**Developer Project Report Outline: Task and Wish Management Application for Children**

**1. Requirements Analysis**

**What is the aim and scope of the project?**

This project aims to develop a console-based application that allows children to track their daily/weekly tasks, earn points, and manage their wishes (products/activities) with family approval. The application will be developed in Java as a console-based program. It will include task management, wish management, points/level tracking, and data persistence. The program will use three input files: Commands.txt, Wishes.txt, and Tasks.txt.

**To-Do List**

* Task Management Module
  + Task addition functionality
  + Task listing functionality
  + Task completion and approval functionality
* Wish Management Module
  + Wish addition functionality
  + Wish listing functionality
  + Wish approval/rejection functionality
* Points and Level System
  + Points tracking
  + Level definition and calculation
* Data Persistence
  + Reading from files (Tasks.txt, Wishes.txt)
  + Writing to files
* Command Processing System
  + Processing commands from Commands.txt

**Who are the ideal users of the system?**

The system has three main user types:

1. **Child**: Can view tasks, complete them, and add wishes.
2. **Parent**: Can assign tasks to the child, approve tasks, approve/reject wishes, and add extra points.
3. **Teacher**: Can add tasks such as school assignments and approve them.

**User Scenarios in an Ideal System**

* **Child Scenario**:
  + The child can view a list of tasks filtered daily or weekly.
  + When tasks are completed, they can mark them as "Done".
  + They can add new wishes (products or activities).
  + They can view their current points and level.
* **Parent Scenario**:
  + The parent can add new tasks for the child (ADD\_TASK).
  + They can approve completed tasks and rate them (TASK\_CHECKED).
  + They can approve or reject the child's wishes (WISH\_CHECKED).
  + They can add extra points to the child (ADD\_BUDGET\_COIN).
* **Teacher Scenario**:
  + The teacher can add tasks like school assignments.
  + They can approve completed tasks and rate them.
* **System Operation**:
  + The user executes commands specified in Commands.txt in order.
  + Initial data is read from Tasks.txt and Wishes.txt.
  + Command outputs are displayed on the console.
  + When new tasks or wishes are added, the relevant files are updated.

**Functional Requirements and Program Behavior (I/O)**

**Inputs:**

* Commands from Commands.txt
* Initial data from Tasks.txt and Wishes.txt
* Command parameters (task/wish IDs, title, description, date, etc.)

**Outputs:**

* Command responses printed to the console
* Task lists
* Wish lists
* Points and level information
* Updated Tasks.txt and Wishes.txt files

**Functional Requirements:**

* Task Management: Adding, listing, completing, approving
* Wish Management: Adding, listing, approving/rejecting
* Points and Level System: Tracking points, calculating level
* Command Processing: Processing commands from Commands.txt
* Data Persistence: File read/write operations

**2. Software Design**

**2.1. Data Structures**

* **ArrayList<LinkedList<Task>>**: Used to store tasks. Provides dynamic sizing and easy access by index.
* **ArrayList<LinkedList<Wish>>**: Used to store wishes. Provides dynamic sizing and easy access.

**2.2. Object-Oriented Design**

Main classes and their responsibilities:

* **Task**: Represents a task.
  + Properties: taskId, title, description, deadline, startTime, endTime, points, status, assignedBy, rating
  + Methods: markAsDone(), approve(rating), calculatePoints()
* **Task1**: Extends Task, represents tasks with only a deadline.
* **Task2**: Extends Task, represents tasks with a specific activity time (start and end).
* **Wish**: Represents a wish.
  + Properties: wishId, title, description, status, requiredLevel
  + Methods: approve(level), reject()
* **Wish1**: Extends Wish, represents product wishes.
* **Wish2**: Extends Wish, represents activity wishes. Contains date and time information.
* **User**: Represents a user in the system.
  + Properties: userId, role
  + Methods: canAddTask(), canApproveTask(), canApproveWish()
* **Child**: Extends User, represents the child user.
  + Properties: points, level, tasks, wishes
  + Methods: completeTask(), addWish(), getFilteredTasks(filterType)
* **Parent**: Extends User, represents the parent user.
  + Methods: addTask(), approveTask(), approveWish(), addBudgetCoin()
* **Teacher**: Extends User, represents the teacher user.
  + Methods: addTask(), approveTask()
* **TaskManager**: Responsible for task management.
  + Methods: addTask(), listAllTasks(filter), markTaskAsDone(), approveTask()
* **WishManager**: Responsible for wish management.
  + Methods: addWish(), listAllWishes(), approveWish(), rejectWish()
* **PointSystem**: Manages the points and level system.
  + Methods: calculateLevel(), addPoints(), printBudget(), printStatus()
* **FileManager**: Manages file read/write operations.
  + Methods: readTasksFromFile(), writeTasksToFile(), readWishesFromFile(), writeWishesToFile(), readCommandsFromFile()
* **CommandProcessor**: Processes commands read from Commands.txt.
  + Methods: processCommand(), executeAddTask(), executeListAllTasks(), executeTaskDone(), etc.

