Iteration Plan Week 1

Our plan is to construct the design of use case 3, which can be seen in the architecture file attached. We will aim to do this by exploring the scenarios of this use case with a domain model, sequence diagrams, arrangement of necessary classes into modules, and a class diagram.

We will not aim for a full implementation of use case 3, “Group Creation”, as this is the most complicated part of our software. Instead, we implement the following parts of use case 3. Our aim is to have the system function as follows:

- System receives course number and maximum number of students per group as input (maybe stored).
- System retrieves list of students (will be stored as text file where each line contains a student name and student number separated by comma) we will use dummy data for this.
- System calculates number of students in class from text file of students
- System calculates number of groups and size of each groups
- System creates empty groups
- System fills empty groups with students at random
- Professor is able to view finalized groups and move students from one group to another

We chose this as our first iteration as we believe this would create the most significant reduction in risk. This is because use case 3 covers the essential feature of our software, creating the groups of students. As well, this implementation will supply us with executable architecture which we will be able to show to the customer.

Use case 3 can be seen below.

Use Case 3,
Group Creation

Actor: The instructor

Description: The instructor initiates the group creation process. The groups are created based on the group parameters entered previously, and the instructor has the ability to modify groups. If modified groups violate the group parameters, then a warning is displayed. A warning is also displayed if the system is unable to create compatible groups.

Flow: See the following page. The first alternate path addresses the case where the instructor wants to create skill-based groups. The second alternate path shows the flow in the case where the created groups violate the original group parameters.

Preconditions: Group parameters exist, all students have completed the self-evaluation questionnaire (if required) and have provided extra-curricular availability information. The deadline for student information being submitted has passed.

Postconditions: Groups have been created, no groups exceed the maximum group size, if groups violate initial group parameters the instructor has acknowledged the warning, groups are available to students.

Use Case 3,
Group Creation

Main Path	Alternate Path	Alternate Path 2
1. instructor chooses to start generating groups		
2. System retrieves class schedule information for each student from the institution's database		
3. System determines the number of groups and the number of students per group		
4. System generates groups based on instructor input, available meeting time, and student preferences (in that order)	4.1 System generates groups based on instructor input, skill balance, available meeting time, and student preferences (in that order)	
		5. System is unable to generate groups with sufficient meeting time. A warning is displayed to the user.
6. instructor reviews and edits generated groups		6.1 Edited groups violate some of the initial criteria. A warning is displayed to the user. Repeat step 6 until instructor decides to confirm.
7. instructor finalizes groups		
8. Groups are made available to students		

Justification: This is the most important feature of our software, as seen in the feature list above, and therefore of utmost importance. As well, it does not appear straightforward to implement. Therefore this satisfies all three of the three Q's of architecture, and it is essential to handle this feature first in order to reduce risk. Therefore we aim to analyze group creation in as many different ways as we can in order to fully understand the core feature of our software.