Project Plan Instructions

“By failing to prepare, you are preparing to fail.” – Benjamin Franklin

Goal: Demonstrate that your team has a thorough understanding of

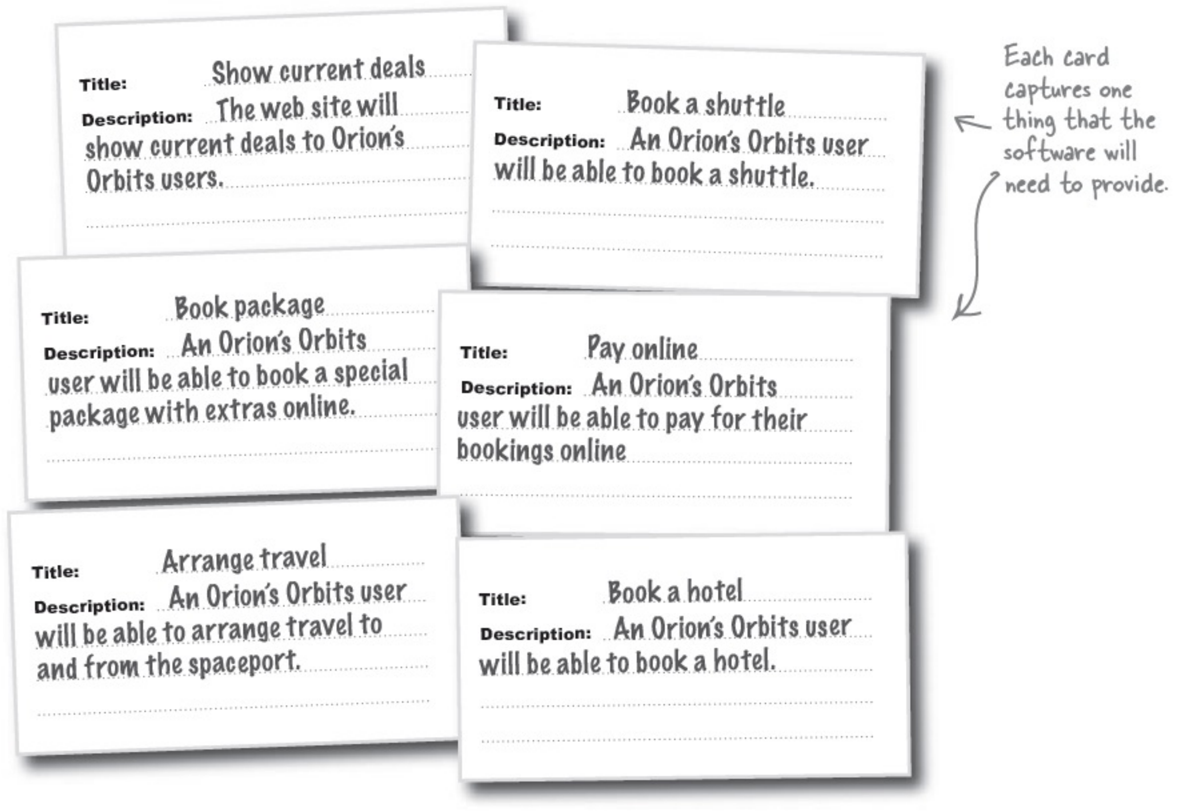
- what features the customer wants,
- what architecture/design the system will have,
- what the user interfaces will look like,
- how you will manage the project,
- how you will assure quality of your product,
- what special resources you will need,
- what schedule the work will follow (best guess), and
- what risks your team must address.

Deliverables: Plan artifacts and a presentation

Plan Components

User Stories

To facilitate communication among your customer, manager, and team, your team must create a set of user stories that describe the system’s functionality. Here are some examples:


Each card captures one thing that the software will need to provide.

Title: Show current deals
Description: The web site will show current deals to Orion’s Orbits users.

Title: Book a shuttle
Description: An Orion’s Orbits user will be able to book a shuttle.

Title: Book package
Description: An Orion’s Orbits user will be able to book a special package with extras online.

Title: Pay online
Description: An Orion’s Orbits user will be able to pay for their bookings online

Title: Arrange travel
Description: An Orion’s Orbits user will be able to arrange travel to and from the spaceport.

Title: Book a hotel
Description: An Orion’s Orbits user will be able to book a hotel.

