# Software Design Document

**DESIGN REPORT FOR SENG-383**

Cem ÖZAL 202228203

Git: https://github.com/cozalss/SENG383-project

## Project 2: KidTask - Task Management App

### 1. Introduction

#### 1.1 Purpose

This document outlines the design for "KidTask," a Java-based GUI application (using Swing or JavaFX). The application allows children to manage tasks and wishes, track points, and level up, all under the supervision and approval of parents or teachers.

#### 1.2 Scope

The system will:

* Provide distinct interfaces and functionalities for three user roles: Child, Parent, and Teacher.
* Allow Child users to view/complete tasks and add wishes.
* Allow Parent users to add/approve tasks and wishes.
* Allow Teacher users to add/rate school-related tasks.
* Track a Child's points and level, displayed visually.
* Persist all data (users, tasks, wishes) to local files (e.g., JSON or CSV).

### 2. System Architecture

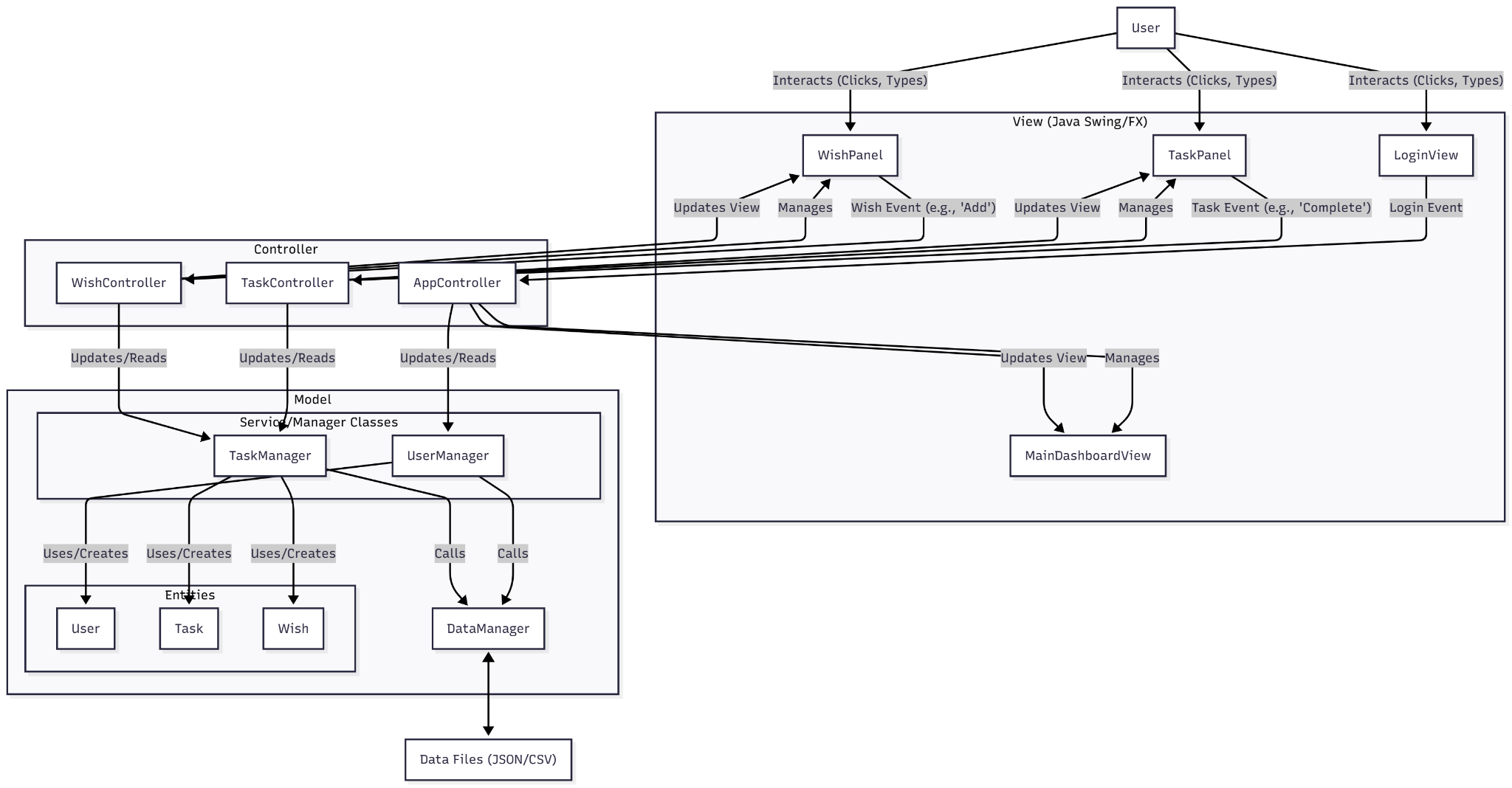
#### 2.1 Architectural Pattern

The **Model-View-Controller (MVC)** pattern will be used. This is a standard and effective pattern for Java Swing/JavaFX applications.

