1. **Team Member Names:**
    a. Daniel Kushner
    b. Matthew Bichay
    c. Vanessa Ulloa
2. **Project Title:**
    a. Wormhole
3. **Brief Description of Project:**
    a. A per-account cloud storage solution for uploading and downloading files stored on the server. Possible implementations of basic post-upload encryption and decryption.

Team Processes

4. **Describe what your team will do to prevent defects or catch defects early in the release. (Pair programming, design and code review meetings, continuous testing, Test Driven Development, other)**
    a. Pair programming and programming together while on google hangouts voice chat / Skype.
    b. Unit testing on all functional components of the project.
    c. Possible regression / benchmark testing on encryption, upload, and download.
5. **How will your team be organized and communicate? (email, daily hangout meetings, WIKI, chat, scrum master, project manager)**
    a. Google Hangouts
    b. Email Chains
    c. Weekly meetings for reviews on changesets and issues we're having.
6. **What programming language(s) and tools will you use?**
    a. Eclipse - Java
    b. JUnit - Unit testing suite
    c. Git/Github

7. **Requirements**

The list of requirements will change during the project. But you need to start with an initial list. Given each requirement a number, keep track of when you added this requirement. Priority is one of: ESSENTIAL, HIGH, LOW. Don't forget about non-functional requirements and FURPS+

ID	Date added	Priority	Description
REQ1	12/07/15	ESSENTIAL	Wormhole shall have a web based or local client for user interaction.
REQ2	12/07/15	HIGH	The client shall have a form of login and registration for new and existing users.
REQ3	12/07/15	ESSENTIAL	The user shall have the ability to upload and download files from a server.
REQ4	12/07/15	HIGH	The user shall be able to download files or upload files from multiple computers.
REQ5	12/07/15	MEDIUM	The files shall be encrypted prior to being uploaded.
REQ6	12/07/15	LOW	Wormhole shall allow the ability to share files amongst users.

If you planning iteration 1, skip the next four questions.

8. **Which requirements were delivered in last the iteration? List the requirement numbers.**

Last week, REQ 1 was partly realized, REQ2, REQ3, REQ4 are essentially complete. We are missing the download functionality, this functionality has taken backburner to getting a proper login system implemented. Currently, we're using a command-line interface. If we can finish the core functionality of REQ 1,2,3,4 by the end of next weekend, a final GUI will be made to tie everything together.

9. **Was code and test cases for the requirements in the last iteration uploaded and committed to GITHUB? (if not, why not?)**

Yes, the code was updated to GitHub.

10. **Did you have to do any extensive refactoring of code in order to keep the code modular when you implemented these new requirements?**

There was a large refactoring effort of the codebase. The code has been written and cleaned for erroneous lines. The code is also neater and functionalized/compartmentalized to avoid writing quick and dirty script-ish looking functions.

11. **Did you use any of the design patterns for the code in this iteration?**

The beginning of a MVC design pattern, this will come to full realization if a GUI interface is constructed for the final iteration of our project.

12. **Plan for Iteration <4>**

We have a rough mockup of how everything is supposed to work. Once we've tied up some loose ends with the login-functionality of the codebase, the final step will be to implement the download functionality and the GUI interface. This week's plan is to solidify the login-process for the codebase, then start testing on the download functionality / GUI preparation of the codebase.

13. **List the requirements that you plan to deliver in this iteration. If a requirement is too big to deliver in one week, then divide it into small requirements and deliver part of it in this iteration.**

Requirement ID	**Programmer(s) working on requirement**
REQ 1/4	**Vanessa Ulloa**
REQ2/3/4	**Matthew Bichay / Daniel Kushner**

14. **Do you see any issues or risks in the project?**
    a. Risks:
        i. Connecting to the Server is unsuccessful
        ii. Server Maintenance times - backup
        iii. Restrictions on the Servers - packages (Admin access)
    b. Issues:
        i. File upload AND download - consistency
        ii. Data encryption
        iii. Tieing together our changes for these iterations will be a huge mess.
        iv. Vanessa and Matt are having major inconsistencies with the way we're able to setup Apache tomcat/SQL on our machines.
        v.
15. **For each requirement in question 13, write a scenario (UML diagram + text) or a user story.**

Use Case UC-#	Use Case #1 - Registering for Wormhole
Related Requirements	REQ1 / REQ2
Initiating Actor	Client/Customer/End-user
Actor's Goal	Register for a unique Wormhole ID and password.

