AcceptList.numMembers + PendingList.numMembers is equal to Program.capacity and Waitlist.numMembers is equal to Waitlist.capacity, set Application.officialStatus to “reject”.

If status is “waitlist” and Waitlist.numMembers is less than to Waitlist.capacity, set Application.officialStatus to “waitlist” and call Waitlist.addMember(application).

If status is “reject” set Applicaton.officialStatus to “reject”.

After the above, notify the messaging facade to send notification to the corresponding Applicant of the application.

4. Method: setOfferStatus(string status)
Return Type: boolean
Parameters: status - the status that Application.offerStatus is to be set. Should be “accepted”, “rejected” or “pending”.
Return value: Success or failure
Pre-condition: application.officialStatus has been set to “accept”.
Post-Condition: Application.offerStatus has been changed to “status” and the appropriate lists have been updated (PendingList and possibly AcceptList).
Attributes read/used: Application.offerStatus
Methods called: PendingList.addMember(), PendingList.removeMember(), AcceptList.addMember()
Processing Logic: If status is “pending”, set Application.offerStatus to “pending” and call PendingList.addMember(application).

If Application.offerStatus is already “pending” and status is “accepted”, set Application.offerStatus to “accepted” and call PendingList.removeMember(application) then AcceptList.addMember(application).

If Application.offerStatus is already “pending” and status is “rejected”, set Application.offerStatus to “rejected” and call PendingList.removeMember(application).

5. Method: getOfficialStatus()
Return type: string
Parameters: none
Return value: Application.officialStatus
Pre-Condition: none
Post-Condition: the attribute Application.officialStatus is returned
Attributes read/used: Application.officialStatus
Methods called: none
Processing logic: Return Application.officialStatus

6. Method: getOfferStatus()
Return type: string
Parameters: none
Return value: Application.offerStatus
Pre-Condition: Application.officialStatus has been set to “accept”.
Post-Condition: the attribute Application.offerStatus is returned
Attributes read/used: Application.offerStatus
Methods called: none

Processing logic: Return Application.offerStatus

7. Method: getDeadline()
Return type: date
Parameters: none
Return value: Application.deadline
Pre-Condition: none
Post-condition: the attribute Application.deadline is returned
Attributes read/used: Application.deadline
Methods called: none
Processing Logic: The value Application.deadline is returned

8. Method: setDeadline(date deadline)
Return type: boolean
Parameters: deadline, the deadline to be set to
Return value: success or failure
Pre-condition: none
Post-condition: The deadline value is updated to the parameters value
Attributes read/used: none
Methods called: none
Processing Logic: The deadline value is overwritten to the parameters passed in.

4.2.2.4 Class: Review

- Purpose: Keeps track of the review a University User gives an application.
- Constraints: Is created by a University User through the UI calling the reviewApplication() method on an application.

Attribute Descriptions:

1. Attribute: comment
Type: string
Description: A string containing the thoughts of the reviewer on the given application. The string is given by the UI-interface.
Constraints:

2. Attribute: score
Type: double
Description: The score given by a University User on an application. Will be truncated to 4 significant digits.
Constraints: Must be greater than or equal to 0 and less than or equal to 100.

3. Attribute: recommendation
Type: string
Description: The status that the reviewer thinks would be best for the applicant.
Constraints: Must be either “accept”, “reject” or “waitlist”

4. Attribute: completed
Type: boolean
Description: Indicator for whether or not the review has been submitted as completed. Will allow or disallow further edits from occurring.

Method Descriptions:

1. Method: addComment(string comment)
Return type: boolean
Parameters: comment - a string to be put in the attribut Review.comment
Return value: Success or failure
Pre-Condition: none
Post-Condition: the attribute Review.comment is now comment or an error is returned.
Attributes read/used: Review.comment, Review.Completed
Methods called: Review.getCompleted()
Processing logic: Set Review.comment to comment if Review.completed is false. Return an error otherwise.