* **Model:** Represents all data and business logic. This includes the User, Task, and Wish data classes, as well as "Manager" classes (UserManager, TaskManager) that hold the data lists and perform operations. It also includes the DataManager for file I/O.
* **View:** The GUI, composed of JPanel or JavaFX scenes. This includes the LoginView, MainDashboardView, and sub-panels like TaskPanel and WishPanel.
* **Controller:** Manages application flow and user actions. An AppController will manage view-switching (e.g., from Login to Dashboard). Sub-controllers will handle logic for specific panels (e.g., TaskController handles "complete task" clicks).

#### 2.2 System Architecture Diagram

This diagram shows the high-level interaction between the MVC components and the data flow from the file system.



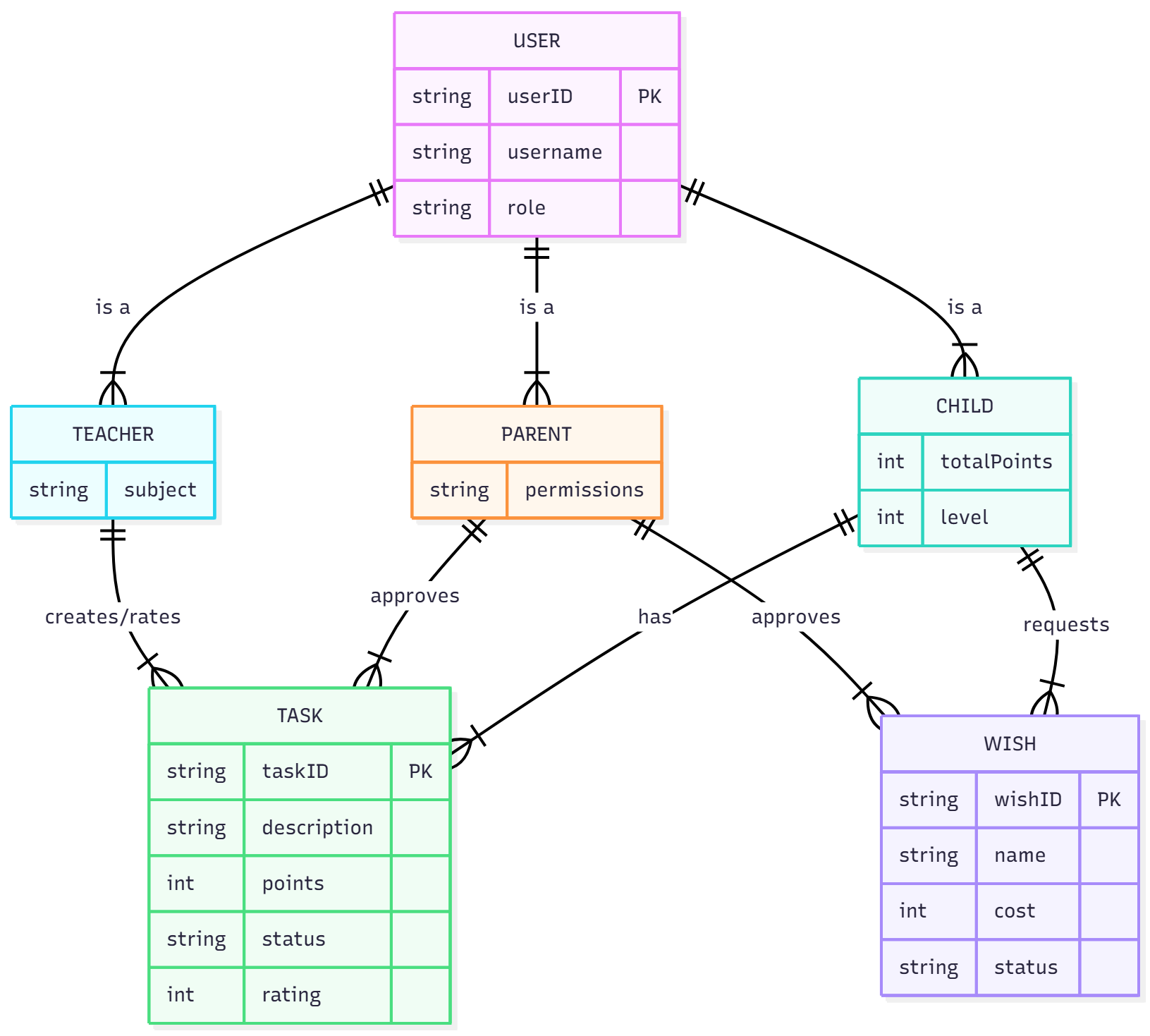
### 3. Data Design

#### 3.1 Data Storage

Data will be persisted in files, as specified. JSON is the recommended format for its ability to store structured objects and lists. The system will use files like users.json, tasks.json, and wishes.json.