**2.3. Algorithms and Flow**

**Task Addition Process (ADD\_TASK):**

[User executes ADD\_TASK command]

↓

[CommandProcessor parses the command and parameters]

↓

[TaskManager creates a new Task object]

↓

[Task is added to the tasks list]

↓

[FileManager updates the Tasks.txt file]

↓

[Confirmation message is shown to the user]

**Task Completion and Approval Process:**

[Child marks a task as completed (TASK\_DONE)]

↓

[TaskManager updates the task status to "Completed"]

↓

[Parent/Teacher checks the task (TASK\_CHECKED)]

↓

[TaskManager marks the task as "Approved" and calculates points]

↓

[PointSystem updates the child's points and level]

↓

[FileManager updates the files]

↓

[Confirmation message is shown to the user]

**Wish Addition and Approval Process:**

[Child adds a wish (ADD\_WISH)]

↓

[WishManager creates a new Wish object and sets status to "Pending Approval"]

↓

[Wish is added to the wishes list]

↓

[FileManager updates the Wishes.txt file]

↓

[Parent checks the wish (WISH\_CHECKED)]

↓

[Parent approves or rejects the wish]

↓

[WishManager updates the wish status]

↓

[FileManager updates the files]

↓

[Confirmation message is shown to the user]

**3. Implementation (Coding Phase)**

**Development Process**

1. Creating the project structure and basic classes
2. Developing file read/write functions
3. Creating the command processing infrastructure
4. Developing task management functions (ADD\_TASK, LIST\_ALL\_TASKS, TASK\_DONE, TASK\_CHECKED)
5. Developing wish management functions (ADD\_WISH, LIST\_ALL\_WISHES, WISH\_CHECKED)
6. Developing the points and level system (ADD\_BUDGET\_COIN, PRINT\_BUDGET, PRINT\_STATUS)
7. Completing the command file reading and processing mechanism
8. Debugging and testing

**Challenges Faced**

* Properly structuring the hierarchy of different task and wish types (Task1/Task2, Wish1/Wish2)
* Ensuring error-free file I/O operations
* Processing date/time formats correctly
* Creating the logic for points and level calculation
* Correctly parsing command parameters, especially for texts in quotes

**Support and Resources**

* Stack Overflow: Help with file operations and date formats
* Java API documentation: For ArrayList, HashMap, and file operations
* ChatGPT: Algorithm design and debugging support
* Team leader: Guidance on system design and error resolution

