**📌 Step 1: Understand Scrum Fundamentals Clearly**

**🔹 What is Scrum?**

Scrum is an **empirical framework** for developing products incrementally. It's suitable for complex projects with unclear requirements and changing conditions.

* **Empirical:** Based on real-world experience and continuous feedback.
* **Incremental:** Delivers small, functional product increments regularly.

**Key idea:**

* Iterative (repeated cycles)
* Incremental (small functional pieces)
* Adaptable (responding to change continuously)

**📌 Step 2: Scrum Roles (Who Does What?)**

Clearly memorize these three roles and their responsibilities:

| **Role** | **Responsibility** |
| --- | --- |
| 🟢 **Product Owner (PO)** | - Decides what features to build and prioritizes them - Maintains the **Product Backlog** |
| 🟢 **Scrum Master (SM)** | - Removes obstacles, ensures the team follows Scrum practices - Does **not** manage the team directly |
| 🟢 **Development Team** | - Self-organizing, cross-functional (coders, testers, designers) - Decides how to accomplish work - Usually 6± members |

**📌 Step 3: Scrum Events (Meetings & Workflow)**

Understand the key events in the Scrum cycle clearly:

| **Event** | **What Happens** | **Time Frame** |
| --- | --- | --- |
| 🟢 **Sprint Planning** | Select Product Backlog Items (PBIs) & tasks for Sprint | Max 8 hours (shorter for shorter Sprints) |
| 🟢 **Daily Scrum** | Quick daily meeting (15 min): What was done, what's next, obstacles | 15 min daily |
| 🟢 **Sprint Review** | Demo the completed Increment, gather feedback | End of Sprint (1–2 hours typically) |
| 🟢 **Sprint Retrospective** | Discuss how team worked and what to improve next time | End of Sprint (45–90 min typically) |
| 🔸 **Backlog Refinement** *(ongoing)* | Clarify and split large PBIs into small tasks | ~10% of team’s Sprint time |

**📌 Step 4: Scrum Artifacts (Items Used in Scrum)**

Understand these clearly:

| **Artifact** | **Definition** | **Managed By** |
| --- | --- | --- |
| 🟢 **Product Backlog** | Prioritized list of all features (User Stories/PBIs) | Product Owner |
| 🟢 **Sprint Backlog** | Tasks team commits to during the current Sprint | Development Team |
| 🟢 **Increment** | Working, tested product delivered each Sprint | Development Team |

**📌 Step 5: Clearly Relate Scrum to Your GUI Project**

Here’s exactly how Scrum translates to your Food Insecurity Notification System Project:

* **Product Backlog:** Your previously-provided user stories (admin setup, subscriber lists, SMS notifications, login).
* **Sprint Backlog:** Select a subset of these stories each "Sprint" (every week or two) to work on.
* **Increment:** Each Sprint results in a functioning piece of the GUI.

**Example GUI Increment:**

| **Sprint** | **Increment (what you'll build)** |
| --- | --- |
| 1 | Create Admin GUI for multiple subscriber lists |
| 2 | Subscriber settings GUI and SMS/email preferences |
| 3 | GUI to send notifications |
| 4 | Web GUI for login, registration, and password reset |

**📌 Step 6: Daily Scrum Meeting (Personal Version)**

Every day, briefly ask yourself these questions (just like the real Daily Scrum):

* **What did I finish yesterday?**
* **What will I do today?**
* **What obstacles might I face?**

This keeps your progress on track and manageable.

**📌 Step 7: Clearly Understand "Definition of Done"**

For each GUI increment you develop, your "Done" criteria must include:

* Properly tested
* Integrated into your existing system (functional)
* Peer-reviewed if applicable (or self-reviewed carefully)
* Documented clearly (comments, README files, etc.)

**📌 Step 8: Retrospectives (Personal Version)**

At the end of each week/Sprint, conduct your own brief **Retrospective**:

Ask clearly and honestly:

* **What worked well this Sprint?**
* **What did not work well?**
* **How can I improve in the next Sprint?**

Use these insights to adapt your approach continuously.

**📌 Step 9: Agile Best Practices (Recommended by Scrum Reference)**

Use these best practices clearly to enhance your workflow:

* **Test-Driven Development (TDD)**: Write tests first, then build your code.
* **Continuous Integration**: Frequently integrate and test your GUI features.
* **Pair Programming**: Occasionally code with a classmate if possible.
* **Mob Programming (optional)**: If working in groups, try coding collaboratively.

**📌 Step 10: Clearly Avoid Common Scrum Mistakes**

As explained by your material:

* **Avoid "technical debt":**  
  Do thorough testing each Sprint.
* **Don’t overcommit:**  
  Select realistic amounts of work.
* **Keep User Stories small:**  
  Break large features into smaller slices clearly.
* **Avoid geographic distribution** (harder to collaborate effectively).

