Answers to Guide 2

1. Software Development Process
   1. Life cycle: Process of dividing software development work into distinct phases to improve design, product management, and project management.
   2. Waterfall Model: A sequential development approach where development is seen as flowing steadily downwards to completion.
      1. Requirements analysis: The tasks required to determine the needs or conditions to meet for a new or altered product or project.
      2. Software design: The process of envisioning and defining software solutions to one or more sets of problems.
      3. Software implementation: the process of designing and building an executable computer program for accomplishing a specific computing task.
      4. Software testing: The investigation conducted to provide stakeholders with information about the quality of the software product or service under test.
      5. System integration (optional): The process of bringing together the component subsystems into one system and ensuring that the subsystems function together as one system.
      6. Software Deployment (optional): The activities that make a software system available for use.
      7. Software Maintenance: The modification of a software product after delivery to correct faults or to improve performance and other attributes.
   3. Waterfall Model:
      1. Project is divided into sequential phases with some overlap splash-back acceptable.
      2. Emphasis on planning, scheduling, deadlines, budgeting, and implementation of an entire system at one time.
      3. Control maintained via extensive documentation, formal reviews, and approval by user/IT management at the end of each phase before beginning the next phase.
      4. Traditional engineering approach applied to software development; inflexible.
   4. Prototyping Model:
      1. Not a standalone complete methodology but an approach to try out particular features in the context of another complete methodology
      2. Some prototypes are made with the expectation of being discarded while others can evolve into a working system.
      3. Used to attempt to reduce project risks by breaking into smaller segments.
      4. The client is involved in the process thereby increasing the likelihood of acceptance of the final system implementation.
   5. Spiral Model:
      1. Combines waterfall and rapid prototyping methodologies in order to gain advantages of both the top-down and bottom-up concepts.
      2. Focus on risk assessment and minimizing risks by breaking project into smaller segments thus provided more ease-of-change during development.
      3. Every cycle progresses through same sequence of steps from overall concept of operation document to coding of each program.
      4. Each trip around the spiral consists of 4 things
         1. Determine objectives, alternatives, and constraints of each iteration
         2. Evaluate alternatives and identify and resolve risks
         3. Develop and verify deliverables for the iteration
         4. Plan the next iteration
      5. Begin each cycle with ID of the needs of the stakeholders and end each cycle with review and commitment.
   6. Rapid prototyping Model:
      1. Focuses on iterative development and rapid construction of prototypes to avoid a lot of pre-planning.
      2. Objective is fast development and delivery of high quality system at relatively low cost.
      3. Reduces project risks by breaking into smaller segments and providing ease-of-change during development.
      4. Emphasis on business needs rather than technological or engineering excellence.
      5. Control via prioritizing development and defining deliverable deadlines.
      6. Emphasis on reducing requirements if over deadlines and budget, not increasing development time.
      7. User involvement is emphasized.
   7. I have relatively little experience working in a team so I would say my methodology is probably at the level of the “code and fix” model.
2. Constructive and Destructive Group Behavior
   1. Strengths
      1. Cooperating
      2. Risk-taking
   2. Weaknesses
      1. Withdrawing
      2. Digressing
   3. Accentuate positive impact:
      1. If I need assistance on something, ask for help instead of trying to tackle it on my own in order to take advantage of working in a team and increase efficiency.
   4. Eliminate negative impact:
      1. Get to know my team members on a personal level in order to better relate to them and hopefully work better as a team.
      2. If I have input, no matter if I feel it might be stupid to say, voice it as it could actually be useful and the worst case is I get a better understanding of why my input was invalid.
   5. “Healthy” conflict is acceptable and desirable on a team in moderation and if it is handled properly.
      1. Team rivalry can weed out inefficiencies and keep people focused
      2. High performers are naturally competitive and this spirit should be fueled.
      3. Don’t confuse feeling with reality
      4. Be creative
      5. Remind team that differences of opinion are inevitable but useful
3. Project Management Tools
   1. Trello
      1. It is not specifically tailored for software development and has a variety of work/personal uses including real estate management, school bulletin boards, lesson planning, accounting, web design, gaming, etc.
      2. Kanban is a lean method to manage and improve work across human systems. In software development, it provides a visual process management system that aids decision making concerning what, when, and how much to produce. It is a methodology.
      3. Kanban relates to Trello in that Trello provides a front-end visual process management system that can utilize this methodology for the software development process.
      4. Quoting time is valuable because it allows for some pre-planning in the software development process in order to budget and create deadlines.
      5. A burn down chart is a graphical representation of work left to do versus time. It is useful for predicting when all the work will be completed.
   2. Slack
      1. Can create any number of channels tailored for a specific need or task.
      2. Team members can join and leave channels as necessary.
      3. Threads prevent side conversations from de-railing the focus of the channel.
      4. Has integrated search feature that allows searching all posted text in all channels or personal messages.
      5. Can connect other services to Slack channels.
      6. Has integrated file sharing, video/voice call capabilities.
      7. Traditional e-mail results in long confusing chains and there is no real-time conversational ability.
      8. Trello seem better for task management during a project by providing a visual interface of all tasks.
4. Android
   1. Layouts, Views, and Resources
      1. MVP = model-view-presenter pattern
         1. Views – user interface elements that display data and respond to user actions.
         2. Presenters – connect the application’s views to the model.
         3. Model – specifies the structure of the app’s data and the code to access and manipulate the data.
      2. Basic Android layouts
         1. LinearLayout – group of child views positioned and aligned horizontally or vertically.
         2. RelativeLayout – group of child views in which each view is positioned and aligned relative to other views within the view group.
         3. ConstraintLayout – group of child views using anchor points, edges, and guidelines to control how view are positioned relative to other elements in the layout.
         4. TableLayout – group of child views arranged into rows and columns.
         5. AbsoluteLayout – group that lets you specify exact x,y, coordinates of its child views.
         6. FrameLayout – group of child views in a stack.
         7. GridLayout – group that places its child screens in a rectangular grid that can be scrolled.
      3. The Android UI layout is specified via the XML layout file. It contains code for all the UI elements and layouts. However, you can also use the layout editor for a visual, non-code, method of creating the UI layout via its sub-components.
      4. Density-independent pixels (dp) are independent of screen resolution.
         1. Abstract unit based on physical density of the screen in dpi (dots per inch)
         2. Automatically scales the number of pixels used to render in order to provide for consistency for UI elements across difference devices.
      5. Scaled-independent pixels (sp)
         1. Similar to “dp” but is also scaled by the user’s font size preference.
         2. Recommended to be used for font sizes to adjust for screen density and user preference.
   2. Text and Scrolling Views
      1. To create a scrolling text view:
         1. Use the ScrollView class for a vertical scrolling view.
         2. Then, add RelativeLayout or another layout type as a child to the ScrollView or use a ViewGroup if you want to add multiple layouts.
         3. Last, add TextViews as necessary to the layout(s).
      2. Toast – provides simple feedback about an operation via a small popup.
         1. Fills only amount of space necessary for the message and disappears after a timeout.
   3. JavaDoc
      1. Will not specify the author and version of each module because more than one person could work on each module?