- You must elicit the user stories from your customer. This may require some back-and-forth communications with your customer.
- It's OK for one user story to depend on another. That is, one may assume that another has already been implemented. You need not denote these dependencies explicitly, but keep them in mind when you're planning work.
- Your customer must approve the set of user stories in your team's plan (prior to the plan presentation and ASAP so you can move forward with the other planning).
- All user stories should be understandable to the customer (no technical/implementation details).
- For your initial plan, you must make the set of user stories as complete as possible (understanding that they will probably change as the project moves along).
- Your team must maintain this set of user stories throughout the project, keeping them up to date.

Software Architecture/Design

Your team must describe what software components you will implement and how those components will interact.

- You have some flexibility in how you communication this. For example, data flow and/or deployment diagrams are probably appropriate.
- Make it clear where the users fit in.
- If your system will use a database, create a database design (i.e., a detailed data model for the database). For example, if you will use a relational database, such as MySQL, design all the tables, keys, etc.
- Specify all hardware/software technologies to be used in implementing your system.
- Specify all programming languages and/or platforms (e.g., Java EE) that you will use, and how they fit into your design.
- Keep your descriptions somewhat high level; for example, I'm not looking for detailed class diagrams.

User Interface Design

Your team must create rough sketches of what the various system interfaces will look like.

- Lo-fi renderings are preferred at this stage. For example, hand-drawn pictures are fine. Sketches done in, say, PowerPoint would also be acceptable.
- Your customer must review and approve your interface designs.

Project Management

Your team must define what policies and procedures will be followed and technologies used for software configuration management, collaborative development of both code and other artifacts, and bug/issue tracking.

Quality Assurance

Your team must define what policies and procedures will be followed and technologies used to ensure that the software built is of good quality—especially from the perspective of the customer.

- A plan for testing must be included (both unit and system testing).
- If part of the plan involves the customer, make sure that the customer is OK with it.

Resources Needed

List any special resources required beyond your individual desktop/laptop computers. Specify how you will get the resource.

- Address resources related to hosting the source repository, bug tracking, system testing/integration machines, and any others particular to your project.
- If you expect the customer to provide a resource, make sure that they agree.

Schedule

Once you have a customer-approved set of user stories, you must estimate how long it will take to fully implement (including testing) each user story.

- Assess each user story independently. It's OK if this results in the same work being counted multiple times. However, if user story *A* depends on user story *B*, then do not count the time required to complete *B* in the estimate for *A*.
- Some team members will be faster/slower than others. Make each estimate an average across all team members.
- Make estimates in days (0.5, 1, 2, 3, 5, 8, 13, 20, 40).
- Planning poker is recommended for this step.
- Don't be afraid to break up user stories with large time estimates, and merge user stories with tiny estimates.

Once you have estimates for each user story, your team must plan which user stories will be completed in the alpha iteration and in the beta iteration. (We'll ignore the release iteration for now.)

- To decide which user stories to implement first, your customer must set the priority of each user story (knowing the cost of that story). He/she may represent the cost as a number (the bigger the number, the higher the priority).
- Your customer must approve your team's plan.
- The manager will tell you how many days per iteration each team member is expected to work.

Note that after you have completed the Project Plan Milestone, you will do additional fine-grained planning at the start of each iteration. Thus, as you estimate the work for each user story, you may want to keep notes on the tasks required to complete the user story. This task information will come in handy when you do the fine-grained iteration planning.

Risks

As your team plans the project, you will find that there are key unknowns, things may not work as planned, or things that could just plain go wrong. These are *risks* to the success of the project, and your team must identify them early.

- Classify the difficulty of each risk: for example, "no idea how to do" versus "probably doable."
- Classify the importance of each risk: for example, "showstopper" versus "nice to have, but not vital." Focus on the most important risks.
- If a risk can be eliminated by talking to customer, do so ASAP.

What artifacts must you produce?

- Set of user stories.
    - Feel free to organize them into categories.
    - Include time estimates.
- Software design/architecture document.
    - Provide supporting text along with diagrams.
    - May be in PPT form (but not meant as presentation).
- User interface design document.
    - Provide supporting text along with figures.
    - May be in PPT form (but not meant as presentation).
- Project management plan document.
    - Mainly, a list of policies/procedures. It should also communicate what technologies will be used.
- Quality assurance plan document.
    - Mainly, a list of policies/procedures. It should also communicate what technologies will be used.

You may create the artifacts as Google documents, wiki/web pages, etc. They should be maintained (updates expected) throughout the project. Each artifact should be understandable to someone not intimately familiar with your project.

What goes in the presentation?

For the presentation, your team must share their biggest worries about the project (i.e., perceived risks), and solicit feedback from the class on how to handle those issues. Have 1–5 slides to support the discussion. Email me (Fleming) your slides by the start of class. Don't forget to provide a brief introduction to your project.

The presentation and discussion must be limited 12 minutes.