SMU
SINGAPORE MANAGEMENT UNIVERSITY

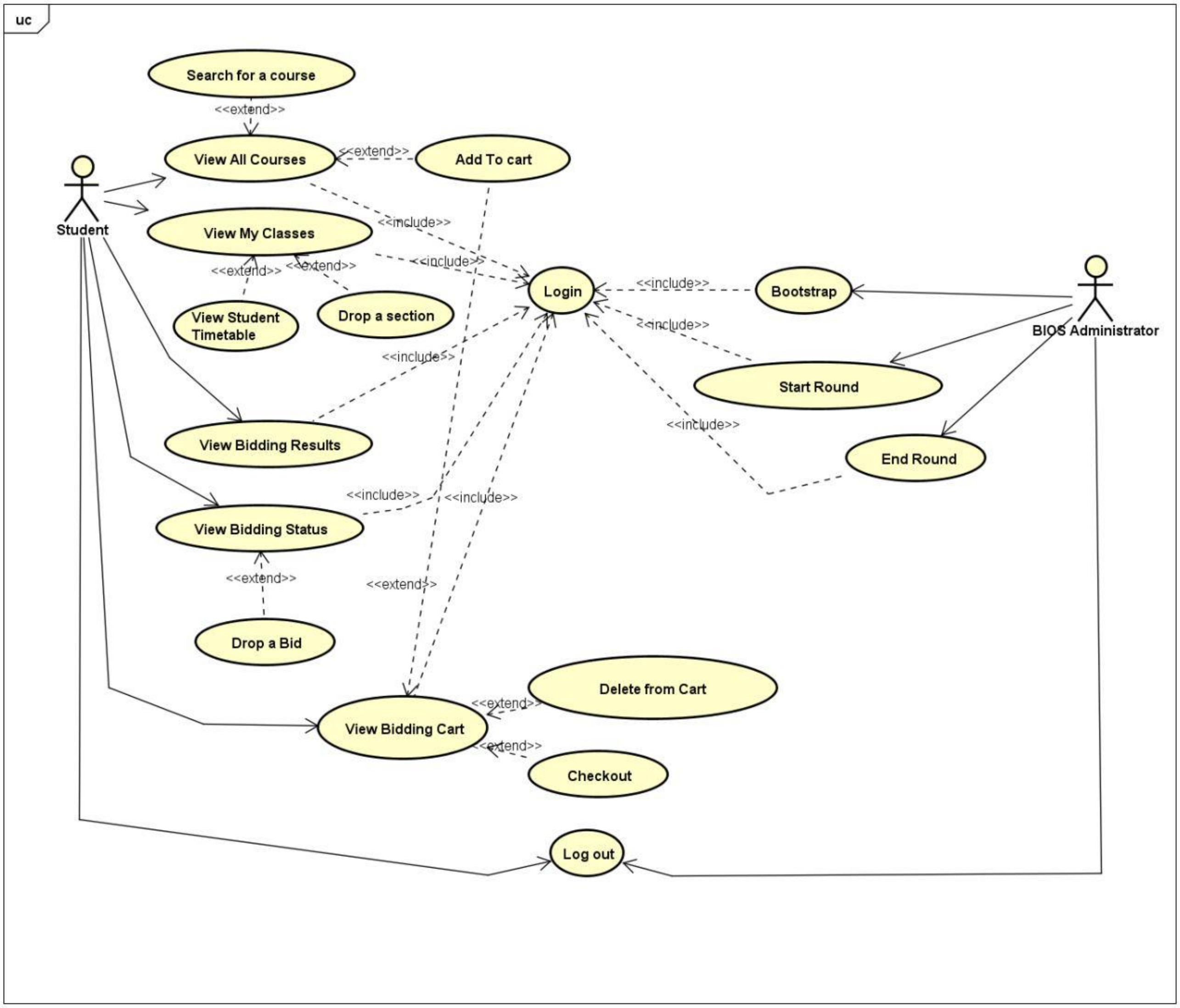
School of
Information Systems

IS203 Software Engineering Final Technical Documents

G6T3
Team Members:

He Yongsheng Wilson
Tan Ming Kwang
Eugene Tan Wei Hong
Cristabel Lau Yuit Theng
Quek Yew Kit

Use Case Diagram



Use Case Description

Bootstrap:

- Insert data into the database. It will conduct validations and return the correct number of entries inserted, as well as the number of rows with errors

Start Round / End round:

- Admin feature. This feature allows admin to start and stop a round and trigger the system to perform round clearing logic to insert successful bids into database and vice versa.

Login:

- Administrator and students are required to log in to the system to continue accessing other functions

Log Out:

- Admin and student can terminate their session and exit the system.

Search for a course:

- Student is allowed to search for a course via Course code or course title and filter the results into a smaller list.

View All courses:

- Student can view all courses to bid for in current round. Round 1 sees pre filtering to only allow student to see and bid for own school courses only

Add to cart:

- Allows student to add bid for a particular section to shopping cart for review before checking out as a bid.

View My Classes:

- Student can view all enrolled sections for the upcoming semester after the round is closed.

View Student timetable:

- Student can view all enrolled courses schedules as part of a timetable format to allow easier reference.

Drop a section:

- Student can drop a particular section and be refunded full amount to re bid for another section of another module

View Bidding results:

- Student can view the results of a bidding round and all his/her bids when the round is closed, and even when round 2 is started in the homepage and in the view my classes page

View Bidding status:

- Student can view the bid status of all the bids placed.

Drop a bid:

- Student can drop his/her bids after placing his/her bids.

View Bidding cart:

- Allows student to see and review his bids before checkout collectively.

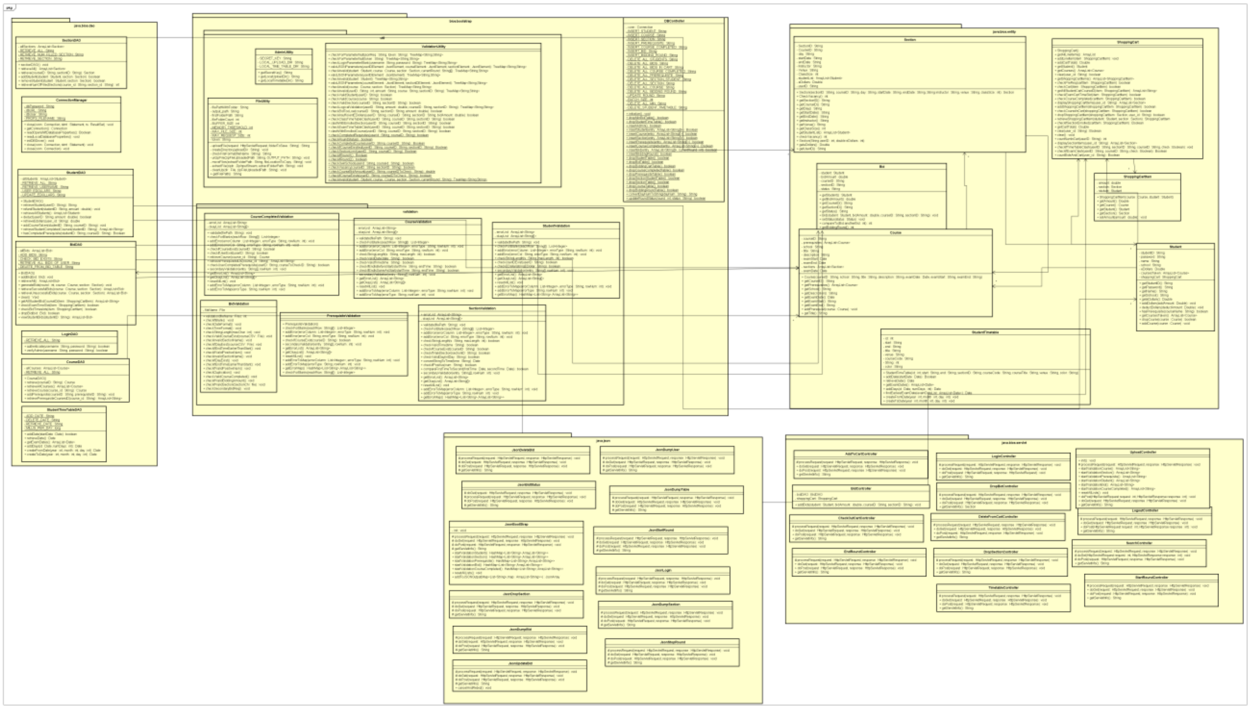
Delete from Cart:

- Allows student to drop a particular shopping cart item to add more items into their cart for checkout

Checkout:

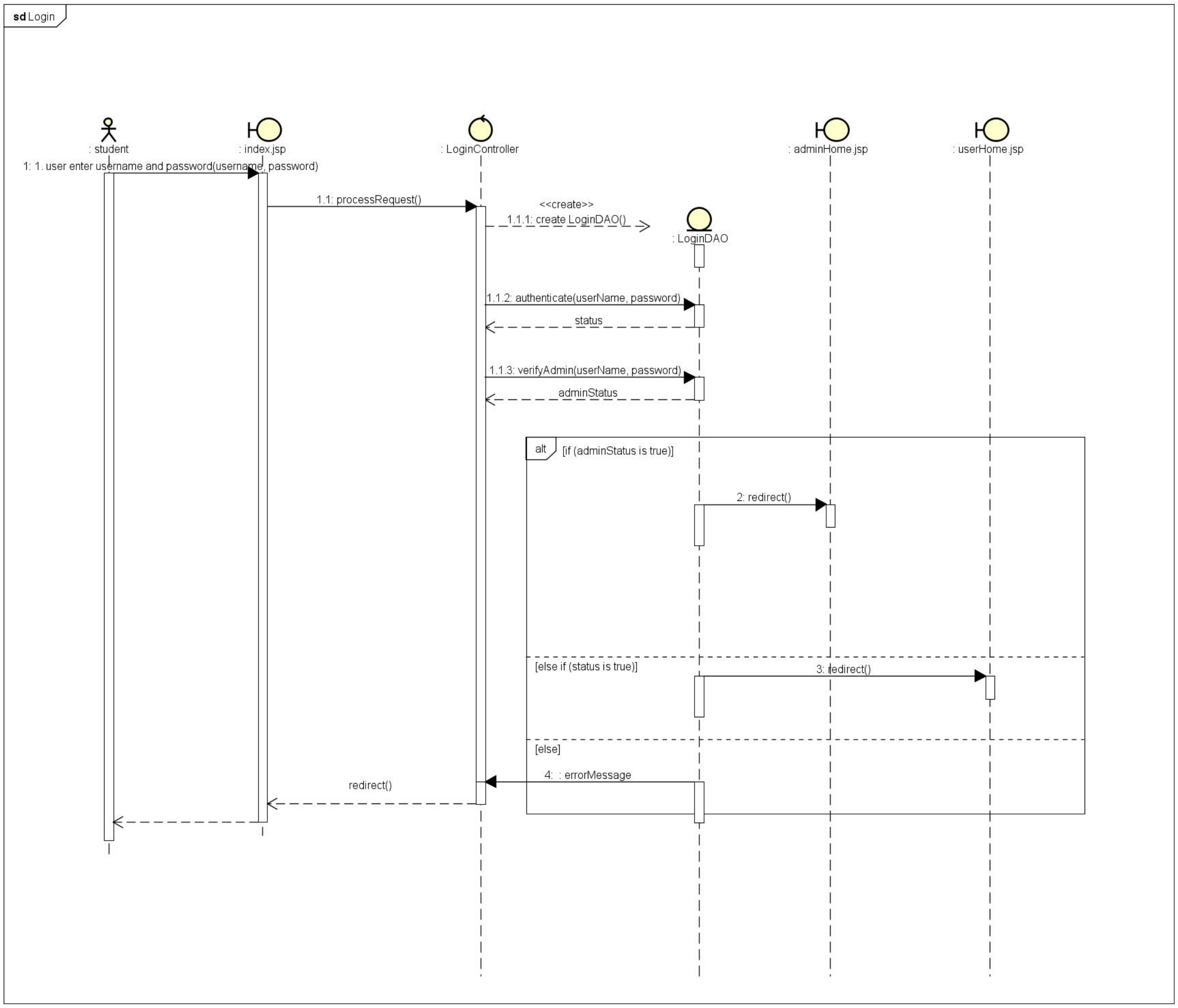
- Allows student to check-out all the bids collectively for easier reference and review.

