By Wednesday, December 3rd:

Since the last implementation deadline, Question and Response GUIs have been built and added to SystemUI. Rather than let the professor type the parameter they will ask about, we'll be implementing a drop-down menu for them to choose from for demonstration purposes. For now, the only parameters being added would be GPA and/or experience, but in a real-world situation, this could be expanded to any number of things. Likewise, the generated groups now output to the interface, rather than to the console. This allows for a more "Real world" display of how the system may function.

Following this, primarily what should be accomplished is that the "Decent Generation" strategy is complete. This would generate groups based on similar GPAs and experience. This functions as a proof-of-concept of the ability to generate organized groups, instead of just at random. In reality, if there was more time to implement the other features, schedules and preferences would also be taken into account. By demonstrating the GPA, it shows that there is a working system for other strategies as well as splitting the students into more fair groups.
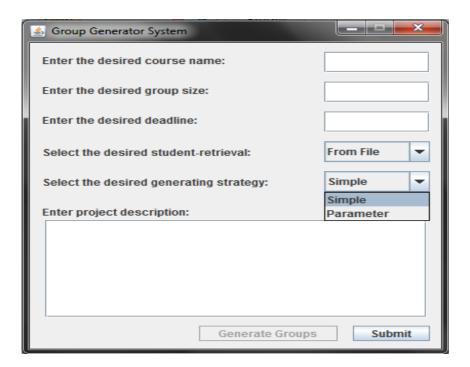
In summary:

1. Create a user interface that allows Professor to enter specifics about course (Features 2,3,4,5)
2. Display generated groups to the user interface (Feature 9)
3. Modify GUIs to allow Professor to ask questions & Students to answer. (Features 6,7,8)
4. Have groups generated based on simple iteration or by GPA (Feature 13)
5. Add documentation (Comments, modify exisiting diagrams, etc...)
6. Any non-essential last touches (Make GUI intuitive, add explanations etc...)
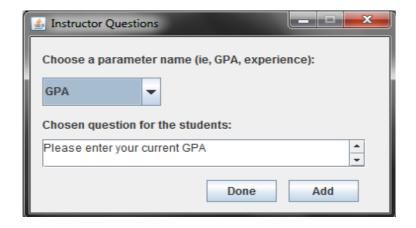
These are the things to be done by Wednesday. More would be feasible if there was more time for the project, but instead it will exist as a proof-of-concept that demonstrates the ability to implement more features if necessary. If possible F7 would be implemented, which would allow for Students to upload schedules, and the groups would be generated off of that.

- Add a GUI window to display generated groups.

- Implement Feature 6 "support adding queries by instructor, to get additional info from students
        (schedules, preferences, experience)"

- If time allows, begin implementing feature 7 "F7-support adding class
and private time schedules by students"

-No scrollbar on generated group window

//NOTE: USER - interface for STUDENT and INSTRUCTOR
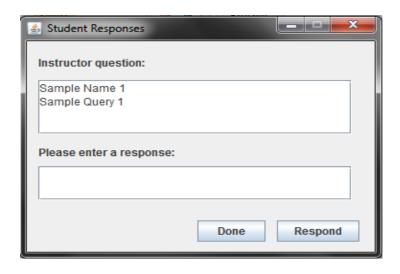//INSTRUCTOR needs to use LIST of PARAMETERS
//each STUDENT needs to have a copy of said PARAMETER LIST
//need GETTERS & SETTERS too
//try to get GUI working with DECENT generator strategy
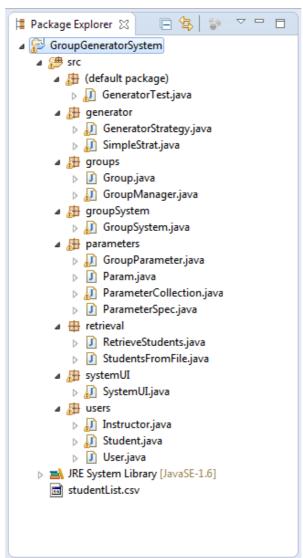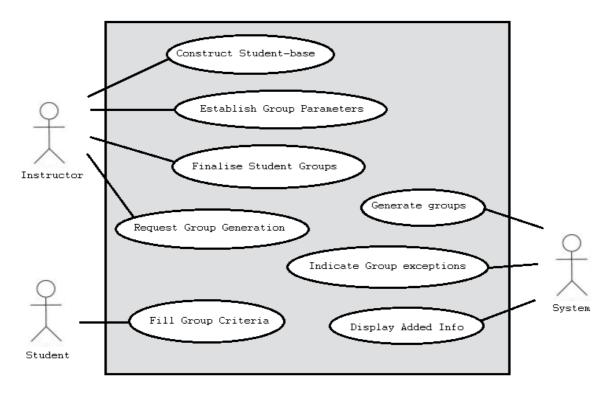//implement simple DECENT strategy

System GUI – main gui

QuestionUI – for instructor to add questions

ResponseUI – for students to respond to instructor questions

Modules and Classes

Rough Use Case Diagram

Use Case Diagram boundary containing:

- Construct Student-base
- Establish Group Parameters
- Finalise Student Groups
- Request Group Generation
- Generate groups
- Indicate Group exceptions
- Fill Group Criteria
- Display Added Info

Actors: Instructor, Student, System

Rough Use Case Diagram

Domain Model

Note: Some explanation written in assignment submission

GroupSystem
- -deadline:String
- +students: Student[*]
- -instructor: Instructor
- -groups: GroupManager
- - description:String
- -params: GroupParamter[*]

SystemUI
- + system:System

GeneratorStrategy
<<interface>>
- generateGroups():Group[*]

SimpleStrat

DecentStrat

Student
- -name:String
- - numID :Number
- +params: GroupParamter[*]

GroupManager
- - groupSize:int
- - strategy: GeneratorStrategy()
- -groups: Group[*]

Instructor
- -name:String
- - numID :Number
- +params: GroupParamter[*]

Note: only instructor can modify group parameters

RetrieveStudents
<<interface>>
- getStudents()

StudentsFromFile

ConcreteClass2

Group
- -stuList:Students[*]

GroupParameter
- -spec:ParameterSpec

ParameterSpec
- -properties:Map

Param (Enum)
- Name
- Query
- Response

Domain Model

Generating Groups

GroupManager | <<interface>> GeneratorStrategy | SimpleStrat | DecentStrat | Group | Student

generateGroups(GeneratorStrategy)

alt

[GeneratorStrategy instanceof SimpleStrat]

generateGroups()

loop [for each Student]

loop [for groupSize]

addStudent(Student)

[GeneratorStrategy instanceof DecentStrat]

generateGroups()

loop [for each Student]

p1: getParameters()

addStudent(Student)

loop [for each Student]

p2: getParameters()

loop [for each Parameter]

compare p1 and p2 RESPONSE

opt [p1.RESPONSE = p2.RESPONSE]

tmpArray.add((index, count++))

loop [for groupSize]

getBestMatch()

addStudent(Student)

Update Sequence Diagram – Generating Groups

Add/Respond to Criteria

User | GPM_UI | System | GroupParameter | ParamSpec | ParamCollection

if [instructor number]

add parameters → add → add parameter(spec) → add property(Enum, key,value)

add Parameter(GroupParameter)

else [student]

respond to queries → add → get parameter()

for [for each parameter]

getSpec()

add property(Enum. response, value)

Sequence Diagram – Add/Respond to Criteria