2. Method: getComment()
Return type: string
Parameters: none
Pre-Condition: none
Post-Condition: Review.comment is returned
Attributes read/used: Review.comment
Methods called: none
Processing logic: Return Review.comment

3. Method: setScore(double score)
Return type: boolean
Parameters: score - the score to be set to Review.officialScore
Pre-Condition: none
Post-Condition: Review.officialScore is now score or an error is returned
Attributes read/used: Review.officialScore, Review.completed
Methods called: Review.getCompleted()
Processing logic: set Review.officialScore to score provided it is between 0 - 100 if Review.completed is false. If not return an error.

4. Method: getScore()
Return type: double
Parameters: none
Return value: Review.score
Pre-Condition: none
Post-Condition: Review.score is returned
Attributes read/used: Review.score

Methods called; none
Processing logic: Return Review.score

5. Method: setRecStatus(string status)
Return type: boolean
Parameters: status - one of “accept”, “reject” or “waitlist”.
Return value: Success or failure
Pre-Condition: none
Post-Condition: Review.recommendedStatus is set to status or an error is returned.
Attributes read/used: Review.recommendedStatus, Review.completed
Methods called: Review.getCompleted()
Processing logic: Set Review.recommendedStatus to status provided it is one of “accept”, “reject” or “waitlist” and Review.completed is false. If not, return an error.

6. Method: getRecStatus()
Return type: string
Parameters: none
Return value: Review.recommendedStatus
Pre-Condition: none
Post-Condition: Review.recommendedStatus is returned
Attributes read/used: Review.recommendedStatus
Methods called: none
Processing logic: Return Review.recommendedStatus

7. Method: submitReview()
Return type: boolean
Parameters: none
Return value: Success or failure
Pre-Condition: Review.score, Review.recommendedStatus have been filled out.
Post-Condition: Review.completed is set to true.
Attributes read/used: Review.score, Review.recommendedStatus, Review.completed
Methods called: Review.getScore(), Review.getRecStatus(), Review.getCompleted()
Processing logic: Check that Review.completed is false. Check that there is a valid score (0 - 100) in Review.score and that Review.recommendedStatus is “accept”, “reject” or “waitlist”. If so, set completed to true.

If Review.completed is already true or Review.recommendedStatus is not one of the above, return an error.

8. Method: getCompleted()

Return type: boolean
Parameters: none
Return value: Review.completed
Pre-Condition: none
Post-Condition: none
Attributes read/used: Review.completed
Methods called: none
Processing logic: Return Review.completed

4.2.3 Classes in the Program Subsystem

4.2.3.1 Class: Program

- Purpose: An instance of this class will stand for a single program within the University of Minnesota (e.g. the Ph.D. program in Computer Science). It serves to keep track of all the applications to the given program as well as tracking the applicants that have pending offers of admittance, those that have accepted offers and the waitlist.
- Constraints: There is one instance for each program and department of study in the University system.

Attribute Descriptions

1. Attribute: name
Type: string
Description: Gives the name of the program the instance of the class is representing (e.g. doctoral-csci).
Constraints: The name will be unique among all instances of the class.

2. Attribute: capacity
Type: int
Description: maximum number of applicants to be admitted to the program.
Constraints: Must be greater than or equal to 0.

3. Attribute: scoreThreshold
Type: double
Description: The recommended minimum score for admittance to the program
Constraints: Must be between 0 and 100.

Method Descriptions

1. Method: setCapacity(int capacity)
Return type: boolean
Parameters: capacity - the number the admitted-capacity is to be set to.
Return value: Success or failure
Pre-Condition: none
Post-Condition: the capacity attribute is set to the value int capacity
Attributes read/used: admitted-capacity
Methods called: none

Processing logic: Set Program.capacity to capacity.