#### 3.2 Entity-Relationship (ER) Diagram

This ERD shows the logical relationships between the system's data entities.



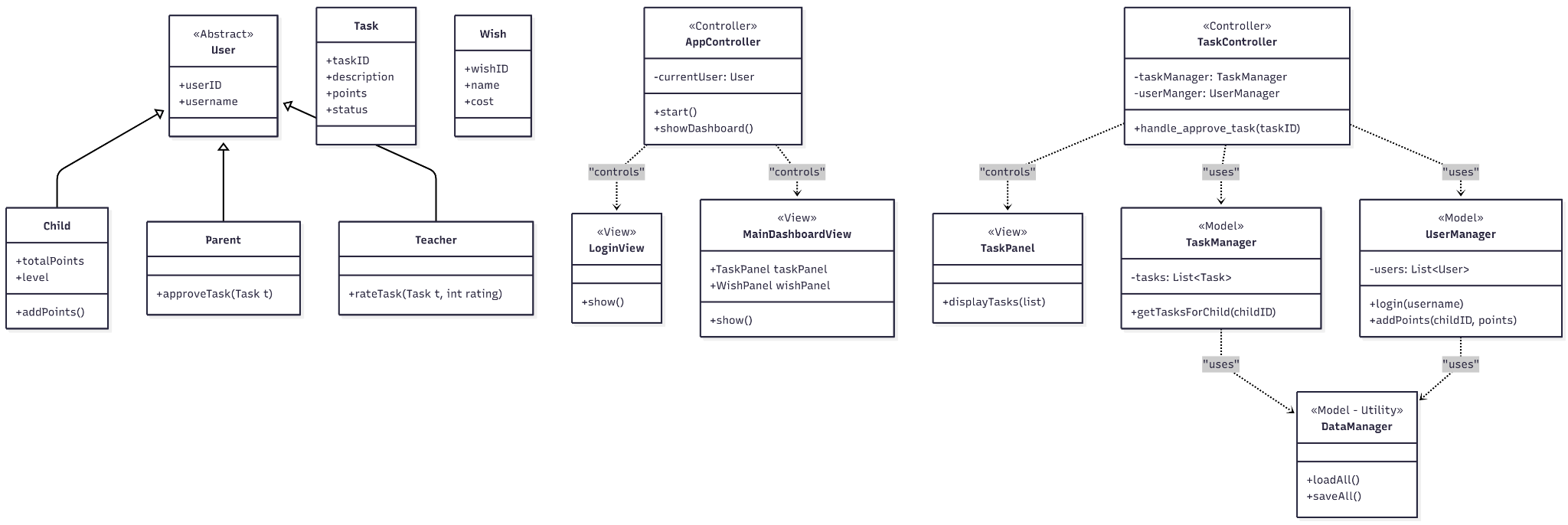
**ERD Explanation:**

* **User (Superclass):** A base entity with userID and username. This is specialized into three roles.
* **Child:** Inherits from User. Has totalPoints and level. A Child *has* many Tasks and *requests* many Wishes.
* **Parent:** Inherits from User. Can approve Tasks and Wishes.
* **Teacher:** Inherits from User. Can create and rate Tasks.
* **Task:** Has taskID, description, points, status ("New," "PendingApproval," "Completed"), and rating. It is assignedTo one Child.
* **Wish:** Has wishID, name, cost (in points), and status ("Pending," "Approved"). It is requestedBy one Child.

### 4. Class & Component Design

#### 4.1 UML Class Diagram

This diagram details the primary classes, highlighting the use of inheritance for user roles and the separation of concerns between managers, controllers, and views.