Participating Actors	Working Tomcat servlet / backend.
Preconditions	Ability to launch our front-end GUI from the web or desktop (currently undecided).
Postconditions	Created a unique user ID and password for Wormhole.
Flow of events for main success scenario	1. The initiating actor selects the register button. 2. The initiating actor chooses a username and password. 3. The server registers the username and associates it with a specific password. 4. The user will then receive confirmation on a successful registration attempt.
Flow of events for extensions	1. User enters pre-existing username, the server will notify the user that the username has been taken and to try a new one. 2. The confirmation password doesn't match the initial specified password, the user will be prompted to to re-enter the password.

Use Case UC-#	Use Case #2 - Login to Wormhole
Related Requirements	REQ1 / REQ2
Initiating Actor	Client/Customer/End-user
Actor's Goal	Log into Wormhole to access/upload/download files
Participating Actors	Server/User Interface
Preconditions	User Registration has been completed
Postconditions	Receive confirmation of Successful Login
Flow of Events for Main Success Scenario	1. User enters username and password into Login screen. 2. Username and Password combination is verified against database information. 3. if Username and Password combination is correct then Login is successful.

Flow of Events for Extensions	1. User enters username and password into Login screen. 2. Username and Password combination is verified against database information. 3. If Username and Password combination is incorrect, then error message of incorrect combination is returned to the user. 4. The Username and Password can be entered again.

Use Case UC-#	Use Case #3 - File Upload to Wormhole
Related Requirements	REQ 2/3
Initiating Actor	Client/Customer/End-user
Actor's Goal	File to be stored on server
Participating Actors	Server/User Interface
Preconditions	User Login is successful
Postconditions	Confirmation of File Upload to server
Flow of Events for Main Success Scenario	1. User is successfully logged into Wormhole 2. User selects file for upload. 3. File is checked for appropriate extension, maximum file size, user has not met maximum uploads. 4. If conditions are met (no disqualifying factors) file is added to the server for that user. 5. Confirmation of successful file upload is displayed.
Flow of Events for Extensions	1. User selects file for upload. 2. File is over the maximum capacity file size or has reached their upload limit. 3. Client relays error message to prompt the user to compress or delete other files to comply with the maximum number of uploads. 4. Upload process is terminated.

Use Case UC-#	Use Case #4 - Access wormhole from any computer.
Related Requirements	REQ 1/2/3
Initiating Actor	Client/Customer/End-user
Actor's Goal	Enact Use-Case 1,2, and 3 from a GUI client to a AWS server.
Participating Actors	Server/User Interface
Preconditions	User Login is successful
Postconditions	Confirmation of File Upload to server and download from a server
Flow of Events for Main Success Scenario	1. User is successfully logged into Wormhole 2. User selects file for upload. 3. File is checked for appropriate extension, maximum file size, user has not met maximum uploads. 4. If conditions are met (no disqualifying factors) file is added to the server for that user. 5. Confirmation of successful file upload is displayed. 6. Task is repeatable from different box.
Flow of Events for Extensions	1. User is successfully logged into the client. 2. User is not connected to the internet. 3. User cannot make connection to the SQL or Apache servlet through HTTP requests 4. Error message displayed.

- **problems your team is having regarding team communication, review of design and code, other issues or risks**

Communication has improved slightly in some instances and have gotten worse in others. Our group's schedule and vision don't align with one another and it's making it very difficult to keep stable communication between the group. In terms of design, we've been constantly modifying what needs to be done based upon our learnings/findings on actually implementing something with a server backend and HTTP requests. The design has been completely refactored to support a more modular and dynamic state, this should make our transition to a graphical interface smoother.

Depending on the overhead of getting the apache tomcat servlet to work on the AWS server handed to everyone, I don't think we'll be able to complete that portion of the deployment process by the deadline. We've had enough issues standardizing our Apache Tomcat7 servers on our local machines in order to test the codebase. I've been reading the "Apache Tomcat 7" textbook which I've checked out from the library nightly.

- **what is plan for next iteration.**

We are still focusing on the most basic and most crucial requirements for this next iteration. Our registration is now done via auto-registration and authentication of user credentials. We are also now able to upload and query the server for available files. The second phase of this will be to make a successful download request to the servlet and test to see if we are able to download a file. If we can complete the download task and solidify the login system, the transition to a better interface and a well-tested stable client will be much smoother with more error checking and input-sanitization.

In layman's terms:

1. Solidify Login (Necessary)
2. Solidify Download functionality (Necessary)
3. Begin input sanitization (Nice to have.)
4. Create basic GUI (Somewhat ambitious)
5. Attempt to deploy on the AWS server. (Ambitious)