Week 7 Assignment

Prioritizing Features for a Developer Tool Based on Customer Feedback

Name: Nagarjuna Reddy Daddanala

STEP 1: Selecting Developer-Focused Tool

GitHub is a web-based platform primarily designed for version control and collaborative software development. It allows developers to host and manage their code repositories using Git, a distributed version control system. The platform is widely used for both open-source and private projects, providing a centralized space where multiple developers can collaborate, share code, track issues, and review changes.

Key features of GitHub include:

1. **Version control with Git:** GitHub tracks code changes over time, making it easy to revert to previous versions or collaborate on updates.
2. **Collaboration tools:** Features like pull requests, code reviews, and issue tracking facilitate teamwork and communication among developers.
3. **Repositories:** GitHub allows users to create public or private repositories to store and organize code.
4. **Branching and merging:** Developers can create branches to work on features independently, then merge them back into the main branch once completed.
5. **Actions and automation:** GitHub Actions enables continuous integration and continuous deployment (CI/CD), automating testing and deployment pipelines.
6. **GitHub Pages:** Users can host static websites directly from GitHub repositories, making it useful for project documentation or portfolio sites.

STEP 2: Crafting Realistic Feedback Answers

What features do users find most valuable?

- Pull requests for collaboration and code review.
- Issue tracking to monitor bugs and tasks.

- GitHub Actions for automating deployment.

What features are missing or could be improved?

- The search function could be more efficient.
- UI for large repositories could be simplified.

How do users feel about the user interface and experience?

- The interface is functional, but beginners sometimes find it difficult to navigate.

Are there any recurring issues or bugs reported by users?

- Issues with real-time notifications not updating.

What suggestions do users have for new features or improvements?

- A visual editor for markdown files.
- More customization options for user profiles or dashboards.

STEP 3: Creating a Prioritization Model

MoSCoW Matrix: Must-have, Should-have, Could-have, and Won't-have features

Category	**Features**	**Description**
Must-have	**Version Control**	Core functionality to track changes, manage branches, and ensure collaboration.
	Repositories (Public/Private)	Essential for hosting and organizing project files.
	Pull Requests	Key for code review and collaboration.
	Branching and Merging	Crucial for managing parallel development.
Should-have	**Issue Tracking**	Important for managing tasks, bugs, and feature requests effectively.
	GitHub Actions (CI/CD)	Facilitates automation of testing and deployment but can be implemented later.
	Code Reviews and Comments	Adds value for quality control and collaboration but isn't mandatory for the tool to function.
Could-have	**GitHub Pages**	Useful for hosting static websites but not critical for core functionality.
	Project Management Tools (Boards)	Helps in organizing workflows but isn’t a critical dependency.

	Insights & Analytics	Provides additional data on project health but isn’t essential.
Won’t-have	**Enterprise-Specific Features (Advanced Security, SAML)**	Relevant for enterprise customers but not necessary for basic usage.
	Marketplace Integrations	Nice for extensibility but not crucial at this time.

Viewing the Presentation and Video

To gain a detailed understanding of the feedback analysis and prioritization process, you can access the following materials:

- PowerPoint Presentation: **https://drive.google.com/file/d/1EhmvLWKq29cm7mE71Vw2s327si1dIvVT/view?usp=sharing**
- Video Demonstration: **https://docs.google.com/presentation/d/1MKk_K5R2JllW8ZMteNR_0dcV1eZfKNpk/edit?usp=sharing&ouid=113944543202524537377&rtpof=true&sd=true**