Answers to Guide 4

1. Requirements Analysis
   1. Analysis Phase: Determining whether the stated requirements are clear, complete, consistent, and unambiguous, and resolves apparent conflicts.
   2. Design Phase: ????
   3. Stakeholders of a project:
      1. Those who operate the system
      2. Those who benefit from the system
      3. Those involved in purchasing or procuring the system
      4. Organizations or people that regulate aspects of the system.
      5. Organizations or people opposed to the system.
      6. Organizations or people responsible for systems that interface with the system under design
      7. Organizations who integrate horizontally with organizations for whom the analyst is designing the system.
   4. How are requirements identified and recorded?
      1. Recording: Documenting requirements in various forms, including summary lists, natural-language documents, use cases, user stories, process specifications, variety of models.
      2. Identifying: Use of business process documentation, stakeholder interviews, workshops, brainstorming, use cases, role playing, prototyping, questionnaires, user observation, etc.
   5. Difference between functional and non-functional requirements.
      1. Functional: Explain what has to be done by identifying the necessary task, action, or activity that must be accomplished. Defines a function of a system or its component, where a function is described as a specification of behavior between outputs and inputs.
      2. Non-functional: specifies criteria that can be used to judge the operation of a system, rather than specific behavior. Often called quality attributes of a system.
   6. Key issues related to requirements and requirements elicitation?
      1. Stakeholder issues: users inhibit requirements gathering.
         1. Users do not understand clearly what they want.
         2. Users do not commit to a set of written requirements.
         3. Users add to requirements after contract is settled.
         4. Slow communication with users.
         5. Users do not understand the development process or technical aspects.
      2. Engineer/developer issues:
         1. Inclination towards writing code and implementation details resulting in inelegant refactoring to meet requirements.
         2. Technical personnel and end-users have different vocabularies
         3. Try to make requirements fit existing system/model instead of developing a system suitable to client needs.
2. Requirements Modeling
   1. Unified Modeling Language (UML)
      1. A general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.
      2. Static (structural) views: emphasizes the static structure of the system using objects, attributes, operations, and relationships. Includes class diagrams and composite structure diagrams.
      3. Dynamic (behavioral) views: emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. Includes sequence diagrams, activity diagrams, and state machine diagrams.
      4. Use case diagrams: a representation of a user’s interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.
      5. Class diagrams: a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations or methods, and the relationship among objects.
      6. Sequence diagrams: shows how objects communicate with each other regarding a sequence of messages.
      7. Deployment diagrams: models the physical deployment of artifacts on nodes.
   2. Use case diagrams
      1. The purpose of use case analysis:
         1. A technique used to identify the requirements of a system and the information used to both define processes used and classes (collection of actors and processes) which will be used both in the use case diagram and the overall use case in the development or redesign of a software system or program.
      2. Elements of Fowler’s use cases:
         1. Title: “goal the use case is trying to satisfy”
         2. Main success scenario: numbered list of steps
            1. Steps: “a simple statement of the interaction between the actor and a system”
         3. Extensions: separately numbered lists, one per extension.
            1. Extension: “a condition that results in different interactions from the main success scenario”
         4. Can be viewed as a simplified variant of the Cockburn template.
      3. What do the actors, use cases, and association links represent in a UML use case diagram?
         1. Actors: Stakeholders that make decisions and interact directly with the system. (need not be human)
         2. Use cases: a list of actions or event steps typically defining the interactions between a role (actor) and a system to achieve a goal.
         3. Association links: indicates that the actor and the use case somehow interacts or communicate with each other.
   3. User story:
      1. Definition: An informal, natural language description of one or more features of a software system. Often written from perspective of end user and stakeholders. They help software teams organize their understanding of the system and its context.
         1. Define what has to be built in the software project.
      2. The “Five W’s” format for a user story:
         1. As <who> <when> <where>, I <want> because <why>
         2. Questions whose answers are considered basic in information gathering or problem solving.
            1. Who was involved?
            2. What happened?
            3. Where did it take place?
            4. When did it take place?
            5. Why did that happen?
            6. Optional – How did it happen?
      3. Compare and contrast user stories and uses cases
         1. User stories
            1. Generally formulated in users’ everyday language and should help reader understand what the software should accomplish.
            2. Provides a small-scale and easy-to-use presentation of information, with little detail, thus remaining open to interpretation, through conversations with on-site customers.
            3. As a <type of user>, I can <some goal> so that <some reason>
         2. Use cases
            1. Written in user’s everyday business language to facilitate stakeholder communications.
            2. Organize requirements to form narrative of how users relate to and use a system, thereby focusing on user goals and how interaction with system satisfied those goals.
            3. Describe sequences of interactions and may be worded in terms of a formal model.
            4. Should provide sufficient detail to be understood on its own.
            5. Title -> Main success scenario -> extensions
3. Android
   1. User Input Controls
      1. Buttons: consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it.
      2. Text Fields:
         1. Text View: user interface element that displays text to the user.
         2. Edit Text: user interface element for entering and modifying text.
      3. Checkboxes: allow the user to select one or more options from a set. Usually presented in a vertical list.
      4. Radio buttons: allow the user to select one option from a set. Should be used for optional sets that are mutually exclusive if user needs to see all available options side-by-side.
      5. Spinners: provides a quick way to select one value from a set. Touching it displays a dropdown menu with all available values and in its default state show its currently selected value.
      6. View focus versus click-ability:
         1. Clickable means the view can be clicked or tapped, while focusable means that the view is allowed to gain focus from an input devices such as a keyboard.
      7. Images as buttons
         1. Can turn any View, such as an ImageView, into a button by adding the android:onClick attribute in the XML layout. (image must be stored in the drawables folder of project)
   2. Menus
      1. Functional areas of the app bar.
         1. App bar or action bar is a dedicated space at the top of each activity screen.
            1. Navigation button or Up button: Nav. Button to open a navigation drawer or up button to navigate through app’s screen hierarchy to the parent activity.
            2. Title: app title, or the name defined in AndroidManifest.xml by android:label attribute for the activity.
            3. Action icons for the options menu: each icon represents one of the option menu’s most frequently used items; those less frequently used appear in overflow options menu.
            4. Overflow options menu: opens a popup with option menu items that are not shown as icons in the app bar.