Class Diagram

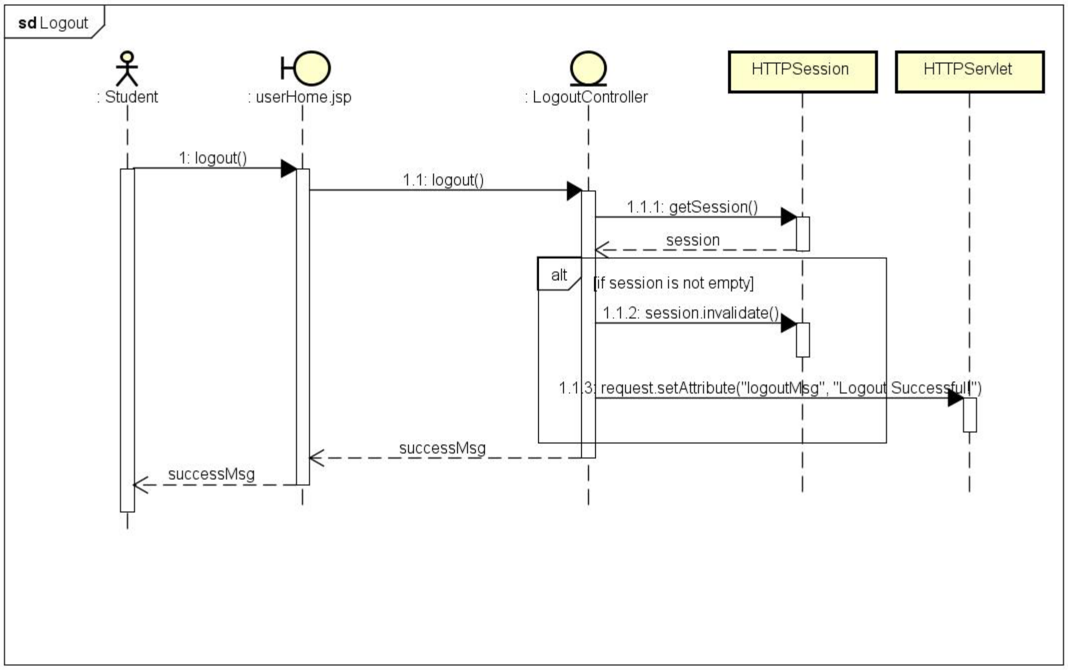


Sequence Diagram

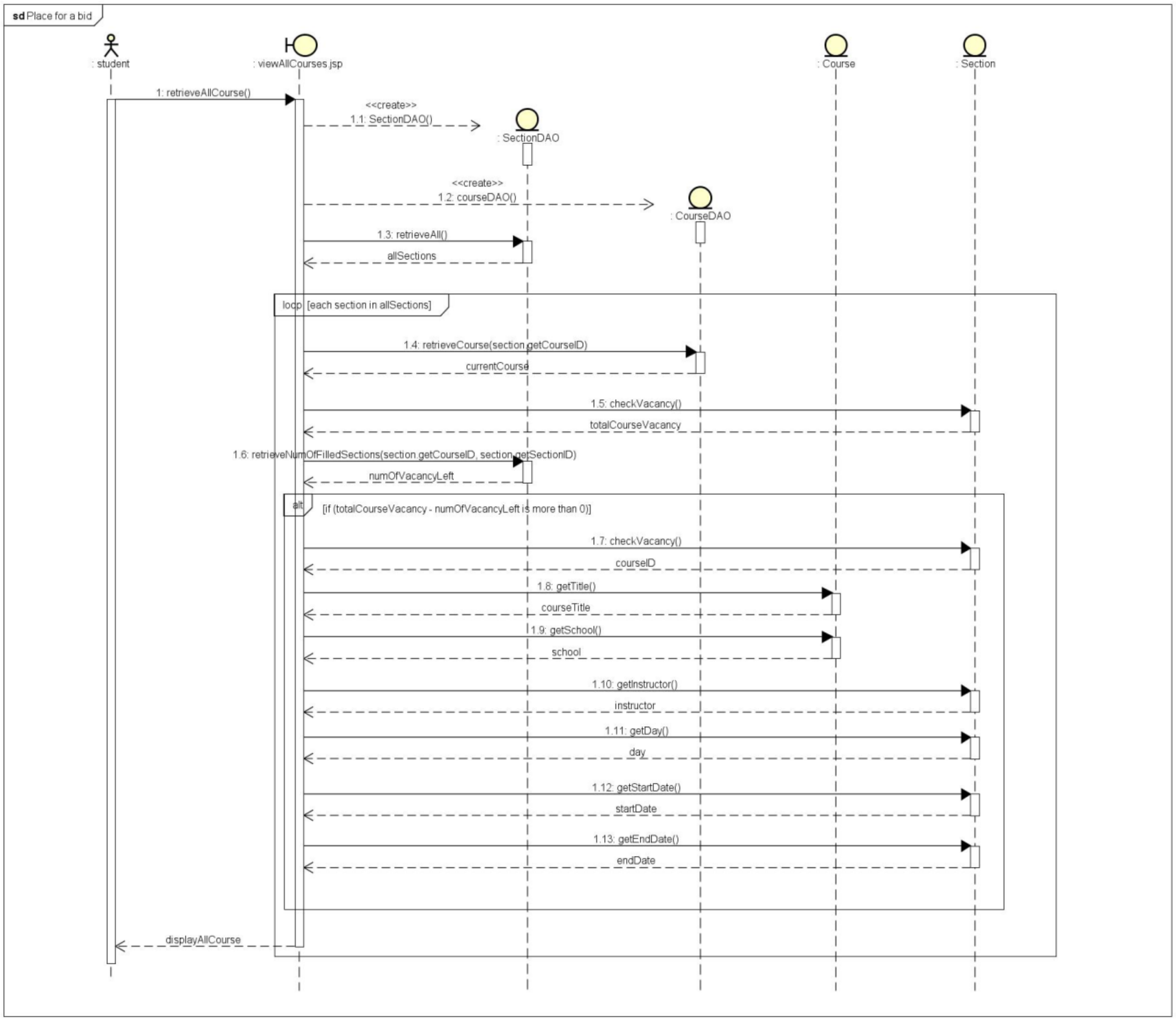
Login



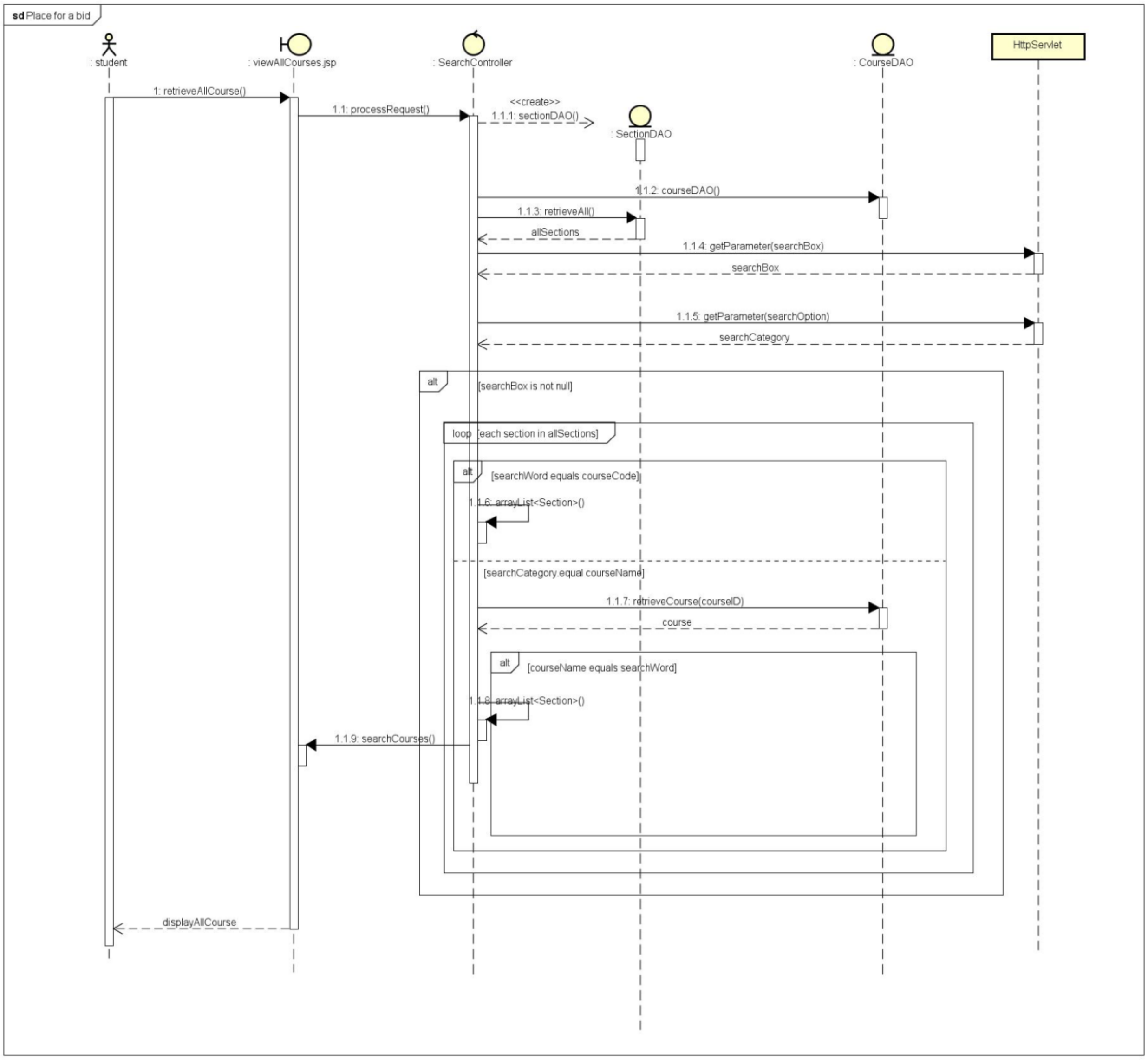
Logout


sd Logout
: Student
: userHome.jsp
: LogoutController
HTTPSession
HTTPServlet
1: logout()
1.1: logout()
1.1.1: getSession()
session
alt
[if session is not empty]
1.1.2: session.invalidate()
1.1.3: request.setAttribute("logoutMsg", "Logout Successful")
successMsg
successMsg