**✅ Scrum Project Workflow (Visual Summary)**

sql

CopyEdit

Start → [Sprint Planning]

↓

[Daily Scrum ←──┐ (every day)

↓ │

[Sprint Work]───┘

↓

[Sprint Review] → Feedback (Demo to instructor/peers/self-review)

↓

[Sprint Retrospective] → Identify improvements

↓

(Repeat each Sprint)

**📝 Quick Checklist for Your Assignment (Module 1)**

| **Task** | **Done?** |
| --- | --- |
| ✅ **Clearly reviewed Agile/Scrum reference card** | ☐ |
| ✅ **Roles (PO, SM, Dev) clearly understood** | ☐ |
| ✅ **Artifacts understood (Backlogs, Increment)** | ☐ |
| ✅ **Scrum events understood & implemented** | ☐ |
| ✅ **Set up initial Product Backlog (user stories)** | ☐ |
| ✅ **Created Sprint Backlog (first tasks)** | ☐ |
| ✅ **Started GUI Increment for first Sprint** | ☐ |
| ✅ **Daily Scrum conducted (personal check-ins)** | ☐ |
| ✅ **Clearly understood "Done" criteria** | ☐ |
| ✅ **Conducted a retrospective (end of Sprint)** | ☐ |

**🎯 What To Do Now (Click-by-Click Summary):**

1. **Carefully review** the Scrum Reference Card you provided.
2. **Memorize Scrum roles, events, and artifacts clearly**.
3. **Clearly organize your provided GUI assignment into User Stories & Sprint tasks** (Backlogs).
4. **Clearly use Agile best practices** like TDD, continuous integration, etc.
5. **Conduct personal Daily Scrums and regular Retrospectives** to ensure Agile/Scrum effectiveness.

**🔑 If you follow these steps carefully:**

* You will clearly meet your Module 1 Learning Objectives.
* You will create your GUI project effectively using Scrum methodology.
* You will clearly understand Scrum and Agile software development in practice.

**Project Goal:**

You're completing **Week 1** of your Agile course, setting up your project for Sprint 1 using terminal commands, aligning to the **Scrum framework** (roles, stories, backlogs, increments), and creating organized folders and files based on your assignment and readings.

**🛠️ Terminal Steps Explained (One by One)**

**✅ Step 1:**

powershell

CopyEdit

cd "C:\Users\Admin\OneDrive\Desktop\School Computer Inforrmation Systems\CIS234A\_Real\_World\_Programming\Calendar\_Weeks\Week 1 - Introduction & Agile Overview"

🧾 **What It Does:**  
Navigates (enters) into your **Week 1 course folder**.  
📦 This is your *working directory* where all project files for this week will be stored.

🎯 **Why:**  
Keeps your project organized and aligns with Agile practice of grouping stories and work by sprint/week.

**✅ Step 2:**

powershell

CopyEdit

mkdir project-backlog

mkdir scrum

mkdir terminal-setup

mkdir html-forms

mkdir sprint-1-code

🧾 **What It Does:**  
Creates 5 folders in your Week 1 directory:

* **project-backlog** → stores your Sprint 1 user stories in markdown
* **scrum** → stores daily logs, reviews, retrospectives
* **terminal-setup** → stores your step-by-step terminal setup documentation
* **html-forms** → stores your GUI forms like register/login
* **sprint-1-code** → contains backend files like server.js for logic

🎯 **Why:**  
This reflects an **n-tier MVC structure** (model-view-controller) and supports **modularity** (important Agile principle). Each folder supports a different layer of development.

**✅ Step 3:**

powershell

CopyEdit

New-Item -Path ".\project-backlog\backlog.md" -ItemType File

New-Item -Path ".\scrum\day1-daily-scrum.md" -ItemType File

New-Item -Path ".\terminal-setup\setup-steps.md" -ItemType File

New-Item -Path ".\html-forms\register.html" -ItemType File

New-Item -Path ".\html-forms\login.html" -ItemType File

New-Item -Path ".\sprint-1-code\server.js" -ItemType File

🧾 **What It Does:**  
Creates individual **files** inside those folders:

* backlog.md → for writing Sprint 1 stories from the project brief
* day1-daily-scrum.md → log what you did, doing, blockers (Scrum standup)
* setup-steps.md → explain your terminal setup process, documented
* register.html + login.html → UI forms (for creating/logging into accounts)
* server.js → backend server (e.g., Express app) to connect logic

🎯 **Why:**

* This matches **Scrum events** (Daily Scrum, Sprint Planning)
* You are **self-organizing your files** by purpose: UI, backend, planning, and logs
* Documentation and modularity align with "Definition of Done" in Agile

**✅ Optional Step:**

powershell

CopyEdit

code .

🧾 **What It Does:**  
Opens the entire **Week 1 folder as a VS Code project**.

🎯 **Why:**

* You now have everything structured.
* Can edit files in their appropriate folders.
* Can run terminal commands in the built-in VS Code terminal.

**🧠 Summary of Structure and Agile Match**

| **Folder** | **Purpose in Agile** | **Matches Scrum Role/Event** | **Your Project Use** |
| --- | --- | --- | --- |
| project-backlog | Product Backlog | Sprint Planning | Write user stories from EPIC |
| scrum | Team Log | Daily Scrum, Retro | Document what you did & plan |
| terminal-setup | DevOps Docs | Transparency | Explain your process (reproducible) |
| html-forms | GUI (View) | Sprint Task, Increment | Build Register/Login UI |
| sprint-1-code | Server (Controller) | Sprint Task, Increment | Write logic (e.g., account creation, messaging) |

**🔚 What You’ve Accomplished by Following These Steps**

✅ Aligned your project folder with Agile best practices  
✅ Broke stories into implementable tasks (Scrum-style)  
✅ Set up initial UI and logic files in a scalable way  
✅ Ready to build your first Sprint deliverables

Would you like me to now:

* 🔨 Fill in register.html, login.html, and server.js with actual Sprint 1 logic?
* 🧪 Show how to write tests or simulate user input?
* 📄 Add sample content for backlog.md or scrum log?