2.Method: getCapacity()
Return type: int
Parameters: none
Return value: Program.capacity
Pre-condition: none
Post-condition: none
Attributes read/used: Program.capacity
Methods called: none
Processing logic: Return Program.capacity.

3. Method: setScoreThreshold(double score)
Return type: boolean
Parameters: the score to be inserted into Program.scoreThreshold
Pre-Condition: none
Post-Condition: Program.scoreThreshold is now set to score.
Attributes read/used: Program.scoreThreshold
Methods called: none
Processing logic: set Program.scoreTheshold to score.

4. Method: getScoreThreshold()
Return type: double
Parameters:
Pre-Condition: none
Post-Condition: Program.scoreThreshold is returned
Attributes read/used: Program.scoreThreshold
Methods called: none
Processing logic: return Program.scoreTheshold

4.2.3.2 Class: Applicant List

- Purpose: Keeps track of certain applicants. Could either be the list of applicants to a certain program, applicants on the waitlist, accept list or pending list.
- Constraints: none

Attribute Descriptions:

1. Attribute: applicationList
Type: list
Description: A list of applications
Constraints: None

2. Attribute: numMembers
Type: int
Description: the number of applications currently on the associated applicationList.

Constraints: Must be greater than or equal to zero.

Method Descriptions:

1. Method: addMember(Application application)
Return Type: boolean
Parameters: application - the application to be added to the list.
Return Value: Success or failure
Pre-Condition: none
Post-Condition: application is now on applicationList and ApplicantList.numMembers is increased by one.
Attributes read: ApplicantList.numMembers, applicationList
Methods called: ApplicantList.setNumMembers(), ApplicantList.getNumMembers()
Processing logic: The method will add the application to applicationList and will increment ApplicantList.numMembers by one.

2. Method: removeMember(Application application)
Return type: boolean
Parameters: application - the application to be removed from the list
Return value: Success or failure
Pre-Condition: The application must already be on the list.
Post-Condition: The number of members on the list is decreased by one, the application is removed from the list.
Attributes read/used: ApplicantList.numMembers, applicationList
Methods called: ApplicanttList.setNumMembers(), ApplicantList.getNumMembers()
Processing logic: The method will remove the application from the applicationList and decrease ApplicationtList.numMembers by one.

3. Method: setNumMembers(int num)
Return type: boolean
Parameters: num - the new number of members of the list.
Return value: Success or failure
Pre-Condition: none
Post-Condition: ApplicantList.numMembers is set to num
Methods called: none
Processing logic: Set the field numMembers to num if not.

4. Method: getNumMembers()
Return type: int
Parameters: none
Return value: Waitlist.numMembers
Pre-Condition: none
Post-Condition: no change
Methods called: none

Processing logic: Return the field numMembers.

4.2.3.2.1 Subclass: Waitlist

- Purpose: Keeps track of the applications on the waitlist and their rankings on the waitlist for a given program.
- Constraints: There is one instance of the waitlist class per instance of the Program class.

Attribute Descriptions:

1. Attribute: capacity
Type: int
Description: maximum number of applicants to be waitlisted for the given program.
Constraints: Must be greater than or equal to zero.

Method Descriptions:

1. Method: addMember(Application application)
Return type: boolean
Parameters: application - the application to be added to the Pending List.
Return value: Success or failure
Pre-Condition: Waitlist.numMembers must be less than Waitlist.capacity
Post-Condition: numMembers is increased by 1 and the application is now on waitList.
Attributes read/used: waitlist.numMembers, waitlist.capacity, waitlist.applicationList
Methods called: Waitlist.setNumMembers(), Waitlist.getNumMembers()
Processing logic: The method will add the application to waitlist.applicationList and increase Waitlist.numMembers by one.