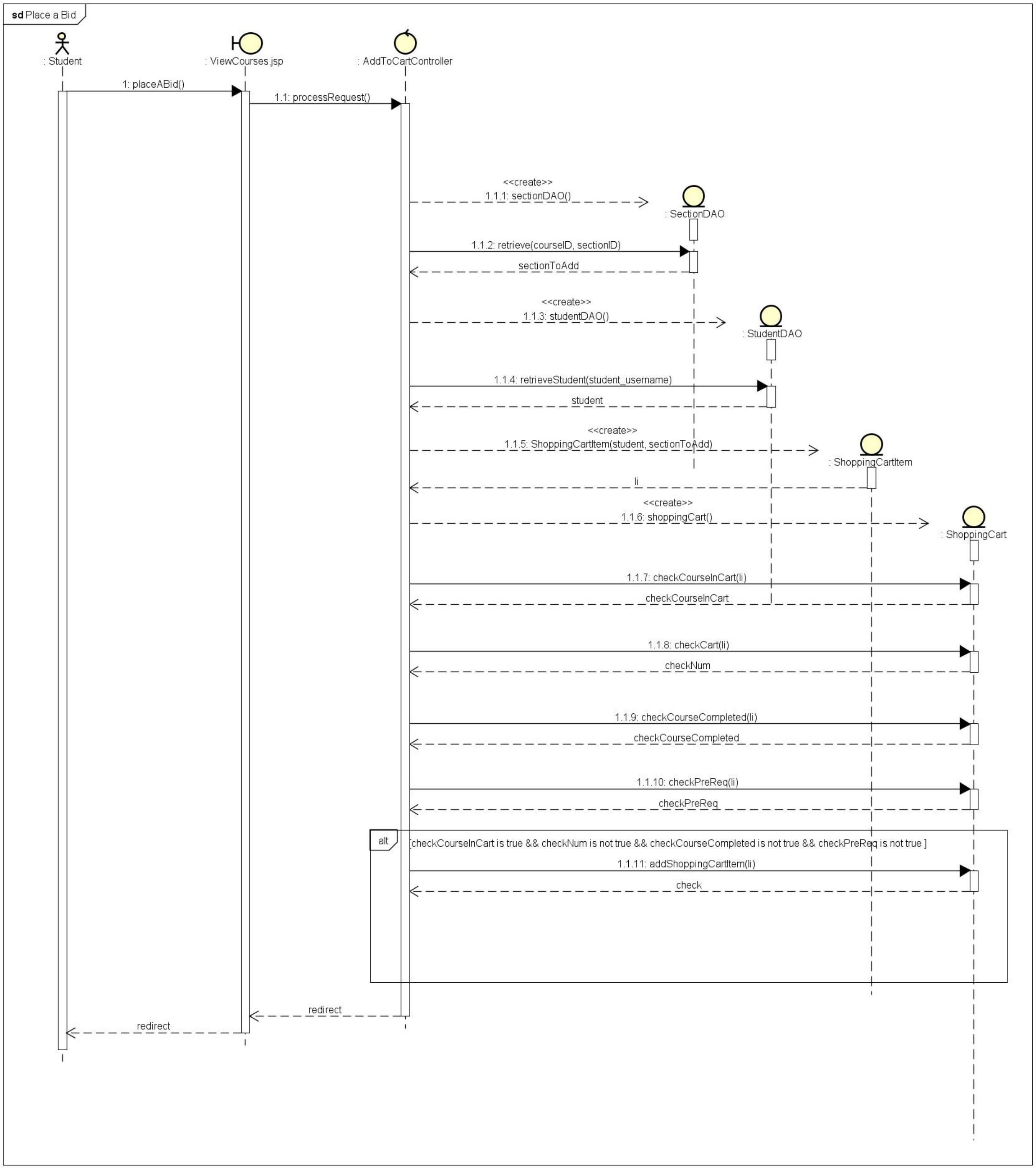
View All Courses



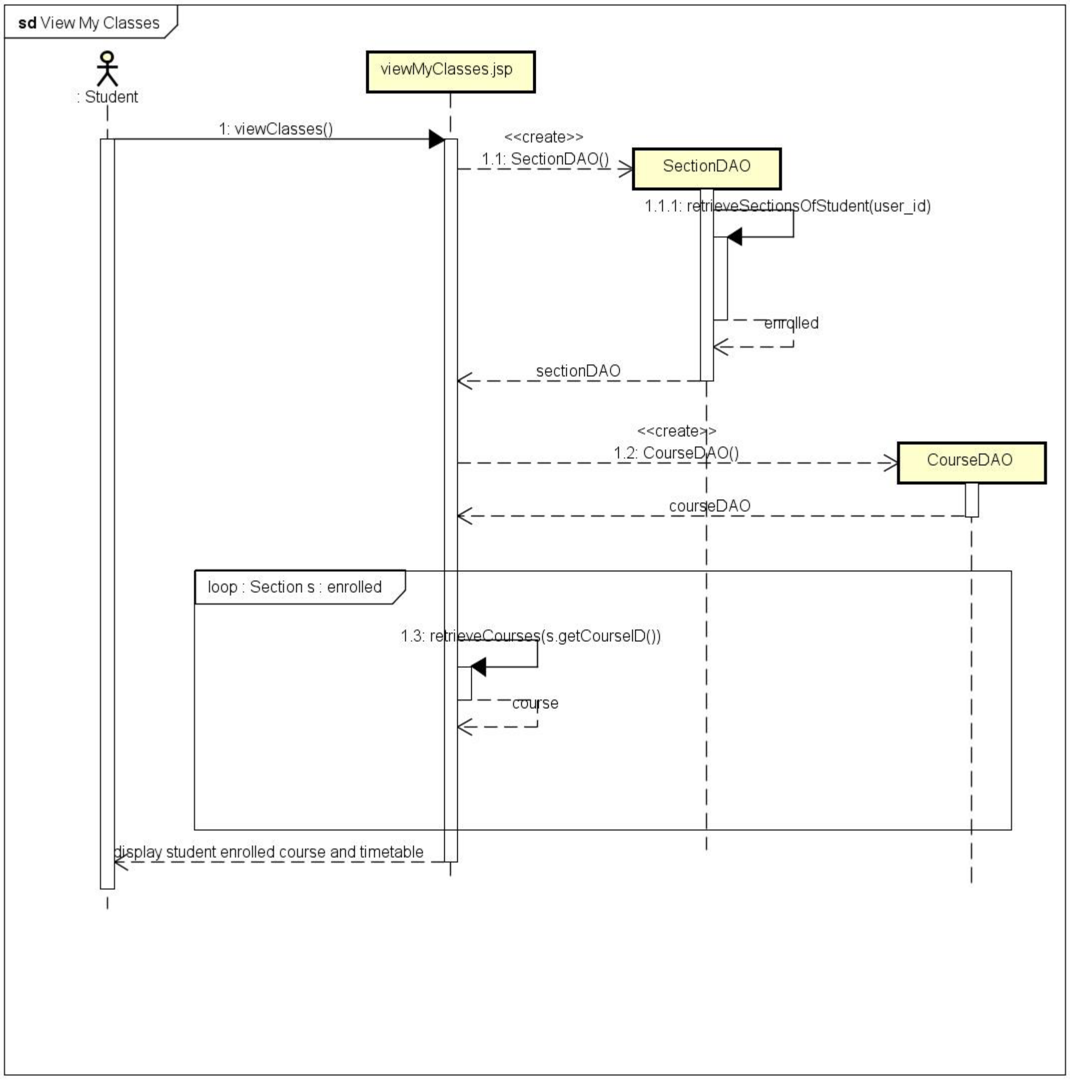
Search for a course



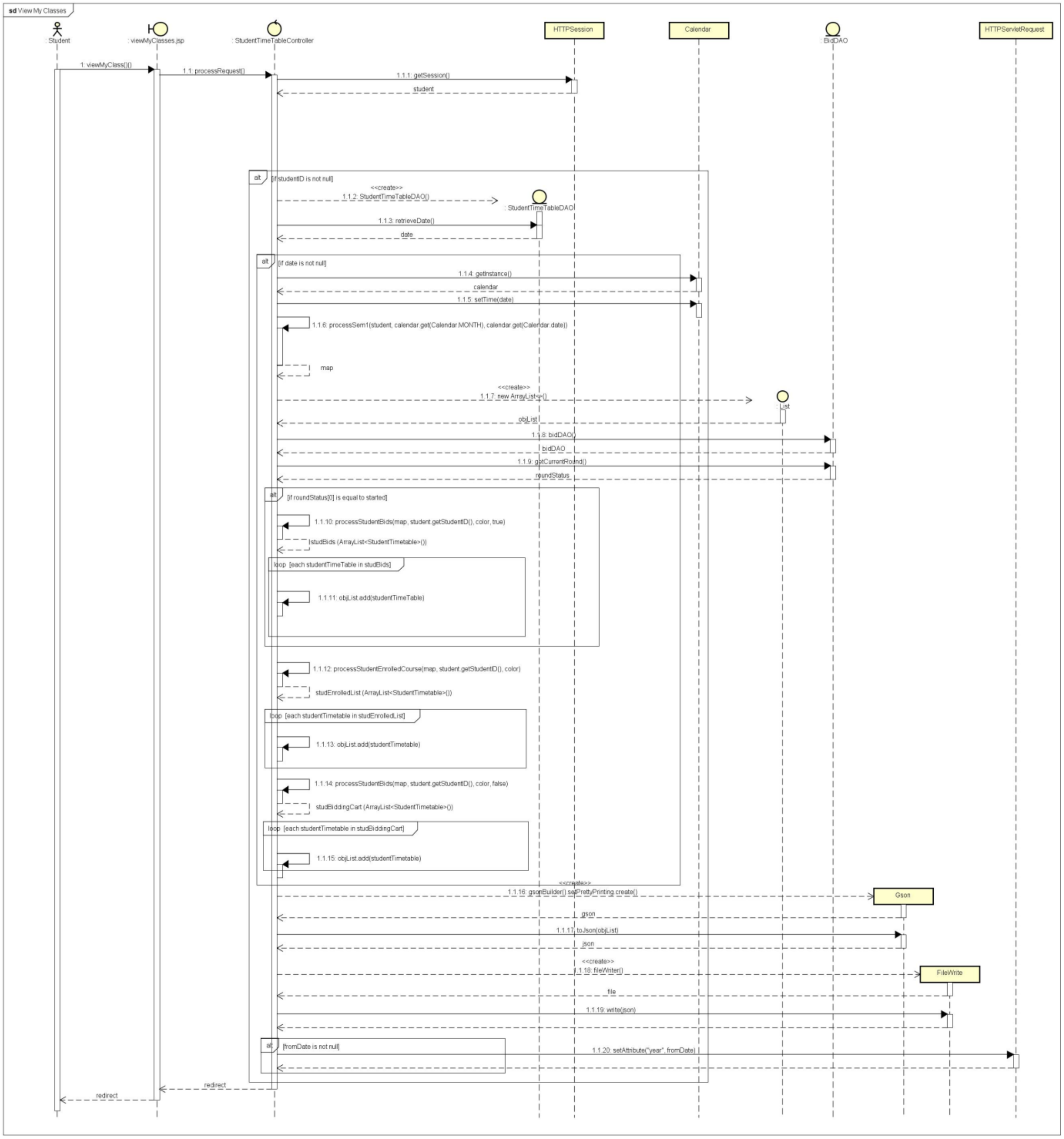
Add to Cart


sd Place a Bid
: Student
: ViewCourses.jsp
: AddToCartController
1: placeABid()
1.1: processRequest()
<<create>>
1.1.1: sectionDAO()
: SectionDAO
1.1.2: retrieve(courseID, sectionID)
sectionToAdd
<<create>>
1.1.3: studentDAO()
: StudentDAO
1.1.4: retrieveStudent(student_username)
student
<<create>>
1.1.5: ShoppingCartItem(student, sectionToAdd)
: ShoppingCartItem
li
<<create>>
1.1.6: shoppingCart()
: ShoppingCart
1.1.7: checkCourseInCart(li)
checkCourseInCart
1.1.8: checkCart(li)
checkNum
1.1.9: checkCourseCompleted(li)
checkCourseCompleted
1.1.10: checkPreReq(li)
checkPreReq
alt
[checkCourseInCart is true && checkNum is not true && checkCourseCompleted is not true && checkPreReq is not true]
1.1.11: addShoppingCartItem(li)
check
redirect
redirect

