**1. Work Breakdown Structure (WBS)**

**WBS Outline**

**Phase 1: Requirements & Planning**  
    **1.1 Requirement Analysis**  
        - **1.1.1:** Gather requirements from the instructor and peers  
        - **1.1.2:** Document requirements and define the project scope  
    **1.2 Project Planning**  
        - **1.2.1:** Define project objectives and deliverables  
        - **1.2.2:** Develop an initial timeline and identify key milestones

**Phase 2: Design**  
    **2.1 UI/UX Design**  
        - **2.1.1:** Sketch basic wireframes for the interface  
        - **2.1.2:** Create detailed UI mockups using design software  
    **2.2 System Architecture Design**  
        - **2.2.1:** Define the overall application architecture (front-end, back-end, database)  
        - **2.2.2:** Select technologies and frameworks (e.g., HTML/CSS, JavaScript, Python/Flask)

**Phase 3: Development**  
    **3.1 Front-End Development**  
        - **3.1.1:** Develop HTML/CSS templates  
        - **3.1.2:** Implement JavaScript functionality for interactivity  
    **3.2 Back-End Development**  
        - **3.2.1:** Set up the server environment and database  
        - **3.2.2:** Develop core functionalities (e.g., add, edit, delete tasks)

**Phase 4: Testing**  
    **4.1 Unit Testing**  
        - **4.1.1:** Write test cases for individual modules  
        - **4.1.2:** Run tests and debug any issues  
    **4.2 User Acceptance Testing (UAT)**  
        - **4.2.1:** Gather feedback from classmates and the instructor  
        - **4.2.2:** Refine features based on feedback

**Phase 5: Deployment**  
    **5.1 Application Deployment**  
        - **5.1.1:** Deploy the application to a web hosting service  
        - **5.1.2:** Verify live functionality with final tests

**Phase 6: Maintenance & Updates**  
    **6.1 Post-Deployment Support**  
        - **6.1.1:** Monitor application performance  
        - **6.1.2:** Schedule periodic updates and bug fixes

**WBS Explanation**

The WBS was developed to decompose the To-Do List Application project into clear, manageable components. Each phase—from requirements and planning to deployment and maintenance—contains specific deliverables and actionable tasks. This structure ensures that all aspects of the project are systematically addressed and that tasks are organized in a logical sequence. Key milestones include completing the design phase, finishing development, and successfully deploying the application. The grouping of tasks reflects dependencies (e.g., development cannot start until design is finalized) and aligns with the project’s overall objectives. The assumptions include a well-defined scope based on class requirements and limited resources available during the semester.

**2. Gantt Chart**

**Gantt Chart Representation**

Below is a simplified textual representation of the project schedule.

| **Task** | **Start Date** | **End Date** | **Duration** | **Dependencies** |
| --- | --- | --- | --- | --- |
| **Phase 1: Requirements & Planning** | 09/01 | 09/07 | 7 days | None |
| - Requirement Analysis | 09/01 | 09/03 | 3 days | None |
| - Project Planning | 09/04 | 09/07 | 4 days | Follows Requirement Analysis |
| **Phase 2: Design** | 09/08 | 09/14 | 7 days | After Phase 1 |
| - UI/UX Design | 09/08 | 09/10 | 3 days | Follows Project Planning |
| - System Architecture Design | 09/08 | 09/14 | 7 days | Follows Project Planning |
| **Phase 3: Development** | 09/15 | 09/28 | 14 days | After Phase 2 |
| - Front-End Development | 09/15 | 09/21 | 7 days | After UI/UX Design |
| - Back-End Development | 09/15 | 09/21 | 7 days | After System Architecture Design |
| **Phase 4: Testing** | 09/29 | 10/05 | 7 days | After Phase 3 |
| - Unit Testing | 09/29 | 10/01 | 3 days | Follows Development |
| - User Acceptance Testing | 10/02 | 10/05 | 4 days | Follows Unit Testing |
| **Phase 5: Deployment** | 10/06 | 10/07 | 2 days | After Phase 4 |
| **Phase 6: Maintenance & Updates** | 10/08 | Ongoing | - | Follows Deployment |

**Gantt Chart Explanation**

The Gantt Chart converts the tasks identified in the WBS into a time-based schedule. Each phase is assigned a start and end date, with durations estimated based on the complexity and available resources. Dependencies are clearly marked, ensuring that tasks such as development and testing commence only after the design phase is completed. This chart helped identify critical milestones—such as the end of the design and development phases—and allowed us to plan for contingencies. Resource constraints, such as the limited number of hours available during the semester, were taken into account by scheduling parallel work streams where possible (e.g., overlapping front-end and back-end development). Overall, the Gantt Chart provided a visual timeline that helped prioritize tasks and monitor progress throughout the project lifecycle.

**3. Kanban Board**

**Kanban Board Layout**

For the agile management of tasks, I used a simple Kanban Board with three main columns: **To Do**, **In Progress**, and **Done**. Below is a textual representation of the board:

**To Do:**

* Gather requirements from instructor and peers
* Document project requirements
* Define project scope
* Create initial project timeline
* Sketch UI wireframes
* Develop detailed UI mockups
* Define system architecture
* Set up server environment and database
* Develop API endpoints for task management
* Write unit test cases
* Prepare for user acceptance testing
* Deploy application to hosting service
* Monitor application performance

**In Progress:**

* Front-End Development: Develop HTML/CSS templates
* Front-End Development: Implement JavaScript functionality
* Back-End Development: Develop core functionalities (add, edit, delete tasks)

**Done:**

* Requirement Analysis (completed on 09/03)
* Project Planning (completed on 09/07)
* UI/UX Design (completed on 09/10)
* System Architecture Design (completed on 09/14)
* Unit Testing (completed on 10/01)

**Kanban Board Explanation**

The Kanban Board provides an agile way to visualize task status throughout the project. Tasks are organized into three columns—**To Do**, **In Progress**, and **Done**—which allow the team to easily track the flow of work. This visualization makes it clear which tasks are pending, which are currently being worked on, and which have been completed. It is especially useful for quickly identifying bottlenecks (for example, if many tasks remain in the “In Progress” column) and reallocating resources as needed. The board’s simplicity facilitates daily stand-up meetings, where the team can update progress and adjust priorities on the fly. Overall, the Kanban Board was instrumental in managing the workload and ensuring steady progress on the project.

**Conclusion**

By applying a **Work Breakdown Structure**, a **Gantt Chart**, and a **Kanban Board** to my programming class project (the To-Do List Application), I was able to comprehensively plan, schedule, and track the project from inception to deployment. These tools provided clarity on task dependencies, milestones, and progress, ultimately ensuring that the project was executed efficiently and delivered on time.