For each feature (i.e, user story) that you pick, you would need to create a copy of the *Template - \_UserStory ID\_ - \_Name of student\_.gdoc* template in its corresponding sprint folder, with the name reflecting the user story number in Mingle: "UserStory # --- / Name of student:.

Keep refining the feature documents that you develop and feel free to take a look at those created by your teammates to make sure that you catch overlapping/redundant concepts as soon as possible. Typically, you get suspicious of such overlapping/redundant concepts when you hear the report given by your teammates during the daily scrum meetings. You can then check the feature documents created by them and ask for a meeting to go over your features together and resolve any inconsistency issues.

Please take a look at the following links:

<http://agilemodeling.com/artifacts/useCaseDiagram.htm>

<http://agilemodeling.com/artifacts/sequenceDiagram.htm>

<http://agilemodeling.com/artifacts/classDiagram.htm>

Keep in mind that at the end of the semester, you will need to consolidate all these feature documents in the Final Document. A template, *Template - \_UserStory ID\_ - \_Name of student\_.gdoc*, is included in this folder for your convenience.

**Link to Mingle for Project Management:**

https://fiu-scis-seniorproject.mingle.thoughtworks.com/

You can login using the username part of your fiu.edu email (e.g., **jdoe001**) and your initial password is username! (e.g., **jdoe001!**). (IF you can’t login, it means i still haven't created mingle accounts for you. Please stay tuned). Please do modify your password after your first login. I have made all of you a member of the project, created Release …. with Sprints 0 to .... All the previously defined user stories are in Carry Over and Defect Backlogs, and you can find the previous release sprints under the Previous Release tab. The new ones for this semester should be defined under the Sprint Planning tab, column Product Backlog. Your product owner, with your help, of course, will be in charge of defining, refining, prioritizing, and assigning user stories to the current/next sprints.

**Link to GitHub Repositories:**

You must create user on GitHub using the **username of your fiu.edu** and **your @fiu.edu email address** at GitHub and so that I can add you to your GitHub repository. Then please stay tuned for my email from github regarding adding you to your project.

Please do make sure to follow the directory structure that we use for our project.

https://github.com/FIU-SCIS-Senior-Projects

**Links to Download Previous Deliverables:**

<https://github.com/FIU-SCIS-Senior-Projects>

please read and watch:

* [Agile Tutorial Document](http://users.cis.fiu.edu/~sadjadi/Teaching/SeniorProject/Tutorials/Agile_Tutorial.pdf) (required)
* [What is Scrum?](https://www.youtube.com/watch?v=XU0llRltyFM) (required)
* [Scrum Tranining Series](http://scrumtrainingseries.com/) (optional)
* [Mingle Tutorial Video](https://www.youtube.com/watch?v=ZkmS6O1SUNQ) (required)
* [Mingle Tutorial for Senior Projects](https://www.youtube.com/watch?v=Z_G2g3zYduE) (required)
* [GitHub Tutorial Video](https://www.youtube.com/watch?v=4B7ZfO0ZOWw) (required)
* [A successful Git branching model](http://nvie.com/posts/a-successful-git-branching-model/) (required)
* [Version Control Tutorial for Senior Projects - Outline](http://users.cis.fiu.edu/~sadjadi/Teaching/SeniorProject/Tutorials/senior_project_git_tutorial_outline.txt) (required)
* [Version Control Tutorial for Senior Projects - Video](https://www.youtube.com/watch?v=6_t5h-LsqUs) (required)

**-Useful guidelines, tips, and heuristics:**

- User story: A user story needs a short title which starts with a verb before user story description. The user story should follow the template" As a.... Who, what, why".

- A use case needs a name.

- A use case needs a use case diagram

**Heuristics for identifying entity objects**

• Terms that developers or users need to clarify in order to understand the use case

• Recurring nouns in the use cases (e.g., Incident)

• Real-world entities that the system needs to track (e.g., FieldOfficer, Dispatcher, Resource)

• Real-world activities that the system needs to track (e.g., EmergencyOperationsPlan)

• Data sources or sinks (e.g., Printer).

**Heuristics for identifying boundary objects**

• Identify user interface controls that the user needs to initiate the use case (e.g.,

ReportEmergencyButton).

• Identify forms the users needs to enter data into the system (e.g., EmergencyReportForm).

• Identify notices and messages the system uses to respond to the user (e.g.,

AcknowledgmentNotice).

• When multiple actors are involved in a use case, identify actor terminals (e.g., DispatcherStation)

to refer to the user interface under consideration.