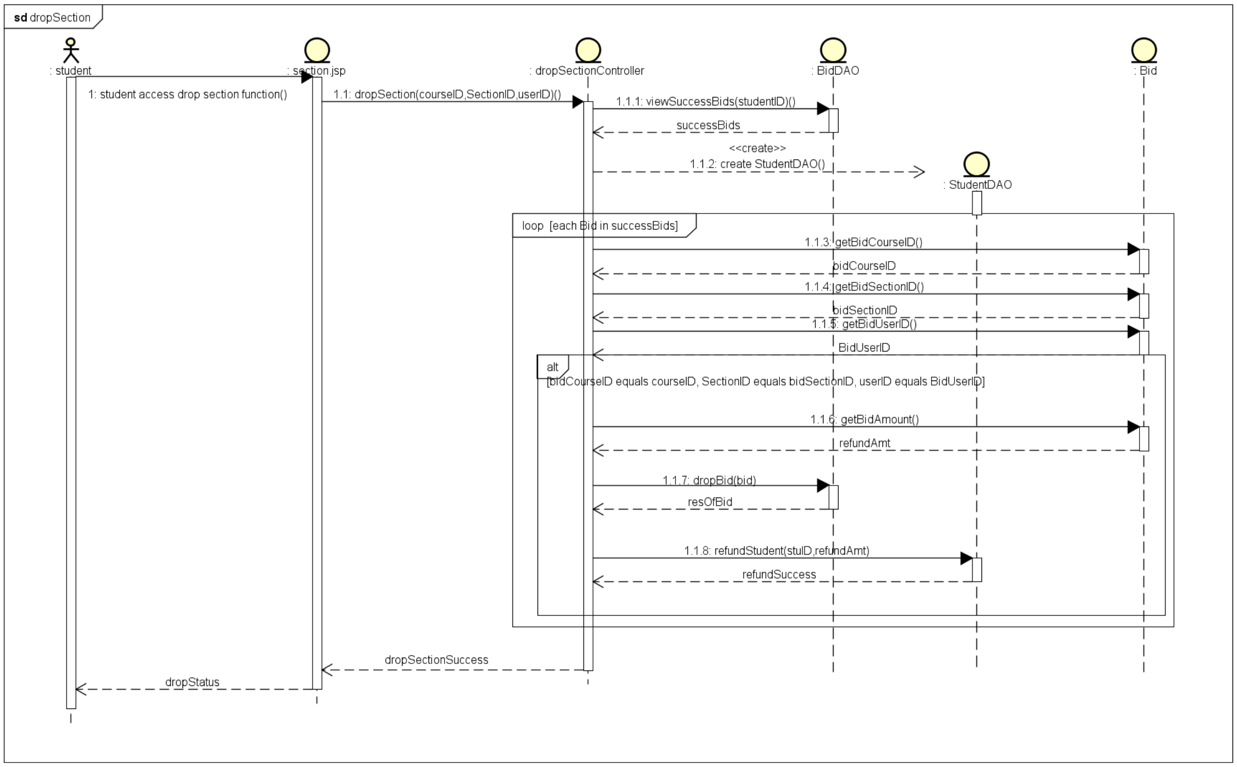
View My Classes



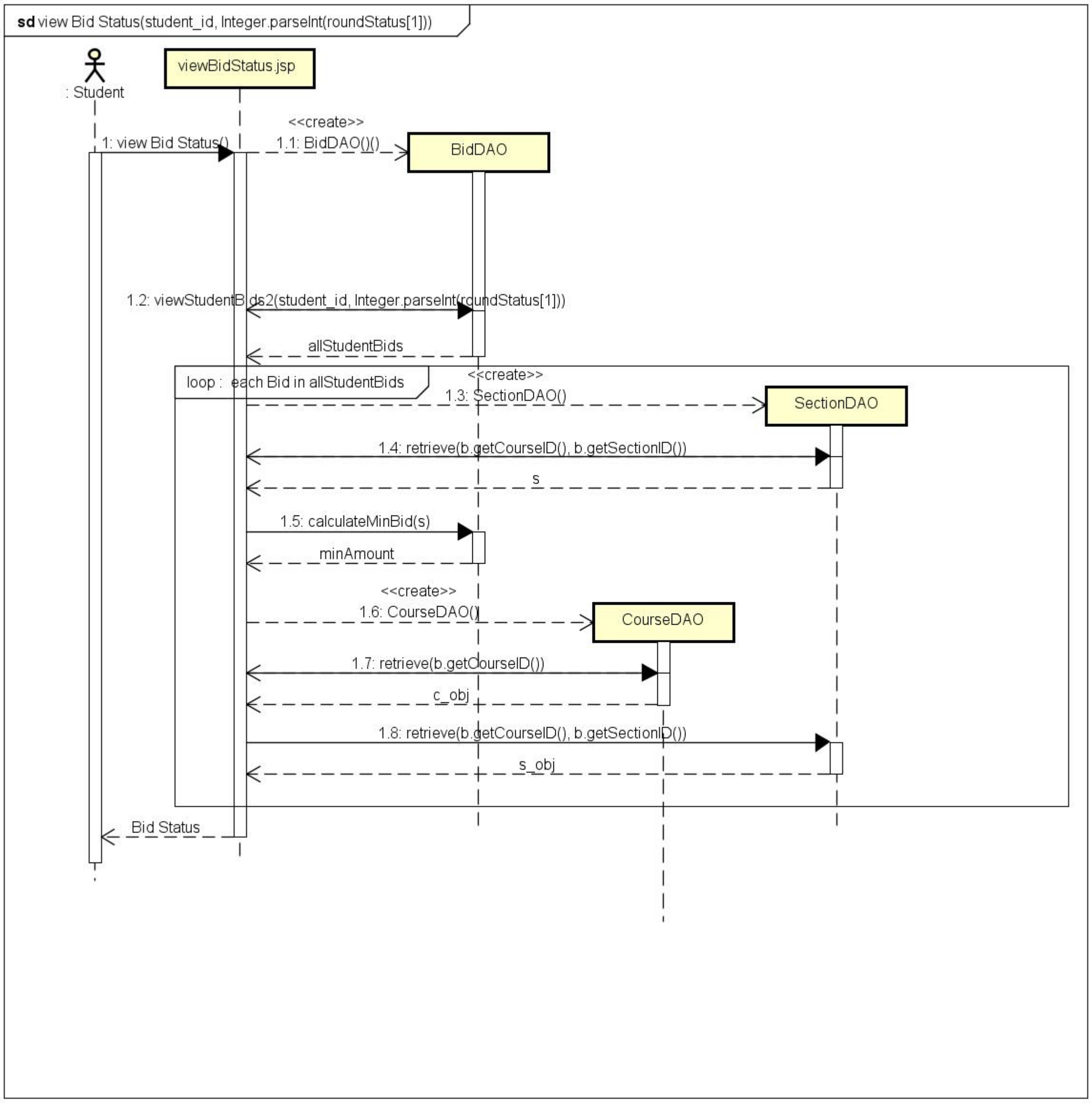
View Student Timetable


sd View My Classes
: Student
: viewMyClasses.jsp
: StudentTimeTableController
HTTPSession
Calendar
: BidDAO
HTTPServletRequest
1: viewMyClass()()
1.1: processRequest()
1.1.1: getSession()
student
alt
[if studentID is not null]
<<create>>
1.1.2: StudentTimeTableDAO()
: StudentTimeTableDAO
1.1.3: retrieveDate()
date
alt
[if date is not null]
1.1.4: getInstance()
calendar
1.1.5: setTime(date)
1.1.6: processSem1(student, calendar.get(Calendar.MONTH), calendar.get(Calendar.date))
map
<<create>>
1.1.7: new ArrayList<>()
: List
objList
1.1.8: bidDAO()
bidDAO
1.1.9: getCurrentRound()
roundStatus
alt
[if roundStatus[0] is equal to started]
1.1.10: processStudentBids(map, student.getStudentID(), color, true)
studBids (ArrayList<StudentTimetable>())
loop [each studentTimeTable in studBids]
1.1.11: objList.add(studentTimeTable)
1.1.12: processStudentEnrolledCourse(map, student.getStudentID(), color)
studEnrolledList (ArrayList<StudentTimetable>())
loop [each studentTimetable in studEnrolledList]
1.1.13: objList.add(studentTimetable)
1.1.14: processStudentBids(map, student.getStudentID(), color, false)
studBiddingCart (ArrayList<StudentTimetable>())
loop [each studentTimetable in studBiddingCart]
1.1.15: objList.add(studentTimetable)
<<create>>
1.1.16: gsonBuilder().setPrettyPrinting.create()
Gson
gson
1.1.17: toJson(objList)
json
<<create>>
1.1.18: fileWriter()
FileWrite
file
1.1.19: write(json)
alt
[fromDate is not null]
1.1.20: setAttribute("year", fromDate)
redirect
redirect

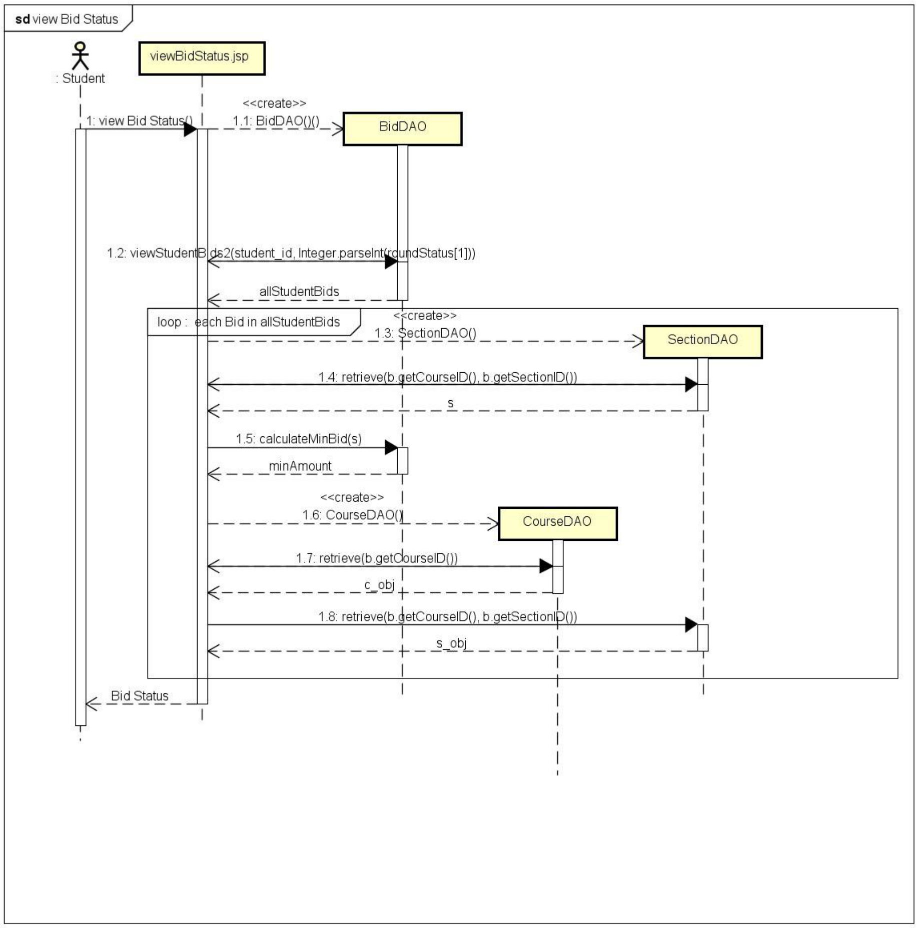
Drop a section



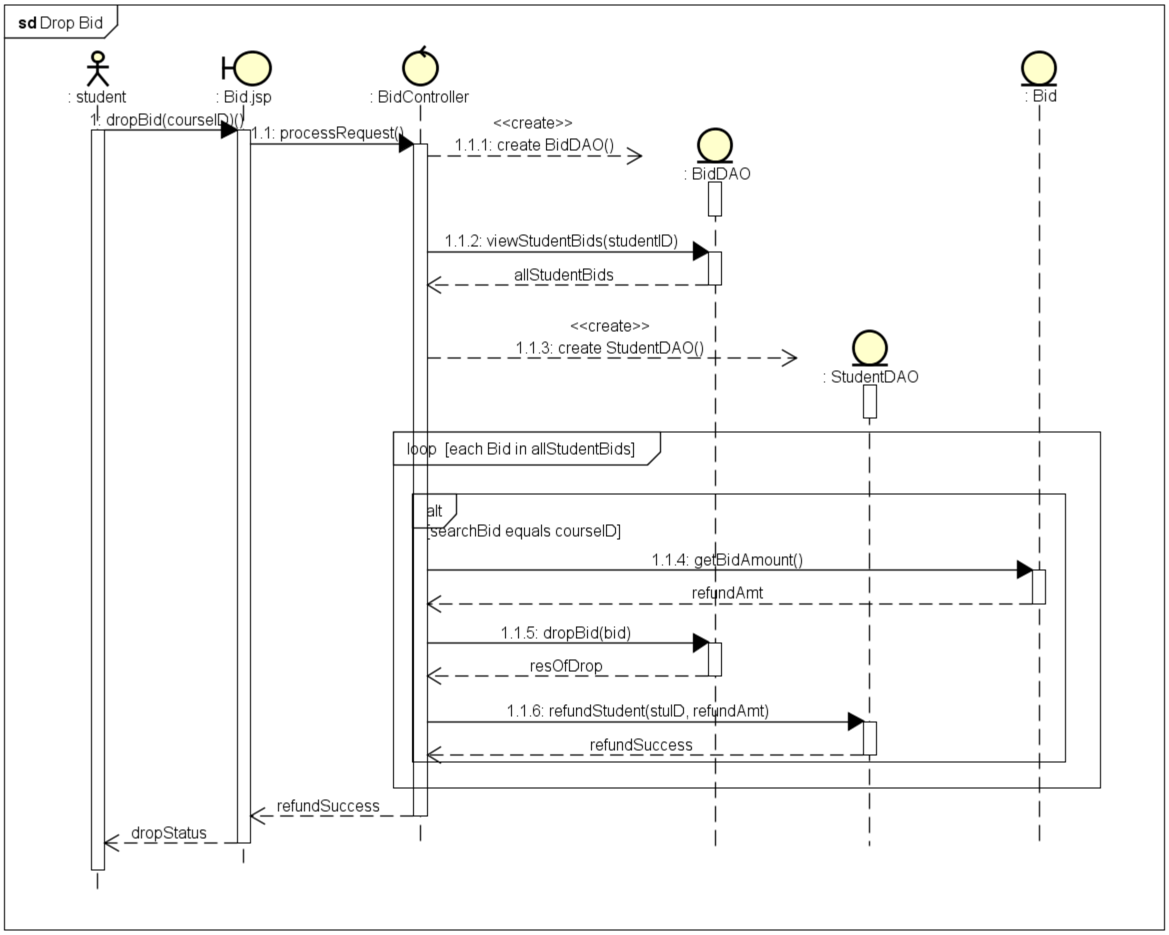
View Bidding results


sd view Bid Status(student_id, Integer.parseInt(roundStatus[1]))
: Student
viewBidStatus.jsp
<<create>>
1: view Bid Status()
1.1: BidDAO()()
BidDAO
1.2: viewStudentBids2(student_id, Integer.parseInt(roundStatus[1]))
allStudentBids
loop : each Bid in allStudentBids
<<create>>
1.3: SectionDAO()
SectionDAO
1.4: retrieve(b.getCourseID(), b.getSectionID())
s
1.5: calculateMinBid(s)
minAmount
<<create>>
1.6: CourseDAO()
CourseDAO
1.7: retrieve(b.getCourseID())
c_obj
1.8: retrieve(b.getCourseID(), b.getSectionID())
s_obj
Bid Status