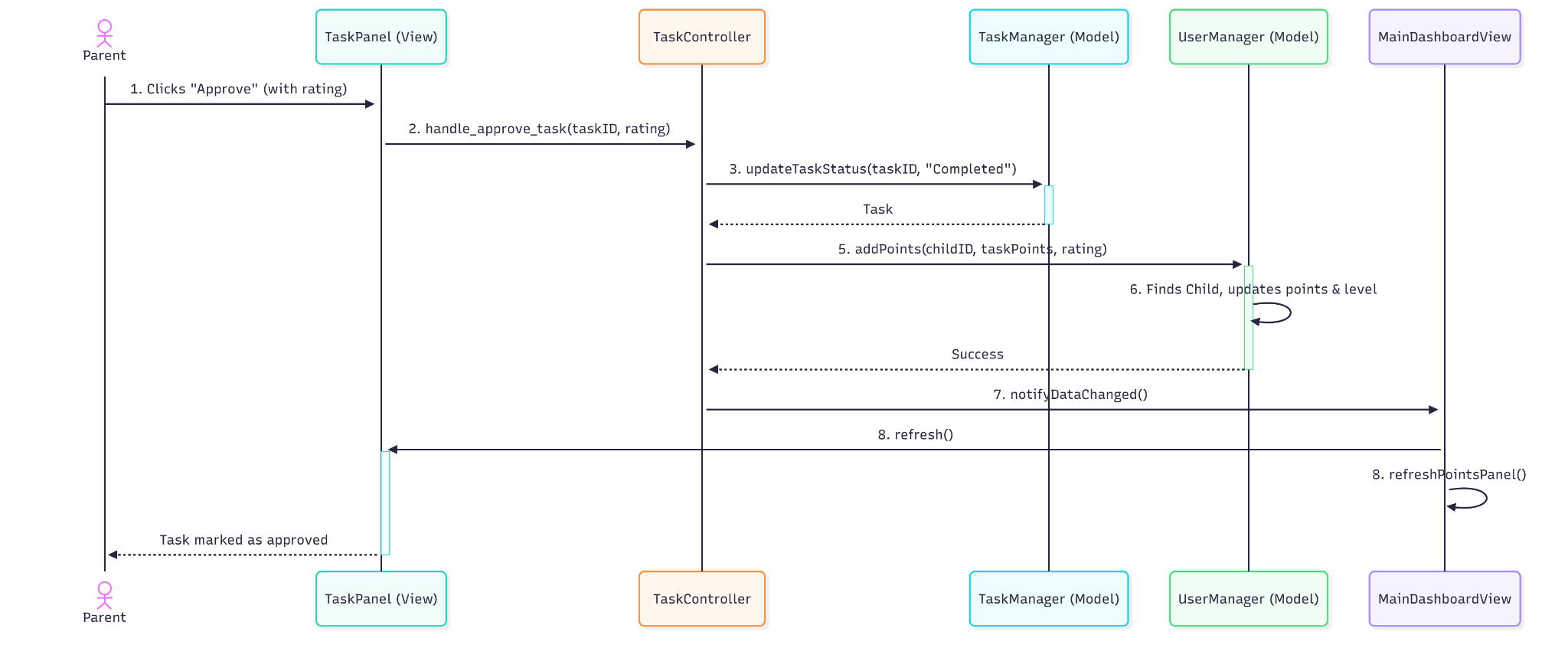


**Class Descriptions:**

* **User (Abstract Class):** Base class for all users.
* **Child, Parent, Teacher (Entities):** Concrete classes inheriting from User, each with its own specific attributes and methods (e.g., Child.addPoints()).
* **Task, Wish (Entities):** Simple data-holding classes.
* **LoginView (View):** The initial screen for role selection/login.
* **MainDashboardView (View):** The main application window, which acts as a container for the other panels (TaskPanel, WishPanel, PointsPanel).
* **TaskPanel (View):** A JPanel that displays a list of tasks with buttons (e.Do, "Complete," "Approve").
* **AppController (Controller):** Manages the "session" (e.g., currentUser) and switches between the LoginView and MainDashboardView.
* **TaskController (Controller):** Handles all logic related to the TaskPanel.
* **UserManager, TaskManager (Model):** Service classes that manage the application's state (e.g., List<User>, List<Task>).
* **DataManager (Model):** Utility class for reading/writing all data to/from JSON files.

### 5. Behavioral Design (Dynamic)

#### 5.1 UML Sequence Diagram: "Parent Approves Task" Use Case

This diagram shows the detailed interaction when a Parent user approves a completed task, which triggers a point update.

