Project Deadlines and Grading Criteria

Item	Iteration 1	Iteration 2	Iteration 3
Requirements	1.1	update	Complete
High-Level Use Case Diagram	1.2	update	Complete
High-Level Use Cases	1.2	update	Complete
Use Case Traceability Matrix	1.2	update	Complete
Tasks Assigned list	1.3	update	Complete
Increment Matrix	1.2	update	Complete
Expanded Use Cases	none	PIM	PIM
Expanded Use Case UI Prototypes	none	PIM	PIM
Domain Model Diagram	1.3	update	Complete
Sequence Diagram	none	PIM	PIM
Design Class Diagram	none	PIM	PIM
Activity Diagram (if needed)	none	none	Complete
Statechart (if needed)	none	none	Complete
Android Project	none	none	Complete
Reports/Presentations	1.3	Yes	Complete
Android Screen Shots	none	Yes	Demo

Term	Description
Complete	Complete product due (w/ submittal)
Demo	Youtube video
None	No submittal required
PIM	Per Increment Matrix (w/ submittal)
Update	Update as Required (at submittal)

The syllabus contains the class dates for the various Iterations. These dates need to be in the Increment Matrix.

Submission needs to be as a PDF file. Please make sure that everything is readable. This file needs to include all presentation materials and artifacts noted in the table above.

Projects will be graded using the following criteria in the syllabus (the project represents 50 percent of the semester grade):

- 5% - Project description
- 20% - Iteration 1 (includes presentation and presentation materials)
- 30% - Iteration 2 (includes presentation and presentation materials)
- 45% - Iteration 3 (includes presentation, presentation materials, and reports)

Grading Criteria for Project Submissions

The following are the criteria for each required item (**consult the table on the previous page for applicability**).

Requirements

1. Does each requirement contain a shall?
2. Is each requirement numbered?
3. Is it easy to determine the required nouns, verbs, attributes to use for the domain diagram?
4. Are there enough requirements that the domain model can be constructed?
5. Are the requirements understandable and complete sentences?
6. Are interface requirements specified and clear?
7. Are attributes consistent with the expanded UC field names, domain attributes, DCD attributes (where applicable)?
8. Are UI requirements consistent with UI prototypes, expanded use cases, etc? UI requirements don't need to be broken out separately - if not then this applies to the portion of the requirements that address UI.
9. Attribute tables should be used to specify vague terms such as car make, or color or model. The tables should spell out what are to be considered. Requirements should reference the tables as needed to make it clear which apply.
10. Each requirement should state the function being performed, the attributes used and the attributes produced. The functional requirement should state some positive result or action being performed by the function. For example, for login - "The software shall provide login facilities for entering user ID and password. When logged in the software shall provide a message of 'login successful.'"

High Level Use Case diagram

1. Does the UCD capture all the verbs from the requirements?
2. Does the UCD capture all the different actors mentioned in the requirements?
3. Are the proper associations between actors and use cases used (e.g. without arrows)?
4. Are all use case "bubbles" inside the context/system boundary box?
5. Do actors have any inheritance relationships depicted or needed?
6. Are multiple diagrams used to improve clarity where needed?
7. Are actor names consistent from one diagram to another?
8. Check the direction of arrows on <<extend>> and <<include>>. Also is the correct dashed line and arrow used? I believe that using these indicates too low-level of detail and that your diagram is not focused on the customer but on software design. If you explain this usage you may be okay.
9. Do names from the UCD agree with names in the high-level use cases and in the expanded UCs?
10. Are use case names in “verb-noun” format?

High-level Use Cases

1. Do the TUCBW and TUCEW have some meaningful descriptions?
2. Is each TUCBW consistent with each expanded UCs?
3. Is each TUCEW consistent with each expanded UCs?
4. Given the context of the use case, does the TUCBW and TUCEW make sense? Does the TUCBW start with a user action and the TUCEW end with a user action.
5. Does each high-level use case correctly scope the specified use case and its context within the system?

Requirements Use Case Traceability Matrix (RUTM)

1. When we go through each requirement, does the indicated UC correctly capture the requirement?
2. Is each requirement fully picked up by a UC or UCs? In other words, no piece of a requirement is left out when mapped across UCs.
3. When we go through each UC does it correctly pickup the requirement it should? Are any left out that should be included?
4. When following the flow of a requirement through the expanded UC, is the requirement consistently used (in terms of function)?
5. Are all requirements identified on the RUTM?
6. Are all UCs identified on the RUTM?
7. Is there a priority for each requirement and for each UC? Do the priorities add correctly (both across and down columns)?
8. Is there a 'priority legend' on the diagram indicating if the lowest number or the highest number indicates the higher priority?

Task assigned (this is simply a list of scheduled tasks for that iteration)

1. Are all scheduled tasks captured in the list?
2. Is each team member shown with tasking assignments?
3. Does each team member have a responsibility?

Increment Matrix

1. Are all the UCs captured in the Iteration planning table?
2. Is each UC mapped to at least one iteration?
3. Are there 3 iterations shown and are the due dates for each iteration shown on the column header?
4. Is the 'depends on' relationship captured in the table? Is it correct?
5. Has the effort been estimated for each UC? Is the total effort depicted? Is there an estimated total effort for each iteration?
6. Does the total effort fit within the course schedule timeframe?