**4. Error Handling & Testing**

**Test Cases and Outputs**

| **Test Case** | **Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- |
| Task Addition | Adding a valid task by a parent | ADD\_TASK1 F 101 "Clean Your Room" "Put away toys" 2025-03-01 15:00 10 | "Task added successfully" | Same | Passed |
| Specific Time Task Addition | Adding a task with activity time | ADD\_TASK2 T 102 "School Project" "Science Fair" 2025-03-05 10:00 2025-03-05 12:00 15 | "Task added successfully" | Same | Passed |
| Invalid Task Addition | Adding a task with missing parameters | ADD\_TASK1 F 103 "Incomplete Task" | "Invalid parameters for ADD\_TASK1" | Same | Passed |
| Task Listing | Listing all tasks | LIST\_ALL\_TASKS | List of all tasks | Same | Passed |
| Task Completion | Child marking a task as done | TASK\_DONE 101 | "Task marked as done" | Same | Passed |
| Task Approval | Parent approving a completed task | TASK\_CHECKED 101 5 | "Task approved with 5 stars" | No output | Failed |
| Wish Addition | Child adding a product wish | ADD\_WISH1 W101 "Lego Set" "Price:150TL" | "Wish added successfully" | Same | Passed |
| Activity Wish Addition | Child adding an activity wish | ADD\_WISH2 W102 "Cinema" "Price:100TL" 2025-03-07 14:00 2025-03-07 16:00 | "Wish added successfully" | Same | Passed |
| Wish Approval | Parent approving a wish | WISH\_CHECKED W101 APPROVED 3 | "Wish approved at level 3" | No output | Failed |
| Wish Rejection | Parent rejecting a wish | WISH\_CHECKED W102 REJECTED | "Wish rejected" | No output | Failed |
| Point Addition | Parent adding points | ADD\_BUDGET\_COIN 50 | "50 coins added to budget" | Same | Passed |
| Budget Display | Showing the child's current points | PRINT\_BUDGET | "Current budget: 75 coins" | Same | Passed |
| Status Display | Showing the child's current level | PRINT\_STATUS | "Current level: 2" | Same | Passed |

**Bugs Encountered and Solutions**

1. **Bug**: Date format parsing error
   * **Cause**: Incorrect date format usage
   * **Solution**: Fixed by using the SimpleDateFormat class with the correct format ("yyyy-MM-dd HH:mm")
2. **Bug**: Issue parsing texts in quotes
   * **Cause**: String splitting operation was incorrectly handling spaces within quotes
   * **Solution**: Developed a custom parsing algorithm that preserves texts within quotes
3. **Bug**: Data loss during file updates
   * **Cause**: Approach of completely rewriting the file
   * **Solution**: Using append mode in file operations and making backups when necessary
4. **Bug**: Inconsistency in point calculation
   * **Cause**: Incorrect addition of points from approved tasks
   * **Solution**: Reviewed and fixed the point calculation logic

**Support and Resources for Bug Fixes**

* Stack Overflow: For date format and file operation issues
* Java documentation: For string operations and SimpleDateFormat usage
* Team leader: Guidance in solving complex errors
* Debugging tools: IDE's debugging features

**5. Version Control**

**Version Control Tool & Strategy**

* No version control for this project.

**6. Discussion / Process Evaluation**

**6.1. Requirements Analysis and Design Process**

**Support from the Team Leader**

* The team leader provided detailed explanations during the requirements analysis process
* Clarified ambiguous points and provided guidance on design decisions
* Support rating: 5/5

**Leadership and Feedback**

* The team leader actively provided feedback during the analysis and design process
* Efficiently managed design meetings
* Provided technical guidance during the prototype development process
* I learned to better understand object-oriented design principles during this process
* I improved myself in developing more detailed class diagrams

**Personal Observations & Reflections**

* Initially, I struggled with fully understanding system requirements and translating them into design
* Throughout the process, I improved my abilities to transition from requirements to technical design
* I made progress in creating UML diagrams and designing class hierarchies

**6.2. Implementation / Coding Phase**

**Support from the Team Leader**

* The team leader provided guidance on coding standards
* Helped with implementing complex algorithms
* Provided feedback through code reviews
* Support rating: 5/5

**Leadership and Feedback**

* The team leader assisted with technical issues and offered solution suggestions
* Coding sprints were efficiently organized
* I gained knowledge about clean code writing and debugging techniques
* I learned to use Java's OOP features more effectively

**Personal Observations & Reflections**

* Initially, I had difficulty creating complex class hierarchies
* Throughout the process, I gained experience with file operations and date/time manipulation in Java
* I made significant progress in error handling and exception management

**6.3. Error Handling & Version Control Process**

**Support from the Team Leader**

* The team leader provided guidance on creating test scenarios
* Support rating: 5/5

**Leadership and Feedback**

* The team leader actively helped with error resolution processes
* I learned about debugging techniques and systematic testing approaches

**Personal Observations & Reflections**

* Initially, I had deficiencies in creating systematic test scenarios

**Attachment**

Weekly meeting notes and other documents related to the project:

metin, diyagram, plan, paralel içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.taslak, çizim, diyagram, çizgi sanatı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.