**Sequence Flow:**

1. The Parent user clicks the "Approve" button for a "PendingApproval" task on the TaskPanel (View).
2. The TaskPanel (View) calls the handle\_approve\_task(taskID, rating) method on the TaskController.
3. The TaskController tells the TaskManager (Model) to update the task's state (task.setStatus("Completed")).
4. The TaskController then retrieves the Task object to get its point value and the associated childID.
5. The TaskController calls addPoints(childID, taskPoints, rating) on the UserManager (Model).
6. The UserManager finds the Child object and updates its totalPoints and recalculates its level.
7. The TaskController then notifies the MainDashboardView that data has changed.
8. The MainDashboardView tells its sub-panels (TaskPanel, PointsPanel) to refresh their data, resulting in the UI updating to show the new point total and the task's completed state.

**6. GUI Design**

The GUI for KidTask was designed entirely using **canva**, leveraging its AI-powered layout generation and refinement capabilities. Gemini provided smart interface suggestions, which were then polished with additional manual adjustments. The goal was to deliver a clean, friendly, and highly intuitive interface tailored for children, parents, and teachers. Below are the main screens produced during the design process.

6.1. Role Selection Screen

The first page users encounter.

It allows them to choose between Child, Parent, and Teacher roles.

metin, çiçek saksısı, oyuncak, çizgi film içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

6.2. Child Dashboard

Displays the child’s level, total points, and next-level progress.

Includes quick statistics for:

-Pending tasks

-Completed tasks

-Approved tasks

-Approved wishes

metin, oyuncak, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

6.3. Child Task List

Shows all tasks assigned to the child.

Uses a minimal layout with an empty-state card when no tasks exist.

metin, çizgi film içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

6.4. Child Wishes Page

Displays child’s wishes ordered by level requirement.

Includes a level-based hint box and a CTA

metin, oyuncak, çizgi film, tablet bilgisayar içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

6.5. Parent Dashboard

A management-focused screen where parents can monitor the child’s progress

metin, oyuncak, işletim sistemi, tablet bilgisayar içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

6.6.Add Task Modal

A pop-up form used by parents to create new tasks

Contains structured inputs for:

-Task name

-Description

-Deadline

-Points

metin, tablet bilgisayar, işletim sistemi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**7. AI Tool Evaluation**

* **Canva**
  + **Purpose:** Creating the UML diagrams.
  + **What Went Well:**
    - Had lots of good-looking, pre-built templates for diagrams.
    - The auto-layout and alignment tools made the diagrams look clean and organized.
  + **What Was Difficult:**
    - The AI feature didn't actually *create* the UML diagrams for us; we had to build them all manually, step-by-step.
    - Many of the best templates were locked behind a Pro account.
    - Connecting arrows and lines in UML was tricky and often needed manual fixing.
  + **Verdict:** Good for making diagrams look nice quickly, but it wasn't the "AI" help we expected. It still required a lot of manual work.
* **Figma (with AI)**
  + **Purpose:** Designing the GUI mockups for KidTask.
  + **What Went Well:**
    - The AI feature was incredibly fast. We just described the screens, and it generated them.
    - It was great at creating a consistent look (colors, fonts) across all the different screens (Child, Parent, Teacher).
    - Made prototyping and auto-spacing components very easy.
  + **What Was Difficult:**
    - It didn't generate *everything*. We still had to go in and tweak components and add details.
  + **Verdict:** Great for GUI design. It saved a huge amount of time and helped us get a professional looking design to follow.

**8. Reflection**

Using AI tools for this design phase was a real eye opener, and the experience was very different between the two tools.

Figma's AI was the clear winner for us. Being able to just describe the KidTask app and see it generate complete, accurate screens was a huge time-saver. It handled the boring parts, like making sure the color palette and layout were the same for all user roles, which let us focus on the user flow. The design it gave us felt friendly and usable from the start.

Canva was more of a mixed bag. We used its components to design the BeePlan GUI, and it helped us create a simple, readable interface (with the yellow accent and gray sidebar) that wasn't too cluttered. But when it came to the UML diagrams, its "AI" wasn't what we expected. It's more of a template tool. We couldn't just *describe* the diagram; we had to build it all manually. The free templates were okay, but we spent a lot of time fixing connectors and adjusting layouts to be proper UML.

Overall, we learned that "AI" means different things. Figma's AI felt like a partner that did the heavy lifting for our GUI, while Canva felt more like a traditional (but good) design tool that just had some AI features. Figma's AI is definitely more advanced and was more helpful for our project.