Expanded Use Cases

1. Does each EUC have a correct actor name and system name? Are the actor names consistent with the UCD?
2. Are there pre-conditions and post-conditions for each EUC?
3. Are there TUCBW/TUCEW for each EUC? Do they align with the HL UCs (i.e., are they the same)?
4. Does each EUC start at Step 0 (system step)? Do they end with an actor step?
5. Is each step numbered sequentially?
6. Are there no blank entries in the use case interactions?
7. Does each actor step clearly and correctly specify the actor input and actions?
8. Does each system step clearly and correctly specify the system responses?
9. Are UI prototypes used for 'system responses' where needed? A UIP is needed when the user is inputting information (attributes). Does the system step provide a clear reference or link to the companion UI prototype? Do the attributes from these UI prototypes match attributes in the domain diagram and in the requirements? (Consider a minimum of 2 UIPs per EUC.)
10. Are non-trivial steps noted?

Expanded Use Case UI Prototypes

1. See step 9 in the Expanded Use Cases above.

Domain Model

1. Does the domain model consist of the important classes in the application domain?
2. Are all nouns, verbs, and attributes from the requirements captured?
3. Are there classes and associations in the domain model that do not appear in the requirements?
4. Are the correct relationships shown - aggregation, inheritance, association?
5. Are there classes on the domain diagram that could be better explained with inheritance?
6. Are multiplicity relationships shown on the diagram?
7. Are methods not depicted on the domain diagram?
8. Are all associations labeled and direction indicated?
9. Do the classes represent real-world items?
10. Is there a consistency between class names on the domain diagram, on the sequence diagram, and on the DCD, where appropriate?
11. Is there a consistency between attribute names on the domain diagram and the DCD, where appropriate?

Sequence Diagram

1. Is each non-trivial step in the EUC captured as a SD? Do steps from the expanded use case match the flow on the SD?

2. Does each actor have the same name on the SD and UCD?
3. Is each message formally specified? Are return values specified as "return_value:=..." instead of as dashed arrows?
4. Does the flow go from top to bottom and left to right? Are all parameters and return values specified?
5. Are fragments used to simplify or clarify the SD? Are fragments correctly labeled? Are they correctly specified (e.g., decision logic)?
6. Are lifelines depicted correctly? Are execution methods depicted correctly on both ends of the reference?
7. Are all actor/object interactions depicted correctly? Are all object/actor interactions depicted correctly? Are stereotypes and dashed lines used correctly?
8. Are there any cases of poor coupling on the SD?
9. Have all design patterns been utilized on the SD?
    a. Has the controller pattern been applied (GUI, Controller, Mgr)?
    b. Has the singleton pattern been identified (Mgr) and with the correct stereotype?
    c. Has the creator pattern been applied as to who should create an object?
    d. Has the expert pattern been applied to show the best provider for the requested data?

Domain Class Diagram

1. Does the DCD capture all project iterations?
2. Do classes from the DCD relate to the following from the SD:
    a. Classes of objects that send or receive messages?
    b. Classes of objects that are passed as parameters?
    c. Classes that serve as return types?
    d. Classes from the domain model (added to clarify the DCD only)?
    e. Same class names from the DCD, where appropriate?
3. Do methods align with messages from the SD? Is each method in the SD correctly captured in the DCD? Do parameters and return types match? Does each incoming edge of the object have a related method?
4. Are all attributes from the SDs called out?
    a. Identify attributes from methods that retrieve objects
    b. Identify attributes from methods that compute a scalar type value
    c. Identify attributes from the parameters to a constructor
    d. Attributes are identified from the domain model
5. Are singletons correctly labeled?
6. Are relationships correctly depicted?
    a. Identify <<create>> relationships from calls to constructors
    b. Identify <<use>> relationships from classes as parameters or return types
    c. Two objects as parameters - association relationship
    d. Identify <<call>> relationships

e. Inheritance and/or aggregation from the domain diagram
f. Are relationships correctly depicted as dashed lines with solid arrows?

Activity Diagram (optional - but may be needed to tie together the top-level UCs)

1. Does each diagram have an initial and end node (this is inferred from a swim-lane)?
2. Are swim lanes provided in the diagram and are they labeled?
3. Are work products of the process/flow labeled?
4. Is the control flow properly represented:
    a. Decisions/merges vs. forks/joins?
    b. Is the behavior consistent with the related expanded UCs?
    c. Do forks represent different threads of control in the code?

State chart (optional)

1. If the state chart is not showing the UI interaction or depicting an event-driven interaction, is the state chart necessary? If the latter, is the information depicted as a state diagram better depicted in other ways?
2. Are all inputs and outputs labeled? Does the state chart have a start state? (Inputs and outputs for the UI interaction state charts are informal in terms of inputs and ouputs).
3. Is each state mutually exclusive to the other states? Is each transition an event?
4. If used for UI interaction, is each major system display represented as a state? Is each transition marked by a UI event? Is there a UI prototype for the states?

User Interface (Android)

1. Does the user interface make the system understandable to use?
2. Is the UI behavior consistent with the expanded use case?
3. Are the labeling of the GUI widgets and descriptive texts clear? Are they readable?
4. Are UI related questions and user-directed information clear?
5. Does the interaction between pages flow smoothly (both to and from)?

Tests/reports (optional)

1. As needed - no specific requirements.

Android screen shots and demo

1. For iteration 2, this includes a screenshot of at least 2 android class project functions accompanied with a snapshot of the android code.
2. For iteration 3, provide a YouTube video demonstrating the operation of the project, all requirements implemented/demonstrated.