Activity 6.1 – Learning Highlights

Technology Stack

- **Back-end:**
  - **Operating System** (e.g. Windows, MacOS, Linux)
  - **Web Server** (e.g. Apache)
  - **Database** (e.g. MySQL, MongoDB)
  - **Programming Language** (e.g. Python, Node.js, C, C#, Java)
  - **Web Framework** (e.g. Flask, Express.js, ASP.NET)
- **Front-end:**
  - **HTML**
  - **CSS**
  - **JavaScript**

Methods for Gathering user requirements and feedback

- **Interviews and Workshops:** Talking directly to stakeholders helps get detailed insights into their needs, what they use, and what's causing them issues.
- **Surveys and Questionnaire:** Great practices for getting feedback from a large group.
- **Usage Analytics:** Tracking how users interact with the system provides real data

Methods for Gathering user requirements and feedback

- **Kano Analysis:** Categorizes features by their impact on user satisfaction, from basic needs to exciting extras. Priorisation is based on user delight.
- **MoSCoW Analysis:** Stands for Must have, Should have, Could have, and Won't have.
- **User Stories and Prioritisation Matrices:** User stories explain what users want, and prioritization matrices rank them based on impact, effort, and business goals.

Techniques for identifying performance bottlenecks and scalability limitations

- **Profiling:** Profiling tools help you see how your app is running by checking which parts of the code are using the most CPU, memory, or other resources.
- **Load Testing:** This tests how well your app handles high traffic or data loads.

- **Stress Testing:** Like load testing, but you push the app past its normal limits to find its breaking point. It's helpful for knowing the max capacity and weak spots.
- **Monitoring and Logging:** Keeping an eye on your app's performance in real time while it's running. You can use logs to track issues like errors and performance dips.
- **APM (Application Performance Management) Tools:** These tools give you detailed info about how your app is performing, tracking things like slow responses or bottlenecks.
- **Code Reviews:** Regular code reviews catch performance issues, like inefficient algorithms or unnecessary operations, before they become big problems.
- **Database Analysis:** Checking queries, indexes, and database settings can help speed things up.
- **Infrastructure Review**: Looking at your servers, networks, and storage can show resource limits or wrong configuration that might hold your app back.
- **User Experience Monitoring:** Real User Monitoring (RUM) tools let you see how users experience the app, tracking things like page load times and error rates from their perspective.
- **Synthetic Monitoring:** These tools simulate user interactions from different locations, helping you catch performance problems that might only show up in certain areas.