2. Method: setNumMembers(int num)
Return type: boolean
Parameters: num - the new number of members of the list.
Return value: Success or failure
Pre-Condition: none
Post-Condition: Waitlist.numMembers is set to num or an error is returned and the value is unchanged.
Methods called: Waitlist.getCapacity()
Processing logic: Check to see that num is less than Waitlist.capacity. If it is, set the field numMembers to num if not, return an error and do not change the value

3. Method: setCapacity(int capacity)
Return type: boolean
Parameters: capacity - the number the waitlist-capacity is to be set to.
Return value: Success of failure
Pre-Condition: none

Post-Condition: the capacity attribute is set to the value int capacity
Attributes read/used: waitlist-capacity
Methods called:
Processing logic: Sets the field Waitlist.capacity to capacity.

4. Method: getCapacity(int capacity)
Return type: int
Parameters: capacity - the number the waitlist-capacity is to be set to.
Return value: Waitlist.capacity
Pre-Condition: none
Post-Condition: the capacity attribute is set to the value int capacity
Attributes read/used: waitlist-capacity
Methods called:
Processing logic: Returns Waitlist.capacity.

4.2.3.2.2 Subclass: Accept List

- Purpose: Keeps track of the applications who have accepted an offer of admittance to a given program.
- Constraints: There is one instance of the Accept List class per instance of the program class. The number of members on AcceptList.applicantList shall not exceed Program.capacity minus PendingList.numMembers

Attribute Descriptions:
Method Descriptions:

1. Method: addMember(Application application)
Return type: boolean
Parameters: application - the application to be added to the Pending List.
Return value: Success or failure
Pre-Condition: The officialStatus of the application must be “accept” and the offerStatus must be “accepted”
Post-Condition: numMembers is increased by 1 and the application is now on acceptList.
Attributes read/used: AcceptList.numMembers, AcceptList.applicationList
Methods called: AcceptList.setNumMembers(), AcceptList.getNumMembers()
Processing logic: The method will check that AcceptList.numMembers + 1 does not exceed Program.capacity - PendingList.numMembers. If so, add the application to AcceptList.applicationList and increase AcceptList.numMembers by one. If not, return an error.

2. Method: setNumMembers(int num)
Return type: boolean
Parameters: num - the new number of members of the list.
Return value: Success or failure
Pre-Condition: none

Post-Condition: AcceptList.numMembers is set to num or an error is returned
Attributes read/used: PendingList.numMembers, Program.Capacity
Methods called: PendingList.getNumMembers(), Program.getCapacity()
Processing logic: Set the field numMembers to num if num is less than or equal to Program.capacity - PendingList.numMembers. If not, return an error and leave numMembers unchanged.

4.2.3.2.3 Subclass: Pending List

- Purpose: Keeps track of the applications who have been given an offer of admittance to the program but have not yet accepted or rejected the offer.
- Constraints: There is one instance of the Pending List class per instance of the program class. PendingList.numMembers shall not exceed Program.capacity minus AcceptList.numMembers

Attribute Descriptions:
Method Descriptions:

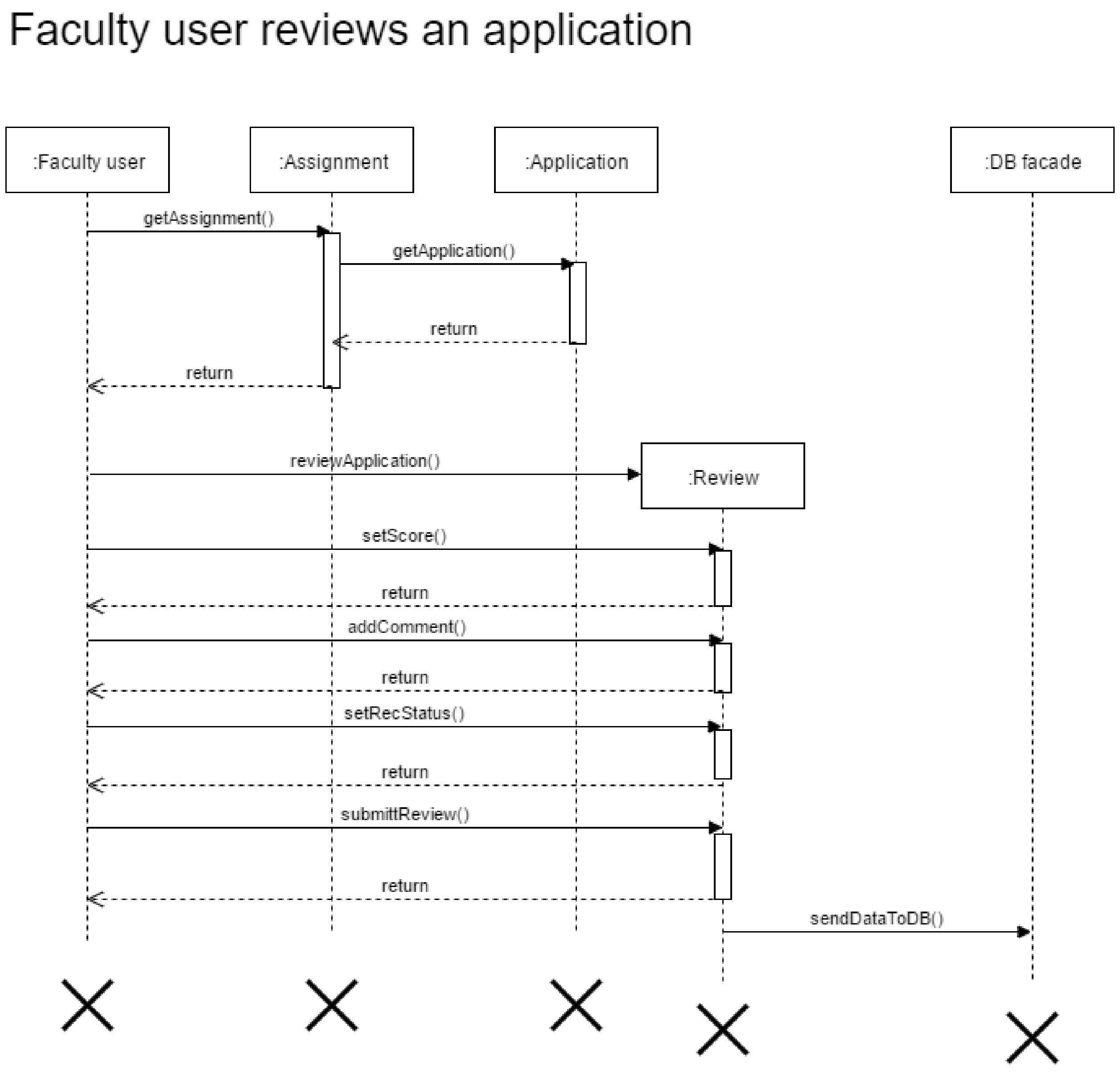
1. Method: addMember(Application application)
Return type: boolean
Parameters: application - the application to be added to the Pending List.
Return value: Success or failure
Pre-Condition: The officialStatus of the application must be “accept”, the offerStatus is “pending”.
Post-Condition: numMembers is increased by 1 and the application is now on pendingList.
Attributes read/used: offerStatus, officialStatus, PendlingList.numMembers, PendingList.applicantList
Methods called: PendingList.setNumMembers(), PendingList.getNumMembers()
Processing logic: The method will check that PendingList.numMembers + 1 does not exceed Program.capacity - AcceptList.numMembers. If so, add the application to PendingList.applicationList and increase the number of members on the list by one. If not, return an error.

2. Method: setNumMembers(int num)
Return type: boolean
Parameters: num - the new number of members of the list.
Return value: Success or failure
Pre-Condition: none
Post-Condition: PendingList.numMembers is set to num or an error is returned
Attributes read/used: AcceptList.numMembers, Program.Capacity
Methods called: AcceptList.getNumMembers(), Program.getCapacity()
Processing logic: Check that num is less than or equal to Program.capacity - AcceptList.numMembers. If it is, set PendingList.numMembers to num. If not, return an error.

