Milestone 0 Checklist

- ❑ Requirements specifications (Section 1.1)
    - ❑ Mentor must have reviewed and signed off on
- ❑ User interface designs (Section 1.2)
    - ❑ Mentor must have reviewed and signed off on
- ❑ Software design (Section 1.3)
    - ❑ Architectural diagram(s)
    - ❑ Database design (if system will have a DB)
- ❑ Quality assurance plan (Section 1.4)
- ❑ Configuration management plan and setup (Section 1.5)
    - ❑ Instructors must have been given access
- ❑ Feedback management plan and setup (Section 1.6)
- ❑ Collaboration plan (Section 1.7)
- ❑ Schedule (Section 1.8)
- ❑ Risks (Section 1.9)
- ❑ Code skeleton (Section 1.10)
- ❑ Individual assignment outcomes (Section 2)
- ❑ Instructions to the instructors, submitted to eCourseware dropbox (Section 3)
- ❑ Teammate evaluations (Section 4)

Milestone 0 Instructions

This milestone concludes the initial iteration in which the team focuses on planning and designing the project, as well as on setting up their development environment.

1. Plan, Design, and Setup Instructions

1.1. Requirements Specifications

To facilitate communication among your mentor, manager, and team, your team **must create a list of requirements** that describes the system's functionality. Follow these criteria:

- **Requirements format**: Your requirements may be documented in the form of user stories, a structured list of natural language specifications, or use cases.
- **Granularity**: Requirements should be documented at a level of granularity appropriate for dividing up and managing the work (e.g., they should be reasonably estimable).
- **Mentor-driven**: You must elicit the requirements from your mentor. This will likely require back-and-forth communications with your mentor.
- **Mentor readable**: All requirements should be understandable to your mentors (no technical jargon or implementation details).
- **Complete set**: For your initial plan, you must make the set of requirements as complete as possible (understanding that they will probably change as the project moves along).
- **Status tracking**: Your team must maintain this set of requirements throughout the project, keeping them up to date. The status of each requirement must be recorded (e.g., "not started", "in progress", "completed").

Your mentor must approve the set of requirements in your team's plan (prior to the milestone deadline).

1.2. User Interface Designs

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

- **Sketch quality**: Lo-fi renderings are preferred at this stage. For example, hand-drawn pictures with notes are fine. Sketches done in, say, PowerPoint would also be acceptable.
- **Well detailed**: Your sketches should be as complete as possible (include all key features, interaction dynamics, effects of button clicks).
- **Basic and advanced versions**: Designs may include different levels of refinement (e.g., "basic" and "advanced" versions), enabling you to incrementally build up the system.

Your mentor must review and approve your interface designs (prior to the milestone deadline).

1.3. Software Architecture and Designs

1.3.1. Architectural Design

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

You have some flexibility in how you communicate this. For example, data flow and/or deployment diagrams are probably appropriate.

- **Model user**: Make it clear where the users fit in.
- **Technologies**: Specify all hardware/software technologies to be used in implementing your system.
- **Languages and platforms**: Specify all programming languages and/or platforms (e.g., Java EE) that you will use, and how they fit into your design.
- **High-level**: Keep your descriptions somewhat high level; for example, I'm not looking for detailed class diagrams.

1.3.2. Database Design

If your system will use a database, create a database design (i.e., a detailed data model of the database).

- **Design notation**: You may represent the model as an object-oriented class diagram, an entity-relationship diagram, or other common database modeling notation.
- **Well refined**: Make your tables/design as complete as possible. For example, if you will use a relational database, such as MySQL, design all the tables, keys, etc.

1.4. Quality Assurance Plan

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

- **Software testing**: A plan for testing must be included (both unit and system testing).
- **Manual vs automated**: Be clear about which verification/validation provisions are automated versus manual.
- **Tools**: Describe any specific tools to be used for verification/validation.
- **Mentor involvement**: If part of the plan involves the mentor, make sure that the mentor is OK with it (prior to plan-review meeting).

1.5. Configuration Management Plan and Setup

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

1.6. Feedback Management Plan and Setup

Your team must define what policies and procedures will be followed in managing and handling feedback received. Also, the team must set up a system for managing feedback and provide the mentor and instructors access.

See the *Feedback Collection and Management* document for detailed instructions.

1.7. Collaboration Plan

Your team must define a plan for communication both among the team members and with the mentor. The plan should address questions, such as

- When and where will the team meet? With the mentor?
- What communication media will the team use?

1.8. Schedule

List the features expected to be completed during each development iteration (and, if relevant, how polished the features will be).

1.9. 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.

- **Difficulty**: Classify the difficulty of each risk: for example, "no idea how to do" versus "probably doable."

- **Importance**: Classify the importance of each risk: for example, "showstopper" versus "nice to have, but not vital." Focus on the most important risks.

- **Eliminate ASAP**: If a risk can be eliminated by talking to mentor, do so ASAP (prior to milestone deadline).

- **Track status**: Plan to keep track of the status of each risk as the project progresses (e.g., "eliminated", "open").

1.10. Code Skeleton

To ensure that your team is ready to "hit the ground running" in Development Iteration 1, it is critical that you have a working code skeleton ready prior to the start of Iteration 1. This skeleton will give your team's members a place to begin adding their functionality. What sort of skeleton is appropriate varies by project, but there should clearly be enough there for teammates to easily begin integrating functionality.

- **Executable skeleton**: Although the code skeleton may have few or no features, it should at least execute cleanly as, for example, a "hello world" sample application.

- **Setup frameworks**: If your application is built upon a significant framework (e.g., Android), the framework must be set up and working (at least sufficiently to support the limited functionality of

your code skeleton). For example, in the case of Android, I would expect that a "hello world" app can be run in both a development emulator as well as on an actual Android device. Similarly, in the case of a web app, the skeleton should be set up such that a hello world page can be served up and displayed in a web browser.

- **Pushed to repos**: Your code skeleton must have been pushed to your central version-control repository so that team members can immediately begin contributing and sharing code.

2. Individual Assignments

Full task planning and outcome reporting must be part of each iteration and milestone. See the *Individual Assignment Specification* document for detailed instructions.

3. Instructions to the Instructors

Since the way to access the various milestone artifacts may vary from team to team, each team must submit a document containing instructions to the instructors regarding how to access each artifact. This document must be submitted to an eCourseware dropbox by the milestone deadline.

4. Teammate Evaluations

At the end of each iteration, each team member must provide an evaluation of each other team member. Instructions and forms for performing these teammate evaluations will be communicated by email near the end of the iteration.

Milestone 0 Mentor Sign-Off Form

Mentors: Please indicate your approval of the following items—but ONLY if you agree 100% with the statement for the item.

If you have ANY disagreement, do not give your approval. Instead, provide the team with feedback, and have them resolve whatever issue is preventing your approval.

- ❑ I have reviewed the requirements specifications, and they meet the following criteria:
  - ❑ I can understand all of them
  - ❑ There are no significant requirements missing
  - ❑ There are no unwanted requirements added by mistake

- ❑ I have reviewed the interface designs, and they meet the following criteria:
  - ❑ I can understand all of them
  - ❑ There are no significant features missing
  - ❑ There are no unwanted features added
  - ❑ I am happy with the design of all the interfaces

- ❑ I have been given access to the version control system that the team will use to manage the source code.

- ❑ I have been given access to the system that the team will manage the feedback that I and others give them.

- ❑ I have reviewed any other parts of the Milestone 0 plans and designs that interest me, and am happy with them.