# The Deadline Crusher

## Team Project

A productivity-focused web application
designed to help students **crush deadlines early**

## Team Members

- **Landen Tomlin** – Team Leader
- **Josh Day** – Backend Development, Database Design
- **Tanner Andrews** – Frontend Development, Calendar & Time Tracking
- **Xander Murphy** – Frontend Development, UI Customization

## Project Description

**The Deadline Crusher** helps students manage assignments, deadlines, and daily responsibilities in a structured and motivating way.

The application combines:

- Smart task organization
- Calendar-based planning
- Gamified customization rewards

## Problem Domain

Many students struggle with:

- Managing multiple deadlines across courses
- Knowing which tasks are most urgent
- Planning work over time
- Staying motivated to complete tasks

Traditional to-do lists lack prioritization and engagement.

## Our Solution

The Deadline Crusher solves these problems by:

- Automatically organizing tasks into smart lists
- Visualizing deadlines using a calendar interface
- Rewarding productivity with unlockable customization options

# Core Features Overview

- Calendar / Time-Based Checklist
- Smart Lists & Custom Views
- Gamified Customization & Rewards

Built using a **3-tier architecture**

# Feature 1: Calendar / Time-Based Checklist

**Owner:** Tanner Andrews

## Description

A calendar interface that helps users plan and track tasks over time.

## Requirements

- Interactive calendar interface
- Tasks associated with due dates
- Upcoming and overdue indicators
- Time tracking per task

# Feature 2: Smart Lists & Custom Views

**Owner:** Josh Day

## Description

Automatically generated task lists and customizable views.

## Requirements

- Auto lists:
  - Today
  - Upcoming
  - Overdue

# Feature 2 Requirements Cont.

- Filters:
  - Tag
  - Priority
  - Due date
- Sorting options
- Search bar
- Save and pin custom views

# Feature 3: Gamified Customization

**Owner:** Xander Murphy

## Description

Users unlock visual customization rewards by completing tasks.

## Requirements

- Unlockable fonts, backgrounds, and themes
- Progress tracking tied to task completion
- Customization menu
- Persistent unlocked rewards

# Non-Functional Requirements

- Three-tier architecture
- Web-based application
- Responsive UI (desktop & mobile)
- Persistent user data
- Scalable and maintainable codebase
- Intuitive and consistent interface

# Data Model Overview

## Core Entities

- User
- Task
- Custom View
- Customization Item

## Relationships

- A user has many tasks
- A user can save multiple custom views
- A user can unlock multiple customization items

## System Architecture

The Deadline Crusher uses a **three-tier architecture**:

1. Presentation Layer
2. Application Layer
3. Data Layer

Each layer is independently maintained and scalable.

## Presentation Layer (Frontend)

- JavaScript with **React**
- Component-based UI
- Handles user interaction and rendering
- Communicates with backend via REST API

## Application Layer (Backend)

- **Node.js** runtime
- **Express.js** framework
- RESTful API endpoints
- Handles business logic and validation

## Data Layer (Databases)

- **MongoDB**
  - Document-based storage
  - Mongoose ODM for schema modeling
- **SQLite**
  - Relational data storage
  - Object-relational database access

# APIs & Data Access

- RESTful API design using Express
- JSON-based client–server communication
- Mongoose ODM for MongoDB
- SQLite APIs for relational data
- Supports both document and relational models

# Testing Strategy

## Acceptance Tests

- Users can create and complete tasks
- Calendar displays tasks correctly
- Smart lists update automatically
- Customization unlocks on completion

# Testing Strategy (Continued)

## Integration Tests

- Frontend communicates correctly with REST API
- Filters and sorting work together
- Custom views persist across sessions

## End-to-End Tests

- Task creation → completion → reward unlock
- Saved views reload correctly
- User data persists across reloads

# Schedule & Milestones

## Sprint 1

- Project setup (React, Node.js, Express)
- Core task CRUD functionality
- Smart lists (Today / Upcoming / Overdue)
- Initial calendar UI
- Database schema design

# Schedule & Milestones

## Sprint 2

- REST API expansion
- Time tracking functionality
- Filters, sorting, and saved views
- Customization unlock system
- UI polish and testing

# Technology Stack

## Frontend

- JavaScript
- React

## Backend

- Node.js
- Express.js
- REST API

## Database

- MongoDB with Mongoose ODM
- SQLite with relational APIs