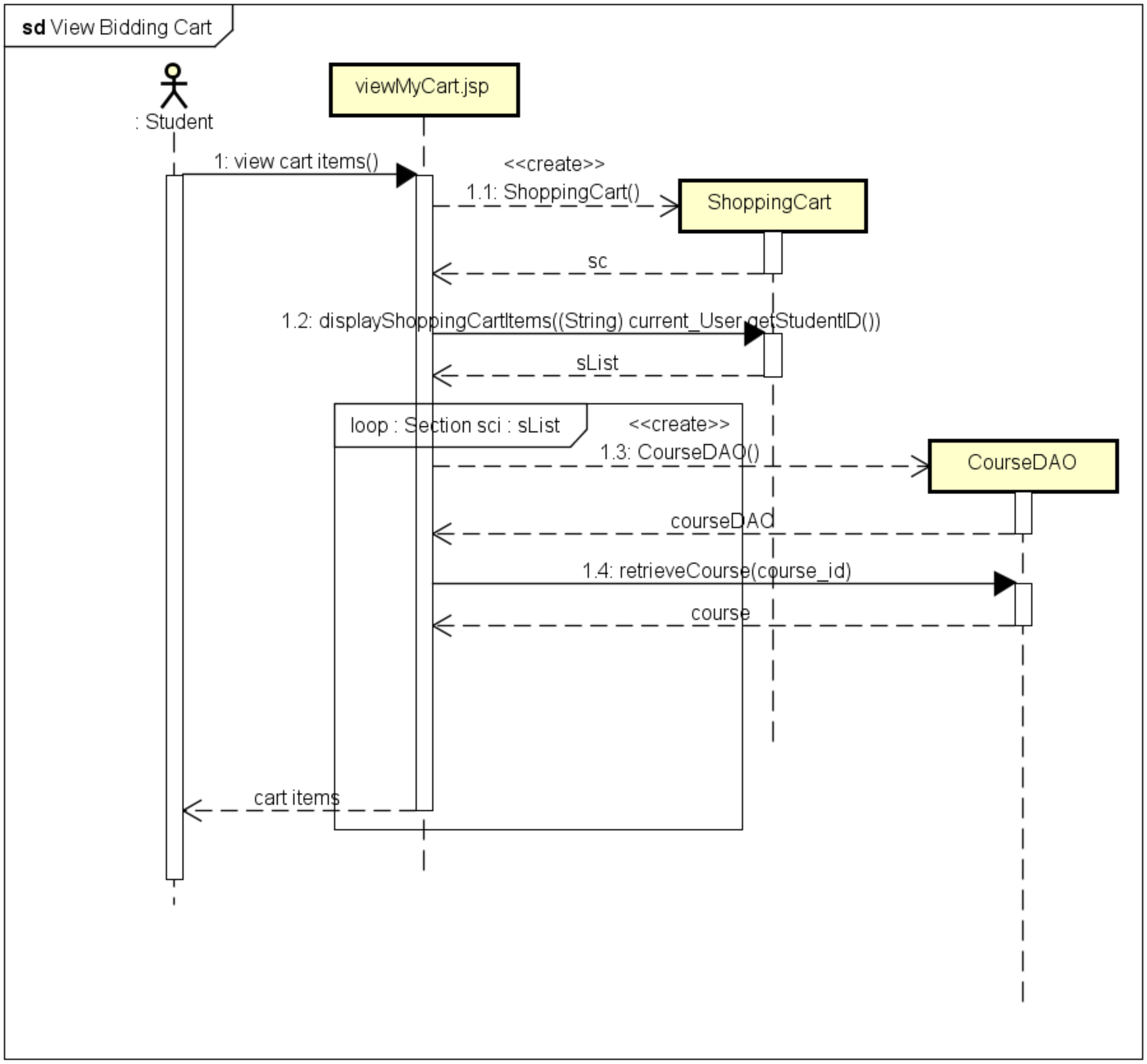
View Bidding Status



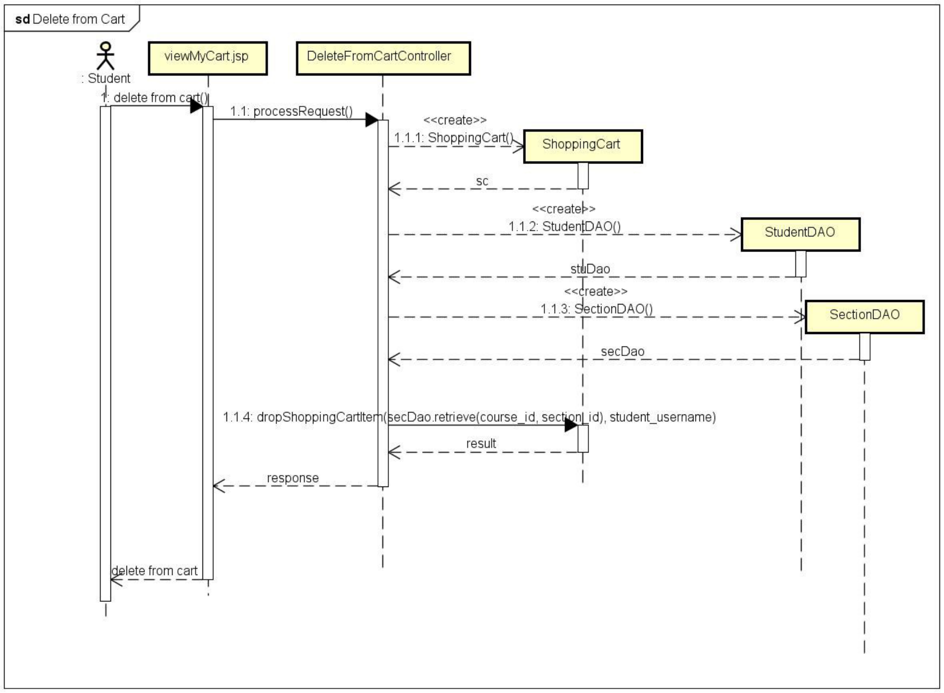
Drop a Bid


sd Drop Bid
: student
: Bid.jsp
: BidController
: Bid
1: dropBid(courseID)()
1.1: processRequest()
<<create>>
1.1.1: create BidDAO()
: BidDAO
1.1.2: viewStudentBids(studentID)
allStudentBids
<<create>>
1.1.3: create StudentDAO()
: StudentDAO
loop [each Bid in allStudentBids]
alt
[searchBid equals courseID]
1.1.4: getBidAmount()
refundAmt
1.1.5: dropBid(bid)
resOfDrop
1.1.6: refundStudent(stuID, refundAmt)
refundSuccess
refundSuccess
dropStatus