5 Dynamic Model

5.1 Scenarios

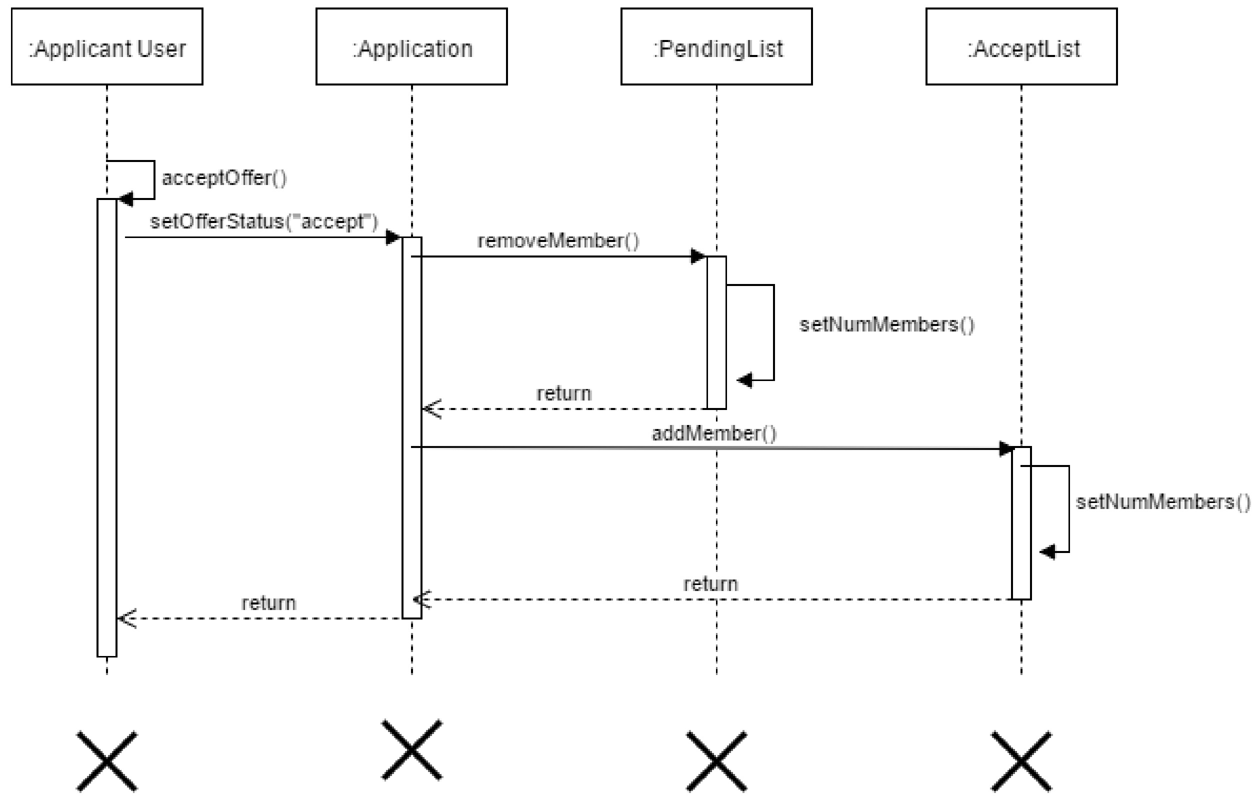
Scenario Name: Faculty user reviews an application
Scenario Description: A faculty user reviews an application. This entails the user setting a score, comment, and a recommendation for acceptance. Once the application is done being reviewed, the faculty user submits the review as done and the department user is notified.
Sequence Diagram:



Scenario Name: Applicant Accepts an offer

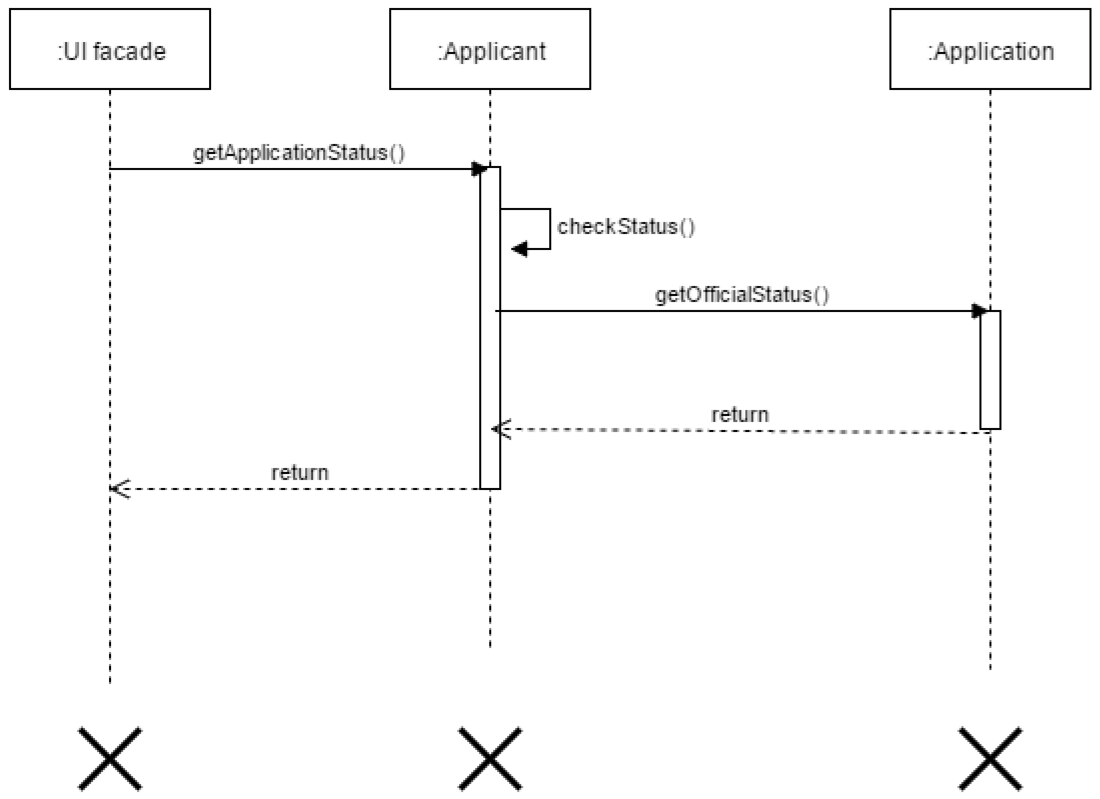
Scenario Description: Once an offer of admittance has been offered to an applicant, the applicant has the choice to accept or reject it. In the case that they want to accept it, through the UI facade, the Applicant User class accepts the offer and this calls the Application to set the offer status to “accept” which in turn removes the applicant from the list of pending offers and adds them to the list of those who have accepted offers of admittance.
Sequence Diagram:

Applicant Accepts an Offer

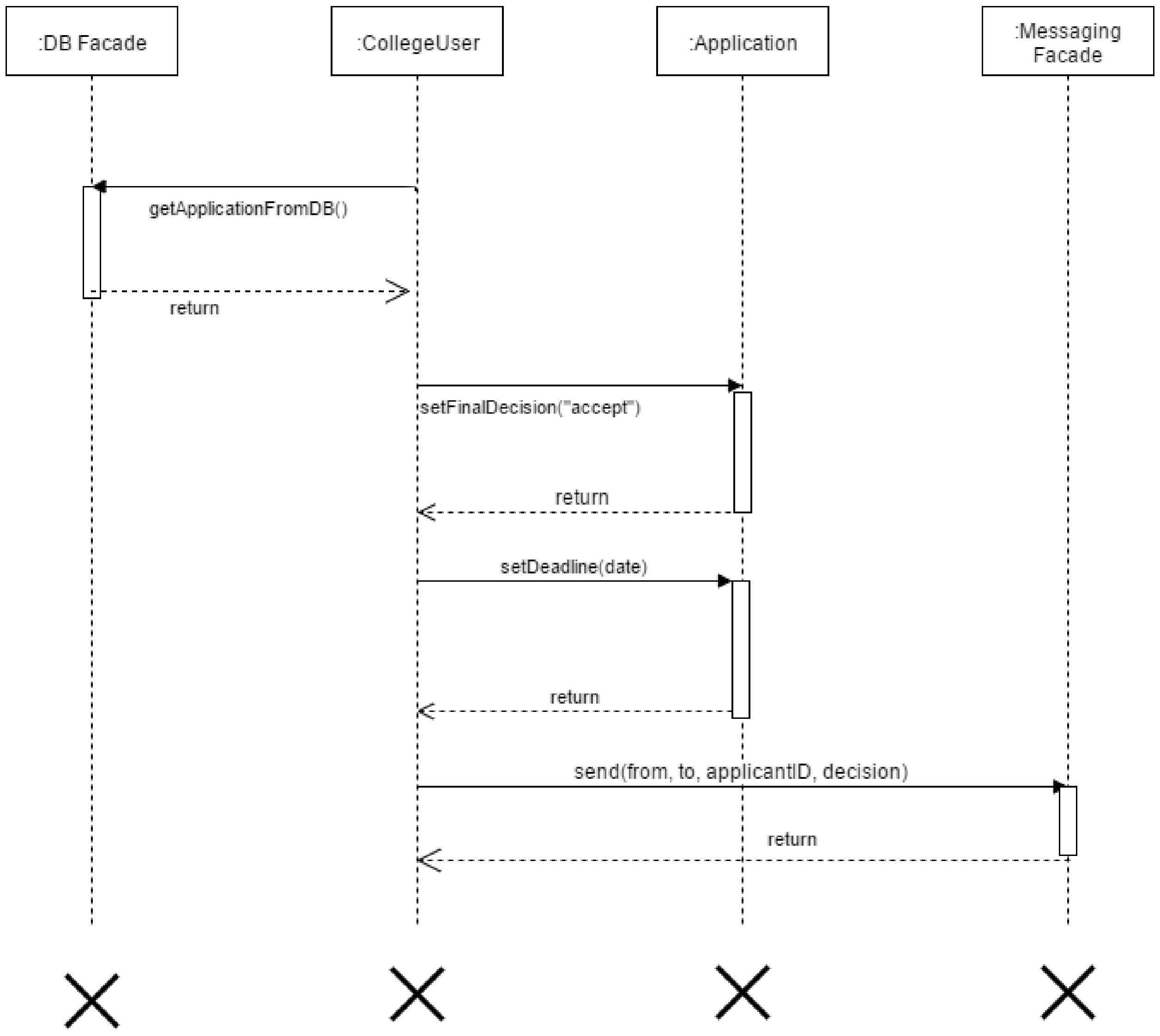


Scenario Name: Applicant checks status of offer
Scenario Description: An applicant checks the official status of their application. The status returns as “accepted”, “rejected”, or “waitlisted”.
Sequence Diagram:

Applicant Checks Status of Offer



Scenario Name: Final decision rendered
Scenario Description: A college user accepts an application into a program.This is done by the college user creating a review for the application, and then setting the officalStatus to “accepted”.
Sequence diagram:


:DB Facade
:CollegeUser
:Application
:Messaging Facade
getApplicationFromDB()
return
setFinalDecision("accept")
return
setDeadline(date)
return
send(from, to, applicantID, decision)
return