• Do not model the visual aspects of the interface with boundary objects (user mock-ups are better

suited for that).

• Always use the end user’s terms for describing interfaces; do not use terms from the solution or

implementation domains.

**Heuristics for identifying control objects**

• Identify one control object per use case.

• Identify one control object per actor in the use case.

• The life span of a control object should cover the extent of the use case or the extent of a user

session. If it is difficult to identify the beginning and the end of a control object activation, the

corresponding use case probably does not have well-defined entry and exit conditions.

**Heuristics for drawing sequence diagrams**

• The first column should correspond to the actor who initiated the use case.

• The second column should be a boundary object (that the actor used to initiate the use case).

• The third column should be the control object that manages the rest of the use case.

• Control objects are created by boundary objects initiating use cases.

• Boundary objects are created by control objects.

• Entity objects are accessed by control and boundary objects.

• Entity objects never access boundary or control objects; this makes it easier to share entity objects

across use cases.

**For UML: You can use Draw.io**

**If you need sample for feature document please take a look at last semester’s documents.**

**https://github.com/FIU-SCIS-Senior-Projects**

**Sprint Meeting Minutes Notes:**

Dear All,

For your three sprint meetings, namely, sprint review, retrospective, and planning meetings, I have created three templates. All minutes must be reflect in these documents, one meeting minute after the other in a sequential manner.

Let me know if you have any questions.

Best,

**Sprint Feedback**

Hi,

I hope that all is well.

Please note that I will be using this sheet to provide you with my feedback regarding your performance in each sprint throughout the semester. You will be earning points that will be counted towards the 40% of your total grade.

Let me know if you have any questions.

Sincerely,

**Fall 2016 - Please Schedule Your Sprint Meetings**

My Dear Students,

Please note that we have 5 two-week long sprints (Sprint 0 to Sprint 7) and as indicated in our Tentative Schedule every other Friday, you are expected to finish each sprint at the same exact time, even if some of the user stories may not be fully done. Each team would need to coordinate with their product owner(s) to come up with common times for sprint meetings. The meetings do not have to be in person and can be online, if need be, but all students MUST attend all sprint meetings.

There are four sprint meetings that would need to be scheduled in the following order: Sprint Review, Sprint Retrospective, Sprint Planning Part I, and Spring Planning Part II. Below are some guidelines for these meetings. Please do make sure that the minutes for each of these meetings are properly reflected in their corresponding meetings minutes documents that I have mentioned before and using the templates that I have provided to you before. For the two parts of sprint planning meetings, please use the one template provided for sprint planning.

**Sprint Review Meeting:** This meeting has to be scheduled at the same exact time at the end of each two-week sprint and must be attended by all students and all product owners. During this sprint, students perform a show and tell, showing the product owners the user stories that were developed and getting their approvals. Any user story that is not fully done and/or not fully approved by the product owner must be placed back in the Product Backlog in Mingle.

**Sprint Retrospective Meeting:** This meeting must be attended by all students and should be scheduled after the sprint review meeting. It is optional for product owners. The students must review their performance, evaluate their work products, and reflect what went right, what went wrong, etc. in the corresponding meeting minutes and make sure that they were able to address those issues indicated in the previous retrospective meetings.

**Sprint Planning, Part I:** This meeting must be attended by all students and the product owner(s), and has to be scheduled after the sprint retrospective meeting. During this meeting, the product owner indicates what user stories have the highest priority and must be done during the next sprint. Students will make sure that all the assigned user stories are well understood and the total estimated work for all the user stories will not go over their sprint velocity. The assigned user stories to the next sprint must be prioritized by the product owner at the end of this meeting.

**Sprint Planning, Part II:** This meeting must be attended by all students and is optional for the product owner(s). It has to be scheduled after the Part I of the sprint planning meeting. Students will discuss who should work on what user stories, in what order, and will come up with a plan of action on how to make sure that all the user stories will be done within the time allocated in this sprint. Note that you guys would need to practice self-organization by being proactive and volunteer for tasks that you may be best suited among your peers to perform them.

Please reflect the schedule of all your sprint meetings in your **Team Schedule** document . You should be aware that from time to time, and definitely at the end of each sprint, I will be checking all your project documents, including all the schedules, time cards, etc., to make sure you are compliant with my instructions and guidelines.

Please watch these series: http://scrumtrainingseries.com/

Should you have any questions, please let me know.