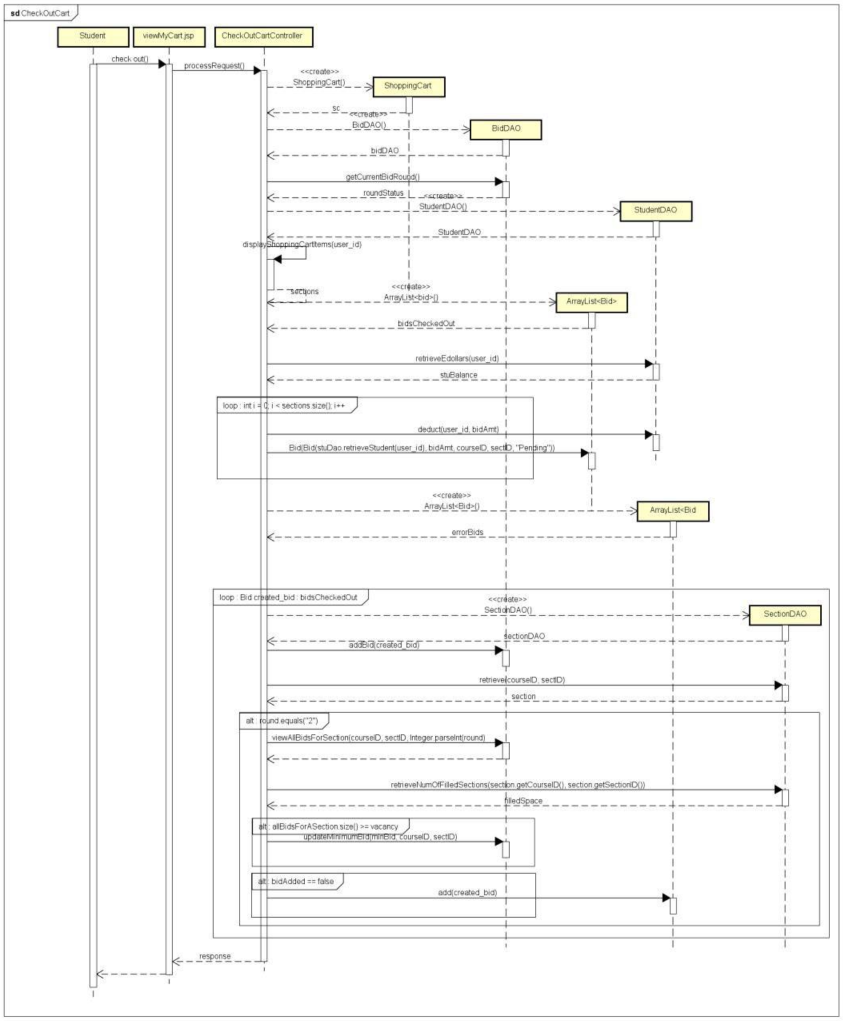
View Bidding Cart



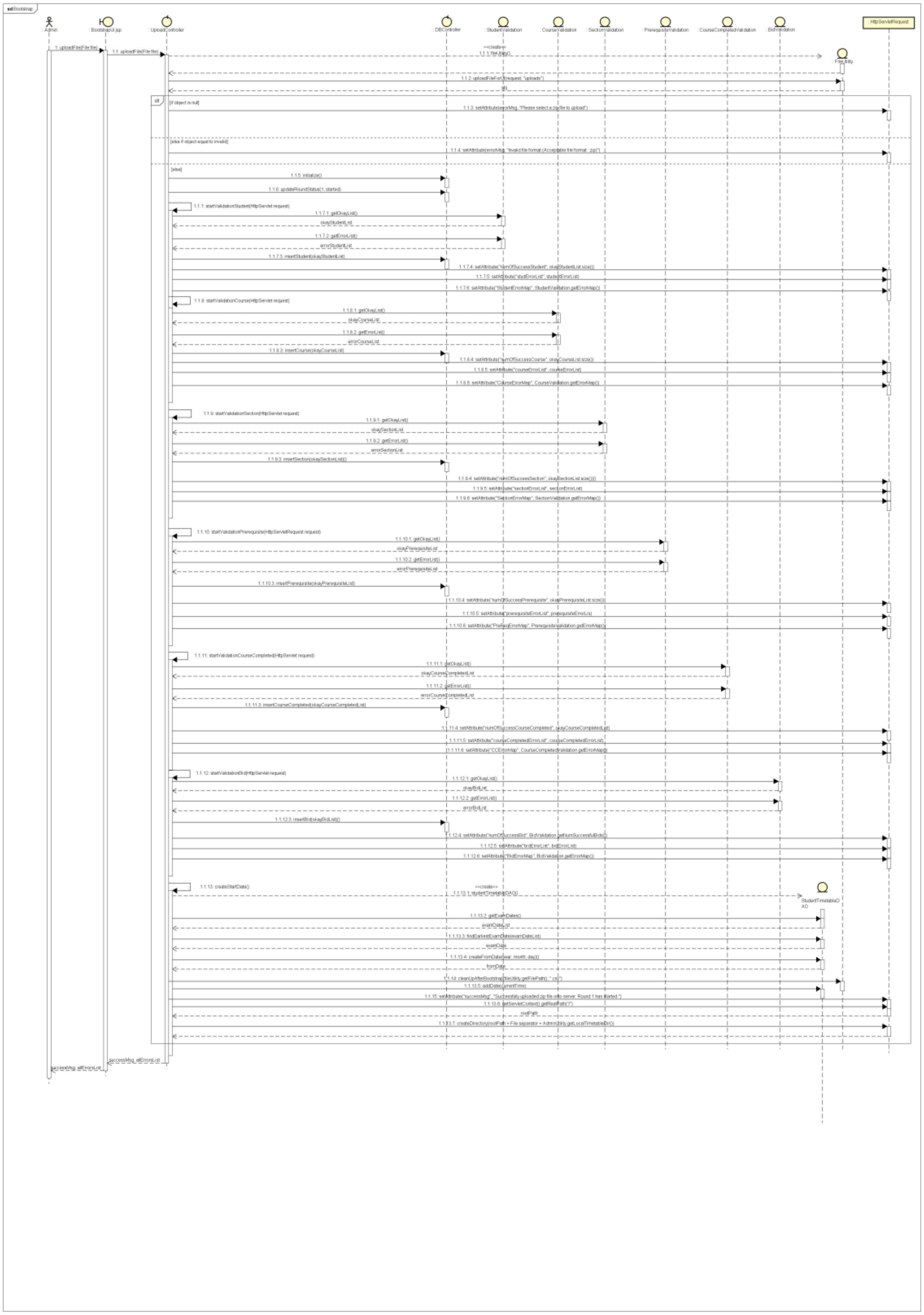
Delete from Cart



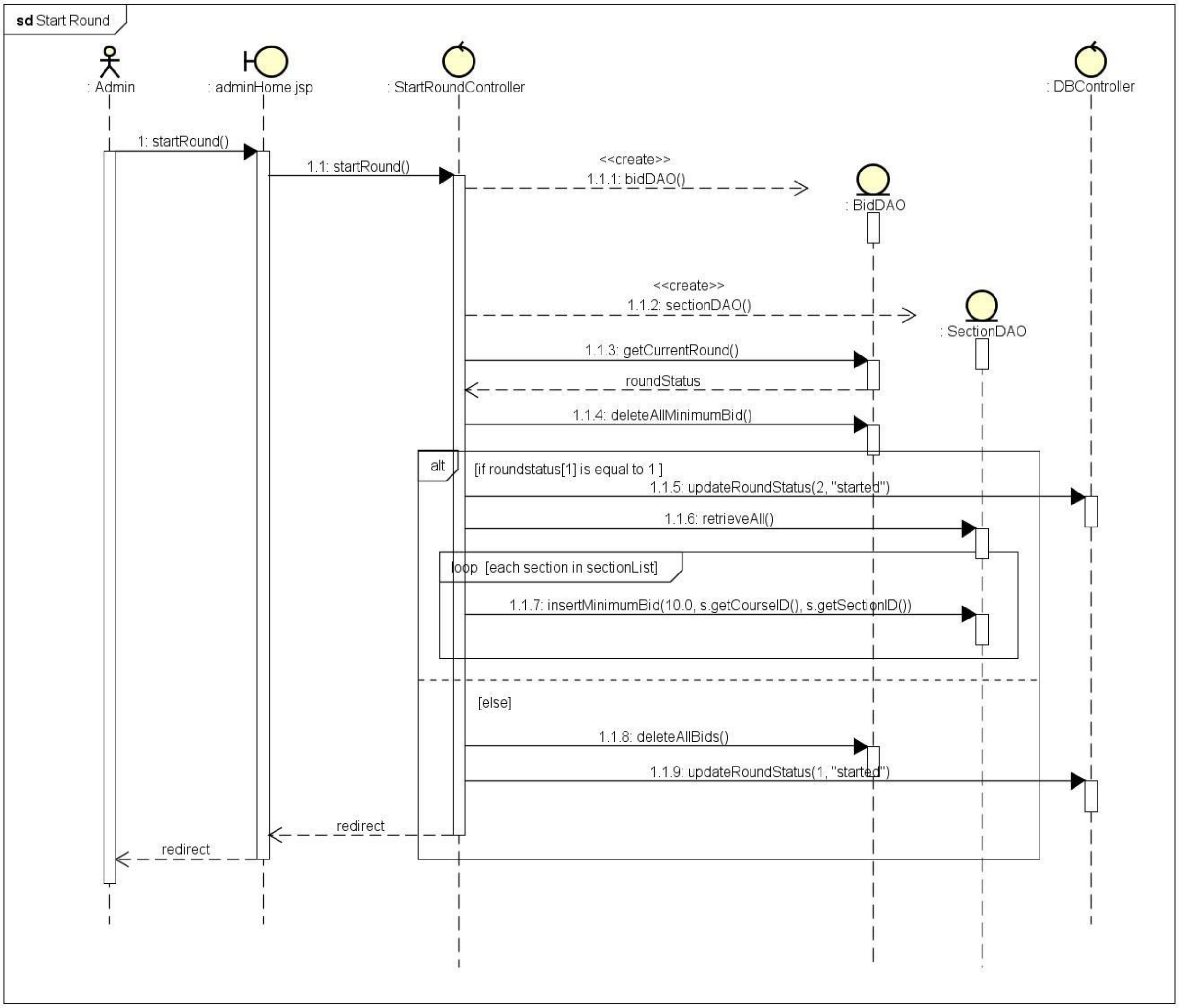
Checkout


sd CheckOutCart
Student
viewMyCart.jsp
CheckOutCartController
check out()
processRequest()
<<create>>
ShoppingCart()
ShoppingCart
sc
<<create>>
BidDAO()
BidDAO
bidDAO
getCurrentBidRound()
roundStatus
<<create>>
StudentDAO()
StudentDAO
StudentDAO
displayShoppingCartItems(user_id)
sections
<<create>>
ArrayList<bid>()
ArrayList<Bid>
bidsCheckedOut
retrieveEdollars(user_id)
stuBalance
loop : int i = 0; i < sections.size(); i++
deduct(user_id, bidAmt)
Bid(Bid(stuDao.retrieveStudent(user_id), bidAmt, courseID, sectID, "Pending"))
<<create>>
ArrayList<Bid>()
ArrayList<Bid
errorBids
loop : Bid created_bid : bidsCheckedOut
<<create>>
SectionDAO()
SectionDAO
sectionDAO
addBid(created_bid)
retrieve(courseID, sectID)
section
alt : round.equals("2")
viewAllBidsForSection(courseID, sectID, Integer.parseInt(round)
retrieveNumOfFilledSections(section.getCourseID(), section.getSectionID())
filledSpace
alt : allBidsForASection.size() >= vacancy
updateMinimumBid(minBid, courseID, sectID)
alt : bidAdded == false
add(created_bid)
response

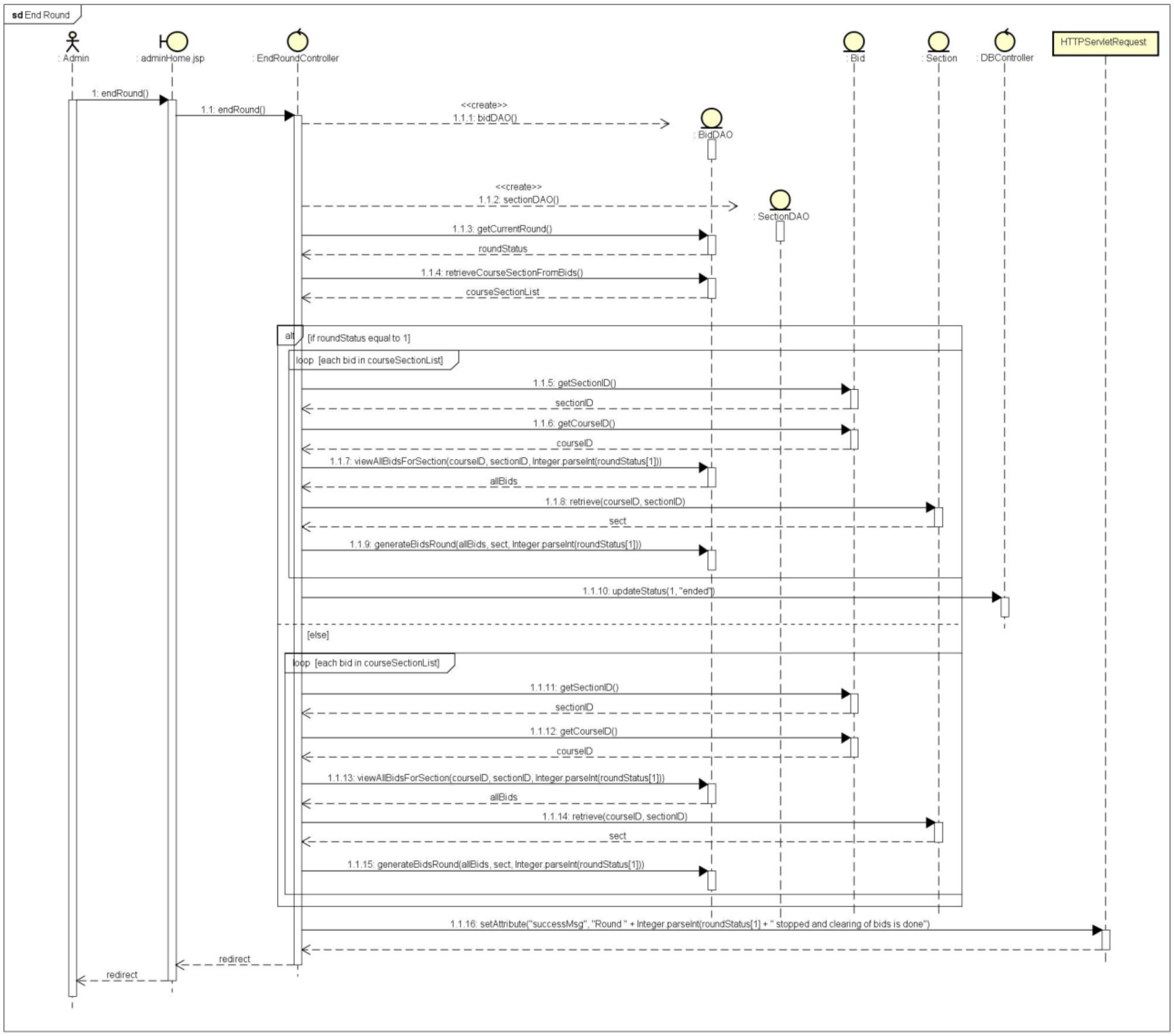
Bootstrap


sd Bootstrap
: Admin
: BootstrapUI.jsp
: UploadController
: DBController
: StudentValidation
: CourseValidation
: SectionValidation
: PrerequisiteValidation
: CourseCompletedValidation
: BidValidation
: HttpServletRequest
1: uploadFile(File file)
1.1: uploadFile(File file)
<<create>>
1.1.1: fileUtility()
: FileUtility
1.1.2: uploadFileForUI(request, "uploads")
obj
alt
[if object is null]
1.1.3: setAttribute(errorMsg, "Please select a zip file to upload")
[else if object equal to invalid]
1.1.4: setAttribute(errorMsg, "Invalid file format (Acceptable file format: .zip)")
[else]
1.1.5: initialize()
1.1.6: updateRoundStatus(1, started)
1.1.7: startValidationStudent(HttpServlet request)
1.1.7.1: getOkayList()
okayStudentList
1.1.7.2: getErrorList()
errorStudentList
1.1.7.3: insertStudent(okayStudentList)
1.1.7.4: setAttribute("numOfSuccessStudent", okayStudentList.size())
1.1.7.5: setAttribute("studErrorList", studentErrorList)
1.1.7.6: setAttribute("StudentErrorMap", StudentValidation.getErrorMap())
1.1.8: startValidationCourse(HttpServlet request)
1.1.8.1: getOkayList()
okayCourseList
1.1.8.2: getErrorList()
errorCourseList
1.1.8.3: insertCourse(okayCourseList)
1.1.8.4: setAttribute("numOfSuccessCourse", okayCourseList.size()
1.1.8.5: setAttribute("courseErrorList", courseErrorList)
1.1.8.6: setAttribute("CourseErrorMap", CourseValidation.getErrorMap())
1.1.9: startValidationSection(HttpServlet request)
1.1.9.1: getOkayList()
okaySectionList
1.1.9.2: getErrorList()
errorSectionList
1.1.9.3: insertSection(okaySectionList))
1.1.9.4: setAttribute("numOfSuccessSection", okaySectionList.size()))
1.1.9.5: setAttribute("sectionErrorList", sectionErrorList)
1.1.9.6: setAttribute("SectionErrorMap", SectionValidation.getErrorMap())
1.1.10: startValidationPrerequisite(HttpServletRequest request)
1.1.10.1: getOkayList()
okayPrerequisiteList
1.1.10.2: getErrorList()
errorPrerequisiteList
1.1.10.3: insertPrerequisite(okayPrerequisiteList)
1.1.10.4: setAttribute("numOfSuccessPrerequisite", okayPrerequisiteList.size())
1.1.10.5: setAttribute("prerequisiteErrorList", prerequisiteErrorLis)
1.1.10.6: setAttribute("PrereqErrorMap", PrerequisiteValidation.getErrorMap())
1.1.11: startValidationCourseCompleted(HttpServlet request)
1.1.11.1: getOkayList()
okayCourseCompletedList
1.1.11.2: getErrorList()
errorCourseCompletedList
1.1.11.3: insertCourseCompleted(okayCourseCompletedList)
1.1.11.4: setAttribute("numOfSuccessCourseCompleted", okayCourseCompletedList)
1.1.11.5: setAttribute("courseCompletedErrorList", courseCompletedErrorList)
1.1.11.6: setAttribute("CCErrorMap", CourseCompletedValidation.getErrorMap())
1.1.12: startValidationBid(HttpServlet request)
1.1.12.1: getOkayList()
okayBidList
1.1.12.2: getErrorList()
errorBidList
1.1.12.3: insertBid(okayBidList))
1.1.12.4: setAttribute("numOfSuccessBid", BidValidation.getNumSuccessfulBids())
1.1.12.5: setAttribute("bidErrorList", bidErrorList)
1.1.12.6: setAttribute("BidErrorMap", BidValidation.getErrorMap())
1.1.13: createStartDate()
<<create>>
1.1.13.1: studentTimetableDAO()
StudentTimetableDAO
1.1.13.2: getExamDates()
examDateList
1.1.13.3: findEarliestExamDate(examDateList)
examDate
1.1.13.4: createFromDate(year, month, day))
fromDate
1.1.14: cleanUpAfterBootstrap(fileUtility.getFilePath(), ".csv")
1.1.13.5: addDate(currentTime)
1.1.15: setAttribute("successMsg", "Successfully uploaded zip file onto server. Round 1 has started.")
1.1.13.6: getServletContext().getRealPath("/")
rootPath
1.1.13.7: createDirectory(rootPath + File.separator + AdminUtility.getLocalTimetableDir())
successMsg, allErrorsList
successMsg, allErrorsList

Start Round


sd Start Round
: Admin
: adminHome.jsp
: StartRoundController
: DBController
1: startRound()
1.1: startRound()
<<create>>
1.1.1: bidDAO()
: BidDAO
<<create>>
1.1.2: sectionDAO()
: SectionDAO
1.1.3: getCurrentRound()
roundStatus
1.1.4: deleteAllMinimumBid()
alt
[if roundstatus[1] is equal to 1]
1.1.5: updateRoundStatus(2, "started")
1.1.6: retrieveAll()
loop [each section in sectionList]
1.1.7: insertMinimumBid(10.0, s.getCourseID(), s.getSectionID())
[else]
1.1.8: deleteAllBids()
1.1.9: updateRoundStatus(1, "started")
redirect
redirect

End Round


sd End Round
: Admin
: adminHome.jsp
: EndRoundController
: Bid
: Section
: DBController
HTTPServletRequest
1: endRound()
1.1: endRound()
<<create>>
1.1.1: bidDAO()
: BidDAO
<<create>>
1.1.2: sectionDAO()
: SectionDAO
1.1.3: getCurrentRound()
roundStatus
1.1.4: retrieveCourseSectionFromBids()
courseSectionList
alt
[if roundStatus equal to 1]
loop [each bid in courseSectionList]
1.1.5: getSectionID()
sectionID
1.1.6: getCourseID()
courseID
1.1.7: viewAllBidsForSection(courseID, sectionID, Integer.parseInt(roundStatus[1]))
allBids
1.1.8: retrieve(courseID, sectionID)
sect
1.1.9: generateBidsRound(allBids, sect, Integer.parseInt(roundStatus[1]))
1.1.10: updateStatus(1, "ended")
[else]
loop [each bid in courseSectionList]
1.1.11: getSectionID()
sectionID
1.1.12: getCourseID()
courseID
1.1.13: viewAllBidsForSection(courseID, sectionID, Integer.parseInt(roundStatus[1]))
allBids
1.1.14: retrieve(courseID, sectionID)
sect
1.1.15: generateBidsRound(allBids, sect, Integer.parseInt(roundStatus[1]))
1.1.16: setAttribute("successMsg", "Round " + Integer.parseInt(roundStatus[1] + " stopped and clearing of bids is done")
redirect
redirect