Sincerely,

**Link to Mingle for Project Management:**

https://fiu-scis-seniorproject.mingle.thoughtworks.com/

You can login using the username part of your fiu.edu email (e.g., **jdoe001**) and your initial password is username! (e.g., **jdoe001!**). (IF you can’t login, it means i still haven't created mingle accounts for you. Please stay tuned). Please do modify your password after your first login. I have made all of you a member of the project, created Release …. with Sprints 0 to .... All the previously defined user stories are in Carry Over and Defect Backlogs, and you can find the previous release sprints under the Previous Release tab. The new ones for this semester should be defined under the Sprint Planning tab, column Product Backlog. Your product owner, with your help, of course, will be in charge of defining, refining, prioritizing, and assigning user stories to the current/next sprints.

**Link to GitHub Repositories:**

You must create user on GitHub using the **username of your fiu.edu** and **your @fiu.edu email address** at GitHub and so that I can add you to your GitHub repository. Then please stay tuned for my email from github regarding adding you to your project.

Please do make sure to follow the directory structure that we use for our project.

https://github.com/FIU-SCIS-Senior-Projects

**Links to Download Previous Deliverables:**

<https://github.com/FIU-SCIS-Senior-Projects>

please read and watch:

* [Agile Tutorial Document](http://users.cis.fiu.edu/~sadjadi/Teaching/SeniorProject/Tutorials/Agile_Tutorial.pdf) (required)
* [What is Scrum?](https://www.youtube.com/watch?v=XU0llRltyFM) (required)
* [Scrum Tranining Series](http://scrumtrainingseries.com/) (optional)
* [Mingle Tutorial Video](https://www.youtube.com/watch?v=ZkmS6O1SUNQ) (required)
* [Mingle Tutorial for Senior Projects](https://www.youtube.com/watch?v=Z_G2g3zYduE) (required)
* [GitHub Tutorial Video](https://www.youtube.com/watch?v=4B7ZfO0ZOWw) (required)
* [A successful Git branching model](http://nvie.com/posts/a-successful-git-branching-model/) (required)
* [Version Control Tutorial for Senior Projects - Outline](http://users.cis.fiu.edu/~sadjadi/Teaching/SeniorProject/Tutorials/senior_project_git_tutorial_outline.txt) (required)
* [Version Control Tutorial for Senior Projects - Video](https://www.youtube.com/watch?v=6_t5h-LsqUs) (required)

**-Useful guidelines, tips, and heuristics:**

- User story: A user story needs a short title which starts with a verb before user story description. The user story should follow the template" As a.... Who, what, why".

- A use case needs a name.

- A use case needs a use case diagram

**Heuristics for identifying entity objects**

• Terms that developers or users need to clarify in order to understand the use case

• Recurring nouns in the use cases (e.g., Incident)

• Real-world entities that the system needs to track (e.g., FieldOfficer, Dispatcher, Resource)

• Real-world activities that the system needs to track (e.g., EmergencyOperationsPlan)

• Data sources or sinks (e.g., Printer).

**Heuristics for identifying boundary objects**

• Identify user interface controls that the user needs to initiate the use case (e.g.,

ReportEmergencyButton).

• Identify forms the users needs to enter data into the system (e.g., EmergencyReportForm).

• Identify notices and messages the system uses to respond to the user (e.g.,

AcknowledgmentNotice).

• When multiple actors are involved in a use case, identify actor terminals (e.g., DispatcherStation)

to refer to the user interface under consideration.

• Do not model the visual aspects of the interface with boundary objects (user mock-ups are better

suited for that).

• Always use the end user’s terms for describing interfaces; do not use terms from the solution or

implementation domains.

**Heuristics for identifying control objects**

• Identify one control object per use case.

• Identify one control object per actor in the use case.

• The life span of a control object should cover the extent of the use case or the extent of a user

session. If it is difficult to identify the beginning and the end of a control object activation, the

corresponding use case probably does not have well-defined entry and exit conditions.

**Heuristics for drawing sequence diagrams**

• The first column should correspond to the actor who initiated the use case.

• The second column should be a boundary object (that the actor used to initiate the use case).

• The third column should be the control object that manages the rest of the use case.

• Control objects are created by boundary objects initiating use cases.

• Boundary objects are created by control objects.

• Entity objects are accessed by control and boundary objects.

• Entity objects never access boundary or control objects; this makes it easier to share entity objects

across use cases.

**For UML: You can use Draw.io**

**If you need sample for feature document please take a look at last semester’s documents.**

**https://github.com/FIU-SCIS-Senior-Projects**